Human Behavior Modeling in Long Videos: Drowsiness Detection and Action Segmentation

Thesis Submitted in Fulfillment of the Requirements for the Degree of
PhD in Computer Science

by

Reza Ghoddoosian
1001390466

Supervised by: Prof. Vassilis Athitsos

Department of Computer Science and Engineering
UNIVERSITY OF TEXAS AT ARLINGTON

May 2022

Abstract

In this thesis we focus on two instances of human behavior modeling in long untrimmed videos: drowsiness detection, and action segmentation. In the first section, we focus on drowsiness detection. Specifically, we introduce a large and public real-life dataset and a baseline temporal model to classify drowsiness into three stages of alert, low vigilant, or drowsy. In the second section, we study action segmentation in instructional videos under weak supervision. In order to save time and cost, weakly supervised methods are trained based on only video-level action sequences as opposed to a fully supervised method which is trained using frame-level labels. We study weakly-supervised action segmentation from multiple aspects. First, we present a duration model to predict the remaining duration of an ongoing action to iteratively align a given sequence of action in an input video. Second, we propose a hierarchical approach to segmentation, where top level tasks are predicted to constrain lower level atomic actions. Third, we introduce the first weakly-supervised online action segmentation model to segment streaming videos online at test time using Dynamic Programming and show its advantages over greedy sliding window approach. Finally, we present a multi-view training strategy to exploit frame-wise correspondence between multiple views as supervision for training weakly-labeled instructional videos. The experimental results on multiple public datasets show the efficacy of our algorithms.

Dedicated to the computer vision research community...

Acknowledgments

While acknowledging the imperfection in my PhD work, I'm lucky to be proud and content about the lessons learned and the honest projects done during my program. A journey that would not have been possible to make without the support of many behind the scenes. I'd like to deeply thank some of those names: My parents, Ali and Noushin, my sister, Nazanin, my aunts, Maryam and Nasrin, my deceased grandfather, Gholam Hossein, my brother from another mother, Mohammad Keshavarzi, my close friends, Carlos Arocha and Mathew Zinke, my cousins Hosein and Yasmin, and finally all the volunteers who helped for data collection.

Furthermore, I would like to acknowledge the contribution of co-authors and lab mates during my PhD projects, specially Saif Sayed and Marnim Galib, as my partners in crime, and Prof. Dr. Vassilis Athitsos as my supervisor. To honor the work of the aforementioned people and others, I use the plural first person pronoun (we/us) across this thesis.

All the good associated with my name is credited to my dear parents Ali Ghoddosian and Noushin Zarifgolzar. I'm a traveler of life with a light backpack. May this journey bring smiles to humanity somewhere, somehow, sometime...

Table of Contents

1 Introduction

Automatic understanding of human behavior has many applications in surveillance and human-machine interaction. Examples of such applications are in driver assistant systems and smart factories where robots and humans interact to assemble a part or complete a task. In this thesis we focus on two instances of human behavior modeling in long untrimmed videos: drowsiness detection, and action segmentation.

In the first section, we focus on drowsiness detection. Specifically, we introduce a large and public real-life dataset of 60 subjects, with video segments labeled as alert, low vigilant, or drowsy. This dataset consists of around 30 hours of video, with contents ranging from subtle signs of drowsiness to more obvious ones. we also benchmark a temporal model for our dataset, which classifies drowsiness into three stages through processing blinking pattern over time.

In the second section, we study action segmentation in instructional videos under weak supervision. In the segmentation problem, the goal is to temporally partition an input video into a sequence of actions. In order to save time and cost, weakly supervised methods are trained based on only video-level action sequences as opposed to a fully supervised method which is trained using frame-level labels. Instructional videos in particular are long videos where a dense number of actions takes place by a human subject to complete a top-level task. Cooking and assembly are specific domains of instructional videos which we focus on in this thesis.

We study weakly-supervised action segmentation from multiple aspects. First, we present a duration model to predict the remaining duration of an ongoing action to iteratively align a given sequence of action in an input video. Second, we propose a hierarchical approach to segmentation. In this approach, we first predict the top-level task of the video and use the predicted task to constrain the segmentation results. We show both the efficiency and the efficacy of low-level segmentation results can be improved upon our proposed hierarchical paradigm. Third, we introduce the first weakly-supervised online action segmentation model to segment streaming videos online at test time using

Dynamic Programming and show its advantages over greedy sliding window approach. Finally, we present a multi-view training strategy to exploit frame-wise correspondence between multiple views as supervision for training weakly-labeled instructional videos. Our experimental results show that such a multi-view supervision improves performance of single view online segmentation models at test time.

In summary, the main contributions in this thesis are as follows:

- We present a large and public real-life dataset of 60 subjects in three drowsiness classes. This dataset consists of around 30 hours of video, with contents ranging from subtle signs of drowsiness to more obvious ones.

- We introduce a Duration Network for action alignment, that is explicitly designed to exploit information from video features and show its edge over the Poisson model used in previous work [1, 2].

- We present a novel top-down approach for weakly-supervised action segmentation, where the video-level task is used to constrain the segmentation output.

- We are the first to address the problem of weakly-supervised online action segmentation in instructional videos, and offer a DP-based framework.

- We use frame-wise multi-view correspondence, during training only, to generate more accurate action pseudo-ground-truth in weakly-labeled videos with no additional annotation cost. Our work is the first to incorporate multi-view video understanding in action segmentation.

Finally, the list of published papers that constitute this thesis is provided below:

- Reza Ghoddoosian, Isht Dwivedi, Nakul Agarwal, Chiho Choi and Behzad Dariush. Weakly-Supervised Online Action Segmentation in Multi-View Instructional Videos. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022

- Reza Ghoddoosian, Saif Sayed, and Vassilis Athitsos. Hierarchical modeling for task recognition and action segmentation in weakly- labeled instructional videos. *In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022

- Reza Ghoddoosian, Saif Sayed, and Vassilis Athitsos. Action duration prediction for segment-level alignment of weakly-labeled videos. *In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2053–2062, 2021

- Reza Ghoddoosian, Marnim Galib, and Vassilis Athitsos. A realistic dataset and baseline temporal model for early drowsiness detection. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019

Part 1:
Early Drowsiness Detection

2 A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection

2.1 Abstract

Drowsy driving can be as dangerous as drunk driving, and it is a constant threat for every driver on the road. It is imperative to find an economical way for the general public to detect the onset of drowsiness. In this chapter, we address early drowsiness detection, which can provide drivers ample time to react. We present a large and public real-life dataset[1] of 60 subjects in three drowsiness classes. This dataset consists of around 30 hours of video, with contents ranging from subtle signs of drowsiness to more obvious ones. We also benchmark a temporal model[2] for our dataset, which is computationally suitable for cell phones. We detect drowsiness via an intermediate regression step that rates the drowsiness level with a score. The core of our proposed method is a Hierarchical Multiscale Long Short-Term Memory (HM-LSTM) network, that is fed by detected blink features in sequence. Our experiments highlight the temporal relationship between blinks. In the experimental results, our baseline method produces higher accuracy than human judgment.

2.2 Introduction

Drowsiness detection is an important problem. Successful solutions have applications in domains such as driving and workplace. For example, in driving, National Highway

[1] Available on: sites.google.com/view/utarldd/home

[2] Code available on: https://github.com/rezaghoddoosian

Figure 2.1: Sample frames from the RLDD dataset in the alert (first row), low vigilant (second row) and drowsy (third row) states.

Traffic Safety Administration in the US estimates that 100,000 police-reported crashes are the direct result of driver fatigue each year. This results in an estimated 1,550 deaths, 71,000 injuries, and $12.5 billion in monetary losses [3]. To put this into perspective, an estimated 1 in 25 adult drivers report having fallen asleep while driving in the previous 30 days [4, 5]. In addition, studies show that, when driving for a long period of time, drivers lose their self-judgment on how drowsy they are [6], and this can be one of the reasons that many accidents occur close to the destination. Research has also shown that sleepiness can affect workers' ability to perform their work safely and efficiently [7, 8]. All these troubling facts motivate the need for an economical solution that can detect drowsiness in early stages. It is commonly agreed [9, 10, 11] that there are three types of sources of information in drowsiness detection: Performance measurements, physiological measurements, and behavioral measurements.

For instance, in the driving domain, performance measurements focus on steering wheel movements, driving speed, brake patterns, and lane deviations. An example is the Attention Assist system by Mercedes Benz [12]. As practical as these methods can be, such technologies are oftentimes reserved for high-end models, as they are too expensive to be accessible to the average consumer. Performance measurements at workplace can be obtained by testing workers' reaction time and short-term memory [8]. Physiologi-

cal measurements such as heart rate, electrocardiogram (ECG), electromyogram (EMG), electroencephalogram (EEG) [13, 14] and electrooculogram (EOG) [14] can be used to monitor drowsiness. However, such methods are intrusive and not practical to use in the car or workspace despite their high accuracy. Wearable hats have been proposed as an alternative for such measurements [15], but they are also not practical to use for long hours.

Behavioral measurements are obtained from facial movements and expressions using non-intrusive sensors like cameras. In Johns's work [16], blinking parameters are measured by light-emitting diodes. However, this method is sensitive to occlusions, where some object such as a hand is placed between the light emitting diode and the eyes.

Phone cameras are an accessible and cheap alternative to the aforementioned methods. One of the goals of this chapter is to introduce and investigate an end-to-end processing pipeline that uses input from phone cameras to detect both subtle and more clearly expressed signs of drowsiness in real time. This pipeline is computationally cheap so that it could ultimately be implemented as a cell phone application available for the general public.

Previous work in this field mostly focused on detecting extreme drowsiness with explicit signs such as yawning, nodding off and prolonged eye closure [17, 10, 18]. However, for drivers and workers, such explicit signs may not appear until only moments before an accident. Thus, there is significant value in detecting drowsiness at an early stage, to provide more time for appropriate responses. The proposed dataset represents subtle facial signs of drowsiness as well as the more explicit and easily observable signs, and thus it is an appropriate dataset for evaluating early drowsiness detection methods.

Our data consists of around 30 hours of RGB videos, recorded in indoor real-life environments by various cell phone/web cameras. The frame rates are below 30 fps, which makes drowsiness detection more challenging, as blinks are not observed as clearly as in high frame-rate videos. The videos in the dataset are labeled using three class labels: alertness, low vigilance, and drowsiness (Fig.3.4). The videos have been obtained from 60 participants. The need for research in early drowsiness detection is further illustrated by experiments we have conducted, where we asked twenty individuals to classify videos from our dataset into the three predefined classes. The average accuracy of the human observers was under 60%. This low accuracy indicates the challenging nature of the early drowsiness detection problem.

In addition to contributing a large and public realistic drowsiness dataset, we also implement a baseline method and include quantitative results from that method in the experiments. The proposed method leverages the temporal information of the video using a

Hierarchical Multiscale LSTM (HM-LSTM) network [19] and voting, to model the relationship between blinking and state of alertness. The proposed baseline method produces higher accuracy than human judgment in our experimental results.

Previous work on drowsiness detection produced results on datasets that were either private [20] or acted [17, 10]. By "acted" we mean data where subjects were instructed to simulate drowsiness, compared to "realistic" data, such as ours, where subjects were indeed drowsy in the corresponding videos. The lack of large, public, and realistic datasets has been pointed out by researchers in the field [11, 17, 10].

Our work is motivated to some extent by the driving domain (i.e., camera angle and distance in our dataset, and the calibration period in our method as explained in Sec. 4.2). However, our dataset has not been obtained from driving and it does not capture some important aspects of driving such as night lighting, camera vibration due to car motion and head movements for monitoring the environment. Given these aspects of our dataset, we do not claim that our dataset and results represent driving conditions. At the same time, the data and the proposed baseline method can be useful for researchers targeting other applications of drowsiness detection, for example in workplace environments.

The proposed dataset offers significant advantages over existing public datasets for drowsiness detection, regardless of whether those existing datasets have been motivated by the driving domain or not: (a) it is the largest to date public drowsiness detection dataset, (b) the drowsiness samples are real drowsiness as opposed to acted drowsiness in [21], and (c) the data were obtained using different cameras. Each subject recorded themselves using their cell phone or web camera, in an indoor real-life environment of their choice. This is in contrast to existing datasets [21, 13] where recordings were made in a lab setting, with the same background, camera model, and camera position.

Other contributions of this chapter can be summarized as follows: (a) introducing, as a baseline method, an end-to-end real time drowsiness detection pipeline based on low frame rates resulting in a higher accuracy than that of human observers, and (b) combining blinking features with Hierarchical Multiscale Recurrent Neural Networks to tackle drowsiness detection using subtle cues. These cues, which can be easily missed by human observers, are useful for detecting the onset of drowsiness at an early stage, before it reaches dangerous levels.

2.3 Related Work

Drowsiness Detection has been studied over several years. In the rest of this section, a review of the available datasets and existing methods will be provided.

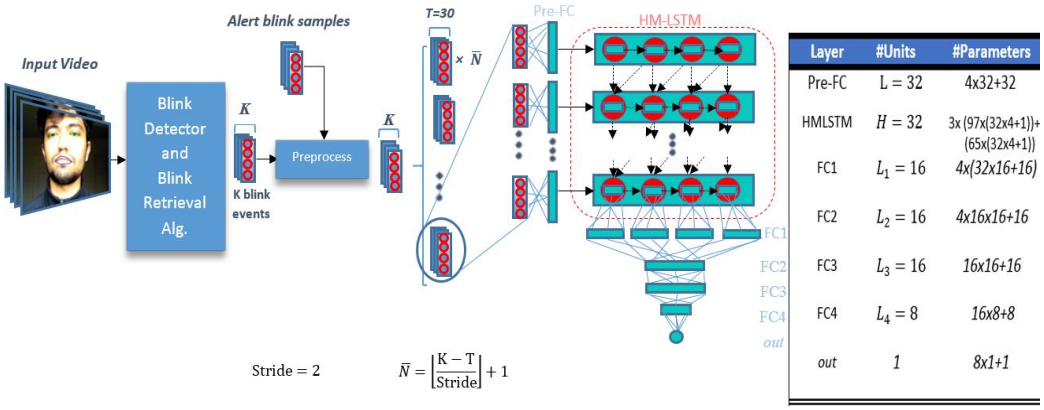


Figure 2.2: The model design and configuration.

2.3.1 Datasets

As pointed out above, there are numerous works in drowsiness detection, but none of them uses a dataset that is both public and realistic. As a result, it is difficult to compare prior methods to each other and to decide what the state of the art is in this area. Several existing methods [16, 22, 23, 9, 24] were evaluated on a small number of subjects without sharing the videos. In some cases [25, 10] the subjects were instructed to act drowsy, as opposed to obtaining data from subjects who were really drowsy.

Some datasets [26, 27, 28] have been created for short and general micro expression detection which are not applicable specifically for drowsiness detection. The NTHU-driver drowsiness detection dataset is a public dataset which contains IR videos of 36 participants while they simulate driving [21]. However, it is based on subjects pretending to be drowsy, and it is an open question whether and to what extent videos of pretended drowsiness are useful training data for detecting real drowsiness, especially at an early stage.

The DROZY dataset [13], contains multiple types of drowsiness-related data including signals such as EEG, EOG and near-infrared (NIR) images. An advantage of the DROZY dataset is that drowsiness data are obtained by subjects who are really drowsy, as opposed to pretending to be drowsy. Compared to the DROZY dataset, our dataset has three advantages: First, we have a substantially larger number of subjects (60 as opposed to 14). Second, for each subject, we have data showing that subject in each of the three predefined alertness classes, whereas in the DROZY dataset some subjects are not recorded in all three states. Third, in DROZY all videos were captured using the same camera position and background, under controlled lab conditions, whereas in our dataset

each subject used their own cell phone and a different background. Compared to DROZY, our dataset also has the important difference that it provides color video, whereas DROZY offers several other modalities, but only NIR video.

Last but not least, Friedrichs and Yang [20], used 90 hours of real driving to train and evaluate their method, but their dataset is private and not available as a benchmark.

2.3.2 Drowsiness Detection Methods

Features in non-intrusive drowsiness detection by cameras are divided into handcrafted features or features learned automatically using CNNs. Regarding handcrafted features, the most informative facial region about drowsiness is the eyes, and commonly used features are usually related to blinking behavior. McIntire *et al.* [22] show how blink frequency and duration normally increase with fatigue by measuring the reaction time and using an eye tracker. Svensson [14] has shown that amplitude of blinks can also be an important factor. Friedrichs and Yang [20] investigate many blinking features like eye opening velocity, average eye closure speed, blink duration, micro sleeps and energy of blinks as well as head movement information to decide the correlation of all these features with alertness, and use the sequential floating forward selection (SFFS) algorithm for feature selection. They report a final classification rate of 82.5% on their own private dataset, using three drowsiness classes as we do in this chapter. Their 82.5% accuracy is noticeably larger than the 65.2% accuracy that we report in our experiments. However, all the features in [20] are extracted using the Seeing Machines sensor [29] that uses not only video information (with the frame rate of 60 fps) but also the speed of the car, GPS information and head movement signals to detect drowsiness. In contrast, in our work the data comes from a cell phone camera or a web camera.

Recent research examines the effectiveness of Deep Neural Networks for end-to-end feature extraction and drowsiness detection, as opposed to the works that use handcrafted features with conventional classifiers or regressors such as regression and discriminant analysis (LDA) [23], or fitting a 2D Gaussian with thresholding [16]. The results of the mentioned studies were not validated based on a large or public dataset.

Park *et al.* [17] fine-tune three CNNs and apply an SVM to the combined features of those three networks to classify each frame into four classes of alert, yawning, nodding and drowsy with blinking. The model is trained on the NTHU drowsiness dataset that is based on pretended drowsiness, and tested on the evaluation portion of NTHU dataset which includes 20 videos of only four people, resulting in 73% drowsiness detection accuracy. We should note that the accuracy we report in our experiment is 65.2%, which is

lower that the 73% accuracy reported in [17]. However, the method of [17] was evaluated on pretended data, where the signs of drowsiness tend to be easily visible and even exaggerated. The work of Park *et al.*does not consider the temporal information in the videos and only classifies each frame independently, thus it can only classify based on the clear signs of drowsiness.

Bhargava *et al.* [10] show how a distillated deep network can be of use for embedded systems. This is relevant to the baseline method proposed in this chapter, which also aims for low computational requirements. The reported accuracy in [10] is 89% using three classes (alert, yawning, drowsy), based on training on patches of eyes and lips. Similar to Park *et al.*'s work, Bhargava *et al.*'s network also classifies each frame independently, thus not using temporal features. The dataset they used is private, and based on acted drowsiness, so it is difficult to compare those results to the results reported in this work.

2.4 The Real-Life Drowsiness Dataset (RLDD)

2.4.1 Overview

The RLDD dataset was created for the task of multi-stage drowsiness detection, targeting not only extreme and easily visible cases, but also subtle cases of drowsiness. Detection of these subtle cases can be important for detecting drowsiness at an early stage, so as to activate drowsiness prevention mechanisms. Our RLDD dataset is the largest to date realistic drowsiness dataset.

The RLDD dataset consists of around 30 hours of RGB videos of 60 healthy participants. For each participant we obtained one video for each of three different classes: alertness, low vigilance, and drowsiness, for a total of 180 videos. Subjects were undergraduate or graduate students and staff members who took part voluntarily or upon receiving extra credit in a course. All participants were over 18 years old. There were 51 men and 9 women, from different ethnicities (10 Caucasian, 5 non-white Hispanic, 30 Indo-Aryan and Dravidian, 8 Middle Eastern, and 7 East Asian) and ages (from 20 to 59 years old with a mean of 25 and standard deviation of 6). The subjects wore glasses in 21 of the 180 videos, and had considerable facial hair in 72 out of the 180 videos. Videos were taken from roughly different angles in different real-life environments and backgrounds. Each video was self-recorded by the participant, using their cell phone or web camera. The frame rate was always less than 30 fps, which is representative of the frame rate expected of typical cameras used by the general population.

1- Extremely alert
2- Very alert
3- Alert
4- Rather alert
5- Neither alert nor sleepy
6- Some signs of sleepiness
7- Sleepy, no difficulty remaining awake
8- Sleepy, some effort to keep alert
9- Extremely sleepy, fighting sleep

Table 2.1: KSS drowsiness scale

2.4.2 Data Collection

In this section we describe how we collected the videos for the RLDD dataset. Sixty healthy participants took part in the data collection. After signing the consent form, subjects were instructed to take three videos of themselves by their phone/web camera (of any model or type) in three different drowsiness states, based on the KSS table [30] (Table 2.1), for around ten minutes each. The subjects were asked to upload the videos as well as their corresponding labels on an online portal provided via a link. Subjects were given ample time (20 days) to produce the three videos. Furthermore, they were given the freedom to record the videos at home or at the university, any time they felt alert, low vigilant or drowsy, while keeping the camera set up (angle and distance) roughly the same. All videos were recorded in such an angle that both eyes were visible, and the camera was placed within a distance of one arm length from the subject. These instructions were used to make the videos similar to videos that would be obtained in a car, by phone placed in a phone holder on the dash of the car while driving (although, as we stated in the introduction, there are other important characteristics of driving conditions that our dataset does not capture). The proposed set up was to lay the phone against the display of their laptop while they are watching or reading something on their computer. After a participant uploaded the three videos, we watched the entire videos to verify their authenticity and to make sure that our instructions were followed. In case of any question, we contacted the participants and asked them to share more details about the situation under which they recorded each video. In some cases, we asked them to redo the recordings and if the videos were clearly not realistic (people faking drowsiness as opposed to being drowsy) or off the standard, we simply ignored those videos for quality reasons. The three classes were explained to the participants as follows:

1) **Alert**: One of the first three states highlighted in the KSS table in Table 2.1. Subjects were told that being alert meant they were experiencing no signs of sleepiness.

2) **Low Vigilant**: As stated in level 6 and 7 of Table 2.1, this state corresponds to subtle cases when some signs of sleepiness appear, or sleepiness is present but no effort to keep alert is required.

3) **Drowsy**: This state means that the subject needs to actively try to not fall asleep (level 8 and 9 in Table 2.1).

2.4.3 Content

This dataset consists of 180 RGB videos. Each video is around ten minutes long, and is labeled as belonging to one of three classes: alert (labeled as 0), low vigilant (labeled as 5) and drowsy (labeled as 10). The labels were provided by the participants themselves, based on their predominant state while recording each video. Clearly there is a subjective element in deciding these labels, but we did not find a good way to remedy that problem, given the absence of any sensor that could provide an objective measure of alertness. This type of labeling takes into account and emphasizes the transition from alertness to drowsiness. Each set of videos was recorded by a personal cell phone or web camera resulting in various video resolutions and qualities. The 60 subjects were randomly divided into five folds of 12 participants, for the purpose of cross validation. The dataset has a total size of 111.3 Gigabytes.

2.4.4 Human Judgment Baseline

We conducted a set of experiments to measure human judgment in multi-stage drowsiness detection. In these experiments, we asked four volunteers per fold (20 volunteers in total) to watch the unlabeled and muted videos in each fold and write down a real number between 0 to 10 estimating the drowsiness degree per video (see Table 2.1). Before the experiment, volunteers (8 female and 12 male, 3 undergraduates and 17 graduate students) were shown some sample videos that illustrated the drowsiness scale. Then, they were left alone in a room to watch the videos (they were allowed to rewind back or fast forward the videos at will) and annotate them. In order to make sure that each judgment was independent of the other videos of the same person, volunteers were instructed to annotate one video of each subject before annotating a second video for any subject. Results of these experiments are demonstrated in section 2.6.3 and compared with the results of our baseline method. Observers (aged 26.1 ± 2.9 (mean $\pm$ SD)) were from computer science, psychology, nursing, social work and information systems majors.

2.5 The Proposed Baseline Method

In this section, we discuss the individual components of our proposed multi-stage drowsiness detection pipeline. The blink detection and blink feature extraction are described first. Then we discuss how we integrate a Hierarchical Multiscale LSTM module into our model, how we formulate drowsiness detection initially as a regression problem, and how we discretize the regression output to obtain a classification label per video segment. Finally, we discuss the voting process that is applied on top of classification results of all segments of a video.

2.5.1 Blink Detection and Blink Feature Extraction

The motivation behind using blink-related features such as blink duration, amplitude, and eye opening velocity, was to capture temporal patterns that appear naturally in human eyes and might be overlooked by spatial feature detectors like CNNs (as it is the case for human vision shown in our experiments). We used dlib's pre-trained face detector based on a modification to the standard Histogram of Oriented Gradients + Linear SVM method for object detection [31].

We improved the algorithm by Soukupová and Cech [32] to detect eye blinks, using six facial landmarks per eye described in [33] to extract consecutive quick blinks that were initially missed in Soukupová and Cech's work. Kazemi and Sullivan's [33] facial landmark detector is trained on an "in-the-wild dataset", thus it is more robust to varying illumination, various facial expressions, and moderate non-frontal head rotations, compared to correlation matching with eye templates or a heuristic horizontal or vertical image intensity projection [32]. In our experiments, we noticed that the approach of [32] typically detected consecutive blinks as a single blink. This created a problem for subsequent steps of drowsiness detection, since multiple consecutive blinks can be a sign of drowsiness. We added a post-processing step (Blink Retrieval Algorithm), and applied on top of the output of [32], so as to successfully identify the multiple blinks which may be present in a single detection produced by [32]. Our post-processing step, while lengthy to describe, relies on heuristics and does not constitute a research contribution. To allow our results to be duplicated, we provide the details of that post-processing step as supplementary material.

The input to the blink detection module is the entire video (with a length of approximately ten minutes in our dataset). In a real-world application of drowsiness detection, where a decision should be made every few minutes, the input could simply consist of the last few minutes of video. The output of the blink detection module is a sequence of blink

events $\{\textbf{blink}_1, ..., \textbf{blink}_K\}$. Each $\textbf{blink}_i$ is a four-dimensional vector containing four features describing the blink: duration, amplitude, eye opening velocity, and frequency. For each blink event $\textbf{blink}_i$, we defined $start_i$, $bottom_i$, and end_i as the "start", "bottom" and "end" points (frames) in that blink (Fig.2.3a) explained in the Blink Retrieval Algorithm. Also, for each frame k, we denoted:

$$EAR[k] = \frac{||\vec{p}_2 - \vec{p}_6|| + ||\vec{p}_3 - \vec{p}_5||}{||\vec{p}_1 - \vec{p}_4||} \tag{2.1}$$

where $\vec{p}_i$ is the 2D location of a facial landmark from the eye region (Fig.2.3b). Using this notation, we define four main scale invariant features that we extract from $\textbf{blink}_i$. These are the features that we use for our baseline drowsiness detection method:

$$\text{Duration}_i = end_i - start_i + 1 \tag{2.2}$$

$$\text{Amplitude}_i = \frac{EAR[start_i] - 2EAR[bottom_i] + EAR[end_i]}{2} \tag{2.3}$$

$$\text{Eye Opening Velocity}_i = \frac{EAR[end_i] - EAR[bottom_i]}{end_i - bottom_i} \tag{2.4}$$

$$\text{Frequency}_i = 100 \times \frac{\text{Number of blinks up to blink}_i}{\text{Number of frames up to } end_i} \tag{2.5}$$

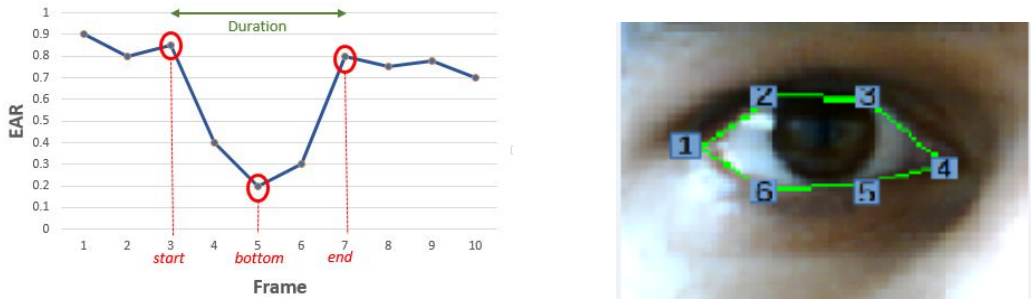
2.5.2 Drowsiness Detection Pipeline

Preprocessing: A big challenge in using blink features for drowsiness detection is the difference in blinking pattern across individuals [20, 25, 14, 34], so features should be normalized across subjects if we are going to train the whole data together at once. In order to tackle this challenge, we use the first third of the blinks of the **alert** state to compute the mean and standard deviation of each feature for each individual, and then use Equation 2.6 to normalize the rest of the alert state blinks as well as the blinks in the other two states of the same person(m) and feature(n):

$$\overline{\text{Feature}}_{n,m} = \frac{\text{Feature}_{n,m} - \mu_{n,m}}{\sigma_{n,m}} \tag{2.6}$$

Here, $\mu_{n,m}$ and $\sigma_{n,m}$ are the mean and standard deviation of feature n in the first third of the blinks of the alert state video for subject m.

We do this normalization for both the training and test data of all subjects and features. A similar approach has been taken in [25, 14]. This normalization is a realistic constraint: when a driver starts driving starts driving a new car or a worker starts working, the camera



(a) **(b)**

Figure 2.3: (a) The EAR sequence during an entire blink and the start, bottom and end points. (b) The eye landmarks to define EAR for each frame.

can use the first few minutes (during which the person is expected to be alert) to compute the mean and variance, and calibrate the system. This calibration can be used for all subsequent trips or sessions. The detector decides the state of the subject relative to the statistics collected during the calibration stage. We should clarify that, in our experiments, the alert state blinks used for normalization are never used again either for training or testing. After the per-individual normalization, we perform a second normalization step, where we normalize each feature so that, across individuals, the distribution of the feature has a mean of zero and a variance of one.

Feature Transformation Layer: Instead of defining a large number of features initially, and then selecting the most relevant ones[20], we let the network use the four main blink features and learn to map them to a higher dimensional feature space to minimize the loss function. The goal of the fully connected layer before the HM-LSTM module is to take each 4D feature vector at each time step as input and transform it to an L dimensional space with shared weights ($W \in \mathbb{R}^{4\times L}$) and biases ($\mathbf{b} \in \mathbb{R}^{1\times L}$) across time steps. Define $\mathbf{T}$ as the number of time steps used for the HM-LSTM Network and $\mathbf{f}_i \in \mathbb{R}^{1\times L}$ for each blink at each time step i , so that:

$$F = \text{ReLU}(BW + \overline{b}), \tag{2.7}$$

where $F = \left[\mathbf{f}_1^T, \mathbf{f}_2^T, ..., \mathbf{f}_\mathbf{T}^T\right]^T$, $\overline{b} = \left[\mathbf{b}^T, \mathbf{b}^T, ..., \mathbf{b}^T\right]^T$, $\overline{b} \in \mathbb{R}^{\mathbf{T}\times L}$ and $B = \left[\mathbf{blink}_1^T, ..., \mathbf{blink}_\mathbf{T}^T\right]^T$.

HM-LSTM Network: Our approach introduces a temporal model to detect drowsiness. The work by[9], using Hidden Markov Model (HMM), suggests that drowsiness features follow a pattern over time. Thus, we used an HM-LSTM network[19] to leverage the temporal pattern in blinking. It is also ambiguous how each blink is related to the other blinks or how many blinks in succession can affect each other. To remedy this challenge, we used HM-LSTM cells to discover the underlying hierarchical structure in a

blink sequence.

Chung *et al.*[19] introduces a parametrized boundary detector, which outputs a binary value, in each layer of a stacked RNN. For this boundary detector, positive output for a layer at a specific time step signifies the end of a segment corresponding to the latent abstraction level for that layer. Each cell state is "updated", "copied" or "flushed" based on the values of the adjacent boundary detectors. As a result, HM-LSTM networks tend to learn fine timescales for low-level layers and coarse timescales for high-level layers. This dynamic hierarchical analysis allows the network to consider blinks both in short and long segments, depending on when the boundary detector is activated for each cell. For additional details about HM-LSTM, we refer the readers to [19].

The HM-LSTM network takes each row of F as input at each time step and outputs a hidden state $\mathbf{h}_l \in \mathbb{R}^{1\times H}$ only at the last time step for each layer l. H is the number of hidden states per layer.

Fully Connected Layers: We added a fully connected layer (with $W_{1,l} \in \mathbb{R}^{H\times L_1}$ as weights and $\mathbf{b}_{1,l} \in \mathbb{R}^{1\times L_1}$ as biases) to the output of each layer l with L_1 units to capture the results of the HM-LSTM network from different hierarchical perspectives separately. Define $\mathbf{e}_{1l} \in \mathbb{R}^{1\times L_1}$ for each layer, so that:

$$\mathbf{e}_{1l} = \text{ReLU}(\mathbf{h}_l W_{1,l} + \mathbf{b}_{1,l}) \tag{2.8}$$

Then, we concatenated $\mathbf{e}_{1l}\ \forall\ l \in \{i|i = 1,2,...,\overline{L}\}$ to form $\mathbf{e}_1 = [\mathbf{e}_{11}, \mathbf{e}_{12}, ..., \mathbf{e}_{1\overline{L}}]$, where $\mathbf{e}_1 \in \mathbb{R}^{1\times(L_1.\overline{L})}$ and $\overline{L}$ is the number of layers.

Similarly, as shown in Fig. 3.3, $\mathbf{e}_1$ is fed to more fully connected layers (with ReLU as their activation functions) in FC2,FC3 and FC4, resulting in $\mathbf{e}_4 \in \mathbb{R}^{1\times(L_4)}$, where L_4 is the number of units in FC4.

Regression Unit: A single node at the end of this network determines the degree of drowsiness by outputting a real number from 0 to 10 depending on how alert or drowsy the input blinks are (Eq.2.9). This 0 to 10 scale helps the network to model the natural transition from alertness to drowsiness unlike the previous works [17, 10], where inputs were classified directly into different classes discretely.

$$out = 10 \times \text{Sigmoid}(\mathbf{e}_4 W_o + b_o) \tag{2.9}$$

Here, $W_o \in \mathbb{R}^{L_4\times 1}$ and $b_o \in \mathbb{R}^{1\times 1}$ are the regression parameters, and $out \in \mathbb{R}^{1\times 1}$ is the final regression output.

Discretization and Voting: When someone is drowsy, it does not mean that all their blinks will necessarily represent drowsiness. As a result, it is important to classify the

drowsiness level of each video as the most dominant state predicted from all blink sequences in that video. As the first step, we used Eq.4.14 to discretize the regression output to each of the predefined classes.

$$\text{class}(out) = \begin{cases} Alert, & 0.0 \leq out < 3.3 \\ LowVigilant, & 3.3 \leq out \leq 6.6 \\ Drowsy, & 6.6 < out \leq 10 \end{cases} \tag{2.10}$$

Suppose there are K blinks in video V. Using a sliding window of length T, each T consecutive blinks form a blink sequence that is given as input to the network (Eq.2.7), resulting in possibly multiple blink sequences. The most frequent predicted class from these multiple sequences would be the final classification result of video V. The positive effect of voting is shown later in our results.

Loss Function: Our model learns not to penalize predictions (out_i) that are within a certain distance $\sqrt{\Delta}$ of true labels (t_i) for all N training sequences, and instead penalizes less accurate predictions quadratically by their squared error. As a result, our model is more concerned about classifying each sequence correctly rather than perfect regression. This attribute helps us to jointly do regression and classification by minimizing the following loss function:

$$loss = \frac{\sum_{i=1}^{N} \max(0, |out_i - t_i|^2 - \Delta)}{N} \tag{2.11}$$

2.6 Experiments

2.6.1 Evaluation Metrics:

We designed four metrics to fully evaluate our model from different views and at various stages of the pipeline.

Blink Sequence Regression Error (BSRE): Using Eq.2.12, the regression results can be interpreted across all M test blink sequences. Eq.2.12 penalizes (only) each wrongly classified blink sequence i quadratic to the distance of the regressed output to the nearest true state border (S_i) defined in Eq.4.14.

$$BSRE = \frac{\sum_{i=1}^{M'} |out_i - S_i|^2}{M} \tag{2.12}$$

M' is the number of wrongly classified sequences in all M blink sequences with BSRE being zero for the correctly classified sequences.

(a) **(b)**

Figure 2.4: The effect of blink sequence size and Δ to the accuracy metrics.

Video Regression Error (VRE): This metric is best interpreted alongside the Video Accuracy (VA) metric across all Q test Videos. A low VRE and high VA means that most of the wrongly classified videos are misclassified closely for the adjacent class.

$$VRE = \frac{\sum_{j=1}^{Q'} |\frac{1}{K_j} \sum_{i=1}^{K_j} (out_{i,j}) - S_j|^2}{Q} \tag{2.13}$$

VRE is zero for all correctly classified videos. K_j is the number of all blink sequences in video j and Q' is the number of wrongly classified videos.

Blink Sequence Accuracy (BSA): This metric evaluates the results before "the voting stage" and after "discretization" across all test blink sequences.

Video Accuracy (VA): "Video Accuracy" is the main metric, that shines light to our final test classification results at the very end of the pipeline after "voting".

2.6.2 Implementation

We used one fold of the UTA-RLDD dataset as our test set, and the remaining four folds for training. After repeating this process for each fold, the results were averaged across the five folds. A sliding window of 30 consecutive blinks was used (Fig. 2.4a) as the input sequence fed to the network (videos with less than 30 blinks were zero padded). If the window size is too large, the long dependency on previous blinks can significantly delay the correct output while transitioning from one state to the other during driving. This window was shifted with the stride of two to create augmented sequences of blinks in time for each video . Then, we annotated all sequences with the label of the video they were taken from. Our model was trained on around 7000 blink sequences (depending on the training fold) using Adam optimizer [35] with a learning rate of 0.000053, Δ of 1.253 (Fig.2.4b), and batch size of 64 for 80 epochs in all five folds. We also used batch

Model	Evaluation Metric			
	BSRE	VRE	BSA	VA
HM-LSTM network	**1.90**	**1.14**	**54%**	**65.2%**
LSTM network	3.42	2.68	52.8%	61.4%
Fully connected layers	2.85	2.17	52%	57%
Human judgment	—	1.80	—	58.4%

Table 2.2: This table numerically compares the performance of our model with two simplified versions of the network and human judgment using four predefined metrics. The above values are the final averaged values across all test folds.

normalization and L2 regularization with a coefficient (λ) of 0.1. The HM-LSTM module has four layers with 32 hidden states for each layer. More details about the architecture is shown in Fig.3.3.

2.6.3 Experimental Results

In this section, we evaluate our baseline method with respect to the human judgment benchmark explained in section 2.4.4, and show that our pipeline results in a higher accuracy. Due to lack of a state-of-the-art method on a realistic and public dataset, we compare our baseline method with two variations of our pipeline to show that the whole pipeline performs best with HM-LSTM cells. The First version has the same architecture, as our network, with typical LSTM cells [36] used instead of HM-LSTM cells. The second version is a simpler version with the same architecture after removing the HM-LSTM module, where the input sequence is fed to a fully connected multilayer network.

The results of our comparison with these two versions and the human judgment benchmark are listed in Table 2.2. This table shows the final cross validation results of drowsiness detection by the predefined metrics. This comparison not only highlights the temporal information in blinks, but also shows the 4% increase in accuracy we gained after switching to HM-LSTM from typical LSTM cells. As indicated by BSRE and VRE metrics in Table 2.2, the margin of error for regression is also considerably lower in the HM-LSTM network compared to the other two. The results for LSTM and HM-LSTM networks suggest that temporal models provide better solutions for drowsiness detection than simple fully connected layers.

As mentioned before, all blink sequences in each video were labeled the same. However, in reality, not all blinks represent the same level of drowsiness. This discrepancy is

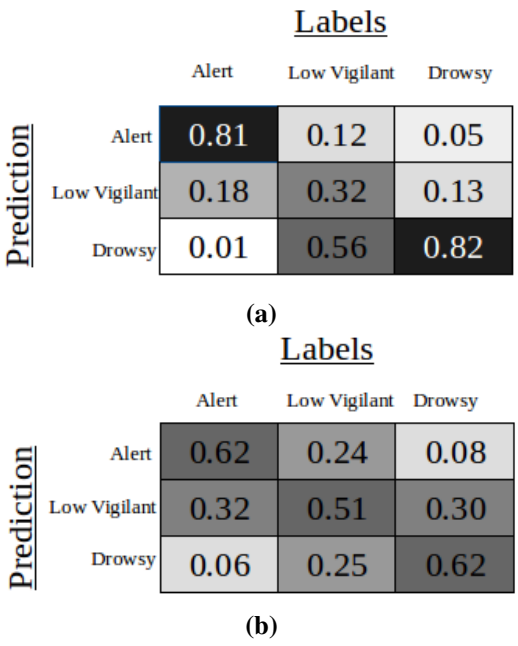


Figure 2.5: Confusion matrices for: (a) our proposed model and (b) human judgment results (video accuracy).

an important reason that BSA is not high, and "voting" makes up for that resulting in a higher accuracy in VA.

Fig.2.5a shows that the middle class (low vigilant) is, as expected, the hardest to classify, where it is mostly misclassified for "drowsy". On the other hand, our model classifies alert and drowsy subjects very confidently with over 80% accuracy, and rarely misclassifies alertness for drowsiness or vice versa. This means, that the results are mostly reliable in practice.

In addition, our model detects early signs and subtle cases of drowsiness better than humans in the UTA-RLDD dataset by just analyzing the temporal blinking behavior. The detailed quantitative results for all folds and the final averaged values are listed in Table 2.3 and Table 2.2 respectively. We also found that, the margin of error in human judgment for our dataset is low relative to its accuracy indicated by VRE in Table 2.2. Finally, the above results introduced a real time drowsiness detection model with around 49,653 trainable parameters, that does not occupy much memory space to run on cell phones.

Case	Metric-Fold									
	A-f1	R-f1	A-f2	R-f2	A-f3	R-f3	A-f4	R-f4	A-f5	R-f5
PM	0.64	2.42	0.61	1.04	0.70	0.58	0.64	0.85	0.67	0.81
HJ	0.62	1.6	0.56	3.05	0.65	1.61	0.53	1.35	0.56	1.41

A-f i: VA for fold i
R-f i: VRE for fold i

Table 2.3: Results of our Proposed Model (PM) and Human Judgment (HJ) measured by VA and VRE

2.7 Conclusion

In this paper, we presented a new and publicly available real-life drowsiness dataset (UTA-RLDD), which, to the best of our knowledge, is significantly larger than existing datasets with almost 30 hours of video. We have also proposed an end-to-end baseline method using the temporal relationship between blinks for multistage drowsiness detection applicable for cell phones. Our results demonstrated that our method outperforms human judgment in two designed metrics on the UTA-RLDD dataset. Future work includes adding a spatial deep network to analyze other features of drowsiness besides blinks in the video. We hope that in future more work will be built on top of our results for a more directed research in this area.

2.8 Supplementary Material

2.8.1 Blink Retrieval Algorithm

In our experiments, we noticed that the approach of Soukupová and Cech [32] typically detected consecutive quick blinks as a single blink. This created a problem for subsequent steps of drowsiness detection, since multiple consecutive blinks can typically be a sign of drowsiness. We added a post-processing step on top of the output of [32], that successfully identifies the multiple blinks which may be present in a single detection produced by [32].

According to[32], define EAR, for each frame, as below:

$$EAR = \frac{||\vec{p}_2 - \vec{p}_6|| + ||\vec{p}_3 - \vec{p}_5||}{||\vec{p}_1 - \vec{p}_4||} \tag{2.14}$$

In the above, each $\vec{p}_i \in \{p_i | i = 1, ..., 6\}$ is the 2D location of a facial landmark from the eye region, as illustrated by Figure 2.6. In [32], an SVM classifier detects eye blinks as a pattern of EAR values in a short temporal window of size 13 depicted in Fig.2.7. This fixed window size is chosen based on the rationale that each blink is about 13 frames long. A single blink takes around 200ms to 400ms on average [34, 37], which translates to six to twelve frames for a video recorded at 30fps. Even if 13 frames is a good estimate for the length of a blink, this approach would not handle consecutive quick blinks.

As depicted in Figure 2.7, each value in this 13 dimensional vector corresponds to the EAR of a frame with the frame of interest located in the middle. The SVM classifier takes these 13D vectors as input and classifies them as "open" or "closed" (more specifically referred to the frame of interest in each input vector). A number of consecutive "closed" labels represent a blink with the length of M. Subsequently, the EAR values of these M frames are stored in $\mathbf{x}$ in order, and fed to the "Blink Retrieval Algorithm", explained in Alg.2, for post-processing (Fig. 2.8a). The sequence of EAR values for **one** blink by [32] will be considered as a **candidate** for **one or more than one** blinks.

This algorithm runs in $\Theta(M)$ time, where M is the number of frames in the video segment that is used as input to the algorithm. In practice, the algorithm runs in real time. In addition, Alg.2 sets a definite frame on when a blink starts, ends or reaches its bottom point based on the extrema of its EAR signal. For better results, x is passed through a median/mean filter to clear the noise and then fed to the algorithm.

At step 1, the derivative of $\mathbf{x}$ is taken. Then, zero derivatives are modified, at steps 2 and 3, so that those derivatives have the same sign as the derivative at their previous

Algorithm 1 Blink Retrieval Algorithm

Input The initial detected EAR signal $\mathbf{x} \in \mathbb{R}^M$, where M is the size of the $\mathbf{x}$ time series, as a candidate for one or more blinks and epsilon=0.01

Output N retrieved blinks, $N \ll M$

1: $\dot{\mathbf{x}}[n] \leftarrow \mathbf{x}[n+1] - \mathbf{x}[n], \forall\, n \in \{i | i = 0, 1, ..., M-2\}$
2: **if** $\dot{\mathbf{x}}[0] = 0$ **then** $\dot{\mathbf{x}}[0] \leftarrow -1 \times$ epsilon
3: $\dot{\mathbf{x}}[n] \leftarrow \dot{\mathbf{x}}[n-1] \times$ epsilon, $\forall\, n \in \{i | \dot{\mathbf{x}}[i] = 0 \wedge i \neq 0\}$ to avoid zero derivatives for steps 4 and 6
4: $\mathbf{c}[n] \leftarrow \dot{\mathbf{x}}[n+1] \times \dot{\mathbf{x}}[n], \forall\, n \in \{i | i = 0, 1, ..., M-3\}$
5: Define $\mathbf{e} \in \mathbb{R}^{P+2}, P \leq M-2$ to store the indices for the P extrema, the first and the last points in $\mathbf{x}$
6: $\mathbf{e}[0] \leftarrow 0$, $\mathbf{e}[P+1] \leftarrow M-1$, supposing the first and last points in $\mathbf{x}$ are maxima
7: $\mathbf{e}[k] \leftarrow n+1, \forall\, (n \in \{i | \mathbf{c}[i] < 0\} \wedge k \in \{i | i = 1, 2, ..., P\})$ ▷ Indices of P+2 extrema, including the first and last points in $\mathbf{x}$ are stored in order
8: Define $THR \leftarrow 0.6 \times \max(\mathbf{x}) + 0.4 \times \min(\mathbf{x})$, as a threshold
9: Define $\mathbf{t} \in \mathbb{R}^{P+2}$, to store +1 or -1 for extrema above and below threshold respectively
10: $\mathbf{t}[0] \leftarrow +1, \mathbf{t}[P+1] \leftarrow +1$, supposing the first and last points in $\mathbf{x}$ are maxima
11: Append +1 in $\mathbf{t}$ for each $n \in \{i | \mathbf{x}[\mathbf{e}[i]] > THR\}$, and append -1 in $\mathbf{t}$ for each $n \in \{i | \mathbf{x}[\mathbf{e}[i]] \leq THR\}$, all in the order of the indices in $\mathbf{e}$
12: Define $\mathbf{z} \in \mathbb{R}^{P+1}$, $\mathbf{z}[n] \leftarrow \mathbf{t}[n+1] \times \mathbf{t}[n]$
13: Define $\mathbf{s}$, to store the indices of all negative values in $\mathbf{z}$, representing the downward and upward movements of eyes in a blink
14: $N \leftarrow \frac{length(\mathbf{s})}{2}$ ▷ N is the number of sub blinks, and $length(\mathbf{s})$ is always an even number
15: **for** $i \leftarrow 0$ to $N-1$ **do** ▷ Define for $blink_i$:
16: StartIndex $\leftarrow \mathbf{e}[\mathbf{s}[2 \times i]]$,
17: EndIndex$\leftarrow \mathbf{e}[\mathbf{s}[2 \times i + 1] + 1]$,
18: BottomIndex$\leftarrow \mathbf{e}[\mathbf{s}[2 \times i + 1]]$

return start, end and bottom points of the N retrieved blinks in $\mathbf{x}$

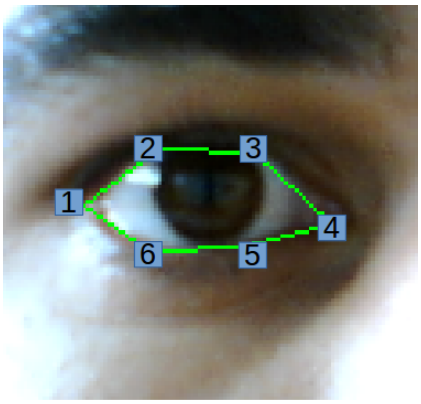


Figure 2.6: Six points marking each eye.

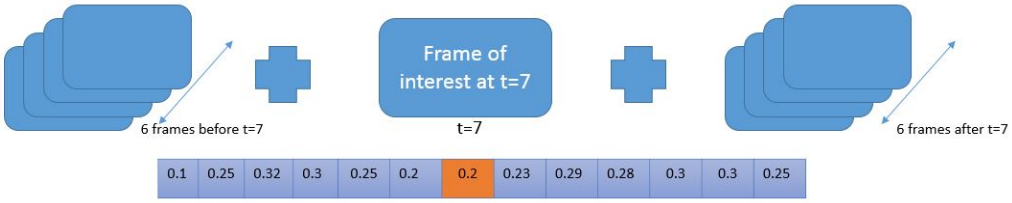


Figure 2.7: Presenting each frame (at t=7) by 13 numbers (EARs) concatenated from 13 frames as a feature vector.

time step. This modification helps to find local extrema, as points where the derivative sign changes (steps 4 to 7). The threshold, defined at step 8, is used to suppress the subtle ups and downs in **x** due to noise and not blinks. The extrema in **x** are circled in Figure 2.8b, and labeled (+1 or -1) relative to the threshold (steps 9 to 11). Each two consecutive extrema are indicative of a downward or upward movement of eyes in a blink if those two are connected, so that the link or links between them pass the threshold line (steps 12 and 13). Fig.2.8c highlights these links in red. Finally, each pairing of these red links corresponds to one blink with start, end and bottom points as depicted in Figure 2.8d (steps 14 to the end).

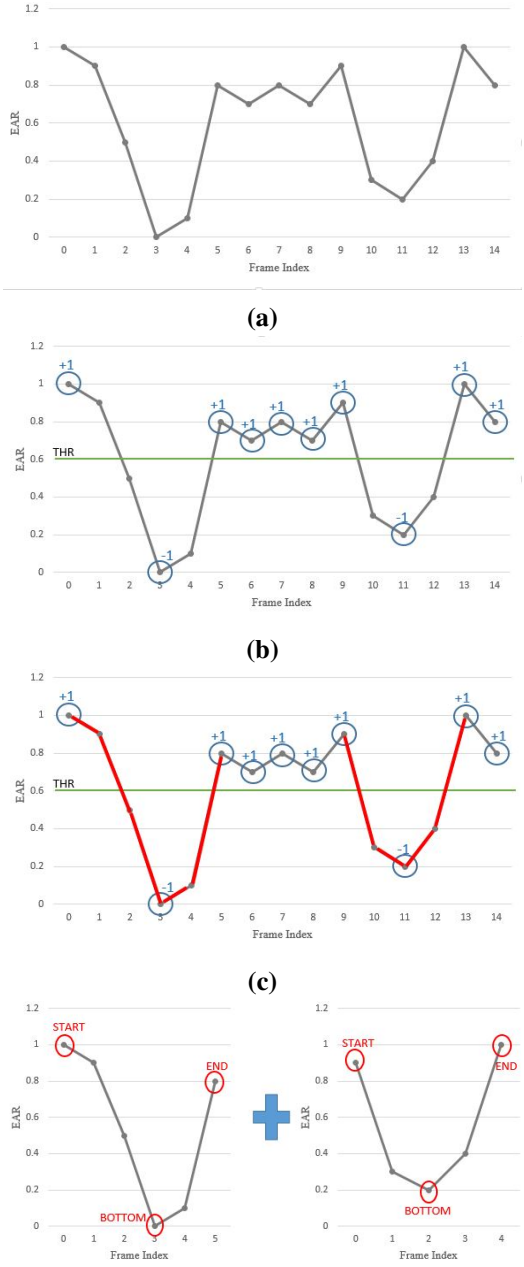


Figure 2.8: The Blink Retrieval Algorithm steps: (a) x with size $M = 15$ as the input for Alg. 2. (b) The indices of circled points form e, and the set of +1 and -1 labels forms t with $P = 8$. (c) The red lines indicate where z values are negative. (d) Two ($N = 2$) blinks are retrieved with definite start, end and bottom points.

Part 2:
Weakly-Supervised
Action Segmentation

3 Action Duration Prediction for Segment-Level Alignment of Weakly-Labeled Videos

3.1 Abstract

This chapter focuses on weakly-supervised action alignment, where only the ordered sequence of video-level actions is available for training. We propose a novel Duration Network[1], which captures a short temporal window of the video and learns to predict the remaining duration of a given action at any point in time with a level of granularity based on the type of that action. Further, we introduce a Segment-Level Beam Search to obtain the best alignment, that maximizes our posterior probability. Segment-Level Beam Search efficiently aligns actions by considering only a selected set of frames that have more confident predictions. The experimental results show that our alignments for long videos are more robust than existing models. Moreover, the proposed method achieves state of the art results in certain cases on the popular Breakfast and Hollywood Extended datasets.

3.2 Introduction

Activity analysis covers a wide range of applications from monitoring systems to smart shopping and entertainment, and it is a topic that has been extensively studied in recent years. While good results have been obtained in recognizing actions in single-action RGB videos [38, 39, 40, 41, 42, 43], there are many real-life scenarios where we want to recognize a sequence of multiple actions, whose labels and start/end frames are unknown.

[1] Code available at: https://github.com/rezaghoddoosian/DurNet

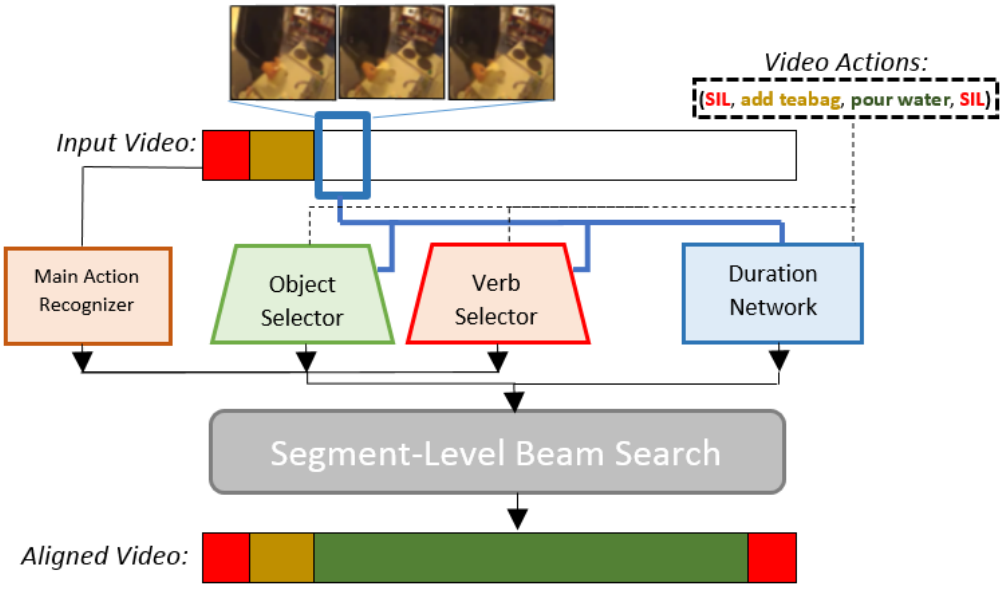


Figure 3.1: An overview of our proposed method. Based on the context of the temporal window, *pour water* is selected and its duration is predicted to align the given video-level actions

Most work done in this area is fully supervised [44, 45, 46, 47, 48, 49, 50, 51, 52], requiring each frame in the training videos to be annotated. Given the need of deep learning algorithms for ever-larger training datasets, frame-level annotation can be expensive and unscalable. "Weak supervision" is an alternative, where each training video is only annotated with the ordered sequence of actions occurring in that video, with no start/end frame information for any action [53, 54, 55, 56, 57, 58, 59, 1].

This chapter focuses on **weakly-supervised action alignment**, where it is assumed that the sequence of video-level action labels are provided as input for training and inference, and the output is the start and end time of each action.

A key challenge in weakly supervised action alignment is correctly predicting the duration of actions. To achieve this goal, we propose a *Duration Network (DurNet)* that, unlike previous methods, takes video features into account. Video features contain valuable information that existing duration models ignore. As an example, video features can capture the pace (slow or fast) at which an action is performed. As another example, video features can capture the fact that an ongoing "frying" action is likely to continue for a longer time if the cook is currently away from the frying pan. Our duration model learns to estimate the remaining duration of an ongoing action based on the current visual

observations. More specifically, the proposed DurNet mainly consists of a bi-directional Long Short-Term Memory (LSTM), which takes as inputs the set of frame features in a short temporal window at a given time, a hypothesized action class and its elapsed duration. The network outputs the probability of various durations (from a discretized set) for the remainder of that action.

We also introduce a Segment-Level Beam Search algorithm to efficiently maximize our factorized probability model for action alignment. This algorithm modifies the vanilla beam search to predict the most likely sequence of action segments without looping through all possible action-duration combination in all frames. Instead, it predicts the action and duration of segments by selecting a small subset of the frames that are significant enough to maximize the posterior probability. The time complexity of our Segment-Level Beam Search is linear to the number of action segments in the video, which is theoretically better than that of other Viterbi based alignment methods [58, 48, 1, 2]. In particular Richard *et al.* [1] considered visual and length models' frame-level outputs and their combinations over all the frames for action alignments. More recently [2] extended Richard *et al.*'s work [1] by incorporating all invalid action sequences in the loss function during training, but follows the same frame-level inference technique as in [1].

The main contributions of this chapter can be summarized as follows: (1) We introduce a Duration Network for action alignment, that is explicitly designed to exploit information from video features and show its edge over the Poisson model used in previous work [1, 2]. (2) We propose a Segment-Level Beam Search that can efficiently align actions to frames without exhaustively evaluating each video frame as a possible start or end frame for an action (in contrast to [57, 58, 1, 2]). (3) In our experiments, we use two common benchmark datasets, the Breakfast [60] and Hollywood Extended [53], and we measure performance using three metrics from [55]. Depending on the metric and dataset, our method leads to results that are competitive or superior to the current state-of-the-art for action alignment.

3.3 Related Work

Weakly-Supervised Video Understanding. Existing methods for video activity understanding often differ in the exact version of the problem that they aim to solve. [61, 62] aim to associate informative and diverse sentences to different temporal windows for dense video captioning. [63, 64, 65] aim to do action *detection*, and are evaluated on videos that consist of typically a single unique action with a large portion of background frames.

Weakly-supervised action segmentation and alignment have been studied under dif-

ferent constraints at training time. Some works utilize natural language narrations of what is happening [66, 67, 68, 69, 70]. [71] use only unordered video-level action sets to infer video frames. Our work is closest to [53, 54, 55, 56, 57, 58, 59, 1, 2], where an ordered video-level sequence of actions is provided for training.

Our chapter focuses on the task of weakly-supervised action alignment, where the video and an ordered sequence of action labels are provided as input, and frame-level annotations are the output.

Duration Modeling. One of the key innovations of our method is in weakly supervised modeling and prediction of action duration. Therefore, it is instructive to review how existing methods model duration. Some methods [54, 55, 56, 59] do not have an explicit duration model; the duration of an action is obtained as a by-product of the frame-by-frame action labels that the model outputs. [72, 73, 74] studied long term duration prediction. However they are fully supervised methods whose results are highly sensitive to ground-truth observations.

Most related to our duration model in action alignment are existing methods that model action duration as a Poisson function [1], or as a regularizer [53, 75, 58, 48] to penalize actions that last too long or too short. Specifically [1] and [2] integrated an action dependent Poisson model into their system which is characterized only by the average duration of each action based on current estimations. The key innovation of our method is that our duration model takes the video data into account. The video itself contains information that can be used to predict the remaining duration of the current action, and our method has the potential to improve prediction accuracy by taking this video information into account.

3.4 Method

In this section, we explain what probabilistic models our method consists of and how they are deployed for our Segment-Level Beam Search.

3.4.1 Problem Formulation

Our method takes two inputs. The first input is a video of T frames, represented by $\mathbf{x}_1^T$, which is the sequence of per-frame features. Feature extraction is a black box, our method is not concerned with how those features have been extracted from each frame. The second input is an ordered sequence $\tau = (\tau_1, \tau_2, ..., \tau_M)$ of M action labels, that list the sequence of all actions taking place in the video.

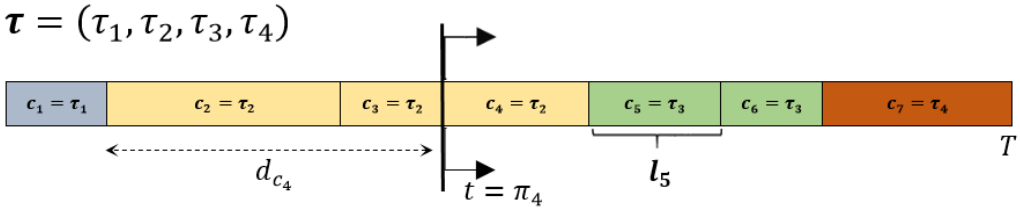


Figure 3.2: A sample segmented $(N = 7)$ video given its video-level labels τ $(M = 4)$. One ground-truth action label τ can correspond to multiple consecutive segments

A partitioning of the video into N consecutive segments is specified using a sequence $\mathbf{c}_1^N$ of action labels (c_n specifies the action label for the n-th segment) and a sequence $\mathbf{l}_1^N$ of corresponding segment lengths (l_n specifies the number of frames of the n-th segment). Given such a partition, we use notation π_n for the first frame of the n-th segment.

Given inputs $\mathbf{x}_1^T$ and τ, the goal of our method is to identify the most likely sequence $\bar{\mathbf{c}}_1^N$ of action labels c_n and corresponding sequence $\bar{\mathbf{l}}_1^N$ of durations l_n:

$$(\bar{\mathbf{c}}_1^N, \bar{\mathbf{l}}_1^N) = \underset{\mathbf{c}_1^N, \mathbf{l}_1^N}{\operatorname{argmax}}\, p(\mathbf{c}_1^N, \mathbf{l}_1^N | \mathbf{x}_1^T, \tau) \tag{3.1}$$

We note that N (the number of segments identified by our method) can be different than M (the number of action labels in input τ). This happens because our method may output the same action label for two or more consecutive segments, and all consecutive identical labels correspond to a single element of τ. We use Ω_n to denote the earliest segment number such that all segments from segment Ω_n up to and including segment n have the same action label. For example, in Fig. 3.2, $\Omega_4 = \Omega_3 = \Omega_2 = 2$.

Consider a frame π_n, that is the starting frame of the n-th segment. We assume that the remaining duration of an action at frame π_n depends on the type of action c_n, the elapsed duration $\mathbf{l}_{\Omega_n}^{n-1}$ of c_n up to frame π_n, and the visual features of a window of α frames starting at frame π_n. We denote this window as $\mathbf{w}_n = \mathbf{x}_{\pi_n}^{\pi_n+\alpha-1}$. Also, we decompose each action label c_n into a corresponding verb v_n and object o_n. For example the action "take cup" can be represented by the $(take, cup)$ pair, where *take* and *cup* are the verb and object respectively. Working with "verbs" instead of "actions" lets us benefit from the shared information among "actions" with the same "verb". This specifically helps in analyzing any weakly-labeled video where the frame-level pseudo ground-truth is inaccurate. Based on the above, we rewrite $p(\mathbf{c}_1^N, \mathbf{l}_1^N | \mathbf{x}_1^T, \tau)$ as:

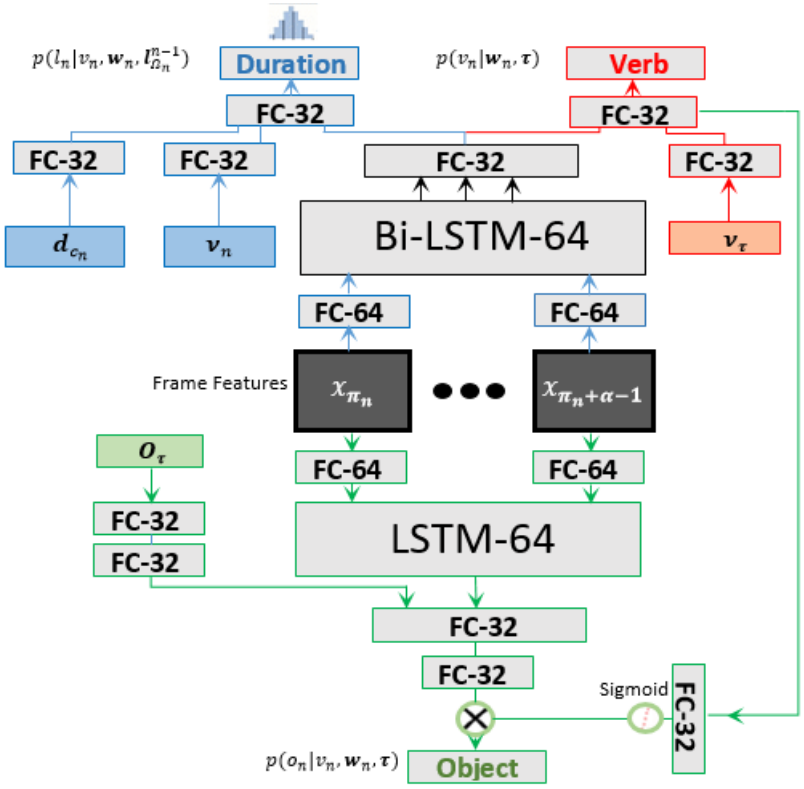


Figure 3.3: Architecture of the Duration, Object and Verb Selector Networks

$$p(\mathbf{c}_1^N, \mathbf{l}_1^N | \mathbf{x}_1^T, \tau) = \prod_{n=1}^{N} p(l_n | \mathbf{w}_n, \mathbf{l}_{\Omega_n}^{n-1}, c_n) \cdot p(c_n | \mathbf{x}_1^T, \tau) \tag{3.2}$$

$$= \prod_{n=1}^{N} p(l_n | \mathbf{w}_n, \mathbf{l}_{\Omega_n}^{n-1}, v_n, o_n) \cdot p(c_n | \mathbf{x}_1^T, \tau) \tag{3.3}$$

$$= \prod_{n=1}^{N} p(l_n | \mathbf{w}_n, \mathbf{l}_{\Omega_n}^{n-1}, v_n) \cdot p(c_n | \mathbf{x}_1^T, \tau) \tag{3.4}$$

We should note that, in the above equations, in the boundary case where $\Omega_n = n$, we define $\mathbf{l}_{\Omega_n}^{n-1}$ to be 0. The Duration and Action Selector Network, described next, will be used to compute the probability terms in Eq. 3.4. Then, using our Segment-Level Beam Search, the most likely segment alignment will be identified.

3.4.2 Duration Network(DurNet)

Previous work [58, 1, 2] has tried to model the duration of actions. Richard *et al.* [1] have used a class-dependent Poisson distribution to model action duration, assuming that the duration of an action only depends on the type of that action. In contrast, we propose a richer duration model, where the length of an action segment depends not only on the type of that action, but also on the local visual features of the video, as well as on the length of the immediately preceding segments if they had the same action label as the current segment (Eq. 3.4).

The proposed model allows the estimate of the remaining length of an action to change based on video features. For example, our model can potentially predict a longer remaining duration for the action "squeeze orange" if the local visual cues correspond to a person just picking up the orange, compared to a person squeezing the orange.

In our method, the range of possible durations of a given action depends on the *verb* of that action. For example, one second could be half of a short action associated with verb "take" and only one-hundredth of a longer action associated with verb "frying". We model this dependency by mapping time length to progress units for each verb. We denote by γ_v the median length of verb v across all training videos, and by L the number of time duration bins. We should note that the system cannot know the true value of γ_v, since frame-level annotations are not part of the ground truth. Instead, our system estimates γ_v based on pseudo-ground truth that is provided using an existing weakly supervised action alignment method, such as [55, 1]. Given this estimated γ_v, we discretize the elapsed and remaining time lengths into verb-dependent bins; i.e. the bin width b_v is calculated based on the type of each verb:

$$b_v = \frac{\gamma_v}{\lfloor \frac{L}{2} \rfloor + 1} \tag{3.5}$$

The above equation assures that the median length of a verb falls on or around the middle bin, which creates a more balanced distribution for learning.

In our method, $p(l_n|\mathbf{w}_n, \mathbf{l}_{\Omega_n}^{n-1}, v_n)$ is modeled by a Bi-LSTM network preceded by a fully-connected layer and followed by fully connected layers and a softmax function σ as shown in Fig. 3.3. The input to this network, for any segment n at a given time π_n , is the one-hot vector representation of the verb $\mathbf{v}_n \in \mathbb{R}^V$ of a given action c_n and its discretized elapsed duration $\mathbf{d}_{c_n} \in \mathbb{R}^L$ as well as the local visual features $\mathbf{w}_n \in \mathbb{R}^{\Gamma \times F}$. Here, V is the total number of verbs, F is the input feature dimension, and Γ is the number of temporally sampled features over α frames starting with frame π_n. At the end, this network outputs the corresponding verb-dependent future progress probability corresponding to each bin. This probability is expressed as an L-dimensional vector $\mathbf{k}_{v_n}$, whose i-th dimension is the

probability that the duration of action c_n falls in the i-th progress unit for verb v_n, given the inputs described above.

During training, we used a Gaussian to represent the progress probability labels as soft one-hot vectors. This representation considers the bins that are closer to the true bin more correct than the further ones. The resulting labels are used to compute the standard cross-entropy loss, as the DurNet loss function.

Finally, we translate this progress indicator back to time expressed as number of frames, according to verb-dependent steps s_v:

$$s_v = \lfloor \frac{\gamma_v}{L} \rfloor \tag{3.6}$$

$$l_{v,i} = (i+1) * s_v, i \in \{0, 1, ..., L-1\} \tag{3.7}$$

Thus, the i-th discretized duration $l_{v,i}$ for verb v corresponds to the i-th dimension of vector $\mathbf{k}_{v_n}$, and the value of $\mathbf{k}_{v_n}$ in the i-th dimension gives the probability of discretized duration $l_{v_n,i}$.

3.4.3 Action Selector Network

This network selects the label of the action occurring at any time in the video. Each action is decomposed as a $(verb, object)$ pair. The importance of objects and verbs in action recognition has been studied before [76, 70]. For example, the verb "take" in both "take bowl" and "take cup" is expected to visually look the same way. These two actions only differ in their corresponding objects. This approach has the advantage that not only the network can access more samples per class (verb/object), but also classification is done over fewer number of classes, because several actions share the same verb/object. This is specifically helpful in weakly-labeled data as the frame-level ground truth is not reliable. The probability of the selected action is obtained by the factorized equation below:

$$p(c_n|\mathbf{x}_1^T, \tau) := \eta[p(o_n|v_n, \mathbf{w}_n, \tau)^\zeta p(v_n|\mathbf{w}_n, \tau)^\beta p(c_n|\mathbf{x}_1^T)^\lambda] \tag{3.8}$$

$\eta[]$ is a normalization function that assures :

$$\sum_{c_n \in \tau} [p(c_n|\mathbf{x}_1^T, \tau)] = 1 \tag{3.9}$$

The Action Selector Network consists of three components: i) The verb selector network. ii) The object selector network. iii) The main action recognizer (Fig. 5.1). The

influence of each network is adjusted by the ζ, β and λ hyper parameters.

i) The Verb Selector Network(VSNet): It focuses only on the local temporal features during the given time frame $[\pi_n, \pi_n + \alpha - 1]$ to select the correct verb v_n for segment n. The video-level verb labels $\mathbf{v}_\tau \in \{0,1\}^V$ are also given as input to the network, where for every $i \in \{0,1,...,V-1\}$, $v_{\tau i} = 1$ if $v_{\tau i}$ is present in the video-level verbs, otherwise $v_{\tau i} = 0$.

ii) The Object Selector Network(OSNet): Similar to the VSNet, using the local temporal features, this module selects the correct segment object o_n from the set of video-level objects $\mathbf{o}_\tau \in \{0,1\}^O$, where O is the number of available objects in the dataset. Selecting the target object is also influenced by the type of the verb for a given action according to Eq. 3.8. In order to model this dependency, latent information from the VSNet flows into the OSNet (Fig. 3.3).

iii) The Main Action Recognizer(MAR): Unlike the other two components, this module produces frame-level probability distribution for the main actions. This network is more discriminative than the other two and particularly helpful in videos with repetitive verbs and objects. Note that the MAR module can be replaced by any baseline neural network architecture like CNNs or RNNs.

Finally, as shown in Eq. 3.8, the probability of a segment action is defined by fusing the output of the three above-mentioned networks. In the special case of $\zeta, \beta = 1$ and $\lambda = 0$, the definition of Eq. 3.8 would be truly probabilistic, and there would be no need for the normalization function η. The contribution of each network is quantitatively shown in Sec. 3.5.2.3. It is noteworthy to mention that our method is equally applicable without the verb-object decomposition assumption. In case there is no specific object associated with actions, our formulation still stands by setting $\zeta = 0$ and working with the actions as our set of verbs.

3.4.4 Segment-Level Beam Search

We introduce a beam search algorithm with beam size B to find the most likely sequence of segments, as specified by a sequence of labels $\mathbf{c}_1^N$ and a sequence of lengths $\mathbf{l}_1^N$. By combining Eq. 3.1 with Eq. 3.4 we obtain:

$$(\bar{\mathbf{c}}_1^N, \bar{\mathbf{l}}_1^N) = \underset{\mathbf{c}_1^N, \mathbf{l}_1^N}{\operatorname{argmax}} \{ \prod_{n=1}^{N} p(l_n | \mathbf{w}_n, \mathbf{l}_{\Omega_n}^{n-1}, v_n) \cdot p(c_n | \mathbf{x}_1^T, \tau) \} \tag{3.10}$$

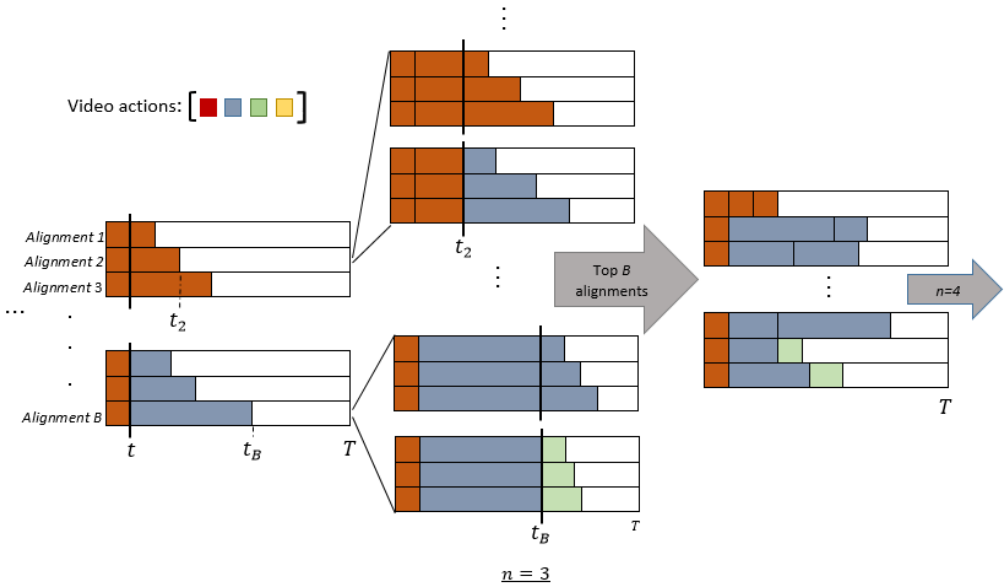


Figure 3.4: Our proposed Segment-Level Beam Search of beam size B during the estimation of the third segment $(n = 3)$. For each alignment, different possibilities of next action and its predicted duration are evaluated. At each point in our method, all B hypothesized alignments consist of the same number of segments

In frame-level beam search, different sequences of action classes are considered at every single frame until the end of the video. In contrast, our Segment-Level Beam Search allows the algorithm to consider such sequences only at the beginning of every segment. This technique is inspired by the fact that actions do not change rapidly from one frame to another.

We introduce the notation $A_i(c,l,t_i)$ to represent the probability of segment-level alignment i until frame t_i for each video, where c and l are the action class and length of the last segment. We also define $\max^B\{a_1,a_2,...,a_n\}$ as the set of B greatest a_i, and calculate $A_i(c,l,t_i)$ of alignment i recursively for every action c_n and length l_n of segment n. Then, the B most probable alignments with n segments are selected over all combinations of c_n and l_n. Algorithm 1 summarizes the procedure for our proposed Segment-Level Beam Search with the following constraints:

- $c_1 = \tau_1, c_N = \tau_M$
- $t_i \leq T, \forall i \in \{1,2,...,B\}$

$\phi(c_{n-1})$ refers to the set of possible actions for segment n. $\phi(c_{n-1})$ is either a repetition of the action c_{n-1} of the previous segment or the start of the next action in τ. The final segment labels $\mathbf{c}_1^N$ and $\mathbf{l}_1^N$ are derived by keeping track of the maximizing arguments c_n and l_n in the maximization steps.

Algorithm 2 Segment-Level Beam Search

Input: Video features $\mathbf{x}_1^T$ and video-level labels τ, beam size B
Output: Action label and length sequences c_1^N and l_1^N.

$n \leftarrow 1$, ▷ first segment

for $l_1 \in \{l_{v_1,0}, ..., l_{v_1,L-1}\}$ **do**:

$A_1(\tau_1, l_1, l_1) = p(l_1|\mathbf{x}_1^\alpha, 0, v_1) \cdot p(\tau_1|\mathbf{x}_1^T, \tau)$

$\mathbb{A}(n) = \max_{l_1}^{B} \{A_1(\tau_1, l_1, l_1)\}$, ▷ set of candidate alignments

while $t_i < T, \forall i \in \{1, 2, ..., B\}$ **do**:

$n \leftarrow n+1$,

for $i \leftarrow 1$ to B **do**:

for all $c_n \in \phi(c_{n-1}); l_n \in \{l_{v_n,0} \; ... \; l_{v_n,L-1}\}$ **do**:

$A_i(c_n, l_n, t_i + l_n) =$ ▷ $A_i(c_{n-1}, l_{n-1}, t_i) \in \mathbb{A}(n-1)$

$A_i(c_{n-1}, l_{n-1}, t_i) \cdot p(l_n|\mathbf{x}_{t_i}^{t_i+\alpha-1}, \mathbf{l}_{\Omega_n}^{n-1}, v_n) \cdot p(c_n|\mathbf{x}_1^T, \tau)$,

$\mathbb{A}(n) = \max_{c_n, l_n}^{B} \{A_i(c_n, l_n, t_i + l_n), \forall i \in \{1, 2, ..., B\}\}$

$A_{\text{final}}(c_N, l_N, T) = \max_{c_N, l_N}^{1} \{A_i(c_N, l_N, t_i), \forall i \in \{i | t_i = T\}\}$

Note that $p(c_n|\mathbf{x}_1^T, \tau)$ in Algorithm 1 is factorized according to Eq. 3.8, and every $c_n \in \phi(c_{n-1})$ is broken down to its corresponding (v_n, o_n) pair. This factorization approach encourages segments that cover the whole duration of an action to avoid the penalty each time a new segment is added. This results in faster alignments with a smaller number of unreasonably short segments.

Time complexity of our Segment-Level Beam Search, for each video, depends on the beam size B, number of segments N and number of length bins L. As B and L are constant values, the time complexity for the algorithm above would be $O(N)$, and only limited to the number of segments per video. Based on our experiments, for the current public action alignment datasets, $N_{max} \approx 70$ is two orders of magnitude less than $T_{max} \approx 9700$. This makes the proposed beam search more efficient than the Viterbi algorithms used in [1, 2] and [58], which have the complexity of $O(T^2)$ and $O(T)$ respectively.

3.5 Experiments

We show results on two popular weakly-supervised action alignment datasets based on three different metrics. We compare our method with several existing methods under different initialization schemes. Further, the contribution of each component of our model is quantitatively and qualitatively justified.

Datasets. 1) *The Breakfast Dataset (BD)* [60] consists of around 1.7k untrimmed instructional videos of few seconds to over ten minutes long. There are 48 action labels demonstrating 10 breakfast recipes with a mean of 4.9 instances per video. The overall duration of the dataset is 66.7h, and the evaluation metrics are conventionally calculated over four splits. 2) *The Hollywood Extended Dataset (HED)* [53] has 937 videos of 17 actions with an average of 2.5 non-background action instances per video. There are in total 0.8M frames of Hollywood movies and, following [53], we split the data into 10 splits for evaluation.

There are four main differences between these two datasets: i) Actions in the *BD* follow an expected scenario and context in each video. However, the relation between consecutive actions in the *HED* can be random. ii) Camera in the *BD* is fixed while there are scene cuts in the *HED*, making the duration prediction more challenging. iii) Background frames are over half of the total frames in the *HED*, while the percentage of them in the *BD* is about 10%, and iv) The inter-class duration variability in the *BD* is considerably higher than the *HED*.

Metrics. We use three metrics to evaluate performance: 1) *acc* is the frame-level accuracy averaged over all the videos. 2) *acc-bg* is the frame-level accuracy without the background frames. This is specifically useful for cases where the background frames are dominant as in the *HED*. 3) *IoU* defined as the intersection over union averaged across all videos. This metric is more robust to action label imbalance and is calculated over non-background segments.

Implementation. For a fair comparison, we obtained the pre-computed 64 dimensional features of previous work [54, 1, 2], computed using improved dense trajectories [77] and Fisher vectors [78], as described in [45]. A single layer bi-directional LSTM with 64 hidden units is shared between the DurNet and VSNet, and a single layer LSTM with 64 hidden units for the OSNet. We followed the same frame sampling as [2], [55] or [1], depending on the method we use for initialization. We use the cross-entropy loss function for all networks, using Adam optimization [35], learning rate of 10^{-5} and batch size of 64. L in the DurNet was set to 7 and 4 for the *BD* and *HED* respectively. In our experiments on the *BD*, we used an $alpha$ of 60 frames and ζ, β, and λ were adjusted

Models	Breakfast (%)			Hollywood Extended (%)		
	acc	acc-bg	IoU	acc	acc-bg	IoU
HTK [57]*	43.9	-	26.6	49.4	-	29.1
ECTC [56]*	∼35	-	-	-	-	-
D^3TW [54]	57.0	-	-	59.4	-	-
TCFPN [55]†	51.7	48.2	33.0	57.6	46.1	28.2
[55]/ [1] pg**	56.4	53.4	36.2	-	-	-
NNViterbi [1]†	63.5	63.0	**47.5**	59.6	53.2	32.4
[1]/ [55] pg**	63.4	62.8	47.3	-	-	-
CDFL [2]	63.0	61.4	45.8	**65.0**†	63.7†	**40.2**†
Ours/ [55] pg	55.7	56.1	36.3	50.1	64.1	31.4
Ours/ [1] pg	63.7	65.0	42.5	56.0	64.3	34.3
Ours/ [2] pg	**64.1**	**65.5**	43.0	59.1	**65.4**	35.6

Table 3.1: Weakly-supervised action alignment results of existing methods on two main datasets. (* from [55], † best results obtained after running the author's source code multiple times, after slight changes to the original source code for the specific task.)**

to 1, 30, and 5 respectively for our selector network. Beam size in our beam search was set to 150 and other hyperparameters were picked after grid search optimization (refer to supplementary material).

Training Strategy. During training, alignment results of a baseline weakly-supervised method, *e.g.* CDFL [2], NNViterbi [1] or TCFPN [55], on the training data is used as the initial pseudo-ground truth. We also adopt the pre-trained frame-level action classifier (visual model) of the baseline (CDFL, NNViterbi or TCFPN) as our main action selector component. The initial pseudo-ground truth is used to train our duration and action selector networks. Then, new alignments are generated through the proposed Segment-Level Beam Search algorithm on the training videos. We call these new alignments the "new pseudo-ground truth". The adopted visual model is finally retrained based on our "new pseudo-ground truth", and used alongside our other components to align the test videos.

3.5.1 Comparison to State-of-the-Art Methods

Comparison Settings. In addition to evaluating existing methods, we also evaluate some combinations of existing methods, as follows: 1,2) *Ours/ [1] pg* and *Ours/ [2] pg*: Ours initialized with NNviterbi [1] and CDFL [2] pseudo-ground truth respectively, and a single layer GRU as the MAR. 3) *Ours/ [55] pg*: Ours initialized with the training results of [55] as our pseudo-ground truth, and the TCFPN [55] network as the MAR. 4) *[55]/ [1] pg*: The ISBA+TCFPN method [55] initialized with NNViterbi [1] pseudo-ground truth. 5) *[1]/ [55] pg*: The NNViterbi method [1] initialized with [55] pseudo-ground truth.

Action Alignment Results. Table 3.1 shows results for weakly-supervised action alignment. Our method produces better or competitive results for most cases on both datasets. Initialized with CDFL, our method achieves state-of-the-art in two of the three metrics for the Breakfast dataset and in one metric on the Hollywood. We compare our method with CDFL [2], NNViterbi [1] and TCFPN [55] more extensively, because they are the best open source methods that follow a similar pseudo-ground approach for training. Also for better comparison, in Table 3.1 we present the results of training NNViterbi on the pseudo ground-truth from TCFPN and vice versa.

In direct head-to-head comparisons with CDFL, NNViterbi and TCFPN, the proposed method often outperforms the respective competitor, and in some cases the head-to-head performance improvement by our method is quite significant. Our method improves action alignment results of TCFPN [55] and NNViterbi [1] in 5 (Table 3.3a) and 4 (Table 3.3b) out of 6 metrics respectively. In addition, we outperform CDFL in frame-level accuracy with and without background on the Breakfast dataset, and when tested on the Hollywood dataset, CDFL accuracy without background is improved while the inference complexity is decreased to $O(N)$ from CDFL's $O(T^2)$ (Table 3.3c).

In Table 3.2, our Segment-Level Beam Search achieves consistent improved results in frame accuracy for both datasets when the background frames are excluded. Considering *acc-bg* is essential especially for the Hollywood dataset as on average around 60% of the video frames are background, so *acc* values can be misleading.

There are two plausible explanations on why the performance of our method for non-background actions is not equally repeated for the background segments. First, there is a lack of defined structure in what background can be, which makes it harder to learn. Second, there are cases where background depicts scenes where a person is still or no movement is happening. It is a tough task for even humans to predict how long that motionless scene would last, so the DurNet can easily make confident wrong predictions resulting in inaccurate alignments of background segments.

Fig. 3.5 shows how alignment results vary with video length on the Breakfast Dataset. The performance of our method compared to NNViterbi and TCFPN improves as video length increases. In longer videos, the DurNet can maintain the same action longer depending on the context, while in [1] any duration longer than the action average length gets penalized.

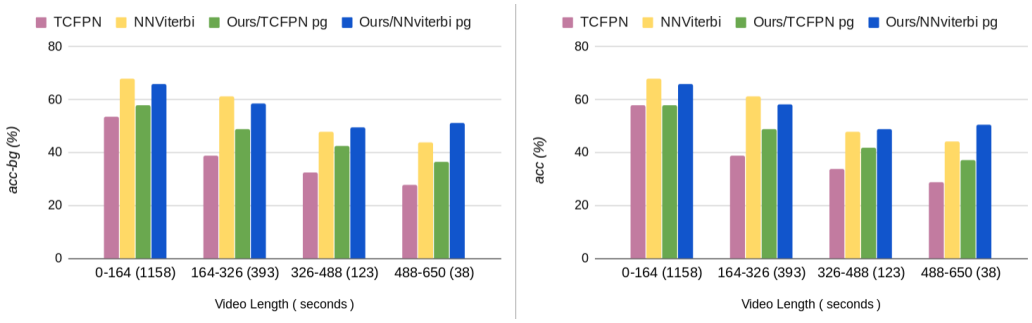


Figure 3.5: Weakly-supervised action alignment accuracy for videos of different lengths. Unlike the other two baselines, ours is more robust to longer videos. We obtained the results on four equal intervals considering the shortest and longest videos. The number of videos for each interval is mentioned in parentheses

3.5.2 Analysis and Ablation Study

All analysis and ablation study is done using the TCFPN [55] pseudo ground-truth initialization. We also ran our ablation study experiments on the Breakfast dataset mainly, because it consists of videos with many actions and high duration variance, so the impact of learning duration can be measured more effectively.

3.5.2.1 DurNet vs. Poisson Duration Model.

We compare our Duration Network with the Poisson length model used in [1, 2]. To compare the two models, we replaced the DurNet in our Segment-Level Beam Search with the Poisson model in [1, 2], while keeping all other parts of our method unchanged.

Table 3.4 quantitatively shows the advantage of using the context of the video, as it has improved the alignment accuracy by more than 1%. One reason for the small improvement, however, could be the imbalanced training set size across the four folds. Unlike the statistical Poisson approach, the performance of DurNet, as in other Neural Networks, depends on the training set size. As Figure 3.6 shows, the bigger the training data size, the better the performance of the DurNet.

3.5.2.2 Duration Step Size Granularity.

As explained in Section 3.4.2, the predicted durations are discretized into a fixed number L of bins, using different step sizes s_v for different verbs. In order to analyze the advantage

Models	Breakfast (%)			Hollywood Extended (%)		
	acc	acc-bg	IoU	acc	acc-bg	IoU
TCFPN [55]†	51.7	48.2	33.0	**57.6**	46.1	28.2
Ours/ [55] pg	**55.7**	**56.1**	**36.3**	50.1	**64.1**	**31.4**

(a)

Models	Breakfast (%)			Hollywood Extended (%)		
	acc	acc-bg	IoU	acc	acc-bg	IoU
NNViterbi [1]†	63.5	63.0	**47.5**	**59.6**	53.2	32.4
Ours/ [1] pg	**63.7**	**65.0**	42.5	56.0	**64.3**	**34.3**

(b)

Models	Breakfast (%)			Hollywood Extended (%)		
	acc	acc-bg	IoU	acc	acc-bg	IoU
CDFL [2]	63.0	61.4	**45.8**	**65.0**†	63.7†	**40.2**†
Ours/ [2] pg	**64.1**	**65.5**	43.0	59.1	**65.4**	35.6

(c)

Table 3.2: Head-to-head action alignment comparisons of the proposed model with the baselines († as specified in Table 3.1).

Models	Alignment	
	acc	acc-bg
Ours+Poisson	54.56%	54.95%
Ours+Duration Net	**55.70%**	**56.10%**

Table 3.4: Comparison between our Duration Network and statistical Poisson length model on the breakfast dataset.

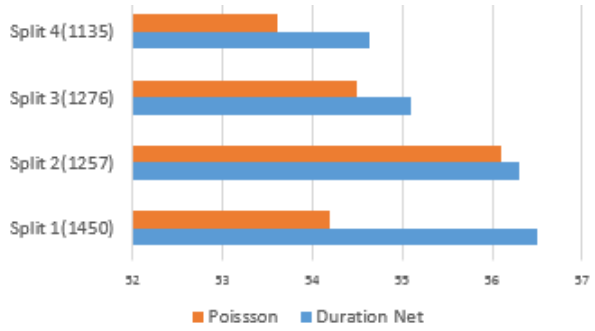


Figure 3.6: Split-wise frame accuracy on the Breakfast dataset. The number of training videos for each split is indicated in parentheses.

	Alignment on Breakfast (%)		
Models	acc	acc-bg	IoU
Fixed steps($s_v = 5$ seconds)	49.9	49.6	32.3
Max-based adaptive steps	48.9	47.6	29.7
Mean-based adaptive steps	54.9	55.4	35.8
Median-based adaptive steps	**55.7**	**56.1**	**36.3**

Table 3.5: The result of fixed step duration modeling with different alternatives of adaptive steps for weakly-supervised alignment.

of this duration modeling, we compare the weakly-supervised alignment results obtained when we replace this approach with fixed step size for all classes, as well as with different alternatives of adaptive steps (Table 3.5); i.e., the predicted duration range of each action can depend on the maximum, mean or median length of that action calculated across all training videos. A fixed step and a step size dependent on maximum duration, both produce poor results. Step sizes dependent on mean and median durations of actions produce comparable results.

3.5.2.3 Analysis of the Action Selector Components.

We evaluate the effect of the OSNet, VSNet and MAR separately. Selecting verbs without objects fails in videos where two actions with the same verb happen consecutively in a video, *e.g.* pour cereal and pour milk (Fig. 3.7). Likewise, excluding the VSNet is problematic when two consecutive actions share the same object. Our experiments show that the VSNet and the MAR have the biggest and smallest contributions respectively(Table 3.6). We also include the results of the special case where we do not use hyperparameters in Eq. 3.8. As we see, a weighted combination of all three components performs best.

Models	Alignment on Breakfast (%)		
	acc	acc-bg	IoU
Special case ($\zeta, \beta = 1, \lambda = 0$)	53.9	54.4	35.4
Action selector w/o main action	55.5	56.1	36.0
Action selector w/o object	54.8	54.6	35.9
Action selector w/o verb	50.9	50.8	32.8
All components	**55.7**	**56.1**	**36.3**

Table 3.6: Contribution of each action selector component. Having all three components gives the best results.

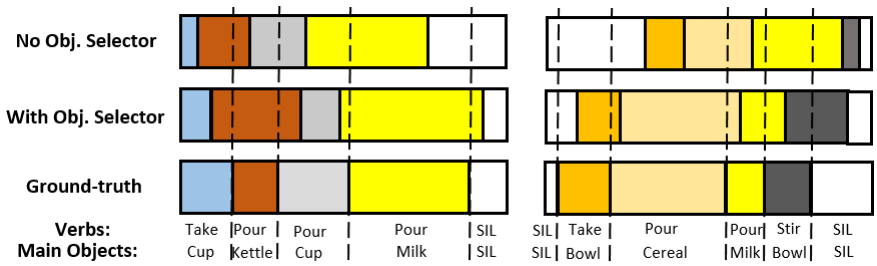


Figure 3.7: Two sample aligned videos, that consist of action labels with the same verb. The object selector component improves the results by aligning the segments with respect to the correct object.

3.5.2.4 Qualitative Segment-Level Alignment Results.

One of the benefits of our Beam Search is predicting the class and length of segments without looping through all possible action-length combinations in all frames. Specifically, by predicting the duration of a segment in advance, only a limited set of more significant frames is processed. This leads to faster alignments with competitive accuracy compared to the frame-level Viterbi in [1, 2] (Table 3.2).

We demonstrate some success and failure cases of our segment predictions in Fig.3.8. It shows how a half minute video can be segmented in a small number of steps. Only a limited window of frames at the start of each step decides the class and length of the corresponding segment. Green and red arrows indicate valid and wrong step duration respectively. Similarly, the correctness of the action selector prediction is shown by the color of the square.

Finally, Fig. 3.9 depicts a case where using visual features for length prediction outperforms the Poisson model in [2]. In this example "frying" is done slower than usual due to the subject turning away from the stove and the flipping of the egg. This makes the peak of the Poisson function temporally far from where "frying" actually ends resulting

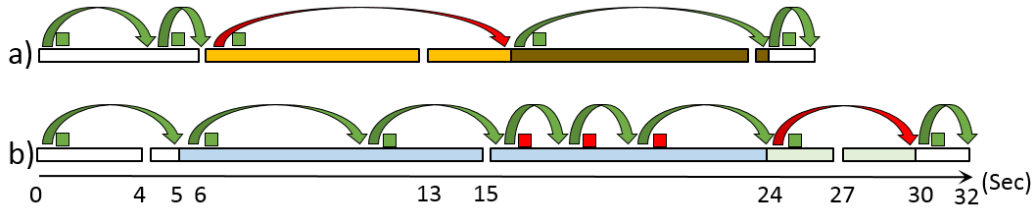


Figure 3.8: Separate segments denote the ground-truth and the color coded ones indicate the predicted segments. White segments are background. No-background actions are *add teabag* and *pour water* in video (a), and *pour cereal* and *pour milk* in video (b).

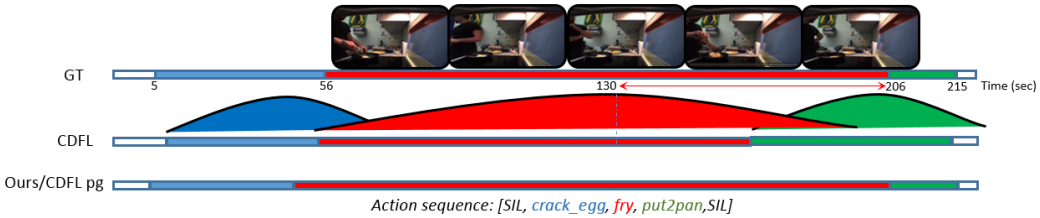


Figure 3.9: Alignment comparison between CDFL and our method for the test video "frying egg". Visual features in DurNet allow the predictions to adapt in duration. The color-coded curves represent the Poisson probability functions characterized by the expected duration of actions in CDFL.

in the premature end of the action as longer predictions have very low probabilities and discouraged by the Poisson model. However, our DurNet takes the visual features into account and adapts to longer than expected action durations.

3.6 Conclusion

We have proposed our *Duration Network*, that predicts the remaining duration of an action taking the video frame-based features into account. We also proposed a Segment-Level Beam Search that finds the best alignment given the inputs from the DurNet and action selector module. Our beam search efficiently aligns actions by considering only a selected set of frames with more confident predictions. Our experimental results show that our method can be used to produce efficient action alignment results that are also competitive to state of the art.

3.7 Supplementary Material

3.7.1 Implementation Details

In this section, we provide additional details about our experiments for both Breakfast [60] and Hollywood Extended [53] datasets.

In all our experiments, we trained our three proposed networks (Duration, Verb and Object Selectors) together with a dropout value of 0.89 and L2 regularization coefficient of 0.0001 for 40 epochs when using [55] as our pseudo ground-truth, and 90 epochs when using [1] and [2] pseudo ground-truth. Our input features were sampled every three frames over $\alpha = 60$ frames, at the start of each segment in time.

3.7.1.1 The Breakfast Dataset Experiments

We set 19 and 14 to be the number of objects and verbs (including background as a separate object/verb) in the Breakfast dataset. ζ, β, and λ were adjusted to 1, 30, and 5 respectively for our selector network using [1] and [2] as the baseline. In experiments where TCFPN results [55] were used as the initial pseudo ground-truth, the aforementioned parameters were slightly changed to 1, 40, and 1.

3.7.1.2 The Hollywood Dataset Experiments

There are 17 actions (including the background) in the Hollywood Extended dataset, and most of these actions do not share verbs or objects with each other. Hence, it would be redundant to decompose the main actions into their verb and object attributes. As a result, for this dataset, we removed the object selector component and used the 17 main actions as our verbs. β, and λ were set to 3 and 1, and 20 and 1 for the TCFPN [55] and NNViterbi [1] baselines respectively. In cases where CDFL [2] were used, β was increased to 50.

Around 60% of the frames are background in this dataset. Therefore, it is worth mentioning that a naive classifier, that outputs "background" for every single frame, can achieve results competitive to the state-of-the-art on the *acc* metric. This is why we emphasize that, specifically for the Hollywood Extended dataset, evaluation using *acc-bg* is more informative. Our method outperforms existing models on this metric while producing better or competitive results on *IoU*.

3.7.1.3 Competitors' Results

During our observations, we realized that the provided frame-level features are missing for a significant amount of frames in four videos[2] in the Breakfast dataset. While TCFPN [55], NNViterbi [1] and CDFL [2] originally trimmed those videos, we decided to remove them for all experiments including our method as well as all baselines [55, 1, 2]. In Tables 1 and 2 of the main content of this chapter, we denote with symbol † the best results that we obtained after running the authors' source code for multiple times. The reason we ran the code multiple times is that each training process is randomly initialized and leads to different final result.

For CDFL [2] in Table 1 and 2, the alignment *acc-bg* on the Hollywood dataset is somewhat different than the one mentioned in the referenced paper. Similarly, for TCFPN [55], in some cases, our reproduced results are not the same as the ones mentioned in [55]. In this case, we reported the results after contacting the authors and having their approval. For a fair comparison in both baselines, we reported the results, that represent the initial pseudo ground-truth in our method.

Without loss of generality, our final accuracy depends on the quality of the initial pseudo ground-truth, so we have provided the initial pseudo ground-truth and pre-trained main action recognizer models (for TCFPN and NNViterbi on the Breakfast dataset) that we used as supplementary material so our results can be reproduced precisely.

[2] 1-P34_cam01_P34_friedegg, 2-P51_webcam01_P51_coffee, 3- P52_stereo01_P52_sandwich, 4-P54_cam01_P54_pancake

4 Hierarchical Modeling for Task Recognition and Action Segmentation in Weakly-Labeled Instructional Videos

4.1 Abstract

This chapter [1] focuses on task recognition and action segmentation in weakly-labeled instructional videos, where only the ordered sequence of video-level actions is available during training. We propose a two-stream framework, which exploits semantic and temporal hierarchies to recognize top-level tasks in instructional videos. Further, we present a novel top-down weakly-supervised action segmentation approach, where the predicted task is used to constrain the inference of fine-grained action sequences. Experimental results on the popular Breakfast and Cooking 2 datasets show that our two-stream hierarchical task modeling significantly outperforms existing methods in top-level task recognition for all datasets and metrics. Additionally, using our task recognition framework in the proposed top-down action segmentation approach consistently improves the state of the art, while also reducing segmentation inference time by 80-90 percent.

4.2 Introduction

Millions of people watch instructional videos online every day, to learn to perform tasks such as cooking or changing a car tire. Also, new models of assistant robots [79] can learn from such videos how to assist humans in their daily lives. Hence, there has been extensive research in recent years on automated understanding of the top-level tasks and their sub-actions in such videos [80, 81, 82, 83].

[1] https://github.com/rezaghoddoosian/Hierarchical-Task-Modeling

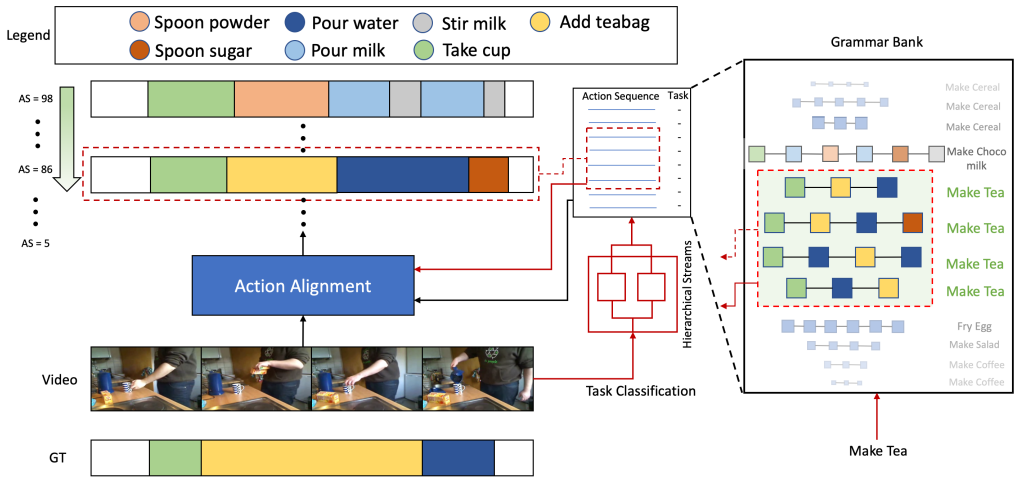


Figure 4.1: The proposed segmentation approach. Initially, the weak understanding of duration leads to the alignment of irrelevant actions with an Alignment Score (AS) of 98. However, a prior prediction of the task allows the alignment module to infer the correct sequence of actions (outlines by red) despite the lower AS (86).

From a theoretical point of view, instructional videos can be seen as videos illustrating hierarchical activities. Each instructional video illustrates a single top-level activity, for which we use the term "task" throughout this paper. Examples of such video-level tasks are "making coffee" or "cooking eggs". Each video-level task is composed of a sequence of lower level activities, such as "pouring milk" or "adding sugar". Throughout the paper, we will refer to such lower-level activities using the term "action". Consequently, using this terminology, each instructional video illustrates a task that consists of a sequence of actions.

For instructional videos, and hierarchical activity videos in general, we would like to have automated systems that both recognize the overall task and also understand what lower-level actions take place, and when those actions start and end. Fully-supervised training would require not only annotating the top level task, but also marking the start and end frame of each lower-level action. With the ever-growing size of instructional video datasets, manually annotating such start and end frames can quickly become a bottleneck. To address this issue, *weakly-supervised action segmentation* methods require, as training data, only the sequence of actions that takes place at each video, and no start/end frame information for those actions [54, 84, 55, 2, 1].

Our goal in this paper is to jointly address the problems of top-level task recognition and lower-level action segmentation, in the weakly supervised setting (given sequences

of actions, not given start/end frames). The main novelty is a method for top-level task recognition that uses, in parallel, two different hierarchical decompositions of the problem. One module models the semantic hierarchy between the top-level task and lower-level attributes. These attributes correspond to either the set of actions, or the set of the object/verb components of those actions, e.g., "take" and "cup" in the action "take cup". This module jointly learns to identify the presence of attributes in the video and to recognize the top-level task based on the estimated attributes.

A parallel second stream models the temporal hierarchy between the entire video and equal-duration subdivisions of the video. Tasks are usually performed in a relative order of stages, and some stages are particularly useful for distinguishing tasks from each other. For example, preparing tea typically involves three stages: taking a cup, adding a tea bag, and pouring water from the kettle. The first and last stages are visually similar with corresponding stages of the "preparing coffee" task. The temporal hierarchy module can capture the importance of adding the tea bag in distinguishing the "preparing tea" task from the "preparing coffee" task. This module learns the relation between stages and their importance in classifying the video task.

We also propose a novel top-down approach for action segmentation (i.e., frame-level action labeling), that combines our task recognition method with existing weakly-supervised segmentation methods. In this approach (Fig.4.1), the video-level task is estimated first, and is subsequently used to constrain the search space for action segmentation. In summary, the contributions of this paper are these:

1) We introduce a two-stream framework that exploits both semantic and temporal hierarchies to recognize tasks in weakly-labeled instructional videos.

2) We provide specific, non-trivial implementations of these two streams. Our ablation studies demonstrate that our implementation choices have a significant impact on performance. A highlight of such an implementation choice is using TF-IDF weights to model the discriminative power of each attribute for each task (see Section 4.4.2.2).

3) We present a novel top-down approach for weakly-supervised action segmentation, where the video-level task is used to constrain the segmentation output.

4) We present results on two benchmark datasets: Breakfast [60] and MPII Cooking 2 [82]. In top-level task recognition, our method significantly outperforms the state of the art on both datasets for all metrics. For weakly supervised action segmentation (frame-level labeling), applying the proposed top-down approach on top of existing methods [2, 1] again leads to state of the art results, and also cuts the inference time by 80-90 percent.

4.3 Related Work

Instructional Video Analysis. In recent years, untrimmed instructional videos have been studied in areas like video retrieval [80, 85, 86], quality assessment [87, 88], future action planning [89], and key-step segmentation [81, 90, 91, 71, 82, 83, 92]. Fully-supervised action segmentation methods [93, 60, 45, 94, 48, 49, 93] learn to identify action segments in the presence of frame-level ground-truth. For example, [83] use a bottom-up technique to aggregate initial action proposal scores to classify the top-level video task, before modifying its preliminary frame labels in a fully supervised way. Also, [82] analyze a host of holistic and regional features to train shared low-level classifiers to recognize tasks and detect fine-grained actions.

Recently, unsupervised learning of instructional videos has seen increased attention [81, 95, 90, 96]. In [90], an unsupervised approach performs video segmentation and task clustering through learned feature embeddings. In [81], a network is trained using only video task labels, for unsupervised discovery of procedure steps and task recognition.

The above-mentioned methods are either fully supervised or unsupervised, and thus they are not direct competitors for our method, which uses weak labels.

In the scope of activity recognition, most works [97, 98, 99] study short-range or trimmed videos. Our work is closest to [100, 101, 102], where the focus is recognizing minutes-long activities. However, unlike them, our paper is on instructional videos, and on how recognition can aid segmentation, so it relies on hierarchical activity labels (top-level task, lower-level attributes as targets for segmentation).

Weakly-Supervised Key-Step Localization. In the context of weakly-labeled instructional videos, many methods [103, 104, 68, 70] are trained under the supervision of narration and subtitle. Directly relevant to our work are [54, 84, 55, 56, 2, 59, 1], where, as in our method, only the sequence of actions is known for each training video. In particular [2, 1] deploy a factorized probabilistic model to tackle the segmentation problem using dynamic programming. Also, [54] formulate a differential dynamic programming framework for end-to-end training of their model.

Recent weakly-supervised segmentation methods [54, 84, 2, 1, 105, 106] are formulated to identify the action taking place at each frame, and not the top-level video task. At the same time, the output of these methods implicitly specifies the top-level task, because only one task is compatible with the detected sequence of actions. We use these implicit task predictions of [2, 1] to compare those methods to ours on task recognition accuracy. In contrast to these bottom-up approaches (going from actions to task), our method ex-

plicitly learns to classify video-level tasks, and this classification is used in a top-down fashion (from task to actions) to constrain the detected action sequence.

We should also mention the methods in [107, 108, 109, 110], which perform weakly-supervised action *detection*. These methods identify and localize occurrences of, typically, a single action in the input video. For completeness, we evaluate extensions of these methods to task classification.

4.4 Hierarchical Task Modeling Method

In this section, we present an overview of our two-stream hierarchical task modeling. Full details of our implementation choices and architecture are provided in Sec. 4.4.2.

As our formulation uses many terms and symbols, the supplementary material provides a glossary of terms and a table of all symbols we use.

4.4.1 Method Overview

Problem Definition. The training set $\mathbb{V} = \{\mathbf{v}_i\}_{i=1}^{N}$ consists of N videos $\mathbf{v}_i$. From each $\mathbf{v}_i$ we extract a feature vector $\mathbf{x}_i \in \mathbb{R}^{F \times T_i}$, that consists of T_i frames of F-dimensional features. We denote by $\mathbb{C} = \{c_i\}_{i=1}^{|\mathbb{C}|}$ the set of all top-level task labels, and by $\mathbb{A} = \{a_j\}_{j=1}^{|\mathbb{A}|}$ the set of all lower-level attribute labels. As an implementation choice, these attributes can be the set of actions in the dataset, or the set of verb/object components of those actions. Each video $\mathbf{v}_i$ is labeled by a task $c_i \in \mathbb{C}$, and also by a set $\mathbb{A}_i \subseteq \mathbb{A}$ of M_i attributes, so that $\mathbb{A}_i = \{a_{i,j}\}_{j=1}^{M_i}$. At test time, given an input video, the system estimates the top-level task.

Semantic Hierarchy Stream (SHS). To recognize the task, one approach is to directly estimate $p(c_i|\mathbf{x}_i)$. However, this approach is prone to overfitting when the number of video samples per task is limited. As attributes can be shared among tasks, the average number of training videos per attribute is typically greater than the average number of videos per task. Using attribute information also helps the model learn similarities and differences of spatio-temporal patterns in different tasks.

Thus, we model task recognition as $p(c_i|\psi_i^a).p(\psi_i^a|\mathbf{x}_i)$, where ψ_i^a is an intermediate vector of attribute scores that is computed for each $\mathbf{x}_i$. The system learns a mapping function $\mathscr{M}_x^a : \mathbb{R}^{F \times T_i} \rightarrow \mathbb{R}^{|\mathbb{A}|}$, that maps each vector $\mathbf{x}_i$ to attribute score vector ψ_i^a. It also learns a function $\mathscr{M}_a^c : \mathbb{R}^{|\mathbb{A}|} \rightarrow \mathbb{R}^{|\mathbb{C}|}$, that maps each attribute score vector ψ_i^a to a task score vector ψ_i^c (Fig.4.2).

Temporal Hierarchy Stream (THS). Tasks in instructional videos are usually performed in a relative order of steps. Understanding the task-discriminative stages of a

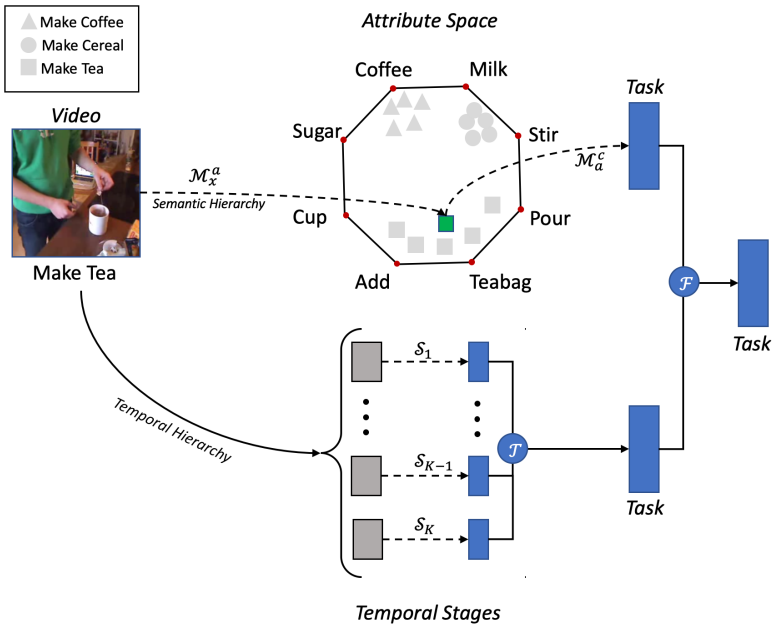


Figure 4.2: An overview of how our recognition model exploits the semantic and temporal hierarchies of tasks. The attribute representation of videos are formed by their discriminative attributes.

video is essential in distinguishing tasks that share similar-looking actions. Thus, we divide each video into K stages of equal duration, and train a classifier $\mathscr{S}_{\kappa} : \mathbb{R}^{F \times \frac{T_i}{K}} \rightarrow \mathbb{R}^{|\mathbb{C}|}$ for each stage κ. The system also learns an aggregation function $\mathscr{T} : \mathbb{R}^{K|\mathbb{C}|} \rightarrow \mathbb{R}^{|\mathbb{C}|}$, that maps stage-wise predictions to classification scores $\vartheta_{i,total}$ of the entire video.

Stream Fusion Module. In the end, we fuse the predictions of the SHS and THS streams to output the final task prediction scores f_i^c of the entire model. A high-level diagram of the overall network is shown on Fig.4.2. The network is optimized using a loss function for the fusion module, as well as separate loss functions for the SHS and THS streams.

4.4.2 Detailed Architecture

In this section we explain in detail the architecture of our two-stream hierarchical model (Fig.4.3), and we derive the three proposed loss functions.

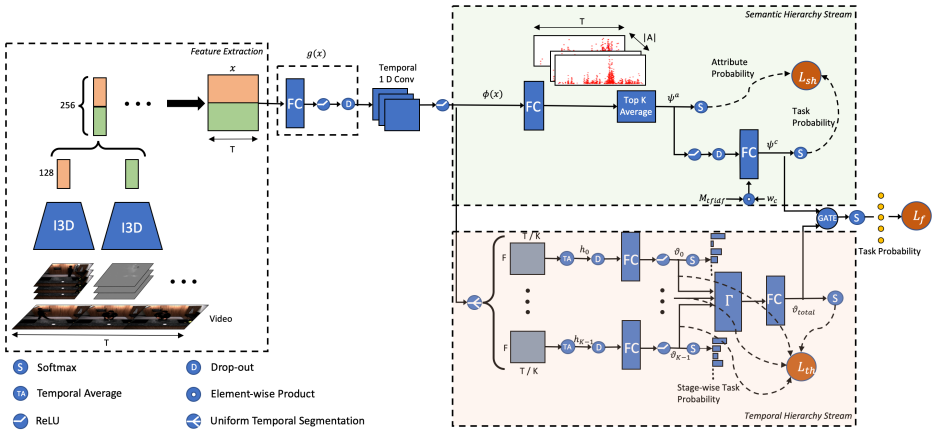


Figure 4.3: Our two-stream model architecture for task classification using RGB and flow frames as input. The semantic hierarchy loss $\mathscr{L}_{sh}$ ensures task classification after clustering videos based on their shared weighted attributes through the TF-IDF mask $\mathbf{M}_{tfidf}$. The THS stream learns to aggregate stage-wise task predictions by $\mathscr{L}_{th}$, and a third loss ($\mathscr{L}_f$) optimizes the fused results of both streams.

4.4.2.1 Feature Extraction

Video task recognition is highly dependent not only on motion patterns, but also on object appearance. Ignoring object appearance can lead to misclassifications when the motion patterns of two tasks are very similar, e.g., making coffee and making tea. Hence, instead of the mostly motion-based iDT features[77] used in [54, 2, 1], we adopt the I3D network, pre-trained on the Kinetics dataset[111]. I3D extracts, for each frame, 1024-dimensional feature vectors respectively from the RGB and optical flow channels. We use PCA separately on RGB and flow features, to reduce the dimensions from 1024 to 128.

The 256-dimensional concatenated RGB and optical flow features of each frame are stored in video-level feature vector $\mathbf{x}_i \in \mathbb{R}^{256\times T_i}$, where T_i is the total number of frames in video $\mathbf{v}_i$. In principle, any spatio-temporal network can be used instead of I3D. In the supplementary material, we show that I3D outperforms iDT for task recognition.

4.4.2.2 Semantic Hierarchy Loss

In order to obtain a data-specific representation of video-level feature vector $\mathbf{x}_i$, we pass each frame-level subvector of $\mathbf{x}_i$ through a fully-connected layer g with bias and output dimension of 256, then apply temporal convolution to the output of g. Using such 1D temporal convolutions with a set of F learnable kernels $\mathbf{k}_\phi \in \mathbb{R}^{L\times 256}$ of size L , $\mathbf{x}_i$ is eventually mapped to a F-dimensional feature encoding $\phi(\mathbf{x}_i) \in \mathbb{R}^{F\times T_i}$.

Let $|\mathbb{A}|$ be the number of unique attributes in the dataset. We pass each frame-level subvector of $\phi(\mathbf{x}_i)$ through a fully connected layer with bias and output dimension of $|\mathbb{A}|$ to obtain $\Psi_i^a \in \mathbb{R}^{|\mathbb{A}| \times T_i}$, which is the sequence of attribute scores for each of the T_i frames. Ψ_i^a is also known as *temporal class activation map* (T-CAM)[112].

Similar to [109, 113], we compute a video-level attribute score vector ψ_i^a by average-pooling the highest $k_i = \lfloor \frac{T_i}{s} \rfloor$ T-CAM scores of each attribute separately over time, where s is a hyperparameter:

$$\psi_i^a[j] = \frac{1}{k_i} \sum_{i=0}^{k_i - 1} \text{top}_\text{k}\{\Psi_i^a[j,:]\} \tag{4.1}$$

Intuitively, these selected k_i scores highlight the most important parts of a task in video i.

To denote the set $\mathbb{A}_i$ of attributes present in video $\mathbf{v}_i$, we define a multihot ground-truth attribute vector $\vec{a}_i \in \{0,1\}^{|\mathbb{A}|}$, where for every $j \in \{0,1,...,|\mathbb{A}|-1\}$, $\vec{a}_{i,j} = 1$ if $a_j \in \mathbb{A}_i$, otherwise $\vec{a}_{i,j} = 0$. However, this representation fails to capture the fact that different attributes have different levels of relevance for recognizing each task. For example, attributes "take out" and "open" are present in most videos, and thus not discriminative. As a second example, for the "preparing avocado" task, "avocado" is a more informative attribute compared to "knife".

Inspired by text retrieval methods [114, 115], we compute the TF-IDF weight matrix $\mathbf{W}_{tfidf} \in \mathbb{R}^{|\mathbb{A}| \times |\mathbb{C}|}$, so that $\mathbf{W}_{tfidf}(j,\tau)$ captures the importance of attribute j for task τ. Initially, we formulate $\text{TF} \in \mathbb{R}^{|\mathbb{A}| \times |\mathbb{C}|}$ and $\text{IDF} \in \mathbb{R}^{|\mathbb{A}|}$ as follows:

$$\text{TF}(j,\tau) = \frac{\sum_{i=0}^{N-1} \vec{a}_{i,j} \cdot \mathbb{1}(\tau = c_i)}{\sum_{i=0}^{N-1} \mathbb{1}(\tau = c_i)} \tag{4.2}$$

$$\text{IDF}(j) = \log\left(\frac{|\mathbb{C}|}{|\{\tau \in \mathbb{C} | \text{TF}(j,\tau) > 0\}|}\right) \tag{4.3}$$

where $\mathbb{1}()$ denotes the indicator function and $\text{TF}(j,\tau)$ and $\text{IDF}(j)$ are, respectively, the percentage of times attribute a_j is present in videos of task $\tau \in \mathbb{C}$, and the log inverse of percentage of all tasks that entail attribute j in at least one of their videos. We then define the elements $\mathbf{W}_{tfidf}(j,\tau)$ of the TF-IDF weight matrix as:

$$\mathbf{W}_{tfidf}(j,\tau) = \frac{\text{TF}(j,\tau) \cdot \text{IDF}(j)}{\varepsilon + \sum_{k=0}^{|\mathbb{A}|-1} \text{TF}(k,\tau) \cdot \text{IDF}(k)} \tag{4.4}$$

with ε set to a very small value to avoid division by zero.

Using these TF-IDF weights, we intorduce the TF-IDF-weighted attribute ground-truth vector $\vec{a^w}_i \in \mathbb{R}^{|\mathbb{A}|}$ as:

$$\vec{a^w}_{i,j} = \frac{\vec{a}_{i,j} \cdot \mathbf{W}_{tfidf}(j, c_i)}{\sum_{k=0}^{|\mathbb{A}|-1} \vec{a}_{i,k} \cdot \mathbf{W}_{tfidf}(k, c_i)} \tag{4.5}$$

We also define a TF-IDF mask $\mathbf{M}_{tfidf} \in \mathbb{R}^{|\mathbb{A}|\times|\mathbb{C}|}$, where $\mathbf{M}_{tfidf}(j,\tau)$ is 1 if the corresponding TF-IDF weight $\mathbf{W}_{tfidf}(j,\tau)$ is nonzero, otherwise it is 0. We use the TF-IDF mask to form a mapping from attribute score vectors ψ_i^a to task probability scores $\hat{\psi}_i^c \in \mathbb{R}^{|\mathbb{C}|}$, as:

$$\hat{\psi}_i{}^c = \mathfrak{s}[(\mathbf{w}_c \odot \mathbf{M}_{tfidf}^T)\mathrm{ReLU}(\psi_i^a)] \tag{4.6}$$

In the above, $\mathfrak{s}[]$ and $\odot$ mean the softmax and element-wise product operations respectively, and $\mathbf{w}_c \in \mathbb{R}^{|\mathbb{C}|\times|\mathbb{A}|}$ are weights to be learned. Using the TF-IDF mask allows the model to focus only on relevant attributes for each task.

Let $\vec{c}_i$ be the one-hot task ground-truth vector and $\hat{\psi}^a = \mathfrak{s}[\psi^a]$. The semantic hierarchy loss $\mathscr{L}_{sh}$ is then defined as:

$$\mathscr{L}_{sh} = -\lambda\mathbb{E}[\vec{a^w}_i^T log(\hat{\psi}_i^a)] - (1-\lambda)\mathbb{E}[\vec{c}_i^T log(\hat{\psi}_i{}^c)] \tag{4.7}$$

$\mathbb{E}$ denotes "expected value", and λ is a design parameter that decides how fast each term is trained comparatively.

4.4.2.3 Temporal Hierarchy Loss

We model the temporal hierarchy by dividing each video into K stages of equal duration d, and training a classifier for each stage. Formally, given the frame-level feature encoding $\phi(\mathbf{x}_i)$ of video i, we define $h_{i,\kappa} \in \mathbb{R}^F$ as the feature summary of the κ-th stage and produce unnormalized task scores (logits) $\vartheta_{i,\kappa} \in \mathbb{R}^{|\mathbb{C}|}$:

$$h_{i,\kappa} = \frac{\sum_{t=\kappa d}^{[(\kappa+1)d]-1} \phi(\mathbf{x}_i)[:,t]}{d} \tag{4.8}$$

$$\vartheta_{i,\kappa} = \mathbf{w}_\kappa h_{i,\kappa} + b_\kappa \tag{4.9}$$

where $\mathbf{w}_\kappa \in \mathbb{R}^{|\mathbb{C}|\times F}$ and $b_\kappa \in \mathbb{R}^{|\mathbb{C}|}$ are parameters of each stage. During the training process, for each stage, the loss function $\mathscr{L}_\kappa = \mathbb{E}[\vec{c}_i^T log(\hat{\vartheta}_{i,\kappa})]$ is defined on the softmax of the stage-wise task prediction logits $\hat{\vartheta}_{i,\kappa}$.

Stage Aggregation Function. As mentioned earlier, certain stages of a task are more discriminative than others. We define the auxiliary function $\Gamma([\vartheta_0\ \vartheta_1\ ...\ \vartheta_{K-1}])$, that maps stage-level task score vectors to a video-level task score vector. While training, $\Gamma(\,)$ randomly masks out one of its K input prediction vectors ϑ_κ entirely and multiplies the rest of the input predictions by $\frac{K}{K-1}$. $\Gamma(\,)$ acts similar to the spatial drop out operation by promoting independence between predictions of each stage, to avoid overfitting to a single stage. We form our stage aggregation function using $\Gamma(\,)$ and aggregation parameters

$\mathbf{w}_{total} \in \mathbb{R}^{K|\mathbb{C}|\times|\mathbb{C}|}$ to produce video-level task probability values $\hat{\vartheta}_{i,total} \in \mathbb{R}^{|\mathbb{C}|}$:

$$\hat{\vartheta}_{i,total}^{T} = \Gamma(\text{ReLU}([\vartheta_{i,0}^{T}\ \vartheta_{i,1}^{T}\ ...\ \vartheta_{i,K-1}^{T}]))\ \mathbf{w}_{total} \tag{4.10}$$

Finally, we present our temporal hierarchy loss $\mathscr{L}_{th}$ to incorporate the aggregated and stage-wise predictions:

$$\mathscr{L}_{th} = -\mathbb{E}[\vec{c}_i^{T}\, log(\hat{\vartheta}_{i,total})] - \sum_{\kappa=0}^{K-1} \mathscr{L}_{\kappa} \tag{4.11}$$

4.4.2.4 Stream Fusion Loss

We explore three different mechanisms for fusing the predictions of the SHS and THS streams, to produce the final task prediction logits f_i^c. We provide experimental results of each in Section 4.6.2.

Average Fusion. Here, we treat results of semantic and temporal hierarchies equally, and we backpropagate the same gradient to both streams at training time.

$$f_i^c = 0.5(\vartheta_{i,total} + \psi_i^c) \tag{4.12}$$

Weighted Average Fusion. Here, the final prediction is a linear combination of streams, whose predictions are weighted by learned weights $\mathbf{w_1}, \mathbf{w_2} \in \mathbb{R}^{|\mathbb{C}|\times|\mathbb{C}|}$. .

$$f_i^c = \mathbf{w_1}\vartheta_{i,total} + \mathbf{w_2}\psi_i^c \tag{4.13}$$

Task-wise Switching Gates. Sometimes, wrong predictions of one stream can negatively impact the final fused classification scores f_i^c at test time. We introduce task-wise switching gates to allow the system enough freedom to learn, for each task independently, to switch between stream predictions. We define switching gate $\alpha = \sigma(w_\alpha)$ as the sigmoid function $\sigma()$ of learnable parameters $w_\alpha \in \mathbb{R}^{|\mathbb{C}|}$. The sigmoid function makes sure our gates stay in the range of 0 to 1. Then, for training and test time, f_i^c is defined as:

$$f_i^c = \begin{cases} \alpha \odot \vartheta_{i,total} + (1-\alpha) \odot \psi_i^c, & \text{training} \\ \mathscr{H}_{0.5}(\alpha) \odot \vartheta_{i,total} + (1-\mathscr{H}_{0.5}(\alpha)) \odot \psi_i^c, & \text{test} \end{cases} \tag{4.14}$$

where $\mathscr{H}_x()$ denotes the Heaviside step function shifted to x. At test time, given task τ, our final prediction is discretely chosen from the SHS stream if $\alpha_\tau < 0.5$ or is selected from the THS stream otherwise.

In cases of the weighted average fusion and switching gates, our fusion loss $\mathscr{L}_f = -\mathbb{E}[\vec{c}_i^{T}\, log(\hat{f}_i^c)]$ is added to the previous losses to form our final loss $\mathscr{L}$ with the design

parameter β. Finally we train the whole network end-to-end but stop the gradients of $\mathscr{L}_f$ flowing back to the streams to isolate the fusion module from the rest.

$$\mathscr{L} = \mathscr{L}_f + \mathscr{L}_{sh} + \beta \mathscr{L}_{th} \tag{4.15}$$

4.5 Top-Down Action Segmentation

We now present our top-down segmentation approach. In the segmentation problem, the goal is to partition a video temporally into a sequence of S action labels δ_1^S and their corresponding durations $\mathbf{l}_1^S$. The input in our approach is a video of T_v frames, represented by $\mathbf{x}_1^{T_v}$, as the sequence of per-frame features. Let $\Pi(\tau)$ be the set of all action sequences in the training set given the top-level task τ. Then, grammar $\pi \in \Pi(\tau)$ lists an ordered sequence of S action labels taking place in the video of task τ. The goal is to identify the most likely sequence of action labels $\overline{\delta}_1^S$ and their durations $\bar{\mathbf{l}}_1^S$ associated with a specific grammar π:

$$(\overline{\delta}_1^S, \bar{\mathbf{l}}_1^S, \overline{\tau}) = \underset{\delta_1^S, \mathbf{l}_1^S, \tau}{\operatorname{argmax}} p(\delta_1^S, \mathbf{l}_1^S, \tau | \mathbf{x}_1^{T_v}) \tag{4.16}$$

$$= \underset{\delta_1^S \in \Pi(\tau), \mathbf{l}_1^S, \tau}{\operatorname{argmax}} p(\mathbf{x}_1^{T_v} | \delta_1^S) p(\mathbf{l}_1^S | \delta_1^S) p(\delta_1^S | \tau) p(\tau) \tag{4.17}$$

where $p(\mathbf{x}_1^{T_v} | \delta_1^S)$ is modeled by a neural network and the Bayes rule as in [59, 1], and $p(\mathbf{l}_1^S | \delta_1^S)$ is any given duration model, e.g., Poisson[1] or DurNet[116].

Eq.5.3 is formulated similarly to the probabilistic model in [1]. However, we explicitly integrate the task variable $\tau \in \mathbb{C}$ into this equation, which dictates the choice of the fine-grained action sequence δ_1^S. Specifically we introduce the task model $p(\tau)$ as the probability output of a task classification network. Without loss of generality, we used the output of our two-stream hierarchical network $\hat{f}^c$, so that $p(\tau) = 1$ for the predicted task $\tau = \operatorname{argmax}(\hat{f}^c)$ and $p(\tau) = 0$ otherwise. In [1], the task is a by-product of the inferred segments ($\overline{\delta}_1^S$,$\bar{\mathbf{l}}_1^S$). In contrast, our proposed approach eliminates all segmentations whose inferred actions do not belong to $\Pi(\tau)$ of the predicted task τ by setting $p(\delta_1^S | \tau)$ to 0 for those segmentations, and to 1 otherwise. We follow the Viterbi algorithm in [1] to solve Eq. 5.3.

4.6 Experiments

We compare our method to several existing methods on two popular instructional video datasets, both for task classification and for action segmentation using weakly-labeled videos as training. Further, in ablation studies we evaluate the contribution of each component of our model.

Datasets. 1) *The Breakfast Dataset (BD)* [60] consists of around 1.7k untrimmed cooking videos of few seconds to over ten minutes long. There are 48 action labels demonstrating 10 breakfast dishes with a mean of 4.9 actions per video, and the evaluation metrics are conventionally calculated over four splits. 2) *The MPII Cooking 2 (C2)* [82] has training and test subject-wise splits of 201 and 42 long and high quality videos respectively. Particularly, these videos are 1 to 40 mins long adding to 27 hours of data from 29 subjects who prepare 58 different dishes. This dataset offers different challenges compared to the *BD* dataset for two main reasons; First, the annotated 155 objects and 67 actions (verbs) are extremely fine-grained, so that there are on average 51.8 non-background action segments per video. Second, despite the great number of frames in the dataset, the number of samples per class is unbalanced and limited.

Metrics. We evaluate task classification performance using two metrics: 1) *t-acc* is the standard mean task accuracy over all videos. 2) *t-mAP* denotes the mean Average Precision of task predictions. *mAP* is used in [82] to assign soft class-wise scores to give insight about how far off the wrong predictions are. Further, we use four metrics as [55] to measure the segmentation results: *acc* and *acc-bg* are the frame-level accuracies with and without background frames, while *IoU* and *IoD* define the average non-background intersection over union and detection, respectively.

Implementation. We extracted I3D features on the *C2* dataset using TV-L1 optical flow [117] on a moving window of 32 frames with stride 2, and the pre-computed I3D features of the *BD* dataset were obtained from [93]. We noticed that it is not necessary to process the whole video. Instead, we followed the sampling strategy in [109] to maintain the length of the videos in a batch to be less than a pre-defined length $T \approx 9$ mins while training. This approach speeds up training, lowers memory demands, and applies temporal augmentation. Also, we divided videos into $K = 3$ stages for the THS stream (analysis in Section 4.6.2).

Our model is trained with a batch size of 10 using the Adam [35] optimizer with 10^{-3} learning rate and 0.005 weight decay for 20k iterations. For both datasets, we adjust λ to 0.9, and β is set to 0.25 and 0.01 for the *BD* and *C2* datasets, respectively. The 1D convolutions are done with $F = 64$ as the number of kernels, and $L = 15$ as their size.

Table 4.1: Task classification results of state-of-the-art methods on two main datasets. Best results reported out of I3D (†) or iDT (‡) features (more in supplementary material and [118]).* results obtained using the author's source code. [90] results for 10 classes.

		Breakfast (%)		Cooking (%)	
Supervision	Models	t-acc	t-mAP	t-acc	t-mAP
Full	Rohrbach *et al.*[82]	-	-	-	57.40
Unsupervised	CTE[90]	31.80‡	-	-	-
Weak	NNViterbi[1]*	70.98‡	-	23.80†	-
	CDFL[2]*	74.86‡	-	28.57†	-
	W-TALC[109]*	76.19†	80.98†	33.33†	43.07†
	3C-Net[107]*	75.23†	80.99†	30.95†	46.30†
	Timeception[100]*	76.37†	80.80†	21.43†	25.14†
	VideoGraph[101]*	78.70†	-	23.80†	-
	Our Method	**80.04**	**86.36**	**45.24**	**54.49**

$s = 8$ and we use a drop-out keep rate of 0.3. The set of verbs and objects are used as our attributes.

4.6.1 Comparison to State-of-the-Art Methods

We used the standard dish labels in both datasets as task labels. All experiments on the *BD* dataset for all models, except the unsupervised CTE [90], were done for 9 tasks after we combined the two dishes of *frying* and *scrambling eggs* as the top-level task of *making eggs*, because both share almost the same set of actions. For CTE, we report the results on the original 10 classes, as given by the authors. We note that CTE is unsupervised and not a direct competitor.

Task Classification. Table 4.1 shows quantitative results on task recognition, for our method as well as other methods that use different types of supervision. Particularly, [2, 1] are the state-of-the-art open-source weakly-supervised segmentation methods. They implicitly identify the task corresponding to the inferred sequence of actions during inference. Our explicit task modeling significantly outperforms them in accuracy by around 5 to 9 percent on the *BD* dataset, and by 16 to 21 percent in the 58 tasks of the *C2* dataset.

Originally, [107, 109] are the state-of-the-art open-source weakly-supervised methods with specific loss functions to classify and localize sparse action instances in videos. To compare with them, we trained both to classify tasks. Also, [100] and [101] classify tasks in long videos by training multi-scale temporal convolutions and graph based representations respectively. Both networks make heavy use of memory and suffer from overfitting specifically in the *C2* dataset, where using low-level attributes is key. While such direct

Table 4.2: Consistent performance gain in weakly-supervised action segmentation following our proposed top-down approach. I3D and iDT features used for experiments on the *C2* and *BD* datasets, respectively. (* as specified in Table 4.1, **: no source code).

	Breakfast (%)				Cooking (%)			
Models	acc	acc-bg	IoU	IoD	acc	acc-bg	IoU	IoD
TCFPN[55]*	38.4	38.4	24.2	40.6	26.9	30.3	9.5	**17.0**
D3TW[54]**	45.7	-	-	-	-	-	-	-
DP-DTW[84]**	50.8	-	**35.6**	45.1	-	-	-	-
NNViterbi[1]*	43.6	42.5	27.8	39.2	23.5	21.2	7.7	10.9
CDFL[2]*	50.2	50.4	33.5	45.6	29.9	32.2	11.0	13.8
NNViterbi+Ours	46.2	46.1	30.2	42.2	26.9	25.0	9.6	12.7
CDFL+Ours	**51.4**	**52.0**	34.5	**46.7**	**31.3**	**34.5**	**12.8**	15.6
CDFL+GT	*59.8*	*63.0*	*41.3*	*55.2*	*35.0*	*39.7*	*14.4*	*17.6*

Table 4.3: Inference run time (minutes) improvement of state-of-the-art following the proposed top-down approach for segmentation.

Models	Breakfast (split 4)	Cooking
NNViterbi[1]	100	840
CDFL[2]	144	1070
NNViterbi+Ours	21	64
CDFL+Ours	25	110

task modelings under weak supervision prove to be more effective than the implicit classification using fine-grained action segments [2, 1], our hierarchical approach outperforms all competitors considerably in all metrics and datasets. Table 4.1 shows our *t-mAP* on the *C2* dataset comes close to the fully-supervised baseline [82], which is trained on frame-level action ground-truth. Comparison results on 10 classes of the *BD* dataset are in the supp. material.

Action Segmentation. Table 4.2 shows results for action segmentation. In our experiments, we combined our two-stream task prediction framework on top of the state-of-the-art weakly-supervised segmentation methods [2, 1] and achieved new state-of-the art results on both datasets, manifested more vividly in *acc-bg*, because background frames are independent of the task. Therefore, excluding background frames highlights the contribution of the correct task label in segmentation. This consistent improvement in all metrics, while decreasing the inference time by 80-90% (Table 4.3), demonstrates the potential of the proposed top-down approach for weakly-supervised segmentation. Moreover, *CDFL+GT* in Table 4.2 represents CDFL segmentation results constrained by ground-truth task labels, which serves as an upper bound for our proposed top-down model.

4.6.2 Analysis and Ablation Study in Task Modeling

Stream-Specific Results. We evaluated the contribution of the SHS and THS streams separately in Table 4.4. The SHS stream is more effective on the *C2* dataset because of two main reasons: First, the average number of videos per task (3.4) is low compared to that of videos per attribute (28.4), so any direct way of task modeling is prone to overfitting. Second, the large number of attributes per task allows the learning of a discriminative attribute-to-task mapping. Meanwhile in the THS stream, despite the weak classification power of the stage-specific classifiers, our hierarchical modeling is able to aggregate stage-wise predictions effectively and produce significantly superior results. This shows that different stages provide complimentary information. Note that the THS stream alone achieves state-of-the-art on the *BD* dataset with only task label supervision.

Semantic Hierarchy Ablation. As shown in Table 4.5a, removing the attributes from the semantic hierarchy loss (Eq.4.7), and directly classifying tasks from the feature encoding $\phi(\mathbf{x})$, leads to around 12% drop in *t-acc* on the *C2* dataset. Simply sharing low-level attributes among tasks is beneficial, and using the TF-IDF weights led to an additional 7% difference in *t-acc* (Table 4.5b).

Temporal Hierarchy Analysis. Modeling tasks as a temporal hierarchy of multiple stages improves the performance compared to the single-stage approach. Furthermore, as indicated by Table 4.6, such an approach is not sensitive to the number of stages ($K > 1$) in the hierarchy. This concludes that these stages provide complimentary information for the stage-aggregation function regardless of their exact positioning or duration in the video.

Comparison of the Stream Fusion Mechanisms. Table 4.7 compares different mechanisms to fuse the predictions of SHS and THS streams. Specifically, the Task-wise Switching Gates are trained to identify the stronger stream per task and perform best, while the vanilla and weighted averaging compromise between both streams and produce sub-optimal results. In the *BD* dataset these gates propagate the results of the THS stream,

Table 4.4: Stream-specific ablation study for task classification.

	Breakfast (%)		Cooking (%)	
Stream	t-acc	t-mAP	t-acc	t-mAP
Semantic hierarchy	73.1	77.2	42.9	52.6
Temporal hierarchy-stage 1	62.7	-	16.7	-
Temporal hierarchy-stage 2	68.9	-	28.6	-
Temporal hierarchy-stage 3	64.7	-	23.8	-
Temporal hierarchy-aggregated	80.0	86.4	31.0	45.2
Two streams fused	**80.0**	**86.4**	**45.2**	**54.5**

Table 4.5: The effect of sharing attributes (a) and using TF-IDF weights (b) in the SHS stream (Cooking dataset).

(a)

Setting	t-acc	t-mAP
Without attributes	33.3	45.4
With attributes	**45.2**	**54.5**

(b)

Setting	t-acc	t-mAP
Without TF-IDF	38.1	49.4
With TF-IDF	**45.2**	**54.5**

Table 4.6: Evaluation of our model under different choices of K on the *BD*.

# Stages	t-acc	t-mAP
1	74.2	82.1
2	79.6	86.4
3	80.0	86.4
4	80.0	86.4
5	79.2	85.8

Table 4.7: Comparison between different fusion mechanisms on the *BD*.

Fusion Type	t-acc	t-mAP
Average	77.2	84.2
Weighted Average	78.4	84.9
Switching Gate	**80.0**	**86.4**

whereas in the *C2* dataset they switch between both for different tasks and combine predictions (see row 1, 5 and 6 of Table 4.4).

4.6.3 Qualitative Results

Fig.4.4 compares results of the THS stream with the single-stage baseline on four challenging videos from the *BD* dataset. Due to similar-looking actions shared between tasks, the single stage baseline misclassifies the video task, whereas our THS stream outputs the correct task under different stage-wise settings. Specifically, in the first and third videos, our model classifies the task correctly although only one of the stage predictions is correct. For example, according to the stage-wise accuracy of the task *making chocolate milk* in Fig.4.4, the last stage of this task is the most discriminative one. Thus, our model learns to put more weight on the predictions of this stage, which compensates for the first two stages outputting the wrong class of *making cereal*.

The two tasks of *making coffee* and *making tea* share similar-looking actions in the first and second videos, so analyzing the entire video in one step produces wrong predictions for both cases. The second video, in particular, provides an interesting case where similar visuals of the action *taking cup* between tasks of *making chocolate milk* and *making tea* led to confusion of the first stage. Also, the later two stages mistakenly predicted the

Figure 4.4: Qualitative task classification results of our THS stream on sample challenging videos from the *BD* dataset. For each video, we show the ground-truth task label, predictions from the single-stage baseline, and the stage-wise and stage-aggregated recognition result. Also, stage-wise accuracy for each of the four tasks is presented on the right side. In the first video, the subject sequentially takes a cup, pours coffee and milk, adds sugar and stirs coffee. In the second video, the subject takes a cup, adds teabag, pours hot water, spoons sugar and stirs. In the third video, the subject takes a cup, pours milk, spoons choc. powder, adds sugar and stirs. In the fourth video, the subject pours cereal, spoons choc. powder, pours milk, and then stirs. Red and green boxes denote wrong and correct predictions, respectively.

task of *making coffee*, because the two actions of *pouring* and *stirring* are shared between both tasks of *making coffee* and *making tea*. Although all three stage-wise predictions are wrong, the aggregated result of those stages is correct. This shows that the proposed hierarchical model not only considers the predicted class of each stage, but also learns the relationship between stages and their fine-grained prediction scores.

In a given stage, the short discriminative part may be dominated by the longer ambiguous section. For example, the final stage of the last video depicts how *stirring* while occluding the bowl dominates the shorter and more discriminative action of *pouring milk*. This effectively resembles the appearance and motion of *flipping a pancake by spatula*, but the complementary information of the first two stages eventually results in the correct aggregated recognition.

4.7 Conclusion

We have introduced a two-stream framework, that exploits semantic and temporal hierarchies to recognize tasks in weakly-labeled instructional videos. We have also proposed

a novel top-down segmentation approach, where the predicted task constrains the fine-grained action labels. We report experimental results on two public datasets. Our two-stream task recognition method outperforms existing methods. Similarly, our top-down segmentation approach improves the accuracy of existing state-of-the-art methods, while simultaneously improving runtime by 80-90%.

4.8 Supplementary Material

4.8.1 Overview

In this supplementary material, we show comparisons of I3D [111] and iDT [77] features in task recognition on two datasets, and present comparison results on the original 10 classes of the Breakfast dataset [60]. We also provide a glossary of terms and a table of symbols we use in the paper.

4.8.2 I3D and iDT Feature Comparison in Task Recognition of Weakly-Labeled Videos

In this section, we compare I3D and iDT features for the purpose of task recognition in weakly-labeled instructional videos. Specifically, we present results of existing models using I3D (Table 4.8) and iDT (Table 4.9) features on the MPII Cooking 2 dataset [82] as well as the first split of the Breakfast dataset [60].

We used the Fisher vectors of iDT features as in [60, 71]. The Fisher vectors for each frame are extracted over a sliding window of 20 frames. They are first projected to a 64-dimensional space by PCA, and then normalized along each dimension. Also, we extracted the I3D features of the Cooking 2 dataset using TV-L1 optical flow [117] on a moving window of 32 frames with a stride of 2, and the pre-computed I3D features of the Breakfast dataset were obtained from [51]. Furthermore, we applied PCA to the extracted I3D features to reduce the dimensionality of RGB and optical flow channels from 1024 to 128. We fed the same features to all competitors except [102] in Table 4.10 whose code is not publicly available, so we compare with their reported result on ResNet101 [119] features.

In the Cooking 2 dataset, we train the models on the training split and test on the test split. However as [2] and [1] take a long time to train and infer the segments, in Tables 4.8 and 4.9, we only use the first split of the Breakfast dataset to evaluate the difference in performance of all models when using I3D and iDT features as input. Note that the reported task recognition results on the Breakfast dataset in the paper are the average of all four splits using the best case for each method.

Explicit task classification methods, e.g., ours, W-TALC [109] and 3C-Net [107], consistently perform better with I3D features on both datasets, whereas the bottom-up inference of tasks in NNViterbi [1] and CDFL [2] produces mixed result. In particular, the performance of [2] and [1] on the Breakfast dataset considerably improves upon using iDT features. Overall, the more significant presence of object information in I3D features helps

	Breakfast (1st split) (%)		Cooking (%)	
Models	t-acc	t-mAP	t-acc	t-mAP
NNViterbi[1]*	57.14	-	23.80	-
CDFL[2]*	66.26	-	28.57	-
W-TALC[109]*	75.79	78.96	33.33	43.07
3C-Net[107]*	75.39	78.50	30.95	46.30
Timeception[100]*	79.50	82.53	21.43	25.14
VideoGraph[101]*	80.06	-	23.80	-
Our Method	81.74	88.30	45.24	54.49

Table 4.8: Task classification results of state-of-the-art methods using I3D features on the Cooking 2 dataset and the first split of the Breakfast dataset. (* results obtained using the author's source code).

	Breakfast (1st split) (%)		Cooking (%)	
Models	t-acc	t-mAP	t-acc	t-mAP
NNViterbi[1]*	71.03	-	16.66	-
CDFL[2]*	77.38	-	21.42	-
W-TALC[109]*	53.17	54.96	19.04	25.85
3C-Net[107]*	56.74	60.36	14.28	27.38
Timeception[100]*	65.87	71.73	9.52	14.36
VideoGraph[101]*	58.93	-	14.28	-
Our Method	60.31	61.72	23.80	27.66

Table 4.9: Task classification results of state-of-the-art methods using iDT features on the Cooking 2 dataset and the first split of the Breakfast dataset. (* results obtained using the author's source code).

to classify top-level tasks more accurately, while detecting fine-grained actions seems to be less affected by such appearance information.

4.8.3 Task Classification Results on 10 Classes of the Breakfast Dataset

Timception [100], VideoGraph [101] and RhyRNN [102] are the latest state-of-the-art methods to classify tasks in minutes-long videos and are the closest competitors to our work. We compared the standard four fold cross validated results of Timeception and VideoGraph over 9 classes of the Breakfast dataset in Table 1 of the paper, however, we could not compare our method to RhyRNN because the source code of RhyRNN is not publicly available to adjust that model to our evaluation settings. Hence, in Table 4.10, we present comparison results of our method with the reported accuracy of this method and different versions of other models over the original 10 classes of the Breakfast dataset.

Models	t-acc	Feature	Test Split
Timeception[100]	71.3	3D-ResNet [120]	Last 8 subjects
Timeception[100]	69.3	I3D (*pre pooling*)	Last 8 subjects
Timeception[100]*	76.6	I3D (*post pooling*)	Split 1
VideoGraph[101]	69.5	I3D (*pre pooling*)	Last 8 subjects
VideoGraph[101]*	79.9	I3D (*post pooling*)	Split 1
RhyRNN[102]	44.3	ResNet101 [119]	Split 1
Our Method	81.5	I3D (*post pooling*)	Split 1
Our Method	85.2	I3D (*post pooling*)	Last 8 subjects

Table 4.10: Task classification results (t-acc) of state-of-the-art methods on the Breakfast dataset for 10 classes. (* results re-implemented using the author's source code).

For a direct comparison with RhyRNN , we show results on the first split as reported in RhyRNN.

Furthermore, Table 4.10 shows the original reported results of Timeception and VideoGraph, which are lower than our re-implemented versions in both cases. Contrary to the standard splitting rule of the Breakfast dataset, both works have used the last 0.15% of subjects in the dataset (8 subjects) to test their performance. Our result on this split significantly outperforms previous methods (Table 4.10). [100] and [101] also use the output before the last average pooling layer (*pre pooling*) in the I3D network as features, unlike us, where we use the features after the pooling layer (*post pooling*). The results in Table 4.10 suggest the superiority of the latter, because the lower dimension after pooling allows each network to be given more features as input, which increases their input temporal range.

Interestingly, the task accuracy for most models, including ours, hardly drops upon evaluation on 10 classes and our method is still superior than different versions of state-of-the-art.

4.8.4 Glossary of Terms and Symbols

As there are similar terms and many symbols used in the paper, here, we provide specific definitions of terms (Table 4.12) and symbols (Table 5.5) for readers to refer to.

Symbol	Definition
$\mathbb{A}$	The set of all attributes
a_j	Attribute j
$a_{i,j}$	Attribute j of video i
$\mathbb{A}_i$	The set of attributes in video i
$\vec{a}_i$	Muiltihot ground-truth attribute vector of video i
$\vec{a}_i^w$	TF-IDF weighted ground-truth attribute vector of video i
$A^T B$	Matrix multiplication of A transposed and B
$a \cdot b$	Scalar multiplication of a and b
β	Importance factor of $\mathscr{L}_{th}$ in the total loss
$\mathbb{C}$	The set of all tasks
c_i	Task label for video i
$\vec{c}_i$	One-hot task ground-truth vector of video i
d	Stage duration
F	Dimension of the feature encoding $\phi(\mathbf{x})$
f_i^c	Final fused classification logits
$g(\mathbf{x})$	The fully connected layer to produce encoding $\phi(\mathbf{x})$
$\mathscr{H}_x(\,)$	Heaviside step function shifted to x
h_κ	Feature summary of stage κ
K	Number of stages in the THS stream
$\mathbf{k}_\phi$	Temporal convolution kernels to produce $\phi(\mathbf{x})$
k_i	Number of selected frames of video i from the top_k operation
L	Kernel length of $\mathbf{k}_\phi$
l_1^S	Sequence of S action durations in a video
$\mathscr{L}_{sh}$	Loss function for the SHS stream
$\mathscr{L}_{th}$	Loss function for the THS stream
$\mathscr{L}_f$	Loss function of the fused streams
M_i	Number of attributes in video i
$\mathbf{M}_{tfidf}$	TF-IDF mask
$\mathscr{M}_x^a$	Mapping function from features to attributes
$\mathscr{M}_a^c$	Mapping function from attributes to tasks
N	Number of videos in the training set/batch
S	Number of segments in a video
$\mathfrak{s}[\,]$	Softmax operation
$\mathscr{S}_\kappa$	Classifier for stage κ in the THS stream
s	The parameter used in the top_k operation
T_i	Number of frames in video i
$\mathscr{T}$	Stage aggregation function in the THS stream
τ	Task variable
ϑ_κ	Task prediction logits of stage κ
ϑ_{total}	Stage-aggregated task prediction logits
$\mathbb{V}$	Set/Batch of training videos
$\mathbf{v}_i$	Video i
$\mathbf{W}_{tfidf}$	TF-IDF weights
$\mathbf{x}_i$	Input feature vector for video i
$\phi(\mathbf{x})$	Learned video feature encoding
ψ_i^a	Attribute score vector of video i in the SHS stream
ψ_i^c	Task score vector of video i in the SHS stream
Ψ_i^a	T-CAM of video i
δ_1^S	Sequence of S action labels in a video
λ	Design parameter in $\mathscr{L}_{sh}$
$\Pi(\tau)$	Set of all action sequences in the training set given task τ
$\sigma(\,)$	Sigmoid operation
$\Gamma(\,)$	Stage-wise drop out in the stage aggregation function
$\mathbb{1}(\,)$	Indicator function
$\odot$	Element-wise product operation

Table 4.11: Definitions of symbols used in the paper.

Term	Definition
Action	Lower level actions happening in the form of segment sequence in instructional videos.
Action alignment	Partitioning the video into sequence of action segments given a sequence of action labels.
Action detection	Classify and localize occurrences of, typically, a single action in the video among considerable background frames.
Action segmentation	Partitioning the video into sequence of action segments.
Attribute	Set of actions or the set of verb/object components of actions.
Fully-supervised classification	Task classification using frame-level and video-level labels.
Instructional videos	Videos with a top-level task and a sequence of fine-grained actions to carry out the underlying task.
Task	The single top-level composite activity present in the video.
Task recognition	Classifying the top-level task in long instructional videos.
Weakly-labeled videos	Videos with no frame-level annotations. In our case, only sequence of video-level action labels is available.
Weakly-supervised classification	Task classification without access to frame-level annotation. We use the term "weak" to distinguish from fully-supervised methods.

Table 4.12: Definitions of technical terms used in the paper.

5 Weakly-Supervised Online Action Segmentation in Multi-View Instructional Videos*

5.1 Abstract

This paper addresses a new problem of weakly-supervised online action segmentation in instructional videos. We present a framework to segment streaming videos online at test time using Dynamic Programming and show its advantages over greedy sliding window approach. We improve our framework by introducing the Online-Offline Discrepancy Loss (OODL) to encourage the segmentation results to have a higher temporal consistency. Furthermore, only during training, we exploit frame-wise correspondence between multiple views as supervision for training weakly-labeled instructional videos. In particular, we investigate three different multi-view inference techniques to generate more accurate frame-wise pseudo ground-truth with no additional annotation cost. We present results and ablation studies on two benchmark multi-view datasets, Breakfast and IKEA ASM. Experimental results show efficacy of the proposed methods both qualitatively and quantitatively in two domains of cooking and assembly.

*This work was done as part of my internship at Honda Research Institute, USA

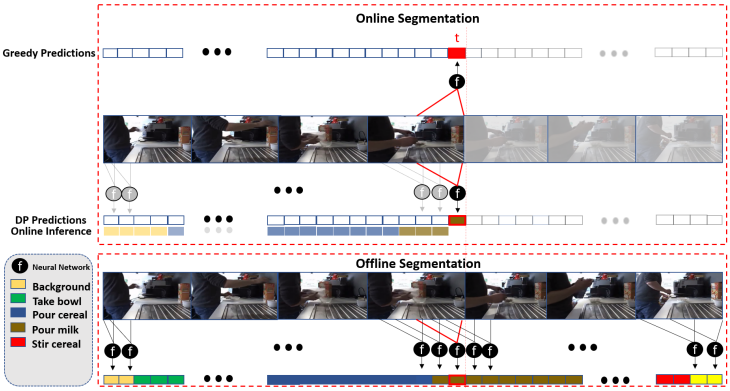


Figure 5.1: Top: online segmentation, where the frame of interest at time t is identified either greedily by the f function or through DP-based online inference based on current and past predictions. Bottom: Offline segmentation after observing the whole sequence.

5.2 Introduction

Action understanding in untrimmed instructional videos is important in many applications, where agents learn by observation of other agents performing complex tasks. Such videos are characterized by composition of a sequence of low-level atomic actions, e.g., *crack eggs* and *whisk eggs*, that form a high-level task, e.g., *making eggs*. This contextual dependency between actions as well as other attributes in instructional videos have inspired new research [88, 82, 83, 89, 121] that has advanced the field.

A fully-supervised training of these videos would require not only the labels for each action, but also their temporal assignment (start and end time) with ordering constraints. However, creating fully annotated clips with action assignments and labels on the temporal boundaries of individual actions is manually intensive and is therefore both time consuming and expensive. This limits the scale and practicality at which fully-supervised video datasets can be created. Furthermore, the subjective nature of labeling the start and end time of each action results in ambiguities and inconsistencies. In *weakly-supervised action segmentation* these limitations are addressed by using only the ordered sequence of action labels per video during training, and forgo subjective labeling of start and end time of each action.

Another important consideration in action understanding relates to requirements for processing the videos online versus offline, which is not addressed in existing weakly-supervised segmentation methods [2, 122, 105]. Online processing with low latency is an increasingly important part of interactive applications where real-time, or near real-time feedback is critical. For example, interactive applications such as human-robot in-

teraction, error correction in manufacturing assembly, and virtual rehabilitation require immediate feedback from the intelligent system as the video streams arrive.

The work presented in this paper considers the two aforementioned aspects in action segmentation: weak-supervision and online processing aimed at temporally partitioning videos into action segments. To our knowledge, our work is the first to address the problem of weakly supervised online action segmentation. Specifically, we present a framework to segment streaming instructional videos online at test time using Dynamic Programming (DP). We show the advantages of using DP as opposed to the greedy sliding window approach that are frequently used in previous online action understanding work [123, 124, 125] (Fig. 5.1).

We also introduce the Online-Offline Discrepancy Loss (OODL). Offline segmentation refers to inference after observing the video in its entirety. Offline segmentation is a non-causal procedure that is generally expected to be more accurate than its online counterpart that makes inference from partial observations. Indeed, there is a trade-off between accuracy of the recognized actions and low-latency (Sec.5.7.2.1). The OODL loss uses the offline segmentation result as a reference and penalizes its difference with online segmentation results generated at each time step in the video. Effectively, this encourages the segmentation results inferred at different observation end points in the video to have higher temporal consistency with respect to each other.

Furthermore, due to lack of frame-level annotation in weakly-labeled videos, frame-wise correspondence between multiple synchronized views of the same recording can provide helpful cues about the temporal location of each action during training. Our work is the first to use the supervision of frame-level correspondence between different views for action segmentation. We compare three ways to exploit this multi-view correspondence to generate more accurate frame-level pseudo ground-truth for weakly-labeled videos. This is in contrast to previous segmentation methods [2, 54, 122], where different views are treated independently, discarding important multi-view information. Note that we only use the multi-view correspondence at training time and our method segments each video independently at test time with no access to other views. Also, our framework utilizes no additional annotation cost, as it is trained independent of the label and number of view points.

In summary, our main contributions are as follows:

1) We are the first to address the problem of weakly-supervised online action segmentation in instructional videos, and offer a DP-based framework.

2) We introduce the Online-Offline Discrepancy Loss (OODL). The OODL loss uti-

lizes the offline segmentation result as a reference to train the online model by minimizing the difference between online and offline inference results.

3) We use frame-wise multi-view correspondence, during training only, to generate more accurate action pseudo-ground-truth in weakly-labeled videos with no additional annotation cost. Our work is the first to incorporate multi-view video understanding in action segmentation.

4) We present results and a detailed ablation study on two benchmark multi-view datasets in domains of cooking and assembly: Breakfast [60] and IKEA [126]. We show quantitatively and qualitatively how our contributions consistently improve various suggested baselines on both datasets.

5.3 Related Work

Weakly-Supervised Action Segmentation. There has been extensive research in action segmentation of instructional videos under different forms of supervision, including fully-supervised [127, 93, 128, 129, 130], unsupervised [131, 132, 96], and time-stamp supervised [133] methods. Methods most similar to ours use only the sequence of action labels as the weak supervision in training [2, 1, 55, 59, 54, 122, 106, 116]. However, all previous methods consider offline segmentation of videos, where future frames are used to make predictions at the current frame. Specifically, [55] encodes the entire video first before decoding it to frame-level action scores. The work in [2, 1, 54, 122, 105] use Dynamic Programming (DP) to infer the most likely actions and their duration given the entire video. Our method also uses a DP-based framework, but to our knowledge, we are the first to introduce a weakly-supervised method to segment a streaming video in an online manner.

Online Action Understanding. Online action understanding has been studied in various problems such as online action detection [134, 124, 125], start of action detection [135, 136] or anticipation [137, 138, 139, 140, 141]. In the context of online action detection, [134] employs knowledge distillation to transfer information from offline to online models and [142, 124, 125, 143] introduce new neural networks to classify current actions in streaming videos using a sliding window approach. Others have focused solely on detecting the start of an ongoing action immediately [135, 136] or with a short delay [144]. However, past methods did not consider instructional videos and, more importantly, required frame-level labels to train.

Most similar to our work is WOAD [123] as the only weakly-supervised online action detection framework. WOAD [123] is different to our framework in two main ways: First,

as a detection model, it is formulated to identify and localize occurrences of, typically, a single action in the input video, while we focus on instructional videos with a series of many unique actions. Second, during test time, we utilize Dynamic Programming and show in our experiments that it outperforms the greedy approach taken in [123].

Multi-View Action Understanding. Using video feeds from multiple view points has improved performance for different problems such as action recognition [145, 146, 147, 148, 149], person identification [150], anomaly detection [151], and video summarization [152, 153, 154]. Similar to our work, [146, 148, 155, 156] limit exploiting multiple views to training time only. In particular, [148, 146] focus on fully-supervised learning of trimmed videos. Meanwhile, [155] explores unsupervised video-to-video alignment, but utilize partial frame-level labels for classification. In addition, [157, 156] study domain adaptation across 3rd and 1st person views. However, unlike us, they rely on view-specific labels for training. Others [158, 147, 149] use multiple data modalities as view points. Specifically, [158] introduces a semi-supervised and view-agnostic framework for trimmed video classification, where multiple modalities are fused to generate video pseudo labels. These pseudo labels are used along with a selected number of ground-truth labels to train a video classifier. In contrast, to our knowledge, we are the first to use multi-view for temporal segmentation in untrimmed videos without frame-level supervision.

5.4 Background

This section describes definitions and background concepts used henceforth. For more clarity, the supplementary material provides a table of all symbols used.

5.4.1 Problem Definition

During training, the input to our model is a video of length T represented by frame-level features $\mathbf{x}_1^T$ and an ordered sequence of actions $\tau = (\tau_1, \tau_2, ..., \tau_M)$ known as the transcript. M is the number of actions in a given video and can vary across videos. Information about the start and end time of each action is not known.

At test time, given the set of action labels in the dataset $\mathbb{A}$, the goal is to identify the action label $a_t \in \mathbb{A}$ at frame t for all $0 < t < T+1$ based on only the past and current observations $\mathbf{x}_1^t$. The final result will be a sequence of N predicted segments identified online by their action a_n and duration l_n, where n refers to the n_{th} segment.

5.4.2 *Offline Inference*

Given the input features $\mathbf{x}_1^T$ of the entire video, a common factorized formulation [1, 2] to model the posterior probability of the sequence of actions a_1^N and their corresponding duration l_1^N is given by:

$$p_{\text{off}}(a_1^N, l_1^N | \mathbf{x}_1^T) \approx p(\mathbf{x}_1^T | a_1^N) p(l_1^N | a_1^N) p(a_1^N). \tag{5.1}$$

To infer the most likely sequence of actions $\bar{a}_1^N$ and their duration $\bar{l}_1^N$ associated with the video transcript τ, we use

$$(\bar{a}_1^N, \bar{l}_1^N) = \underset{a_1^N, l_1^N}{\text{argmax}} \left\{ p_{\text{off}}(a_1^N, l_1^N | \mathbf{x}_1^T) \right\}, \tag{5.2}$$

$$= \underset{a_1^N, l_1^N}{\text{argmax}} \{ \prod_{t=1}^{T} p(x_t | a_{n(t)}) \prod_{n=1}^{N} p(l_n | a_n) p(a_1^N) \}, \tag{5.3}$$

where $n(t)$ is the segment number at frame t. While training, $\bar{a}_1^N = \tau$ and $N = M$, since the sequence of action labels is already given in the transcript. $p(x_t|a)$ is modeled by a GRU [159] and the Bayes rule as in [1]. The GRU can be optionally replaced by any other neural network as a black box. $p(l|a)$ is a Poisson distribution modeling the duration of a given action and is parameterized by the mean length of action a. Finally, $p(a_1^N) = 1$ if the sequence of action labels a_1^N exist in the training set transcripts and 0 otherwise.

5.4.3 *Offline Segmentation Energy Score*

We revisit the definition of *energy score* $\mathscr{E}$ introduced in offline segmentation [2]. Specifically, based on the inferred segments (Eq.5.3), we define $(\bar{a}_1^N, \bar{l}_1^N)$ as the unique valid path π^+ and $(\grave{a}_1^N, \bar{l}_1^N)$ as an invalid path $\pi^- \in \mathbb{P}^-$, where $\grave{a}_n \in \mathbb{A} \setminus \{\bar{a}_n\}$ and $\mathbb{P}^-$ is the set of all invalid paths given the inferred durations $\bar{l}_1^N$. Accordingly, we define the segment-level energy score of the valid action $\bar{a}_n$ with length $\bar{l}_n$ at segment n as $e_n(\bar{a}_n, \bar{l}_n) = \prod_{t \in \eta(n)}^{\eta(n)+\bar{l}_n-1} p(\bar{a}_n | x_t)$, and the segment-level energy score of an invalid action $\grave{a}_n$ is given as $e_n(\grave{a}_n, \bar{l}_n) = \prod_{t \in \eta(n)}^{\eta(n)+\bar{l}_n-1} p(\grave{a}_n | x_t)$. Here, $\eta(n)$ is a function that maps an input segment number to the starting frame number of that segment. Note that the start of each segment occurs immediately after the end of the previous segment and $p(a|x_t)$ is the output of the GRU. Further, in order to exclusively focus on hard invalid actions, the segment energy score of hard invalid actions denoted by $e_n^-(\grave{a}_n, \bar{l}_n)$ is defined as follows:

$$e_n^-(\grave{a}_n, \bar{l}_n) = \begin{cases} e_n(\grave{a}_n, \bar{l}_n), & \text{if } e_n(\grave{a}_n, \bar{l}_n) > e_n(\bar{a}_n, \bar{l}_n) \\ 1, & \text{otherwise} \end{cases}.$$

Finally, $\mathscr{E}_{\pi^+} = \prod_1^N e_n(\bar{a}_n, \bar{l}_n)$ is the energy score of the valid path, and $\mathscr{E}_{\pi^-} = \prod_1^N e_n^-(\grave{a}_n, \bar{l}_n)$ is the energy score of the invalid path π^-. Calculation of these energy scores is done in the log space using DP as explained in [2].

5.5 Weakly-Supervised Online Segmentation

In this section, we introduce our framework for causal action inference and present how the relation between online and offline inference is exploited to derive a loss function for weakly-supervised online action segmentation.

5.5.1 Online Inference

Since online action inference is a causal process, we cannot directly use Eq. 5.3 to infer the action label at the current frame t'. A straightforward causal solution is to employ the GRU in a sliding window fashion and apply argmax$\{p(a_{t'}|x_t)\}$ as the output of the GRU with the highest probability [123]. However, as shown in Fig.5.5, this greedy approach does not consider the context and predictions of previous time steps and is therefore sub-optimal. In order to fully account for the past actions and their duration, we formulate the marginal causal (or online) probability $p_{\text{on}}(a_{t'}|\mathbf{x}_1^{t'})$ of the present action $a_{t'} = a_{n(t')}$ at segment $n' = n(t')$ over all previous actions $a_1^{n'-1}$ if $n' > 1$, and duration $l_1^{n'}$. The inferred present action $\hat{a}_{t'}$ is derived as follows:

$$\hat{a}_{t'} = \underset{a_{t'} \in \mathbb{A}}{\text{argmax}} \big\{ p_{\text{on}}(a_{t'}|\mathbf{x}_1^{t'}) \big\}, \tag{5.4}$$

$$= \underset{a_{n'} \in \mathbb{A}}{\text{argmax}} \big\{ \sum_{a_1^{n'-1}, l_1^{n'}} p_{\text{on}}(a_1^{n'}, l_1^{n'}|\mathbf{x}_1^{t'}) \big\}. \tag{5.5}$$

To improve computational efficiency, we empirically approximated Eq. 5.5 by the maximum joint probability value:

$$\hat{a}_{t'} \approx \underset{a_{n'} \in \mathbb{A}}{\text{argmax}} \big\{ \max_{a_1^{n'-1}, l_1^{n'}} p_{\text{on}}(a_1^{n'}, l_1^{n'}|\mathbf{x}_1^{t'}) \big\}. \tag{5.6}$$

Eq.5.6 involves two steps. The first is to find the most likely sequence of actions $\tilde{a}_1^{n'}$ with duration $\tilde{l}_1^{n'}$ until time t'. The second involves taking only the last segment label $\tilde{a}_{n'} = pop(\tilde{a}_1^{n'})$ to infer the label of the current frame t', where $pop()$ is a function that outputs the last element of a list. To execute the first step, online inference of the most likely sequence of past action segments $(\tilde{a}_1^{n'}, \tilde{l}_1^{n'})$ is formulated as argmax$\big\{p_{\text{on}}(a_1^{n'}, l_1^{n'}|\mathbf{x}_1^{t'})\big\}$,

where $p_{\text{on}}(a_1^{n'}, l_1^{n'} | \mathbf{x}_1^{t'})$ for $n' > 1$[1] is derived below:

$$p_{\text{on}}(a_1^{n'}, l_1^{n'} | \mathbf{x}_1^{t'}) = \Gamma(l_{n'}|a_{n'}) \prod_{t=1}^{t'} p(x_t|a_{n(t)}) \prod_{n=1}^{n'-1} p(l_n|a_n) \cdot p(a_1^{n'}). \quad (5.7)$$

$p(a_1^{n'}) = 1$ if $a_1^{n'}$ is a sub-sequence of any of the transcripts in the training set, and 0 otherwise, and $\Gamma(l|a)$ is a half Poisson function to model the duration $l_{n'}$ of the current action $a_{n'}$ at the last observed segment, given by

$$\Gamma(l|a) = \begin{cases} 1 & \text{if } l < \lambda_a \\ \frac{\lambda_a^l \exp(-\lambda_a)}{l!} & \text{otherwise} \end{cases},$$

where λ_a is the estimated mean length of action a.

Inclusion of $\Gamma()$ in the online inference of the current action is essential as it accounts for the two following cases: First, using the full Poisson distribution of Eq.5.3 to model the duration of the current observed action leads to penalizing the current actions with a short duration, $l_{n'} < \lambda_{a_{n'}}$. However, since we do not have foresight about the duration of the current segment, any conclusion about the current segment length would be premature. Second, $\Gamma()$ still allows us to penalize the current action if its duration is longer than expected since this can be concluded solely based on the observed segment of the action.

At test time, the final online segmentation result in a streaming video when the current time t' changes from 1 to any given time T is the sequence of frame-level actions $(\hat{a}_1, .., \hat{a}_T)$, where each $\hat{a}_{t'} \leftarrow \tilde{a}_{n'} = pop(\tilde{a}_1^{n'})$ is inferred by Eq.5.7 using the Viterbi algorithm.

5.5.2 Online-Offline Discrepancy Loss (OODL)

Offline action segmentation is expected to be more accurate than online segmentation because segments are inferred from information contained in the entire length of the video, including transcripts as well as prior knowledge of the video end. Thus, offline segmentation results provide a rich source of supervision for training online segmentation models. Ideally, the sequence of actions inferred online from the initial frame to any point in the video is expected to be a sub-sequence of the offline inference result as shown in Fig. 5.2. Consequently, this encourages all frame-level action sequences $\{\tilde{a}_1^t\}_{t=1}^T$ to be temporally consistent, where each sequence $\tilde{a}_1^t$ is inferred online at time t .

[1] For $n' = 1$, the Poisson factor $p(l|a)$ is excluded.

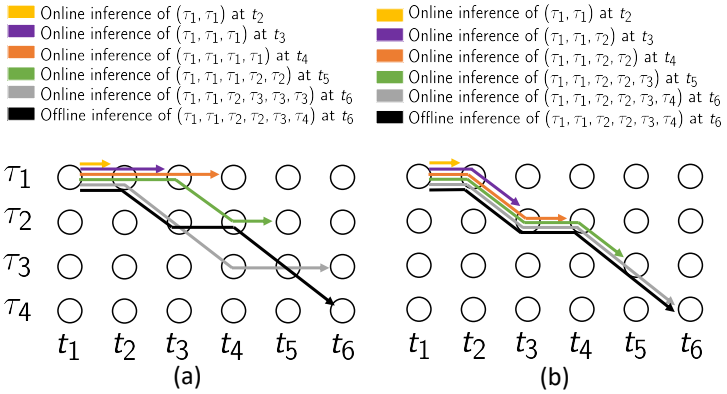


Figure 5.2: Given the video transcript τ=$(\tau_1, \tau_2, \tau_3, \tau_4)$, the OODL loss encourages the online segmentation results in (a) to become a sub-sequence of the offline result for each time step as in (b).

We present the Online-Offline Discrepancy Loss (OODL) $\mathscr{L}_{\text{OODL}}$ in Algorithm 3 to minimize the difference between online and offline segmentation scores. Specifically, we first use Eq.5.7 to infer the set of online paths $\{\tilde{a}_1^t\}_{t=1}^T$ after $n(t)$ pairs of segment-level labels $(\tilde{a}_1^{n(t)}, \tilde{l}_1^{n(t)})$ are converted into t frame-level action labels $\tilde{a}_1^t$ for each time step t. Then, we use the hinge loss function to penalize any online inference result that has a higher energy score $\mathscr{E}_{\text{on}}$ than the energy score $\mathscr{E}_{\text{off}}$ of the offline path $\overline{a}_1^t \subseteq \overline{a}_1^T$ inferred from Eq. 5.3. The OODL ultimately discourages all frame-level predictions that contribute to the discrepancy between the intermediate online inference results and the most likely sequence of actions inferred offline at the end of the video.

$\mathscr{L}_{\text{OODL}}$ is added to the baseline offline segmentation loss $\mathscr{L}_b$ [2] to form our final loss function $\mathscr{L}_f$:

$$\mathscr{L}_f = \mathscr{L}_b + \mathscr{L}_{\text{OODL}}. \tag{5.8}$$

Minimizing the offline segmentation loss $\mathscr{L}_b$ effectively corresponds to maximizing the decision margin between offline valid and hard invalid paths derived in Sec. 5.4.3.

$$\mathscr{L}_b = -\log(\mathscr{E}_{\pi^+}) + \log(\sum_{\pi^- \in \mathbb{P}^-} \mathscr{E}_{\pi^-}). \tag{5.9}$$

We iteratively utilize the offline and online segmentation pseudo labels inferred by Eq. 5.3, and 5.7, respectively, as well as the loss in Eq.5.8 to train the GRU until convergence.

Algorithm 3 Online-Offline Discrepancy Loss (OODL)

Require: Video features $\mathbf{x}_1^T$ of T frames and the offline inference result $(\overline{a}_1^N, \overline{l}_1^N)$ of N segments

Ensure: L_T as the OODL loss $\mathscr{L}_{\text{OODL}}$

for $t \leftarrow 1$ to T **do**:
  $\tilde{a}_1^{n(t)}, \tilde{l}_1^{n(t)} = \text{argmax}\left\{p_{\text{on}}(a_1^{n(t)}, l_1^{n(t)} | \mathbf{x}_1^t)\right\}$ ▷ Eq.5.7
  $\tilde{a}_1^t = (\tilde{a}_1, ..., \tilde{a}_{n(t)}) = \text{convert}(\tilde{a}_1^{n(t)}, \tilde{l}_1^{n(t)})$
  $\mathscr{E}_{\text{on}}(t) = \prod_{a_t \in \tilde{a}_1^t} p(a_t | x_t)$
$\overline{a}_1^T = (\overline{a}_1, ..., \overline{a}_{n(T)}) = \text{convert}(\overline{a}_1^N, \overline{l}_1^N)$
$L_0 = 0$
for $t \leftarrow 1$ to T **do**:
  $\mathscr{E}_{\text{off}}(t) = \prod_{a_t \in \overline{a}_1^t} p(a_t | x_t)$
  $d = \max\left(0, \log(\mathscr{E}_{\text{on}}(t)) - \log(\mathscr{E}_{\text{off}}(t))\right)$
  $L_t = L_{t-1} + \frac{d}{t}$ ▷ Averaging d over time t
return L_T

5.6 Multi-View Supervision

Due to lack of frame-level action labels at training time, it is imperative to maximize the functional capacity of the training data available. We do so by leveraging the correspondence between multiple unknown views to infer more accurate frame pseudo labels.

Concretely, consider a training set of K videos $\{v_i\}_{i=1}^K$ and their corresponding view adjacency matrix $V \in \mathbb{R}^{K \times K}$, where each element $v_{i,j}$ in V is 1 if v_i and v_j are two different views of the same recording, and 0 otherwise. During training, we pair each video v_i, as the anchor video, with an auxiliary video v_j, which is randomly sampled from the anchor view's adjacent set $\mathbb{V}_i = \{v_k | V_{i,k} = 1 \wedge k \neq i\}$. As shown in Fig. 5.3, each video pair is given as an input to a multiview-inference module to generate pseudo labels[2], which are used to train the GRU with respect to the anchor video i. In this section, we discuss three different multi-view inference techniques employed during training:

Sequence Voting (SV). Given synchronized video features ${}_i\mathbf{x}_1^T$ and ${}_j\mathbf{x}_1^T$ of any two given views, we define the result of voting as the sequence of actions $\overline{a}_1^N$ with durations $\overline{l}_1^N$ that have the highest product of sequence probability over both views:

$$(\overline{a}_1^N, \overline{l}_1^N) = \underset{a_1^N, l_1^N}{\text{argmax}} \left\{p(a_1^N, l_1^N | {}_i\mathbf{x}_1^T) p(a_1^N, l_1^N | {}_j\mathbf{x}_1^T)\right\}. \tag{5.10}$$

[2] Interchangeably named as offline valid path or offline inference result

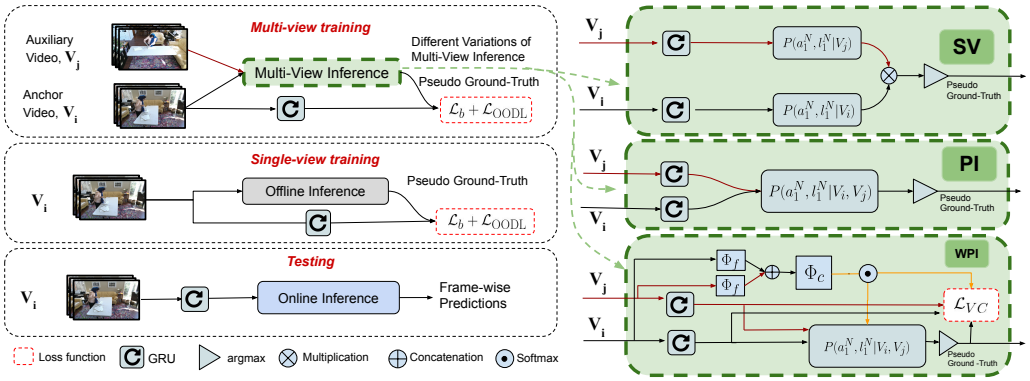


Figure 5.3: An overview of the single-view and multi-view training schemes can be seen on the left. More details of the three proposed multi-view inference techniques are depicted on the right. Notice how a single view is always used to segment the video at test time.

In this case, the inferred sequence must have high probabilities (votes) in both views, as inconsistent probabilities (votes) diminish the overall score of any segmentation.

Probabilistic Inference (PI). Instead of combining multi-view results at the video level as in the SV technique, here we fuse frame-level scores to infer the sequence that maximizes the posterior probability $p(a_1^N, l_1^N | {}_i\mathbf{x}_1^T, {}_j\mathbf{x}_1^T)$ given the two views:

$$p(a_1^N, l_1^N | {}_i\mathbf{x}_1^T, {}_j\mathbf{x}_1^T) \approx p({}_i\mathbf{x}_1^T | a_1^N) p({}_j\mathbf{x}_1^T | a_1^N) p(l_1^N | a_1^N) p(a_1^N). \tag{5.11}$$

The argmax of the above equation can be solved by integrating

$p(x_t | a_{n(t)}) = p({}_ix_t | a_{n(t)}) p({}_jx_t | a_{n(t)})$ in Eq.5.3.

Weighted Probabilistic Inference (WPI). The Probabilistic Inference model in Eq.5.11 assumes equal contribution from each view. However, a more appropriate formulation is to compare the two views and provide a higher confidence weight on the more reliable view. Hence, we introduce the class agnostic confidence weight ${}_ic_t \in [0,1]$ for the anchor view i at time t as follows:

$${}_ic_t, 1 - {}_ic_t = \text{Softmax}\Big(\Phi_c\big([\Phi_f({}_i\mathbf{x}_{t-w}^t)\ \Phi_f({}_j\mathbf{x}_{t-w}^t)]\big)\Big), \tag{5.12}$$

where $\Phi_f() : \mathbb{R}^{w \times F_1} \rightarrow \mathbb{R}^{F_2}$ is a function that embeds a temporal window of the past w frame features $\mathbf{x}_{t-w}^t$ for each view independently, and $\Phi_c() : \mathbb{R}^{F_2} \rightarrow \mathbb{R}^2$ is the compare function that takes the concatenated view embeddings $[\Phi_f({}_i\mathbf{x}_{t-w}^t)\, \Phi_f({}_j\mathbf{x}_{t-w}^t)]$ and outputs the relative confidence weight of the anchor view i with respect to the auxiliary view j at time t. F_1 and F_2 are the dimensions of each frame feature and the window embedding respectively.

Having defined the view confidence weight ${}_ic_t$, we rewrite the likelihood $p(x_t | a_{n(t)})$ as

follows and use Eq.5.3 to infer the pseudo labels $(\overline{a}_1^N, \overline{l}_1^N)$:

$$p(x_t|a_{n(t)}) = p({}_ix_t|a_{n(t)})^{{}_ic_t} p({}_jx_t|a_{n(t)})^{(1-{}_ic_t)}. \tag{5.13}$$

We incorporate a new loss $\mathscr{L}_{\text{vc}}$ in our final loss function $\mathscr{L}_f$ in order to learn the parameters of the view confidence weight ${}_ic_t(\theta_c)$, where θ_c is the set of all parameters in the compare and embedding functions of Eq. 5.12. In addition, θ_a denotes the set of parameters (i.e. GRU) required to predict frame-level action probability $p(a_t|x_t;\theta_a)$. Given the inferred pseudo labels $(\overline{a}_1^N, \overline{l}_1^N)$, we define the weighted energy score of the pseudo labels as :

$$\tilde{\mathscr{E}}_{\theta_c} = \prod_{t=1}^{T} p(\overline{a}_{n(t)}|{}_ix_t)^{{}_ic_t(\theta_c)} p(\overline{a}_{n(t)}|{}_jx_t)^{(1-{}_ic_t(\theta_c))}, \tag{5.14}$$

where we freeze θ_a and allow $\mathscr{L}_{\text{vc}}(\theta_c) = -\log(\tilde{\mathscr{E}}_{\theta_c})$ to be optimized with respect to the view confidence weight ${}_ic_t(\theta_c)$, so that the weighted energy score $\tilde{\mathscr{E}}_{\theta_c}$ of the correct path $(\overline{a}_1^N, \overline{l}_1^N)$ is maximized:

$$\mathscr{L}_f(\theta_c, \theta_a) = \mathscr{L}_b(\theta_a) + \mathscr{L}_{\text{OODL}}(\theta_a) + \mathscr{L}_{\text{vc}}(\theta_c). \tag{5.15}$$

Note that the embedding and compare functions, $\Phi_f()$ and $\Phi_c()$, are utilized only in training. Besides, $\mathscr{L}_b$ and $\mathscr{L}_{\text{OODL}}$ are computed using just the anchor video after taking the multi-view inference pseudo labels as their valid path and offline inference result respectively.

5.7 Experiments

Datasets. *The Breakfast Dataset (BD)* [60] contains approximately 1.7k cooking videos, recorded from multiple views, ranging from a few seconds to over ten minutes long. Both the angle and number of views differ across recordings. The dataset consists of 48 action labels demonstrating 10 breakfast dishes with a mean of 6.9 action segments per video. The evaluation metrics are calculated over four splits. *The IKEA ASM Dataset (IAD)* [126] has 371 recordings of assembly for four types of furniture. Each assembly is recorded from three consistent view points, providing 1113 total videos. Videos in this dataset contain a dense number of action segments (mean of $\approx$ 23) per shorter videos with a mean duration of 1.9 min. There are 32 action classes after combining the *NA* and *other*

classes as *background*. We report results over 5 splits, where each split belongs to one of the five recording environments as suggested by [126].

Metrics. Similar to previous work [55, 2, 122], we use four metrics to evaluate performance: 1) *acc* is the frame-level accuracy averaged over all the videos. 2) *acc-bg* is the frame-level accuracy without the background frames. 3) *IoU* defined as the intersection over union, which is particularly useful for imbalanced datasets such as IKEA. 4) *IoD* denotes the intersection interval over the detected interval averaged across all videos. This metric tends to overrate over-segmentation results. As in [55], both *IoD* and *IoU* are calculated over non-background segments.

Implementation. We extracted I3D features [111] for the *IAD* dataset using TV-L1 optical flow [160] on a moving window of 16 frames. Final dimensions of the features were reduced to 400 by PCA. Meanwhile, for the *BD* dataset, we obtained the Fisher vectors [78] of iDT features [77] as in [161]. We implemented the embedding function $\Phi_f()$ as temporal convolution and max pooling, while two fully connected layers were used as the compare function $\Phi_c()$. Also, we set $F_2 = 64$ and $\omega = 21$. For a fair comparison we used the same random seed in all our experiments. The model was trained for around 70k and 6k iterations on the *BD* and *IAD* datasets, respectively, following the training setup of [2].

5.7.1 Comparison to the Baseline Methods

Baselines. We implemented the Greedy baseline following the strategy of [123], where a recurrent network is trained using the pseudo labels generated by an offline segmentation method. At test time, the network takes a greedy approach and identifies actions in a sliding window fashion. Also, DP_{on} represents the proposed online inference (Eq.5.7), and DP_{off} denotes the offline segmentation baseline of Eq.5.3.

Quantitative Results. Table 5.1 compares the Greedy and DP_{on} methods in online segmentation. The Greedy baseline shows poor performance specially in the *BD* dataset largely due to poor video quality that makes isolated predictions error-prone. However, Greedy shows decisively high *IoD* values. In general, high *IoD* with low *IoU* indicates over-segmentation, which leads to overrating the result. The presence of the Γ function in our online modeling is important. Its omission leads to about 1% and 3% drop in all metrics in the *BD* and *IAD* datasets, respectively. The best result is achieved by including the OODL loss and multi-view training. This leads to a 2.6% and 3.3% IoU improvement of the DP_{on} baseline in the *BD* and *IAD* datasets, respectively. Overall, improvements on the *IAD* dataset are better represented by *IoU* since frame accuracy is dominated by the

Training		Test	Breakfast (%)				IKEA ASM (%)			
M	$\mathscr{L}_{\text{OO}}$	Inference	acc	acc-bg	IoU	IoD	acc	acc-bg	IoU	IoD
×	×	Greedy [123]	20.4	15.9	7.4	**58.1**	55.6	56.2	30.9	**53.5**
×	×	DP_{on} w/o Γ	34.3	31.4	21.4	45.1	52.8	54.6	30.0	39.3
×	×	DP_{on}	35.1	32.3	22.4	46.9	55.3	57.8	33.3	44.1
✓	✓	$\text{DP}^{*}_{\text{on}}$	**36.6**	**34.7**	**25.0**	49.1	**56.9**	**59.7**	**36.8**	48.0
✓	✓	$\text{DP}^{*}_{\text{off}}$	50.4	46.8	33.3	44.9	60.3	63.5	41.7	52.0

Table 5.1: Comparison of our multi-view supervised segmentation model with various baselines in online action segmentation. ***M*** **refers to multi-view training.* We report the WPI and PI multi-view results for the *BD* and *IAD* datasets respectively.**

action "*spin leg*", which occupies nearly 45% of the frames. We include offline results to show the performance gap between online and offline segmentation. The smaller gap between DP_{off} and the Greedy method in the *IAD* dataset highlights the challenge of weakly-supervised learning in videos with a dense number of action segments.

Qualitative Results. Incorporating multi-view supervision in training makes the GRU more robust to bad lighting, occlusion and scene variations as demonstrated in Fig. 5.4. Particularly, the top figure shows results of coffee table assembly by two people in the *IAD* dataset. This is a challenging case since nearly all tasks in both datasets are completed by one person. Hence, the baseline DP_{on} has missed the third instance of "*spinning the leg*", which is correctly detected by our final segmentation model trained using $\mathscr{L}_{\text{OO}}$ and multi-view inference. The bottom figure compares different segmentation methods under dark lighting and occlusion in a sample cooking video of the *BD* dataset. Notice the over-segmented results of the Greedy baseline in both cases. More examples included in the supplementary material.

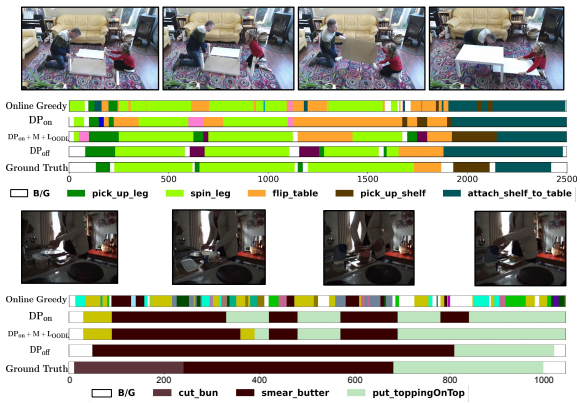


Figure 5.4: Segmentation results of various methods on the IKEA (top) and Breakfast (bottom) datasets. Legend is shown only for the ground-truth classes.

	Breakfast (%)				IKEA ASM (%)			
Training Approach	acc	acc-bg	IoU	IoD	acc	acc-bg	IoU	IoD
DP_{on}	35.1	32.3	22.4	46.9	55.3	57.8	33.3	44.1
DP_{on} + $\mathscr{L}_{OODL}$	**35.5**	**32.5**	**23.4**	**48.0**	**55.3**	**57.9**	**34.3**	**45.5**

Table 5.2: Impact of the OODL loss on weakly-supervised online segmentation results for the *BD* and *IAD* datasets.

5.7.2 Analysis and Ablation Study

All experiments in this section are reported as an average over all splits unless stated otherwise. Run-time and complexity, as limitations of the proposed algorithm, are discussed in the supplementary material.

5.7.2.1 Online-Offline Discrepancy Analysis

Impact of the OODL Loss. Addition of OODL loss leads to consistent improvement in both datasets as shown in Table 5.2. This improvement is manifested more vividly in *IoU* and *IoD* because *IoU*, in particular, is most appropriate in evaluating alignment quality between predicted and ground-truth segments.

Fig. 5.5 further demonstrates the role of the OODL loss in decreasing the online-offline segmentation discrepancy in the *BD* dataset. It shows the non-background frame accuracy of multiple segmentation approaches at five different observation end points in the video. Upon comparison of DP_{on} and DP_{on} + $\mathscr{L}_{OODL}$, it can be seen that the loss has improved mostly the early predictions in the video, where it is hardest to identify actions. This is mainly due to lack of past context in early stages of a task. With the passage of time, more information regarding the past actions becomes available. Consequently, this leads to more accurate online predictions of the current frame. On average, after observing the first 20% of the video, the performance of the DP_{on} is more similar to the Greedy baseline than the Offline model. However, the DP_{on} approach starts resembling the Offline model more just after the 60% point. In comparison, such a behavior is highlighted much less by the Greedy approach due to its limitation in capturing long-range past context.

Evaluation of Semi-Online Segmentation. Online segmentation offers practical advantages over offline inference in interactive applications that require immediate feedback. However, this comes with a 10% and 13% compromise over *acc-bg* and *IoU*, respectively, as indicated in Fig. 5.6. In some applications, certain degree of latency can be tolerated. In order to evaluate the trade-off between latency and accuracy, we implemented a semi-online variation of our framework, where predictions are made after a fixed

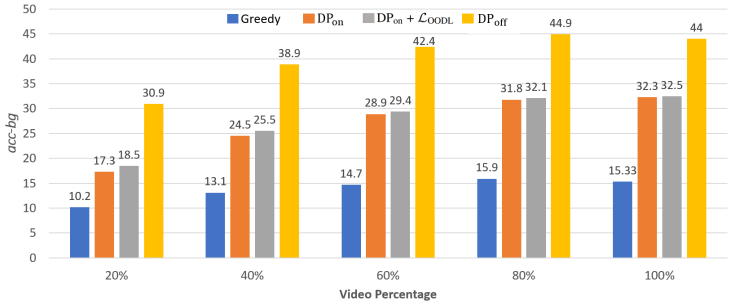


Figure 5.5: Average segmentation results (*acc-bg*) at five observation end points during the course of the video on the *BD* dataset.

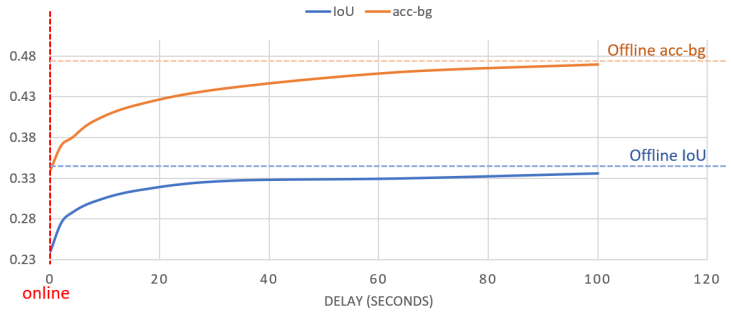


Figure 5.6: Accuracy vs. delay on split 2 of the *BD* dataset.

time delay. Fig. 5.6 shows that accuracy improves with larger latency and converges to the offline result. Importantly, we can achieve approximately 90% of the offline performance with 10 seconds of delay on the *BD* dataset.

5.7.2.2 Multi-View Supervision

Weakly-Supervised Online Segmentation. We evaluate the online segmentation performance of different multi-view inference techniques in Table 5.3. Regardless of the approach, using multi-view correspondence to generate pseudo ground-truth improves

	Breakfast (%)				IKEA ASM (%)			
Training Approach	acc	acc-bg	IoU	IoD	acc	acc-bg	IoU	IoD
Single-View	35.5	32.5	23.4	48.0	55.3	57.9	34.3	45.5
SV	36.4	34.7	24.8	48.6	55.7	58.3	34.6	45.9
PI	36.2	34.2	24.4	48.1	**56.9**	**59.7**	**36.8**	**48.0**
WPI	**36.6**	**34.7**	**25.0**	**49.1**	56.4	59.0	35.9	47.2
Fully-Supervised	41.6	41.2	30.4	52.9	63.5	67.36	44.5	56.9

Table 5.3: Comparison of online segmentation results under different pseudo ground-truth generation techniques (all with $\mathscr{L}_{\text{OODL}}$).

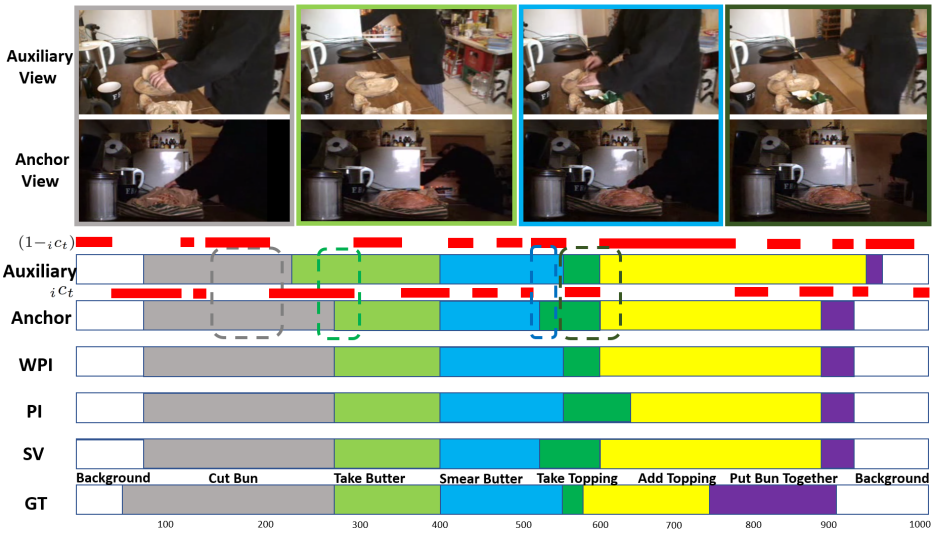


Figure 5.7: Pseudo ground-truth generated by 3 multi-view inference techniques during training. The red bar above each single-view inference result (anchor/auxiliary) indicates their learned view confidence weight c_t at each frame. Each pair of frames corresponds to the time enclosed by its color-coded dashed box.

	Breakfast (%)				IKEA ASM (%)			
Model	acc	acc-bg	IoU	IoD	acc	acc-bg	IoU	IoD
CDFL [2]	49.2	44.2	31.0	43.7	59.9	62.0	39.5	50.4
Multi-View CDFL	**50.4**	**46.8**	**33.3**	**44.9**	**60.3**	**63.5**	**41.7**	**52.0**

Table 5.4: Offline segmentation results with and without multi-view supervision (same multi-view approaches as in Table5.1). [2] results obtained after running the authors' code on our machine.

performance over the single-view method in all metrics and datasets. We also provide the *Fully-Supervised* baseline as the upper bound, where the pseudo ground-truth is 100% accurate.

In the *BD* dataset, different views provide more complementary information as compared to the *IAD* dataset. This is attributed to many instances of challenging lighting or occlusion conditions in the *BD* dataset. Fig. 5.7 underscores this fact by showing the pseudo ground-truth generated during training. Specifically, the bread in the *anchor* view occludes the two actions of "cutting bread" and "smearing butter". The resulting view confidence weight (red bar) of the *auxiliary* view becomes high for these frames. This allows the model to exploit the more visible view in the *auxiliary* video during these actions, while the *anchor* view is recognized as more reliable when the subject "takes an object". Notice how the view confidence weights dictate the selection of single-view re-

sults to form the multi-view WPI outcome. Meanwhile, the three views of the *IAD* dataset remain fairly similar in terms of lighting and no considerable occlusion occurs in most videos. Hence, weighing views equally in the PI approach leads to the best results in the *IAD* dataset.

Weakly-Supervised Offline Segmentation. Table 5.4 shows how the advantage of multi-view training is further generalized to weakly-supervised "offline" segmentation. We selected the open source state-of-the-art offline segmentation method [2] as our baseline. For this experiment, we trained [2] twice, with and without multi-view supervision, under the same parameters and random seed.

5.8 Conclusion

We introduced a framework to address a new problem of weakly supervised online action segmentation in multi-view instructional videos. The proposed solutions are formulated with the insight that offline and multi-view results provide a rich source of supervision during training which in-turn improves performance of single view online segmentation models at test time. Extensive experiments on two benchmark datasets demonstrate efficacy of our algorithms.

5.9 Supplementary Material

In this supplementary material we first provide the definitions of all the terms used in the paper, explain complexity as a limitation, and then show and discuss more qualitative segmentation results of different methods.

5.9.1 Glossary of Symbols

We provide specific definitions of symbols in Table 5.5 for readers to refer to.

5.9.2 Limitation

Here we discuss the practical run-time and computational complexity of our method, both in test and training, and compare it with CDFL [2] as an offline baseline. The lower frame rate compared to a greedy approach and the lengthy training-time are notable limitations of the proposed online segmentation method. We hope further work can mitigate this limitation.

5.9.2.1 Runtime Frame Rate Analysis

Both our method and the greedy approach[123] compute optical flow (OF) and use the I3D network to extract features. We extracted features on 320×240 frames of the *BD* dataset recorded at 15 fps. On a single GeForce GTX 1080, the OF and I3D network process videos at 90 and 20 fps, respectively. Practically, if online inference is done every 15 frames, then our method segments videos at 10+ fps. While this is less than the 100 fps of the greedy alg., it leads to considerably more accurate results

5.9.2.2 Computation Complexity of Online vs. Offline

The complexity of the proposed online inference to fully segment a video of length T and maximum transcript length of N is the same as the offline inference of CDFL ($O(T^2N)$). In other words, online inference at each time step takes $O(TN)$. This is due to DP as the inference at time t depends on the optimal results of previous time steps which have already been obtained as part of DP.

With this in mind, the training complexity of our method over K classes is the sum of complexities for the offline inference ($O(T^2N)$), baseline offline segmentation loss $\mathscr{L}_b$

($O(\Delta^2 NK)$) and $\mathscr{L}_{OODL}$. $\Delta \ll T$ is a small window size of 10 [2]. A naive implementation of OODL has complexity of $O(T^2N)$. However, if the online and offline inferences are done together outside Alg.1 and $\mathscr{E}(t)$ is summed over segments rather than frames, the OODL complexity becomes $O(TN)$. Hence, regardless of the implementation choice, our overall training has the same complexity as CDFL ($O(T^2N + \Delta^2 NK)$). Our time complexity during test time with M training transcripts is $O(T^2NM)$ which is also the same as that of CDFL. In order to quantitatively support our calculations, we tested both methods on the 4th split of the *BD* dataset. Our method and CDFL took 26 and 21 hrs to train, respectively. Meanwhile at test time, ours and CDFL took 2.7 and 2.4 hrs to run, respectively.

5.9.3 Qualitative Results

In Figure 5.8, we present two segmentation examples on the IKEA [126] (top) and Breakfast [60] (bottom) datasets. We demonstrate how training using multiple view points has let to more robust segmentation results against full occlusion (top) and extremely bad lighting (bottom). Specifically, the top figure depicts a task where the subject assembles a "side table". This assembly consists of four instances of "spinning leg", where the last instance is fully occluded by the subject's body. The baseline method DP_{on} , that is trained on a single viewpoint, misses most of the action, while training on multi-view correspondence and the OODL loss has enabled our final model ($\text{DP}_{\text{on}} + M + \mathscr{L}_{\text{OODL}}$) to capture nearly the full segment.

In the second example, the dark lighting makes it even hard for a human observer to recognize the ongoing action. Our final method is able to identify the action of "adding tea bag", where both the offline method DP_{off} and online baselines DP_{on} fail. This is an interesting case, where our model is able to outperform even the offline method. One reason is the flexibility of the proposed online segmentation model in switching between different transcripts in a series of online inferences across different time steps. This allows the predicted sequence of actions to potentially come from a transcript not observed at training time. In contrast, in offline segmentation the sequence of inferred action labels is limited only to the training transcripts.

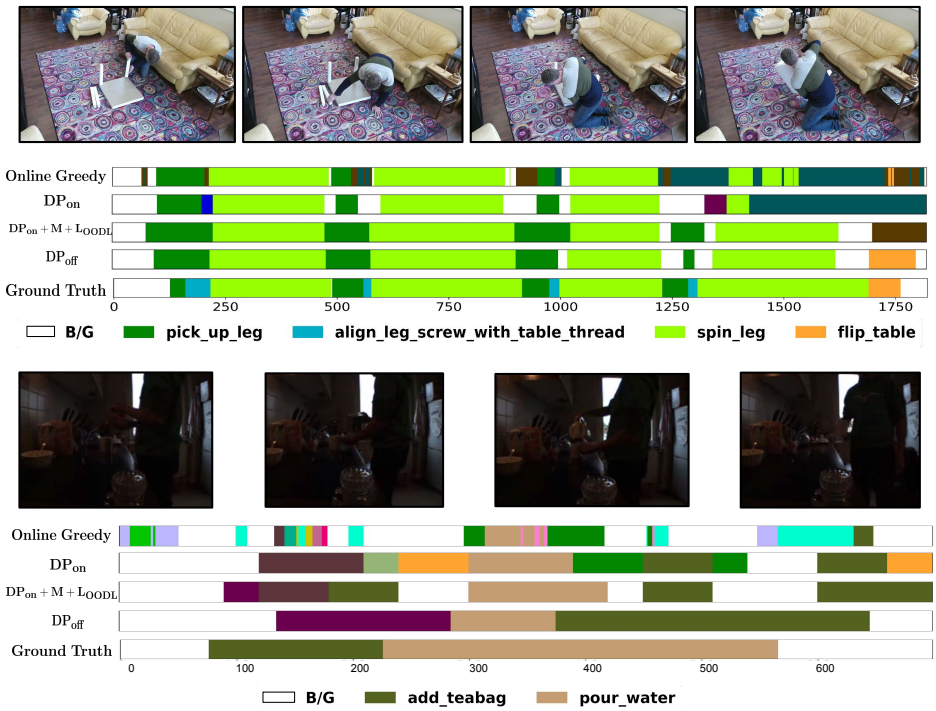


Figure 5.8: This figure shows segmentation results of various methods on the IKEA (top) and Breakfast (bottom) datasets. Subjects in the top and bottom figures assemble a side table and prepare tea respectively. Legend is shown only for the ground-truth classes.

Symbol	Definition
$\mathbb{A}$	The set of all actions in the dataset
a_n	Action variable at segment n
a_t	Action variable at frame t
$\hat{a}_t$	Predicted action at time t in an online way
$(\bar{a}_1^N, \bar{l}_1^N)$	Offline inference result
$(\tilde{a}_1^{n(t)}, \tilde{l}_1^{n(t)})$	Online inference result until time t
${}_ic_t$	View confidence weight of video i (anchor) at time t
e_n	Energy score of segment n
$\mathscr{E}_{\pi^+}$	Energy score of the valid path π^+
$\mathscr{E}_{\pi^-}$	Energy score of the invalid path π^-
$\mathscr{E}_{\text{off}}(t)$	Energy score of the offline or valid path until time t
$\mathscr{E}_{\text{on}}(t)$	Energy score of the online path until time t
$\tilde{\mathscr{E}}$	Weighted energy score of a path
F_1	Input feature dimension
F_2	Embedding dimension
K	Total number of videos in the data set
l_n	Duration variable of action a_n
$\mathscr{L}_f$	Final loss
$\mathscr{L}_b$	Baseline offline segmentation loss
$\mathscr{L}_{vc}$	View confidence loss to train WPI
M	Number of action labels in the transcript
N	Number of predicted segments in the video
$p_{\text{on}}()$	Causal probability
$p_{\text{off}}()$	Non-causal probability
$\mathbb{P}^-$	The set of all invalid paths
T	Total number of frames in the video
τ	Video transcript
v_i	Video i
$\mathbb{V}_i$	View adjacency set of video i
V	$K \times K$ view adjacency matrix
ω	Feature window size for Φ_f
$\mathbf{x}_1^T$	Sequence of T frame features
${}_i\mathbf{x}_1^T$	Features of video i
$\eta(n)$	Mapping function from segment to frame number
$\Gamma()$	Half Poisson function
λ_a	Estimated mean length of action a
Φ_c	Compare function
Φ_f	Feature embedding function
θ_c	Parameters of Φ_f and Φ_c
θ_a	Parameters of the action classifier, i.e. GRU
π^+	Valid path or offline segmentation action sequence
π^-	Invalid path

Table 5.5: Definitions of symbols used in the paper.

6 Conclusion

In this thesis, we studied long-range video understanding in two domains of Drowsiness Detection and Weakly-Supervised Action Segmentation. In the first part (Drowsiness Detection) we presented a new and publicly available real-life drowsiness dataset (RLDD) with almost 30 hours of video. We have also proposed an end-to-end baseline method using the temporal relationship between blinks for multistage drowsiness detection. The proposed method has low computational and storage demands. The results demonstrated that our method outperforms human judgment in two designed metrics on the RLDD dataset.

In the second part of this thesis, we addressed weakly-supervised action segmentation in instructional videos from multiples aspects. Concretely, we proposed a duration model to predict the remaining duration of an ongoing action to iteratively align a given sequence of action in an input video. Furthermore, we proposed a hierarchical approach to segmentation where the top-level task is predicted to constrain the segmentation results. We showed both the efficiency and the efficacy of this approach. Finally, we introduced a framework to address a new problem of weakly supervised online action segmentation in multi-view instructional videos. The proposed solutions are formulated with the insight that offline and multi-view results provide a rich source of supervision during training which in turn improves performance of single view online segmentation models at test time.

References

[1] A. Richard, H. Kuehne, A. Iqbal, and J. Gall, "Neuralnetwork-viterbi: A framework for weakly supervised video learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7386–7395.

[2] J. Li, P. Lei, and S. Todorovic, "Weakly supervised energy-based learning for action segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6243–6251.

[3] "'Facts and Stats'," May 2018, [Online]. Available: http://drowsydriving.org/about/facts-and-stats/. [Accessed: 20- May- 2018].

[4] C. D. F. E. C. J. Wheaton AG, Shults RA, "Drowsy driving and risk behaviors—10 states and puerto rico, 2011-2012." *MMWR Morb Mortal Wkly Rep.*, vol. 63, pp. 557–562, 2014.

[5] P.-C. L. C. J. R. D. Wheaton AG, Chapman DP, "Drowsy driving – 19 states and the district of columbia, 2009-2010." *MMWR Morb Mortal Wkly Rep.*, vol. 63, p. 1033, 2013.

[6] E. A. Schmidt, W. E. Kincses, M. Scharuf, S. Haufe, R. Schubert, and G. Curio, "Assessing drivers' vigilance state during monotonous driving," 2007.

[7] "'Sleepy and unsafe'," May 2014, [Online]. Available: https://www.safetyandhealthmagazine.com/articles/10412-sleepy-and-unsafe-worker-fatigue. [Accessed: 8- Sep- 2018].

[8] K. Sadeghniiat-Haghighi and Z. Yazdi, "Fatigue management in the workplace," *Industrial psychiatry journal*, vol. 24, no. 1, p. 12, 2015.

[9] E. Tadesse, W. Sheng, and M. Liu, "Driver drowsiness detection through hmm based dynamic modeling," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4003–4008.

[10] B. Reddy, Y.-H. Kim, S. Yun, C. Seo, and J. Jang, "Real-time driver drowsiness detection for embedded system using model compression of deep neural networks," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017, pp. 438–445.

[11] M. Ngxande, J.-R. Tapamo, and M. Burke, "Driver drowsiness detection using behavioral measures and machine learning techniques: A review of state-of-art techniques," in *Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech), 2017*. IEEE, 2017, pp. 156–161.

[12] "'Attention Assist'," 2018, [Online]. Available: https://www.mbusa.com/mercedes/technology/videos/detail/title-safety/videoId-710835ab8d127410VgnVCM100000ccec1e35RCRD/. [Accessed: 20- May- 2018].

[13] Q. Massoz, T. Langohr, C. François, and J. G. Verly, "The ulg multimodality drowsiness database (called drozy) and examples of use," in *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, 2016, pp. 1–7.

[14] U. Svensson, "Blink behaviour based drowsiness detection," Tech. Rep., 2004.

[15] "'Ford Brazil tests drowsiness-detecting cap'," Nov. 2017, [Online]. Available: http://www.transportengineer.org.uk/transport-engineer-news/ford-brazil-tests-drowsiness-detecting-cap/164875/. [Accessed: 20- May- 2018].

[16] M. Johns *et al.*, "The amplitude-velocity ratio of blinks: a new method for monitoring drowsiness," *Sleep*, vol. 26, no. SUPPL., 2003.

[17] S. Park, F. Pan, S. Kang, and C. D. Yoo, "Driver drowsiness detection system based on feature representation learning using various deep networks," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 154–164.

[18] P. Smith, M. Shah, and N. da Vitoria Lobo, "Monitoring head/eye motion for driver alertness with one camera," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 4. IEEE, 2000, pp. 636–642.

[19] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," *arXiv preprint arXiv:1609.01704*, 2016.

[20] F. Friedrichs and B. Yang, "Camera-based drowsiness reference for driver state classification under real driving conditions," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 101–106.

[21] C.-H. Weng, Y.-H. Lai, and S.-H. Lai, "Driver drowsiness detection via a hierarchical temporal deep belief network," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 117–133.

[22] L. K. McIntire, R. A. McKinley, C. Goodyear, and J. P. McIntire, "Detection of vigilance performance using eye blinks," *Applied ergonomics*, vol. 45, no. 2, pp. 354–362, 2014.

[23] M. Suzuki, N. Yamamoto, O. Yamamoto, T. Nakano, and S. Yamamoto, "Measurement of driver's consciousness by image processing-a method for presuming driver's drowsiness by eye-blinks coping with individual differences," in *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, vol. 4. IEEE, 2006, pp. 2891–2896.

[24] H. Yin, Y. Su, Y. Liu, and D. Zhao, "A driver fatigue detection method based on multi-sensor signals," in *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, 2016, pp. 1–7.

[25] J. Jo, S. J. Lee, K. R. Park, I.-J. Kim, and J. Kim, "Detecting driver drowsiness using feature-level fusion and user-specific classification," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1139–1152, 2014.

[26] W.-J. Yan, Q. Wu, Y.-J. Liu, S.-J. Wang, and X. Fu, "Casme database: a dataset of spontaneous micro-expressions collected from neutralized faces," in *Automatic face and gesture recognition (fg), 2013 10th ieee international conference and workshops on*. IEEE, 2013, pp. 1–7.

[27] W.-J. Yan, X. Li, S.-J. Wang, G. Zhao, Y.-J. Liu, Y.-H. Chen, and X. Fu, "Casme ii: An improved spontaneous micro-expression database and the baseline evaluation," *PloS one*, vol. 9, no. 1, p. e86041, 2014.

[28] X. Li, T. Pfister, X. Huang, G. Zhao, and M. Pietikäinen, "A spontaneous micro-expression database: Inducement, collection and baseline," in *Automatic face and gesture recognition (fg), 2013 10th ieee international conference and workshops on*. IEEE, 2013, pp. 1–6.

[29] "'Gaurdian'," 2018, [Online]. Available: http://www.seeingmachines.com/guardian/. [Accessed: 20- May- 2018].

[30] T. Åkerstedt and M. Gillberg, "Subjective and objective sleepiness in the active individual," *International Journal of Neuroscience*, vol. 52, no. 1-2, pp. 29–37, 1990.

[31] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[32] T. Soukupova and J. Cech, "Real-time eye blink detection using facial landmarks," in *21st Computer Vision Winter Workshop (CVWW'2016)*, 2016, pp. 1–8.

[33] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1867–1874.

[34] S. ROSTAMINIA, A. MAYBERRY, D. GANESAN, B. MARLIN, and J. GUMMESON, "ilid: Low-power sensing of fatigue and drowsiness measures on a computational eyeglass," 2017.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] M. Singh and G. Kaur, "Drowsy detection on eye blink duration using algorithm," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 4, pp. 363–365, 2012.

[38] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[39] C. R. De Souza, A. Gaidon, E. Vig, and A. M. López, "Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 697–716.

[40] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Temporal residual networks for dynamic scene recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4728–4737.

[41] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "Actionvlad: Learning spatio-temporal aggregation for action classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 971–980.

[42] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[43] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*. Springer, 2016, pp. 20–36.

[44] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 780–787.

[45] H. Kuehne, J. Gall, and T. Serre, "An end-to-end generative framework for video segmentation and recognition," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–8.

[46] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[47] D. Oneata, J. Verbeek, and C. Schmid, "The lear submission at thumos 2014," 2014.

[48] A. Richard and J. Gall, "Temporal action detection using a statistical language model," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3131–3140.

[49] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 1194–1201.

[50] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta, "Asynchronous temporal fields for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 585–594.

[51] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1961–1970.

[52] N. N. Vo and A. F. Bobick, "From stochastic grammar to bayes network: Probabilistic parsing of complex activity," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2641–2648.

[53] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic, "Weakly supervised action labeling in videos under ordering constraints," in *European Conference on Computer Vision*. Springer, 2014, pp. 628–643.

[54] C.-Y. Chang, D.-A. Huang, Y. Sui, L. Fei-Fei, and J. C. Niebles, "D3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3546–3555.

[55] L. Ding and C. Xu, "Weakly-supervised action segmentation with iterative soft boundary assignment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6508–6516.

[56] D.-A. Huang, L. Fei-Fei, and J. C. Niebles, "Connectionist temporal modeling for weakly supervised action labeling," in *European Conference on Computer Vision*. Springer, 2016, pp. 137–153.

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

[58] ——, "A hybrid rnn-hmm approach for weakly supervised temporal action segmentation," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[59] A. Richard, H. Kuehne, and J. Gall, "Weakly supervised action learning with rnn based fine-to-coarse modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 754–763.

[60] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 780–787.

[61] X. Duan, W. Huang, C. Gan, J. Wang, W. Zhu, and J. Huang, "Weakly supervised dense event captioning in videos," in *Advances in Neural Information Processing Systems*, 2018, pp. 3059–3069.

[62] Z. Shen, J. Li, Z. Su, M. Li, Y. Chen, Y.-G. Jiang, and X. Xue, "Weakly supervised dense video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1916–1924.

[63] P. Nguyen, T. Liu, G. Prasad, and B. Han, "Weakly supervised action localization by sparse temporal pooling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6752–6761.

[64] K. K. Singh and Y. J. Lee, "Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 3544–3553.

[65] L. Wang, Y. Xiong, D. Lin, and L. Van Gool, "Untrimmednets for weakly supervised action recognition and detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 4325–4334.

[66] J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien, "Unsupervised learning from narrated instruction videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4575–4583.

[67] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," 2008.

[68] J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy, "What's cookin'? interpreting cooking videos using text, speech and vision," *arXiv preprint arXiv:1503.01558*, 2015.

[69] O. Sener, A. R. Zamir, S. Savarese, and A. Saxena, "Unsupervised semantic parsing of video collections," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4480–4488.

[70] D. Zhukov, J.-B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic, "Cross-task weakly supervised learning from instructional videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3537–3545.

[71] A. Richard, H. Kuehne, and J. Gall, "Action sets: Weakly supervised action segmentation without ordering constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5987–5996.

[72] T. Mahmud, M. Hasan, and A. K. Roy-Chowdhury, "Joint prediction of activity labels and starting times in untrimmed videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5773–5782.

[73] Y. Abu Farha, A. Richard, and J. Gall, "When will you do what?-anticipating temporal occurrences of activities," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5343–5352.

[74] Q. Ke, M. Fritz, and B. Schiele, "Time-conditioned action anticipation in one shot," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9925–9934.

[75] P. Bojanowski, R. Lajugie, E. Grave, F. Bach, I. Laptev, J. Ponce, and C. Schmid, "Weakly-supervised alignment of video with text," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4462–4470.

[76] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta, "Asynchronous temporal fields for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 585–594.

[77] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 3551–3558.

[78] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–8.

[79] "'Robotic kitchen assistant on a rail'," May 2020, [Online]. Available: https://www.roboticsresear.ch/articles/19606/robotic-kitchen-assistant-on-a-rail. [Accessed: 15- November- 2020].

[80] H. Doughty, I. Laptev, W. Mayol-Cuevas, and D. Damen, "Action modifiers: Learning from adverbs in instructional videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 868–878.

[81] E. Elhamifar and D. Huynh, "Self-supervised multi-task procedure learning from instructional videos," *European Conference on Computer Vision*, 2020.

[82] M. Rohrbach, A. Rohrbach, M. Regneri, S. Amin, M. Andriluka, M. Pinkal, and B. Schiele, "Recognizing fine-grained and composite activities using hand-centric features and script data," *International Journal of Computer Vision*, vol. 119, no. 3, pp. 346–373, 2016.

[83] Y. Tang, D. Ding, Y. Rao, Y. Zheng, D. Zhang, L. Zhao, J. Lu, and J. Zhou, "Coin: A large-scale dataset for comprehensive instructional video analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1207–1216.

[84] X. Chang, F. Tung, and G. Mori, "Learning discriminative prototypes with dynamic time warping," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8395–8404.

[85] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman, "End-to-end learning of visual representations from uncurated instructional videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9879–9889.

[86] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic, "Howto100m: Learning a text-video embedding by watching hundred million narrated video clips," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 2630–2640.

[87] H. Doughty, W. Mayol-Cuevas, and D. Damen, "The pros and cons: Rank-aware temporal attention for skill determination in long videos," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7862–7871.

[88] P. Parmar and B. T. Morris, "What and how well you performed? a multitask learning approach to action quality assessment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 304–313.

[89] C.-Y. Chang, D.-A. Huang, D. Xu, E. Adeli, L. Fei-Fei, and J. C. Niebles, "Procedure planning in instructional videos," *arXiv preprint arXiv:1907.01172*, 2019.

[90] A. Kukleva, H. Kuehne, F. Sener, and J. Gall, "Unsupervised learning of action classes with continuous temporal embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 066–12 074.

[91] J. Li and S. Todorovic, "Set-constrained viterbi for set-supervised action segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 820–10 829.

[92] L. Zhou, C. Xu, and J. J. Corso, "Towards automatic learning of procedures from web instructional videos," *arXiv preprint arXiv:1703.09788*, 2017.

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

[94] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[95] E. Elhamifar and Z. Naing, "Unsupervised procedure learning via joint dynamic summarization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6341–6350.

[96] F. Sener and A. Yao, "Unsupervised learning and segmentation of complex activities from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8368–8376.

[97] C. Feichtenhofer, "X3d: Expanding architectures for efficient video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 203–213.

[98] B. Jiang, M. Wang, W. Gan, W. Wu, and J. Yan, "Stm: Spatiotemporal and motion encoding for action recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2000–2009.

[99] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.

[100] N. Hussein, E. Gavves, and A. W. Smeulders, "Timeception for complex action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 254–263.

[101] ——, "Videograph: Recognizing minutes-long human activities in videos," *arXiv preprint arXiv:1905.05143*, 2019.

[102] T. Yu, Y. Li, and B. Li, "Rhyrnn: Rhythmic rnn for recognizing events in long and complex videos," in *European Conference on Computer Vision*. Springer, 2020, pp. 127–144.

[103] J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien, "Unsupervised learning from narrated instruction videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4575–4583.

[104] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," 2008.

[105] Y. Souri, M. Fayyaz, L. Minciullo, G. Francesca, and J. Gall, "Fast weakly supervised action segmentation using mutual consistency," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[106] Y. Souri, Y. A. Farha, F. Despinoy, G. Francesca, and J. Gall, "Fifa: Fast inference approximation for action segmentation," *arXiv preprint arXiv:2108.03894*, 2021.

[107] S. Narayan, H. Cholakkal, F. S. Khan, and L. Shao, "3c-net: Category count and center loss for weakly-supervised action localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8679–8687.

[108] P. X. Nguyen, D. Ramanan, and C. C. Fowlkes, "Weakly-supervised action localization with background modeling," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5502–5511.

[109] S. Paul, S. Roy, and A. K. Roy-Chowdhury, "W-talc: Weakly-supervised temporal activity localization and classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 563–579.

[110] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*. Springer, 2016, pp. 20–36.

[111] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[112] P. Nguyen, T. Liu, G. Prasad, and B. Han, "Weakly supervised action localization by sparse temporal pooling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6752–6761.

[113] M. Rashid, H. Kjellstrom, and Y. J. Lee, "Action graphs: Weakly-supervised action localization with graph convolution networks," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 615–624.

[114] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[115] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of research and development*, vol. 2, no. 2, pp. 159–165, 1958.

[116] R. Ghoddoosian, S. Sayed, and V. Athitsos, "Action duration prediction for segment-level alignment of weakly-labeled videos," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2053–2062.

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

[118] Y. Souri, A. Richard, L. Minciullo, and J. Gall, "On evaluating weakly supervised action segmentation methods," *arXiv preprint arXiv:2005.09743*, 2020.

[119] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[120] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6546–6555.

[121] R. Ghoddoosian, S. Sayed, and V. Athitsos, "Hierarchical modeling for task recognition and action segmentation in weakly-labeled instructional videos," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2022, pp. 1922–1932.

[122] X. Chang, F. Tung, and G. Mori, "Learning discriminative prototypes with dynamic time warping," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8395–8404.

[123] M. Gao, Y. Zhou, R. Xu, R. Socher, and C. Xiong, "Woad: Weakly supervised online action detection in untrimmed videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1915–1923.

[124] H. Eun, J. Moon, J. Park, C. Jung, and C. Kim, "Learning to discriminate information for online action detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 809–818.

[125] M. Xu, M. Gao, Y.-T. Chen, L. S. Davis, and D. J. Crandall, "Temporal recurrent networks for online action detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5532–5541.

[126] Y. Ben-Shabat, X. Yu, F. Saleh, D. Campbell, C. Rodriguez-Opazo, H. Li, and S. Gould, "The ikea asm dataset: Understanding people assembling furniture through actions, objects and pose," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 847–859.

[127] S.-H. Gao, Q. Han, Z.-Y. Li, P. Peng, L. Wang, and M.-M. Cheng, "Global2local: Efficient structure search for video action segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 805–16 814.

[128] Y. Ishikawa, S. Kasai, Y. Aoki, and H. Kataoka, "Alleviating over-segmentation errors by detecting action boundaries," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2322–2331.

[129] Z. Wang, Z. Gao, L. Wang, Z. Li, and G. Wu, "Boundary-aware cascade networks for temporal action segmentation," in *European Conference on Computer Vision*. Springer, 2020, pp. 34–51.

[130] F. Sener, D. Singhania, and A. Yao, "Temporal aggregate representations for long-range video understanding," in *European Conference on Computer Vision*. Springer, 2020, pp. 154–171.

[131] S. Sarfraz, N. Murray, V. Sharma, A. Diba, L. Van Gool, and R. Stiefelhagen, "Temporally-weighted hierarchical clustering for unsupervised action segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 225–11 234.

[132] S. Kumar, S. Haresh, A. Ahmed, A. Konin, M. Z. Zia, and Q.-H. Tran, "Unsupervised activity segmentation by joint representation learning and online clustering," *arXiv preprint arXiv:2105.13353*, 2021.

[133] Z. Li, Y. Abu Farha, and J. Gall, "Temporal action segmentation from timestamp supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8365–8374.

[134] P. Zhao, L. Xie, Y. Zhang, Y. Wang, and Q. Tian, "Privileged knowledge distillation for online action detection," *arXiv preprint arXiv:2011.09158*, 2020.

[135] M. Gao, M. Xu, L. S. Davis, R. Socher, and C. Xiong, "Startnet: Online detection of action start in untrimmed videos," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5542–5551.

[136] Z. Shou, J. Pan, J. Chan, K. Miyazawa, H. Mansour, A. Vetro, X. Giro-i Nieto, and S.-F. Chang, "Online detection of action start in untrimmed, streaming videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 534–551.

[137] J. Gao, Z. Yang, and R. Nevatia, "Red: Reinforced encoder-decoder networks for action anticipation," *arXiv preprint arXiv:1707.04818*, 2017.

[138] T. Mahmud, M. Hasan, and A. K. Roy-Chowdhury, "Joint prediction of activity labels and starting times in untrimmed videos," in *Proceedings of the IEEE International conference on Computer Vision*, 2017, pp. 5773–5782.

[139] Y. Abu Farha and J. Gall, "Uncertainty-aware anticipation of activities," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[140] Q. Ke, M. Fritz, and B. Schiele, "Time-conditioned action anticipation in one shot," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9925–9934.

[141] A. Furnari and G. M. Farinella, "What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6252–6261.

[142] S. Qu, G. Chen, D. Xu, J. Dong, F. Lu, and A. Knoll, "Lap-net: Adaptive features sampling via learning action progression for online action detection," *arXiv preprint arXiv:2011.07915*, 2020.

[143] M. Xu, Y. Xiong, H. Chen, X. Li, W. Xia, Z. Tu, and S. Soatto, "Long short-term transformer for online action detection," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[144] B. Zhang, H. Chen, M. Wang, and Y. Xiong, "Online action detection in streaming videos with time buffers," *arXiv preprint arXiv:2010.03016*, 2020.

[145] D. Wang, W. Ouyang, W. Li, and D. Xu, "Dividing and aggregating network for multi-view action recognition," in *ECCV*, 2018, pp. 451–467.

[146] S. Vyas, Y. S. Rawat, and M. Shah, "Multi-view action recognition using cross-view video prediction," in *ECCV*. Springer, 2020, pp. 427–444.

[147] Y. Liu, L. Wang, Y. Bai, C. Qin, Z. Ding, and Y. Fu, "Generative view-correlation adaptation for semi-supervised multi-view learning," in *European Conference on Computer Vision*. Springer, 2020, pp. 318–334.

[148] A. Piergiovanni and M. S. Ryoo, "Recognizing actions in videos from unseen viewpoints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4124–4132.

[149] L. Wang, Z. Ding, Z. Tao, Y. Liu, and Y. Fu, "Generative multi-view human action recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6212–6221.

[150] C. Fan, J. Lee, M. Xu, K. Kumar Singh, Y. Jae Lee, D. J. Crandall, and M. S. Ryoo, "Identifying first-person camera wearers in third-person videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5125–5133.

[151] K. Deepak, G. Srivathsan, S. Roshan, and S. Chandrakala, "Deep multi-view representation learning for video anomaly detection using spatiotemporal autoencoders," *Circuits, Systems, and Signal Processing*, vol. 40, no. 3, pp. 1333–1349, 2021.

[152] H.-I. Ho, W.-C. Chiu, and Y.-C. F. Wang, "Summarizing first-person videos from third persons' points of view," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 70–85.

[153] J. Meng, S. Wang, H. Wang, J. Yuan, and Y.-P. Tan, "Video summarization via multi-view representative selection," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1189–1198.

[154] R. Panda and A. K. Roy-Chowdhury, "Multi-view surveillance video summarization via joint embedding and sparse optimization," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2010–2021, 2017.

[155] S. Haresh, S. Kumar, H. Coskun, S. N. Syed, A. Konin, Z. Zia, and Q.-H. Tran, "Learning by aligning videos in time," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5548–5558.

[156] G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari, "Actor and observer: Joint modeling of first and third-person videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7396–7404.

[157] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1134–1141.

[158] B. Xiong, H. Fan, K. Grauman, and C. Feichtenhofer, "Multiview pseudo-labeling for semi-supervised learning from video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7209–7219.

[159] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

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

[161] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 780–787.