UNIVERSITÄT BONN

Temporal Action Segmentation: Weak Supervision and Efficiency

DISSERTATION

zur Erlangung des Doktorgrades (*Dr. rer. nat.*)

der Mathematisch-Naturwissenschaftlichen Fakultät

der Rheinischen Friedrich–Wilhelms–Universität, Bonn

vorgelegt von

YASER SOURI

aus

Toyserkan, Iran

Bonn, 2023

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich–Wilhelms–Universität Bonn

1. Gutachter / 1st Advisor: Prof. Dr. Juergen Gall
2. Gutachterin / 2nd Advisor: Prof. Dr. Angela Yao
Tag der Promotion / Day of Promotion: 19 January 2023
Erscheinungsjahr / Year of Publication: 2023

Abstract

by Yaser Souri

for the degree of

Doctor rerum naturalium

Action segmentation deals with the temporal understanding of actions that occur in an untrimmed video. The output of action segmentation is an action label for each frame of the input video. This task has recently received a lot of attention from both academy and industry as it has a lot of important applications such as home monitoring systems, guidance systems in assembly lines, and semantic video retrieval. While existing methods for action segmentation already achieve encouraging results, most require fully annotated datasets for training. Obtaining fully annotated untrimmed videos for training action segmentation models is quite expensive. This has resulted in a lot of recent progress on weakly supervised approaches for action segmentation from transcripts and timestamps. Most of the existing weakly supervised action segmentation approaches are very inefficient during training and testing while some others make unrealistic assumptions about the type of weak supervision available during training. In this dissertation, we focus on weakly supervised approaches for action segmentation that are efficient and fast during training and testing. We also focus on more realistic weak supervision in the form of timestamps without restrictive assumptions.

First, we propose a novel end-to-end approach for weakly supervised action segmentation from transcripts based on a two-branch neural network. The two branches of our network predict two redundant but different representations for action segmentation and we propose a novel mutual consistency (MuCon) loss that enforces the consistency of the two redundant representations. Using the MuCon loss together with a loss for transcript prediction, our proposed approach achieves the accuracy of state-of-the-art approaches while being 14 times faster to train and 20 times faster during inference. This approach provides a unified architecture that supports both fully and weakly supervised training and makes mixed supervision training possible.

Second, we introduce FIFA, a fast approximate inference method for action segmentation and alignment. Unlike previous approaches, FIFA does not rely on expensive dynamic programming for inference. Instead, it uses an approximate differentiable energy function that can be minimized using gradient-descent. FIFA is a general approach that can replace exact inference improving its speed by more than 5 times while maintaining its performance. FIFA is an anytime inference algorithm that provides a better speed vs. accuracy trade-off compared to exact inference.

Third, we focus on weak supervision in the form of timestamps. Timestamp supervision is a promising type of weak supervision as it is less expensive to obtain compared to full supervision and results in accurate action segmentation models. Previous works assume that every action segment is annotated with a timestamp which is an unrealistic and restrictive assumption. Robustness against missing annotations is of crucial importance as such mistakes can occur during the annotation process. We proposed an approach for action segmentation

from timestamps that is robust to missing timestamp annotation of some action segments.

In robotics applications after recognizing the actions, the goal is to perform an action and complete a task. To this end, our final contribution is task-driven object detection. When humans have to solve everyday tasks, they simply pick the most suitable objects. While trivial for humans, this issue is not addressed by current benchmarks and approaches for object detection that focus on detecting object categories. We, therefore, introduce the COCO-Tasks dataset which comprises about 40,000 images where the most suitable objects for 14 tasks have been annotated. We furthermore propose an approach that detects the most suitable objects for a given task. The approach builds on a Gated Graph Neural Network to exploit the appearance of each object as well as the global context of all present objects in the scene.

We evaluate our approaches on various benchmarks and datasets. Extensive evaluations and comparisons with state-of-the-art and various baselines show the effectiveness of our proposed approaches and their efficiency.

Keywords: Video Understanding, Action Segmentation, Weakly Supervised Action Segmentation, Efficient Action Segmentation, Task Driven Object Detection

Acknowledgements

I would like to thank my dear advisor, Prof. Juergen Gall, for all the guidance and support during the ups and downs of my Ph.D. Prof. Gall provided me with the freedom to work on the ideas that I was interested in and supported me throughout my journey.

I would like to thank my funding partners from Toyota Motor Europe, Gianpiero Francesca, Luca Minciullo, and Fabien Despinoy for their trust in me and their help in the latter half of my Ph.D.

I am also grateful to my colleagues and friends in the Computer Vision Group at University of Bonn. You provided a warm and welcoming environment for someone like me. I will never forget the happy memories we had. I would like to thank my co-authors, Johann, Mohsen, Sovan, Yazan, Emad, and Alexander. The work we did would have not been possible without your help and effort.

I would like to thank Rodrigo Benenson and Vittorio Ferrari from Google Research Zurich for the great internship I had with them. I learned many valuable lessons during my internship.

Finally, I would like to thank my family. I would like to thank my wife and partner through the past 12 years Azadeh for her unconditional support. Azadeh's patience, care, kindness, and sacrifice made it all possible. I would like to thank my two wonderful daughters Zeinab and Zahra for bringing joy and meaning to my life and giving purpose to me. Last but not least, I want to thank my parents and brother. I was very lucky to have amazing parents who selflessly supported me throughout my life and especially during my Ph.D. Thank you.

In memory of *Mohammad Salehe*

Mohammad was one of the best friends I ever had.
He was a great engineer, scientist, educator, brother, and friend.
Mohammad left a positive impact everywhere he was and with anyone he interacted with. I will forever be grateful for the time that I knew him, worked with him, and learned from him.
Such a combination of scientific and engineering competence with unique personality traits such as generosity, kindness, faithfulness, and optimism is exceptional and hard to find.
Mohammad passed away while he was still a Ph.D. student at University of Toronto, onboard the PS752 flight that was shot down in January 2020.
During his short life, he achieved great accomplishments in industry and academia.
We will forever miss Mohammad.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Challenges . . . 3
1.2.1 Long-Range Temporal Dependencies . . . 3
1.2.2 Supervision . . . 3
1.2.3 Smooth and Structured Prediction . . . 4
1.2.4 Efficiency . . . 4
1.2.5 Detecting the Most Suitable Object . . . 5
1.3 Contributions . . . 5
1.3.1 MuCon: Mutual Consistency for Action Segmentation . . . 5
1.3.2 FIFA: Fast Approximate Inference . . . 6
1.3.3 Robust Learning from Timestamp Supervision . . . 6
1.3.4 Finding the Most Suitable Object . . . 7
1.4 Thesis Structure . . . 7

2 Background **9**
2.1 Supervised Learning . . . 9
2.1.1 Weakly Supervised Learning . . . 10
2.2 Representation . . . 11
2.2.1 Video Representation . . . 11
2.2.2 Temporal Action Segmentation Target Representation . . . 13
2.3 Neural Networks . . . 14
2.3.1 Building Blocks . . . 14
2.3.2 Network Training . . . 19
2.4 Temporal Action Segmentation . . . 19
2.4.1 Problem Statement . . . 20
2.4.2 Supervision . . . 20
2.4.3 Evaluation Metrics . . . 21
2.5 Datasets . . . 22

3 Literature Review **25**
3.1 Action Recognition . . . 25
3.2 Temporal Action Segmentation . . . 26
3.2.1 Fully Supervised Temporal Action Segmentation . . . 26
3.2.2 Weakly Supervised Temporal Action Segmentation . . . 29
3.3 Timestamp Supervision for Recognizing Activities . . . 30
3.4 Sequence to Sequence Learning . . . 31
3.5 Energy-Based Inference . . . 31

4 Mutual Consistency for Action Segmentation **33**
4.1 Introduction . . . 34
4.2 Weakly Supervised Action Segmentation . . . 36
4.3 Proposed Method . . . 36
4.3.1 Temporal Backbone . . . 38
4.3.2 Frame Classification Branch . . . 38
4.3.3 Segment Generation Branch . . . 38
4.3.4 Mutual Consistency Loss . . . 40
4.3.5 Differentiable Mask Generation . . . 41
4.3.6 Regularization . . . 43
4.3.7 Fully Supervised and Mixed Training . . . 44
4.3.8 Inference . . . 44
4.4 Experiments . . . 45
4.4.1 Evaluation Protocols and Datasets . . . 45
4.4.2 Implementation Details . . . 45
4.4.3 Ablation Experiments . . . 46
4.4.4 Statistical Analysis . . . 48
4.4.5 Training and Inference Time . . . 49
4.4.6 Further Comparisons . . . 50
4.4.7 Qualitative Evaluation . . . 52
4.5 Conclusion . . . 55

5 Approximate Action Segmentation Inference **57**
5.1 Introduction . . . 58
5.2 Background . . . 59
5.2.1 Action Segmentation . . . 59
5.2.2 Inference in Action Segmentation . . . 60
5.2.3 Exact Inference . . . 61
5.3 FIFA: Fast Inference Approximation . . . 61
5.3.1 Approximate Differentiable Energy E^* . . . 63
5.3.2 Time Complexity . . . 65
5.4 Experiments . . . 66
5.4.1 Evaluation Protocols and Datasets . . . 66
5.4.2 Implementation Details . . . 66
5.4.3 Results and Discussions . . . 66
5.4.4 Comparison to State of the Art . . . 69
5.4.5 Qualitative Results and Visualization . . . 72
5.5 Conclusion . . . 79

6 Robust Action Segmentation from Timestamp Supervision **81**
6.1 Introduction . . . 82
6.2 Proposed Method . . . 84
6.2.1 Timestamp Supervision for Action Segmentation . . . 85
6.2.2 Robust Pseudo Ground Truth Generation . . . 85

6.3 Experiments . . . 88
6.3.1 Datasets and Metrics . . . 88
6.3.2 Implementation Details . . . 89
6.3.3 Impact of Missing Annotations . . . 89
6.3.4 Comparison with Baselines . . . 90
6.3.5 Impact of Initialization . . . 92
6.3.6 Evaluation of Different Timestamps Selection Strategies . . . 92
6.3.7 Sensitivity to β . . . 95
6.3.8 Performance on Timestamps without Missing Annotations . . . 95
6.3.9 Localization Experiment . . . 96
6.3.10 Speed of Pseudo Ground Truth Generation . . . 96
6.4 Limitations . . . 98
6.5 Conclusion . . . 98

7 What Object Should I Use? - Task Driven Object Detection 99
7.1 Introduction . . . 100
7.2 Related Work . . . 101
7.3 COCO-Tasks Dataset . . . 102
7.4 Task Driven Object Detection . . . 109
7.4.1 Proposed Method . . . 110
7.4.2 Implementation Details . . . 111
7.5 Experiments . . . 112
7.5.1 Comparison to Baselines . . . 112
7.5.2 Ablation Experiments . . . 114
7.5.3 Class Bias Analysis . . . 115
7.5.4 Task Wise Difficulty Analysis . . . 116
7.5.5 Scene Context Learned by GGNN . . . 116
7.6 Conclusion . . . 123

8 Conclusion 125
8.1 Summary . . . 125
8.2 Future Work . . . 126
8.2.1 Larger Datasets . . . 126
8.2.2 Incorporating FIFA Inference in MuCon Training . . . 126
8.2.3 Temporal Action Segmentation From Multiple Levels of Supervision . . . 127
8.2.4 Feature Learning for Temporal Action Segmentation . . . 127
8.2.5 Moving Beyond Fixed Tasks for Task Driven Object Detection . . . 127

List of Figures

1.1 The top row shows video frames of a pancake cooking tutorial (*Tasty*, 2019). The bottom row visualizes the actions being performed in the video. A human is naturally able to predict the actions and their durations by watching the video. In temporal action segmentation, the goal is to predict the actions that are being performed in each frame given the input untrimmed video. 2

1.2 A comparison of framewise and structured prediction for temporal action segmentation. The first row shows input video frames while the second and third-row visualize the framewise and structured predictions from the MuCon model (Chapter 4). Making framewise predictions usually results in noisy and undesirable output while structured predictions are easier to interpret. 4

2.1 Improved Dense Trajectories (IDT) feature extraction pipeline. First feature points are densely sampled at different scales (left). Then using optical flow, the feature points are tracked in subsequent frames constructing the trajectories (middle). Finally different discriptors such has histogram of oriented gradients (HOG) are used to encode the trajectories. The figure is taken from *Wang and Schmid* (2013). 11

2.2 Feature extraction from a video x using the I3D (*Carreira and Zisserman*, 2017) network. A window of size w from the frames of the video x is used as an input for the I3D (*Carreira and Zisserman*, 2017) network. The output of the final convolutional layer is used as a feature representation of that window ϕ after removing the spatial and temporal dimensions of the network output using average pooling. This process is repeated for the whole video using in a sliding window fashion with stride s. The final representation for the whole video is the concatenation of the representations of each window ϕ_t. 12

2.3 Two different representations for temporal action segmentation output. In *frame-level* representation, the output is represented as a list of action labels for each frame. On the other hand, in *segment-level* representation the output is represented as a segment and its corresponding length. 13

2.4 A schematic representation of layers of a neural network. Each layer f_l receives the input from the previous layer and transforms it. The final layer generates the output of the model. 14

2.5 A schematic representation of a fully connected layer. A fully connected layer gets an input of size D and produces an output of size O. Each output is a function of every input. 15

2.6 A schematic representation of a 2D convolutional input and output. The 2D convolutional layer gets as input a 3D tensor and outputs another 3D tensor with different dimensions. The depth is also known as the number of channels. For a 3-channel RGB color image, the number of channels is 3. The depth of the output depends on the parameters of the convolutional layer. The width and height of the output change depending on the width and height of the input. 16

2.7 Visualization of a 2d convolutional operation for a single channel. The input (shown in blue) is filtered with a shared kernel (shown in dark blue) to produce the output (shown in green). The figure is taken from *Dumoulin and Visin* (2016). 17
2.8 (a) Visualization of a Recurrent cell. (b) Visualization of sequence processing using a Recurrent cell. 17
2.9 Visualization of the different types of temporal action segmentation supervision. The first row visualizes the full supervision, which is the case that the action label for each frame is known. The second row visualizes the timestamp weak supervision. In this case, only a single frame is annotated for the action segments. The final row visualizes the transcript weak supervision. In this setting, only the ordered list of actions that occur in the video is known and no information is available regarding the temporal boundary and size of the action segments. 20
2.10 Visualization of the ambiguity in the transcript weak supervision for temporal action segmentation. All of the action segmentations visualized have the same transcript. . 21

3.1 Overview of a dilated residual block in the MS-TCN architecture. The figure is taken from *Abu Farha and Gall* (2019). 27
3.2 Overview of a the full MS-TCN network architecture, consisting of multiple stages of several dilated residual convolutional blocks. The figure is taken from *Abu Farha and Gall* (2019). 28

4.1 Average inference time per video (seconds) vs. average mean over frames (MoF) accuracy (%) for weakly supervised action segmentation approaches on the Breakfast dataset (*Kuehne et al.*, 2014). The average MoF is calculated over 5 training/inference iterations. The proposed approach (MuCon-full) provides the best trade-off between inference time and accuracy. More details are provided in Section 4.4.5. 35
4.2 Our proposed network consists of three subnetworks (gray). The temporal backbone f_t embeds the input features in the hidden representation Z which is used for two branches. While the frame classification branch f_c predicts framewise class probabilities Y for action segmentation, the segment generation branch f_s predicts the segment representation S for action segmentation. We train our network using two loss functions. While the transcript prediction loss $\mathcal{L}_t$ enforces that the predicted transcript A matches the ground-truth transcript $\hat{A}$, our proposed mutual consistency (MuCon) loss $\mathcal{L}_\mu$ enforces that the two representations are consistent. 37
4.3 Visualization of the segment generation branch and the loss functions. 39
4.4 Examples of different masks for three consecutive action segments. The top row shows regular masks with different shapes while the bottom row shows masks generated with added 10% overlap. The left, middle and right figures depict box, bell, and trapezoid shaped masks. 41
4.5 Using beam search to reduce the inference time of CDFL. The blue squares show the performance of CDFL with different values for the beam size. The x-axis is the total inference time for split 1 of the Breakfast dataset in log scale. 51

4.6 MoF accuracy (%) for training with mixed supervision. The x-axis denotes the percentage of videos that are fully annotated. The average accuracy for 5 runs on split 1 of the Breakfast dataset is reported. 53
4.7 Qualitative examples for weakly supervised action segmentation on the Breakfast dataset. Each figure visualizes a different video from the test set of split 1. We compare the results from MuCon, NNV, and CDFL with the ground truth (GT). Each row shows the result for the entire duration of a video and the colored segments show when the actions occur in the video. 54
4.8 Qualitative examples for different levels of supervision on the Breakfast dataset. Each figure visualizes a different video from the test set of split 1. *GT* visualizes the ground truth segmentation and *Weak*, *Mixed*, and *Full* visualize the output of MuCon trained with weak, mixed (10%), or full supervision, respectively. 55

5.1 Overview of the FIFA optimization process. At each step in the optimization, a set of masks are generated using the current length estimates. Using the generated masks and the frame-wise negative log probabilities, the observation energy is calculated in an approximate but differentiable manner. The length energy is calculated from the current length estimate and added to the observation energy to calculate the total energy value. Taking the gradient of the total energy with respect to the length estimates we can update it using a gradient step. 62
5.2 Speed vs. accuracy trade-off of different inference approaches applied to the MuCon method. Using FIFA we can achieve a better speed vs. accuracy trade-off compared to frame sampling or hypothesis pruning in exact inference. 67
5.3 Effect of the length energy multiplier for Laplace and Gaussian length energy. Accuracy is calculated on the Breakfast dataset using FIFA applied to the MuCon approach trained in the weakly supervised action segmentation setting. 68
5.4 Effect of the optimization step size on the performance of weakly supervised action segmentation using FIFA applied on the MuCon approach. Accuracy is calculated on the Breakfast dataset. 69
5.5 The value of the approximate energy during FIFA optimization for SGD and Adam optimizers for the same inference. 71
5.6 Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon 73
5.7 Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon 73
5.8 Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon 73
5.9 Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon 74
5.10 Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon 74
5.11 Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon, Failure Case 74
5.12 Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL 75
5.13 Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL 75
5.14 Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL 75
5.15 Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL 76
5.16 Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL 76
5.17 Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL, Failure Case 76

5.18 Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN 77
5.19 Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN 77
5.20 Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN 77
5.21 Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN 78
5.22 Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN 78
5.23 Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN, Failure Case 78

6.1 For an input video with 3 actions A, B, and C, the annotator has correctly annotated the actions A and C, but action B is not annotated. In this setting, *Li et al.* (2021) assumes that all actions are annotated and generates a pseudo ground truth that is incorrect for the entire region of action B. We propose an approach that takes into account the possibility of missing segments and correctly ignores the region corresponding to the missed action in the generated pseudo ground truth. 83
6.2 Accuracy of different action segmentation approaches trained using timestamp supervision with different percentages of the action segments being annotated with timestamps. The top figure depicts results on the 50Salads dataset (*Stein and McKenna*, 2013), while the bottom depicts results on the Breakfast dataset (*Kuehne et al.*, 2014). Our approach consistently achieves higher accuracy compared to the approach of *Li et al.* (2021), and the gap between our approach and the approach of *Li et al.* (2021) increases as the percentage of annotated segments decreases. In the “Oracle" setting the correct boundary is known for each annotated timestamp. 84
6.3 Given the timestamps, the temporal region of the video is divided into multiple sections. The region between the two consecutive timestamps p_1 and p_2 has been divided into 3 parts. The first part r_1 will take the same label as p_1 and the last part l_2 will take the same label as p_2. The gray colored regions g_i will be ignored and not used for training. 86
6.4 The plateau function (6.5) with center parameter λ^c and width parameter λ^w. 87
6.5 Qualitative results for comparing the generated labels by our approach and *Li et al.* (2021) on two videos from the 50Salads dataset. The top row shows the ground truth frame-wise labels and the annotated timestamps in black. Missing annotated segments are highlighted with red dashed ellipses. The pseudo labels generated by our approach are shown in the second row, where the ignored regions are colored in gray. The pseudo labels generated by *Li et al.* (2021) are shown in the third row. For the missing segments, our approach successfully assigned these frames to the ignored region. On the contrary, the approach of *Li et al.* (2021) densely generates labels that result in wrong pseudo labels for these segments. 91
6.6 Impact of β on different percentages of the annotated segments on the 50Salads dataset. For each experiment, the mean and standard deviation of three runs are shown. 94
6.7 Average length of the ignore regions and the number of the ignore regions with length 0 for different values of β. The numbers are reported for the training set of split 1 of the 50Salads dataset after the initial 30 epochs. 95

6.8 Qualitative results showing a limitation of our proposed pseudo ground truth generation approach for small action segments. In this example a large portion of frames were ignored around small action segments. 98

7.1 What object in the scene would a human choose to serve wine? In the left image, the wine glass is preferred to other drinking glasses. In the right image, neither a wine glass nor other drinking glasses are present. The cup is therefore chosen by the human. 101

7.2 Distribution of chosen objects for all tasks across COCO categories. 103

7.3 Whether an object should be chosen for a task depends on the object properties as well as the presence of better alternatives. The image in the first row shows the objects (green bounding boxes) that have been chosen by the majority of the annotators for the task `step on something to reach top of a shelf`. While in the first image the table is chosen, the chair is preferred instead of the table in the second image. In the last image, one of the two beds is chosen. The second row shows examples for the task `sit comfortably`. In the first image, both beds are selected. In the second image, the comfortable chair is preferred over the bed and and the stool. In the third image, the real toilet is selected. Note that the reflection of the toilet in the mirror is annotated as object in COCO as object, but it is not selected since one cannot sit on it. 105

7.4 Distribution of the number of preferred objects per image for tasks 2 and 8, which are the tasks with highest and lowest number of selected instances per image, respectively. 107

7.5 Overview of the annotation tool used by the annotators. At the top (blue section) the user can see the description of the task with an image which clarifies the intention of the task. Below is the image with already selected preferred objects (as purple). The annotator should select any object that is suitable by clicking on it. If the annotator hovers on an COCO annotated instance in the image, it will become highlighted with yellow color. No COCO class information is provided to the annotator. The user is able to click on an already selected instance to deselect it. On the right hand side of the image is an area over that the annotator can hover with the mouse to view all COCO annotated instances in the image (all with the same color). If there are no preferred instances for the task in the image, the user is able to mark a check box at the bottom and save the annotation result for the image. 108

7.6 Overview of the proposed method. Given an image containing a number of objects (3 in this example shown with colors green, blue and red), our method first extracts ResNet features from each bounding box containing the object. (a) Using the extracted features and one-hot encoding of the detected class of the object ($\hat{c}_i$), we compute the initial hidden state of the graph node corresponding to that object using (7.1). (b) Using the hidden states of all of the other graph nodes, we aggregate the scene information using (7.2) and update the node's hidden state as in (7.3). (c) After T iterations of the GGNN, we combine each node's initial and final hidden states using (7.4) to compute the probability of that object being favored for the task. (d) Finally to make the features learned by the ResNet discriminative we also force the network to estimate suitability scores only from visual features of a single object. At test time, we average the two estimated probabilities. 109
7.7 Accuracy of the predicted categories when retrieving nearest neighbors based on h_i^0 vs. retrieving based on h_i^T. 118
7.8 Average difference of the number of same instances between the query image and the nearest neighbor images based on h_i^0 vs. h_i^T. 118
7.9 A qualitative example from Task 1 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 119
7.10 A qualitative example from Task 2 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 119
7.11 A qualitative example from Task 3 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 119
7.12 A qualitative example from Task 4 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 120
7.13 A qualitative example from Task 5 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 120
7.14 A qualitative example from Task 6 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 120
7.15 A qualitative example from Task 7 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 120
7.16 A qualitative example from Task 8 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 121
7.17 A qualitative example from Task 9 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 121
7.18 A qualitative example from Task 10 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 121
7.19 A qualitative example from Task 11 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 121
7.20 A qualitative example from Task 12 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 122
7.21 A qualitative example from Task 13 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 122
7.22 A qualitative example from Task 14 test set comparing nearest neighbors according to h_i^0 vs. h_i^T. 122

List of Tables

4.1 Transcript prediction accuracy on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. The first four rows show the matching score of the segment generation branch for different dilation factors used for the temporal backbone. In these settings, the mutual consistency loss $\mathcal{L}_\mu$ is not used. The last row shows the proposed setting with the mutual consistency loss. 46

4.2 Impact of mask generation on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. As shown in Figure 4.4, different shapes with or without overlap can be used for the mask generation. We report the MoF accuracy, matching score, and F1 score. 46

4.3 Impact of regularizers on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. The first three rows denote settings where the length regularizer $\mathcal{L}_\ell$ for the segment generation branch, the smoothing loss $\mathcal{L}_s$ for the frame classification branch, or both are omitted. 47

4.4 Impact of fusing both branches on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. While MuCon-Y and MuCon-S use either the frame classification branch or the segment generation branch for inference, MuCon-full fuses both branches for inference. 47

4.5 Impact of teacher forcing on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. The first row denotes a setting where teacher forcing is not used for training. The second row denotes a setting where teacher forcing is used only for the first 70 epochs. The last row denotes a setting where teacher forcing is used for all epochs. 48

4.6 Comparison of accuracy, training, and inference time. We report the average and standard deviation of MoF and matching score for 5 runs on the entire Breakfast dataset (*Kuehne et al.*, 2014). For ISBA and MuCon-Y, the matching score is not calculated as they predict frame-wise labels and not the transcript. Training and inference time is measured as wall clock time. The training time is measured for training on split 1 of the Breakfast dataset (*Kuehne et al.*, 2014). The inference time is measured as the average inference time for a video from the test set of split 1. . . 49

4.7 Friedman statistical test results. The top row and the leftmost column indicate different approaches and their MoF accuracy and matching score from Table 4.6. Note that the test is symmetrical. If the p-value is smaller than 0.05 the difference is considered as significant (indicated with ✓); otherwise the difference is not considered as significant (indicated with ✗). 50

4.8 Comparison to the state-of-the-art. The highest score is indicated by a bold font and the second highest is underlined. $*$ indicates that the reported metric is averaged over 5 runs. 52

4.9 Effect of the MuCon loss on the accuracy in the fully supervised setting. The results are reported for split 1 of the Breakfast dataset. 52

4.10 Comparison with the state-of-the-art for fully supervised action segmentation on the Breakfast dataset. (* obtained from *Ding and Xu* (2018)). 53
4.11 Training with mixed supervision. The average MoF accuracy and standard deviation for 5 runs on split 1 of the Breakfast dataset is reported. 53

5.1 Impact of the length initialization for MuCon using exact inference and FIFA for weakly supervised action segmentation on the Breakfast dataset. 68
5.2 Results for weakly supervised action segmentation on the Breakfast dataset. * indicates results obtained by running the code on our machine. 70
5.3 Results for fully supervised action segmentation setup on the Breakfast dataset. . . . 70
5.4 Results for weakly supervised action segmentation on the Hollywood extended dataset. Time is reported in seconds. * indicates results obtained by running the code on our machine. 71
5.5 Results for fully supervised action segmentation on the Hollywood extended dataset. EI stands for Exact Inference. 72
5.6 Results for weakly supervised action alignment on the Breakfast dataset. 72

6.1 Impact of missing segments on the 50Salads dataset. 90
6.2 Impact of missing segments on the Breakfast dataset. 92
6.3 Comparison with different baselines on the 50Salads dataset. 93
6.4 Impact of initialization on the 50Salads dataset. 93
6.5 Comparing different selection strategies of timestamps using 90% of the segments on the 50Salads dataset. Our approach outperforms *Li et al.* (2021) using every strategy on every metric. 94
6.6 Results on timestamps without missing segments on the 50Salads dataset. * results obtained from *Li et al.* (2021). 96
6.7 Results on timestamps without missing segments on the Breakfast dataset. * results obtained from *Li et al.* (2021). 97
6.8 Comparison for action localization with timestamp supervision on the GTEA and BEOID datasets. 97

7.1 List of the 14 tasks in the COCO-Tasks dataset and some statistics. Selected object categories (column 3) are COCO object categories for which there exists at least one instance chosen by the majority of the annotators for a given task. Column 4 reports how many instances of each of the selected categories are in the images. Column 5 provides the numbers of object instances that are chosen for each task. Column 6 counts the number of instances of categories in an image where at least one instance but not all instances of the same category are selected. Examples of such cases are shown in the last column of Figure 7.3. The last column reports the probability that two annotators agree if an object is preferred or not. Overall, we have a very high annotation consistency. 104
7.2 We ensured that for each task images with certain COCO supercategories are over-represented in the dataset. This table shows the respective supercategories. 106

7.3 Comparison of the proposed method to several baselines on ground truth bounding boxes as well as Faster-RCNN (*Ren et al.*, 2015b) detections. The classification baseline is the strongest one but achieves 12.6% lower mAP on ground truth bounding boxes and 3.8% lower mAP on detections compared for our proposed approach. . . . 112
7.4 Evaluation of the components of our proposed method. We start with a task wise classifier, (a) then add joint training, (b) add COCO classes as input, (c) introduce the GGNN, (d) add weighted aggregation, (e) add the discriminatory loss and (f) perform fusion. Further ablation experiments (g) and (h) reveal the impact of the visual information. 114
7.5 Results on the subset of the test images where an object belonging to the most frequently chosen category was neglected in favor of an object from a less frequently chosen category. 116
7.6 We present the task wise numbers of the proposed method on Faster-RCNN (*Ren et al.*, 2015b) detections, the Yolov2 (*Redmon and Farhadi*, 2017) detection and on ground truth bounding boxes. 117

Nomenclature

Abbreviations

An alphabetically sorted list of abbreviations used in the thesis:

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
GPU	Graphical Processing Unit
RGB	Red Green Blue (3 channels of a color image)
SGD	Stochastic Gradient Descent
IDT	Improved Dense Trajectory
PCA	Principal Component Analysis
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
ReLU	Rectified Linear Unit
MoF	Mean over Frames
IoD	Intersection over Detection
IoU	Intersection over union
Acc	Accuracy
BG	Background
TP	True positive
FP	False positive
FN	False negative
mAP	mean Average Precision
HMM	Hidden Markov model
GNN	Graph Neural Network
GGNN	Gated Graph Neural Network
GCN	Graph Convolutional Network
GTEA	Georgia Tech Egocentric Activities
BEOID	Bristol Egocentric Object Interactions Dataset
CTC	Connectionist Temporal Classification
DTW	Dynamic Time Warping
NNV	Neural Network Viterbi

List of Publications

The thesis is based on the following publications:

- **What Object Should I Use? - Task Driven Object Detection**
  Johann Sawatzky*, Yaser Souri*, Christian Grund, Juergen Gall
  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
  (* indicates equal contribution (alphabetically ordered))

- **Fast Weakly Supervised Action Segmentation Using Mutual Consistency**
  Yaser Souri*, Mohsen Fayyaz*, Luca Minciullo, Gianpiero Francesca, Juergen Gall
  IEEE Transactions on Machine Intelligence and Pattern Analysis (PAMI), 2021.
  (* indicates equal contribution)

- **FIFA: Fast Inference Approximation for Action Segmentation**
  Yaser Souri, Yazan Abu Farha, Fabien Despinoy, Gianpiero Francesca, Juergen Gall
  DAGM German Conference on Pattern Recognition (GCPR), 2021.

- **Robust Action Segmentation From Timestamp Supervision**
  Yaser Souri*, Yazan Abu Farha*, Emad Bahrami*, Gianpiero Francesca, Juergen Gall
  The British Machine Vision Conference (BMVC), 2022.
  (* indicates equal contribution)

CHAPTER 1

Introduction

Contents

1.1 Motivation . . . 1
1.2 Challenges . . . 3
1.2.1 Long-Range Temporal Dependencies . . . 3
1.2.2 Supervision . . . 3
1.2.3 Smooth and Structured Prediction . . . 4
1.2.4 Efficiency . . . 4
1.2.5 Detecting the Most Suitable Object . . . 5
1.3 Contributions . . . 5
1.3.1 MuCon: Mutual Consistency for Action Segmentation . . . 5
1.3.2 FIFA: Fast Approximate Inference . . . 6
1.3.3 Robust Learning from Timestamp Supervision . . . 6
1.3.4 Finding the Most Suitable Object . . . 7
1.4 Thesis Structure . . . 7

1.1 Motivation

The production, availability, and consumption of video data are increasing every day at an exponential rate. The cost of video recording and storage hardware is decreasing, which has resulted in widely available video recording devices such as smartphones and smart home cameras. It has been reported that in 2019 around 500 hours of video are uploaded to Youtube every minutes (*Statista*, 2022) and this amount has been growing exponentially in the past ten years. With such growth comes the need to analyze, monitor, annotate and learn from this huge amount of often publicly available data. The research community has therefore shown great interest in video understanding (*Poppe*, 2010; *Zhu et al.*, 2020). Early work in video understanding was focused on classifying actions performed by humans in short and trimmed video clips (*Schuldt et al.*, 2004; *Wang and Schmid*, 2013) which is referred to as *action recognition*. The success of Deep Neural Networks resulted in a paradigm shift and the performance of action recognition approaches improved considerably by adopting the use of neural networks (*Simonyan and Zisserman*, 2014; *Wang et al.*, 2019; *Carreira and Zisserman*, 2017). With the availability of large scale datasets (*Kay et al.*, 2017; *Carreira et al.*, 2019; *Diba et al.*, 2020), current deep learning-based approaches in action recognition have achieved remarkable performance (*Feichtenhofer et al.*, 2019; *Feichtenhofer*, 2020; *Arnab et al.*, 2021). The current state-of-the-art approach of *Yan et al.* (2022) achieves remarkable top-1 classification accuracy of 89% between 400 classes of the Kinetics dataset (*Kay et al.*, 2017).



Figure 1.1: The top row shows video frames of a pancake cooking tutorial (*Tasty*, 2019). The bottom row visualizes the actions being performed in the video. A human is naturally able to predict the actions and their durations by watching the video. In temporal action segmentation, the goal is to predict the actions that are being performed in each frame given the input untrimmed video.

Although these results on trimmed video clips are important, in many real-world scenarios we are often faced with untrimmed videos that contain multiple actions and the goal is to recognize the actions that occur in the videos, their order, and their lengths. This task is known as *temporal action segmentation* which is the main focus of this dissertation.

Humans are naturally capable of temporal understanding of actions. When a person is watching a cooking tutorial video they are naturally able to understand and infer the actions that occurred in the video, their duration, and order (Figure 1.1).

Temporal action segmentation has many important applications in monitoring, web search, robotics, and more. For example, the assembly line monitoring system in a factory where the worker has to perform multiple consecutive actions can benefit a lot from temporal action segmentation (*Kumar et al.*, 2021). The system will be able to alert the worker if any steps are missed. Furthermore, the analysis of the duration of actions can help refine the assembly process for efficiency. In web search, temporal action segmentation can be used to semantically retrieve videos. A user can query the search engine for a specific action and the search engine can retrieve specific sections of the videos where that action is being performed. Temporal action segmentation also allows the possibility of more complicated queries such as one action followed by another action.

There have been many works in computer vision focusing on temporal action segmentation. Early approaches for temporal action segmentation rely on multi-scale sliding window processing (*Spriggs et al.*, 2009; *Rohrbach et al.*, 2012) or Markov models on top of frame-classifiers (*Kuehne et al.*, 2016; *Lea et al.*, 2016). More recent approaches for temporal action segmentation aim to capture long-range temporal dependencies using temporal convolutions (*Abu Farha and Gall*, 2019; *Lea et al.*, 2017). In this dissertation, we build on previous work and try to address some of their shortcomings.

Despite the progress in temporal action segmentation in recent years, a major bottleneck in their adaptation remains the expensive training annotation cost. Training temporal action segmentation approaches from full supervision is considerably expensive. Human annotators have to spend time recognizing each action in the video and accurately determine their start and end position. Therefore there has been a lot of work focusing on training action segmentation models from weaker forms of supervision (*Richard et al.*, 2018b; *Li et al.*, 2019; *Souri et al.*, 2021b; *Richard et al.*, 2018a; *Fayyaz and Gall*, 2020; *Li et al.*, 2021). One of the most popular types of weak supervision is *transcript* supervision where only the ordered list of actions is provided during training without any information regarding their start and end time. Although obtaining transcript supervision is less expensive than full supervision, the performance of temporal action segmentation approaches that use transcript supervision is still far lower than fully supervised approaches. To address this

issue, recently timestamp supervision (*Moltisanti et al.*, 2019) was used to train temporal action segmentation models (*Li et al.*, 2021). Timestamp supervision is only marginally more expensive to obtain compared to transcript supervision (*Ma et al.*, 2020) but the resulting model achieves much higher performance compared to transcript supervision (*Li et al.*, 2021).

In this thesis, we contribute to temporal action segmentation with a focus on weak supervision and training and testing efficiency. We propose a novel approach for end-to-end and efficient weakly supervised temporal action segmentation. We further address the inference problem in temporal action segmentation and propose a fast approximate but accurate approach for inference. Furthermore, we focus on timestamps and propose a robust approach for temporal action segmentation from timestamps. In the last project, we aim to learn what object can be used to execute an action. In case of steering coffee, it would be a spoon. But what object can we use if a spoon is not available? The question is thus what is the best object for executing an action in particular, if the default option object (in this case a spoon) is not available. The means we need to identify what objects are available and identify what is the best option, e.g. using a fork as a replacement of a missing spoon. To address this problem, we propose an object detection approach to detect the most suitable object to execute an action.

1.2 Challenges

Analyzing long untrimmed videos of human actions is a challenging task. In this section, we will discuss some of the challenges that need to be addressed for a temporal action segmentation model to be used for real-world scenarios. At the end we discuss the challenge of recognizing the most suitable object to execute an action.

1.2.1 Long-Range Temporal Dependencies

In an untrimmed video which can be multiple minutes in duration, there are dependencies between the actions that occur at different points of time thus the frames should not be treated independently. For example, when a person opens the cabinet and picks up something, later the person will likely close the cabinet. Initial temporal action segmentation approaches noticed that Recurrent Neural Networks are not good at modeling such long-range dependencies and proposed to use Hidden Markov Models for long-range temporal modelling (*Richard et al.*, 2017; *Kuehne et al.*, 2016). Recently multiple layers of temporal convolutional networks with exponentially growing dilation factors have been proposed to help with long-range dependencies (*Abu Farha and Gall*, 2019; *Li et al.*, 2020).

1.2.2 Supervision

It has been observed that part of the success of deep learning is dependent on having large-scale and preferably annotated datasets. Current state-of-the-art approaches for temporal action segmentation also rely on annotated datasets (*Li, AbuFarha, Liu, Cheng, and Gall*, 2020; *Yi, Wen, and Jiang*, 2021). In temporal action segmentation, the cost of obtaining full supervision (*i.e.* recognizing the actions, and annotating the exact temporal boundary of each action) is considerably large. This has prevented the adaptation of temporal action segmentation technology in large-scale scenarios. Researchers in temporal action segmentation have adopted the use of weak supervision to address this issue.

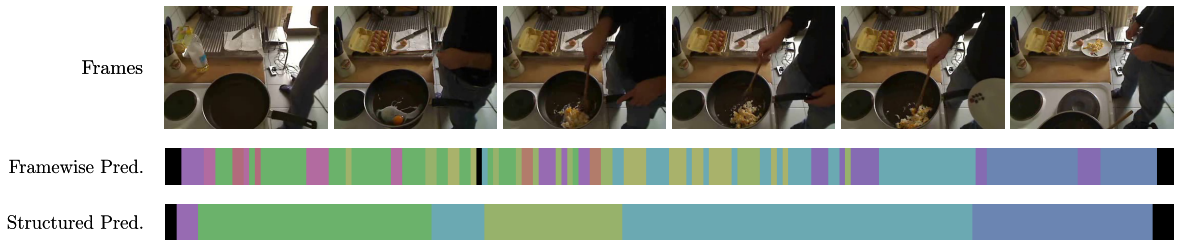


Figure 1.2: A comparison of framewise and structured prediction for temporal action segmentation. The first row shows input video frames while the second and third-row visualize the framewise and structured predictions from the MuCon model (Chapter 4). Making framewise predictions usually results in noisy and undesirable output while structured predictions are easier to interpret.

Transcript supervision which is the most widely used form of weak supervision in temporal action segmentation has shown promising results (*Souri et al.*, 2021a,b; *Li et al.*, 2019) but the gap between the performance of full supervision and transcript supervision approaches remains large. Timestamp supervision (*Moltisanti et al.*, 2019) has recently been adopted for temporal action segmentation with success (*Li et al.*, 2021) since it provides a much more effective form of weak supervision compared to transcript but the cost of obtaining it is still low (*Ma et al.*, 2020). The usage of timestamp supervision for temporal action segmentation in current approaches remains limited by unrealistic assumptions. Particularly the current approach assumes that every action is annotated and there are no missing annotations which is an unrealistic assumption.

1.2.3 Smooth and Structured Prediction

In temporal action segmentation, it is important that our models make smooth and *structured* predictions. Figure 1.2 visualizes the two types of predictions for a video from the Breakfast dataset (*Kuehne et al.*, 2014) from the MuCon model (Chapter 4). In structured prediction the model predicts smooth and interpretable output while in the framewise prediction the action class for each frame is predicted independently which can result in noisy and undesirable predictions. To make structured prediction, *Richard et al.* (2018b) proposed to use an inference algorithm which guarantees a smooth output. On the other hand, (*Abu Farha and Gall*, 2019) took a more implicit approach to penalize non-smooth predictions during training. But still, making smooth and structured prediction remains one of the major challenges of temporal action segmentation.

1.2.4 Efficiency

One of the main applications of temporal action segmentation is to analyze and understand the vast amounts of video recordings available. In a real-world scenario it is thus important that the action segmentation approach used is efficient and fast. Many of the recent approaches in temporal action segmentation (specially the weakly supervised approaches) are too slow for real-world usage (*Souri et al.*, 2020, 2021b). Unfortunately the more accurate the models, the slower they are. Furthermore the major bottleneck in temporal action segmentation is the inference stage, which as discussed above, is crucial in applicability of these models.

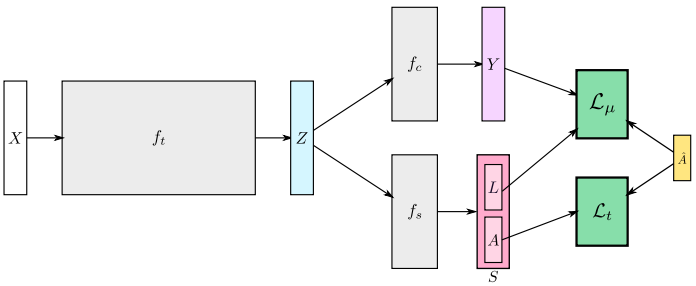
1.2.5 Detecting the Most Suitable Object

Typically, object detection methods focus on making local predictions, where the detection of an object in one region of an image is independent of other regions and the objects they contain. While this approach suffices for standard object detection tasks, it falls short when aiming to identify the most suitable object for executing an action. In such cases, global understanding and prediction becomes crucial. The presence of an object in the detection results should depend on the evaluation of all regions of the image, considering the possibility of a more suitable object existing in a different region.

1.3 Contributions

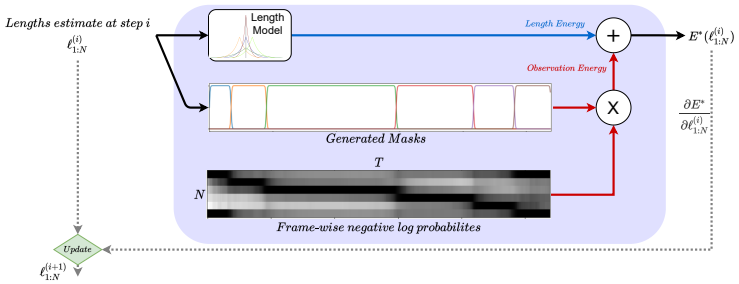
In this thesis, we propose approaches for temporal action segmentation that tackle some of the challenges mentioned above. Below we summarize our main contributions.

1.3.1 MuCon: Mutual Consistency for Action Segmentation



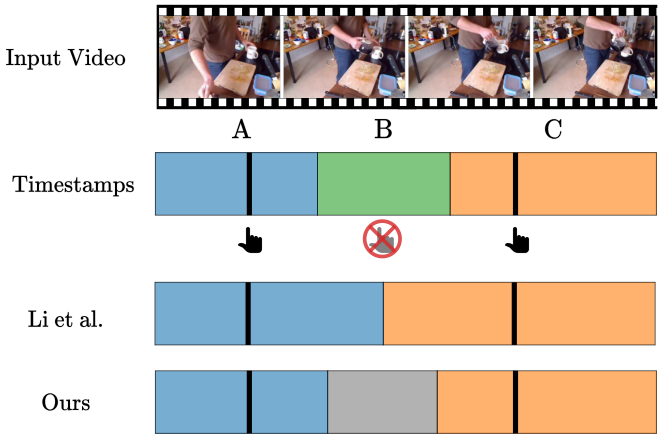
Our first contribution is a novel end-to-end architecture for temporal action segmentation named MuCon. MuCon proposes a two-branch architecture. The two branches of the MuCon network predict two redundant but different representations for temporal action segmentation and we propose a novel mutual consistency loss that enforces the consistency of the two representations. Contrary to previous approaches for temporal action segmentation, MuCon predicts the transcript for the input video. Predicting the transcript at test time allows for much faster inference speeds and generalizability to unseen transcripts. The proposed approach achieves the accuracy of state-of-the-art approaches on weakly supervised temporal action segmentation while being 14 times faster to train and 20 times faster during inference. We furthermore propose a novel mixed supervision setting where a small percentage of the videos are fully annotated and the remaining videos are only weakly annotated by transcripts. We observe that having 10% of the videos fully annotated, the accuracy gap between the mixed setting and the fully supervised setting is already reduced by about 50%.

1.3.2 FIFA: Fast Approximate Inference



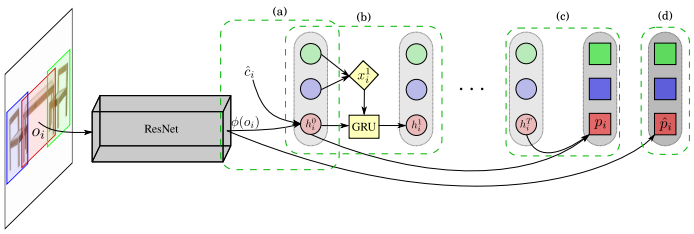
Although MuCon was able to considerably speed up temporal action segmentation, the inference step remains the efficiency bottleneck. In our second contribution, we focus on the inference problem in temporal action segmentation specifically. We first formulate the objective that is optimized with dynamic programming during the inference algorithm. Furthermore, we propose a differentiable approximate for that objective. Having a differentiable objective allows for solving the inference problem using gradient descent. Thus we introduce FIFA, a gradient-descent-based approach for solving the inference problem in temporal action segmentation. FIFA is a general approach that can replace exact inference improving its speed by more than 5 times while maintaining its performance. FIFA is an anytime inference algorithm that provides a better speed vs. accuracy trade-off compared to exact inference. We apply FIFA on top of state-of-the-art approaches for weakly supervised temporal action segmentation and alignment as well as fully supervised temporal action segmentation. FIFA achieves state-of-the-art results on most metrics on two temporal action segmentation datasets.

1.3.3 Robust Learning from Timestamp Supervision



Transcript supervision was the main focus of our previous contributions. In this work, we focus on timestamp supervision for temporal action segmentation. Timestamp supervision was shown to be an effective alternative to weak supervision since the cost of obtaining it is slightly higher than transcript supervision but results in more accurate models. Previous works assume that every action segment is annotated with a timestamp which is a restrictive assumption. In this work, we try to relax this restrictive assumption and take missing annotations for some segments into account. Robustness against missing annotations is of crucial importance as such mistakes occur during the annotation process. We show that our approach achieves much better action segmentation results compared to competitive approaches and various baselines.

1.3.4 Finding the Most Suitable Object



In the previous works, we focused on recognizing past actions. The next step after recognition is taking action and achieving a goal. This is of crucial importance in many applications such as robotics. The task of finding the most suitable object to perform a task has not been addressed in previous works. We, therefore, introduce the COCO-Tasks dataset which comprises about 40,000 images where the most suitable objects for 14 tasks have been annotated. We furthermore propose an approach that detects the most suitable objects for a given task. The approach builds on a Gated Graph Neural Network to exploit the appearance of each object as well as the global context of all present objects in the scene. In our experiments, we show the effectiveness of our approach compared to various baselines.

1.4 Thesis Structure

The rest of this dissertation is organized as follows:

Chapter 2 gives a brief overview of the basic concepts that this thesis relies on. It starts with an overview of supervised and weakly supervised machine learning. Later discussed are the main building blocks or layers used in neural network design followed by a formal definition of the temporal action segmentation problem and an overview of the datasets.

Chapter 3 provides a review of the video understanding literature with a focus on temporal action segmentation.

Chapter 4 proposes the MuCon approach for temporal action segmentation with weak, full, and mixed supervision. This chapter is based on *Souri et al.* (2021b).

Chapter 5 introduces the FIFA approach for temporal action segmentation inference. FIFA is an approximate, gradient-descent-based inference algorithm that is fast and accurate. This chapter is based on *Souri et al.* (2021a).

Chapter 6 introduces the robust temporal action segmentation approach for timestamp supervision. This approach enables training from natural timestamp supervision which can include missing annotations.

Chapter 7 introduces the COCO-Tasks dataset for detecting the most suitable object to achieve a task. We further introduce an approach based on graph neural networks to solve this problem. This chapter is based on *Sawatzky et al.* (2019).

Finally, conclusions are given in **Chapter 8** along with suggested directions for future work.

CHAPTER 2

Background

In this chapter, we discuss some of the core concepts used in this thesis. We start with an overview of supervised learning and weakly supervised learning. Then we discuss the input and output representations in temporal action segmentation. Then we briefly describe the fundamental building blocks in neural networks and the learning algorithm. Finally, we formally define the temporal action segmentation task, the datasets used, and the evaluation metrics.

Contents

2.1 Supervised Learning . . . 9
2.1.1 Weakly Supervised Learning . . . 10
2.2 Representation . . . 11
2.2.1 Video Representation . . . 11
2.2.2 Temporal Action Segmentation Target Representation . . . 13
2.3 Neural Networks . . . 14
2.3.1 Building Blocks . . . 14
2.3.2 Network Training . . . 19
2.4 Temporal Action Segmentation . . . 19
2.4.1 Problem Statement . . . 20
2.4.2 Supervision . . . 20
2.4.3 Evaluation Metrics . . . 21
2.5 Datasets . . . 22

2.1 Supervised Learning

In this thesis, we rely on machine learning, more specifically supervised machine learning to perform temporal action segmentation. In supervised learning the goal is to estimate a function $f(.)$ that takes a vectorized input x and estimates a vectorized output y, *i.e.* $y = f(x)$. There are four main components in a supervised learning approach which we will summarize below.

Data $\{x, \hat{y}\}_i$ known as the training set is the information that should be used to achieve the goal of learning. In supervised learning, the data consists of the input x and its corresponding target $\hat{y}$. In the training data, $\hat{y}$ is also known as the supervision or ground truth, since it is the label that the learning algorithm approach will use. i is the index of the training example. Usually, there is a finite number of training examples also known as the training data size. In our approaches, we assume that the input x and the target $\hat{y}$ are vectorized *i.e.* can be represented in vector form. For instance a 3 channel color image with height H and width W can be represented as a 3 dimensional matrix

i.e. $x \in \mathbb{R}^{H \times W \times 3}$. In a classification problem where the goal is to classify the input into one of the N number of classes, the target $\hat{y}$ can be represented as an integer number between 1 and N *i.e.* $\hat{y} \in [1, \ldots, N]$. In Section 2.2 we will discuss the input and output representation for temporal action segmentation.

Model f is the function that performs the mapping from the input to output. The model is parametric and the goal of the learning is to estimate the parameters of this model *i.e.* $f(x; \theta) = y$ where θ is the parameter of the model and y is the estimated output of the model for the input x given the parameters. The model f is usually a deep neural network where the parameters θ are the weights of the neural network model.

Loss function $\mathcal{L}$ is a measure that is designed to be minimized during training. The loss function measures the distance between the estimated output y and the ground truth target $\hat{y}$ as we want the model to predict the ground truth. The loss function takes the output of the model and the ground truth and outputs a scalar value *i.e.* $\mathcal{L}(f(x; \theta), \hat{y}) \in \mathbb{R}$. The lower the loss value the better. During training usually the value of the loss function is estimated from a batch of the training examples $\{x, \hat{y}\}_{i=1,\ldots,B}$ *i.e.* $\frac{1}{B} \sum_{i=1}^{B} \mathcal{L}(f(x_i; \theta), \hat{y}_i))$. Usually, the size of the batch B is much smaller than the size of the training set N. For example in neural networks, the size of the batch is limited by the size of the memory of the hardware used for training. If a *GPU* is used for training the neural network (which is usually the case because of the parallelization benefits it provides) the amount of memory available is a few tens of gigabytes which results in small batch sizes.

Optimization is the algorithm that is used to estimate the parameters of the model such that it minimizes the loss function during training. The optimization takes the form of

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f(x_i, \theta), \hat{y}_i). \tag{2.1}$$

In neural networks, this optimization is usually a variant of stochastic gradient descent (SGD). The loss is estimated over a batch of the training data and the weights are changed in the negative direction of the gradient of the loss with respect to those weights. More details are provided in Section 2.3.2.

During training, the optimization algorithm minimizes the loss function by finding the best parameters. These parameters are then fixed and used for estimating the output for test instances or evaluation. Usually, there is an evaluation metric that is different from the loss function which is used for evaluating the accuracy of the learned model parameters.

2.1.1 Weakly Supervised Learning

Weakly supervised learning is a type of supervised learning where the supervision provided is of a weaker type compared to the full supervision in regular supervised learning. Usually weakly supervised learning is used because it is cheaper to obtain that weaker type of supervision as compared to the full supervision. Formally the target label space has two parts $\hat{y} = (\hat{y}_o, \hat{y}_h)$. $\hat{y}_o$ is the weak supervision and is observed during training while $\hat{y}_h$ is not available during training meaning that the training data is $\{(x, \hat{y}_o)_i\}$. It is important to emphasize that the model f should predict both parts of the label space, the observed and the hidden part. Although weak supervision is beneficial because

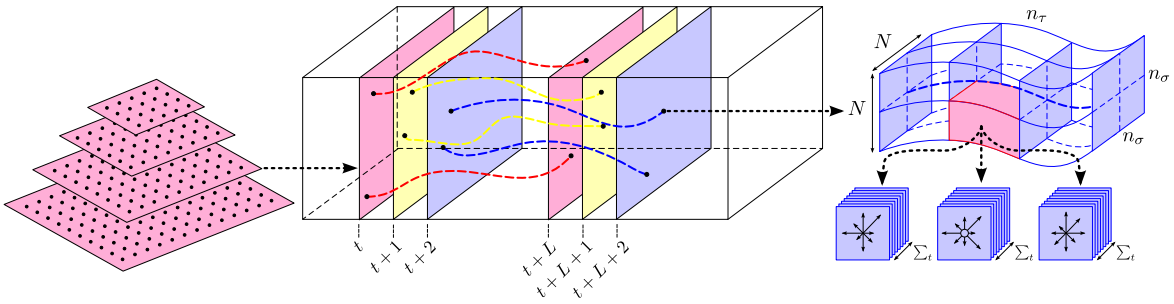


Figure 2.1: Improved Dense Trajectories (IDT) feature extraction pipeline. First feature points are densely sampled at different scales (left). Then using optical flow, the feature points are tracked in subsequent frames constructing the trajectories (middle). Finally different discriptors such has histogram of oriented gradients (HOG) are used to encode the trajectories. The figure is taken from *Wang and Schmid* (2013).

usually, the cost of obtaining the weak supervision is less than full supervision, the loss function and optimization get more complicated and require specific design for each approach. Furthermore, usually the weakly supervised approaches are less accurate compared to fully supervised approaches. We formally define weakly supervised temporal action segmentation in Section 2.4.2.

2.2 Representation

As discussed before, there is a need to represent the input and target of the machine learning model with a vector representation. In temporal action segmentation, the input is an untrimmed video and the output is the action label for each frame of the video. Below we discuss the input and output representation in more detail.

2.2.1 Video Representation

A video with T frames with height H and width W can be represented as a 4 dimensional vector *i.e.* $x \in \mathbb{R}^{T \times H \times W \times 3}$ where 3 is the number of channels in a typical RGB frame. This representation which is similar to the image representation discussed previously is the raw representation as it corresponds to the raw data itself. Although this raw 4D representation is used in action recognition where the number of frames is small (currently usually less than 100) when the number of frames grows to thousands it is no longer possible to use this raw representation. As it is required to have a large model (*e.g.* a large neural network with multiple layers) to process the raw input and extract semantic features from it, the memory of the computational device becomes a limiting factor. For example, using a GPU it will not be possible to store all the intermediate representations from the layers of the network for a long untrimmed video in the GPU memory. As a result, in temporal action segmentation, an intermediate representation of the video is used as input instead of its raw representation. Below we discuss the two main representations currently used in temporal action segmentation.

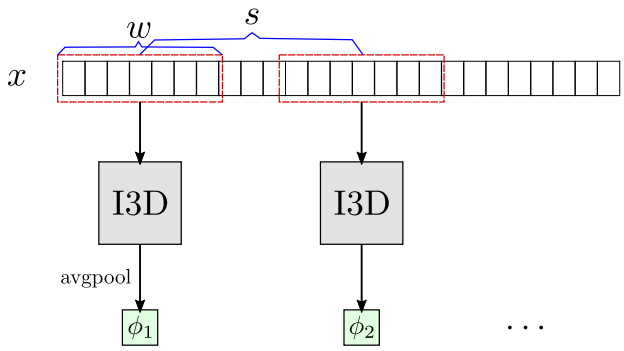


Figure 2.2: Feature extraction from a video x using the I3D (*Carreira and Zisserman*, 2017) network. A window of size w from the frames of the video x is used as an input for the I3D (*Carreira and Zisserman*, 2017) network. The output of the final convolutional layer is used as a feature representation of that window ϕ after removing the spatial and temporal dimensions of the network output using average pooling. This process is repeated for the whole video using in a sliding window fashion with stride s. The final representation for the whole video is the concatenation of the representations of each window ϕ_t.

2.2.1.1 Improved Dense Trajectories (IDT)

Improved Dense Trajectories (IDT) (*Wang and Schmid*, 2013) is a hand-engineered approach for video feature representation. IDT has been proposed for action recognition and at the beginning, it showed promising results. Figure 2.1 visualizes the pipeline involved in extracting IDT features from video. A dense optical flow is calculated and pre-processed for smoothness and camera motion. Then feature points are densely sampled at different scales and tracked using the dense optical flow. Using tracked points of subsequent frames the trajectories are formed. Finally, histograms of oriented gradients (HOG), histogram of optical flow (HOF), and motion boundary histograms (MBH) descriptors are used to encode the trajectories.

In temporal action segmentation, *Kuehne et al.* (2016) proposed the use of IDT features. First PCA is used to reduce the dimensionality of the IDT features to 64. Then Fisher vectors (*Perronnin et al.*, 2010) are used to produce the final representation. Inspired by *Sánchez et al.* (2013) the Fisher vectors are finally power and ℓ_2 normalized. Finally after dimensionality reduction using PCA, This process results in 64 dimensional features for each frame of the video which means that the input video with length T is represented as $x \in \mathbb{R}^{T\times 64}$.

2.2.1.2 Deep Learned Features

Another approach for representing videos is using a pre-trained deep neural network. *Sharif Razavian et al.* (2014) showed that a well pre-trained deep neural network (for example pretrained on the ILSVRC dataset (*Russakovsky et al.*, 2015)) can be used as a feature extractor for many visual recognition tasks. By freezing the weights of the network, the output of the penultimate layer of the network can be used as a feature representation for an input.

A similar approach can be used for extracting features from video. First, a deep neural network

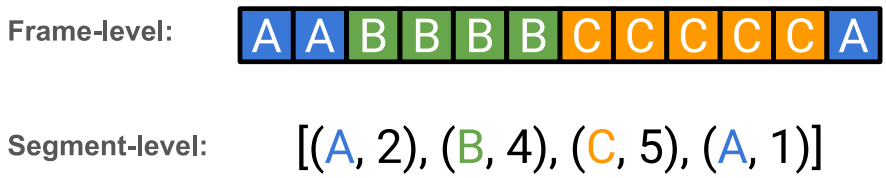


Figure 2.3: Two different representations for temporal action segmentation output. In *frame-level* representation, the output is represented as a list of action labels for each frame. On the other hand, in *segment-level* representation the output is represented as a segment and its corresponding length.

that is trained to perform action recognition is obtained, *e.g.* the I3D network (*Carreira and Zisserman*, 2017) pretrained on the Kinetics dataset (*Kay et al.*, 2017). Then the output of the final convolutional layer of the network is used as a representation as the feature is extracted using a sliding window fashion (Figure 2.2). This results in a representation of the video that depends on the stride of the sliding window and the dimensionality of the neural network output.

The I3D network (*Carreira and Zisserman*, 2017) pretrained on the Kinetics dataset (*Kay et al.*, 2017) for the task of action segmentation is a popular choice for input representation in temporal action segmentation (*Abu Farha and Gall*, 2019; *Souri et al.*, 2021b; *Li et al.*, 2020; *Yi et al.*, 2021). Specifically, both the RGB and optical flow variants of the I3D network are used and the representations from each modality are concatenated. In this case, the video with length T is represented as $x \in \mathbb{R}^{T \times 2048}$ where 2048 is two times 1048 which is the feature dimension of the last convolutional layer of the I3D network.

2.2.2 Temporal Action Segmentation Target Representation

The goal of temporal action segmentation is to predict the action label for each frame of the input video. The obvious way to represent this is as a list of action labels for each frame of the video, *i.e.* $\hat{Y}_{1:T} = [\hat{y}_1, \dots, \hat{y}_T]$, where $\hat{y}_t \in \mathcal{C}$ and $\mathcal{C}$ is the set of action classes and T is the length of the video. This representation is termed *frame-level* representation since each frame is represented (Figure 2.3 top).

In temporal action segmentation, a *segment* is the set of neighboring frames that contain the same action. So the output of temporal action segmentation can be represented as a set of segments and their corresponding lengths as well (Figure 2.3 bottom). Formally, action labels $\hat{Y}_{1:T}$ can also be represented as an ordered sequence of M segments $\hat{S}_{1:M}$ where each segment $\hat{s}_m$ is defined as an action label $\hat{a}_m \in \mathcal{C}$ and its corresponding length $\hat{\ell}_m \in \mathbb{R}_+$. This representation is termed *segment-level* representation since each segment is represented.

2.2.2.1 Frame-level and Segment-level Representation

It is important to note that the two representations for action segmentation, frame-level, and segment-level, are equivalent. From one, the other one can be calculated. This redundancy is the main idea behind our MuCon contribution which will be discussed in Chapter 4. Furthermore, segment-level representation is a more compact, more structured representation for the output of action segmentation. Making predictions using the frame-level representation is more likely to be faced with noisy and over-segmented predictions.

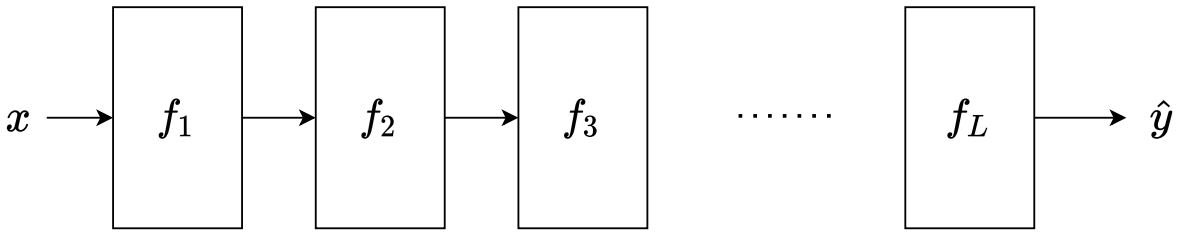


Figure 2.4: A schematic representation of layers of a neural network. Each layer f_l receives the input from the previous layer and transforms it. The final layer generates the output of the model.

2.3 Neural Networks

Neural Networks are currently the most popular type of model used in supervised learning and machine learning in general. They can represent an expressive set of models which can be learned effectively from data. A neural network consists of a set of layers (or building blocks) that are stacked on top of each other (Figure 2.4).

It is useful to not think of neural networks as neurons with connections, but as learnable transformations on a data representation. Each layer f_l, where l is the index of the layer, transforms the data. Most layers have weights which are the parameters of that layer. Imagine a network with L layers. The neural netwok model can then be seen as a composition of all the functions f_l that represent the layers, *i.e.* $\hat{y} = f_L(f_{L-1}(\dots f_1(x)))$.

2.3.1 Building Blocks

Below we will discuss some of the common neural network building blocks used in this thesis.

2.3.1.1 Fully Connected Layer

A fully connected layer in a neural network also known as a linear layer, performs a linear transformation on the data. Given an input $x \in \mathbb{R}^D$ a fully connected layer produces the output as

$$\hat{y} = xA^T + b, \tag{2.2}$$

where $A \in \mathbb{R}^{O \times D}$ is the learnable weights of the transformation and $b \in \mathbb{R}^O$ is the learnable bias. O is the dimensionality of the output, *i.e.* $\hat{y} \in \mathbb{R}^O$.

A fully connected layer can be seen as a layer that takes in D inputs and produces O outputs. The name *fully connected* comes from the fact that every single output is a function of all the inputs *i.e.* as a network each output is connected to all the inputs.

2.3.1.2 Convolutional Layer

A convolutional layer is especially suitable to be used with images. A convolutional layer can be seen as a set of filters that are applied to the input using the convolution operation. Each filter has its weight and bias and the weights and the bias are shared between spatial locations during the filtering operation.

Here we first discuss convolutional layers applied to image data or 2D convolutional layers. It is better to think of the 2D convolutional layer as transforming a 3D tensor input into another 3D tensor

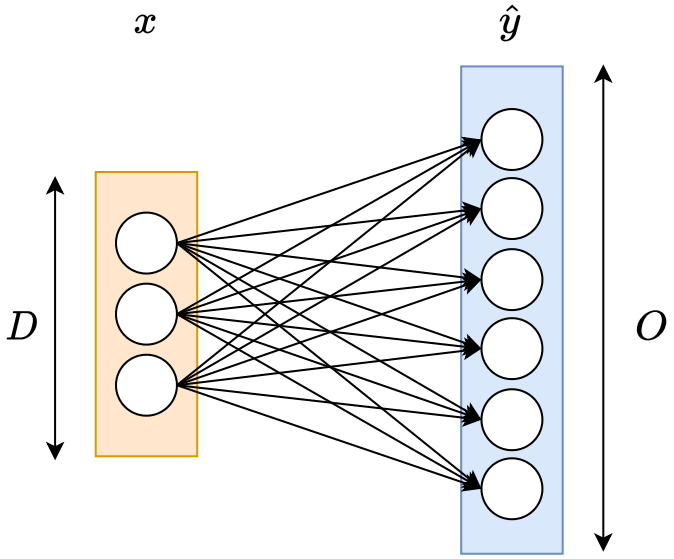


Figure 2.5: A schematic representation of a fully connected layer. A fully connected layer gets an input of size D and produces an output of size O. Each output is a function of every input.

input (Figure 2.6). For example, a regular 3-channel RGB color image can be seen as a 3D tensor where the depth is 3. The depth of the output depends on the number of filters that the convolutional layer has. The width and height of the output depend on the width and height of the input as well as the type of the convolutional layer.

A 2D convolutional layer with a filter size of $k \times k$ and the number of filters D_o, transforms the 3D input $x \in \mathbb{R}^{H_i \times W_i \times D_i}$ into the output $\hat{y} \in \mathbb{R}^{H_o \times W_o \times D_o}$. Each element of the output is calculated using a filtering operation as

$$\hat{y}[h, w, d] = \sum_{k=1}^{D_i} \sum_{i,j=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} x[h-i, w-j, k]\, W[i, j, k, d] + b[d], \tag{2.3}$$

where $W \in \mathbb{R}^{k \times k \times D_i \times D_o}$ is the 4 dimensional weight and $b \in \mathbb{R}^{D_o}$ is the bias of the convolutional layer. This weight sharing filtering operation is visualized in Figure 2.7 for a single channel.

2.3.1.3 Temporal Convolution

Temporal convolution or 1D convolution is a special case of the 2D convolution where the input is only 2 dimensional (*e.g.* width of the input is equal to 1). 1D temporal convolution can be used to process time-series data or in our use case, feature representation of videos.

A 1D convolutional layer with a filter size of k and the number of filters D_o, transforms the 2D input $x \in \mathbb{R}^{L_i \times D_i}$ into the output $\hat{y} \in \mathbb{R}^{L_o \times D_o}$. Each element of the output is calculated using a filtering operation as

$$\hat{y}[l, d] = \sum_{k=1}^{D_i} \sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} x[l-i, k]\, W[i, k, d] + b[d], \tag{2.4}$$

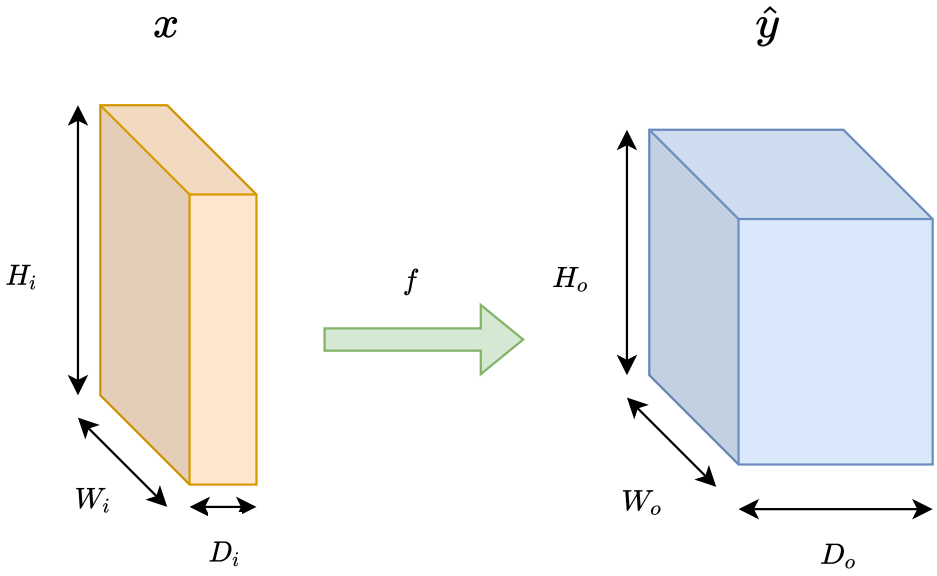


Figure 2.6: A schematic representation of a 2D convolutional input and output. The 2D convolutional layer gets as input a 3D tensor and outputs another 3D tensor with different dimensions. The depth is also known as the number of channels. For a 3-channel RGB color image, the number of channels is 3. The depth of the output depends on the parameters of the convolutional layer. The width and height of the output change depending on the width and height of the input.

Where $W \in \mathbb{R}^{k \times D_i \times D_o}$ is the 3 dimensional weight and $b \in \mathbb{R}^{D_o}$ is the bias of the convolutional layer.

2.3.1.4 Recurrent Layer

Recurrent Layers in neural networks are special types of layers that process sequences of data and output sequences. Because of their design, recurrent layers have memory, therefore previous inputs, as well as the current input, can affect the output.

A *recurrent cell* is the unit that processes the data. A cell takes as input the previous output (if not available it is heuristically provided) and the current input (Figure 2.8 (a)). Given the previous output $\hat{y}_{t-1}$ and the current input x_t, the current output $\hat{y}_t$ is generated using

$$\hat{y}_t = f(W_x x_t + W_y \hat{y}_{t-1} + b), \tag{2.5}$$

where f is a non-linearity function (*e.g.* $tanh$ or sigmoid), W_x and W_y are the weights, and b is the bias. The formulation in (2.5) is known as *Vanilla RNN* cell.

Using a recurrent cell, it is possible to perform sequence processing (Figure 2.8 (b)). During sequence processing using a recurrent cell, the input sequence is transformed into a sequence of outputs with the same length. It is important to note that the output at each time step is a function of the input at the current time step and all the inputs from the previous time steps. In other words, the recurrent cell has a memory mechanism to remember all the previous inputs when producing the current output. This makes recurrent layers suitable for sequence modeling tasks. Researchers have shown despite the theoretical ability of the RNN cell to perform long-term dependency modeling

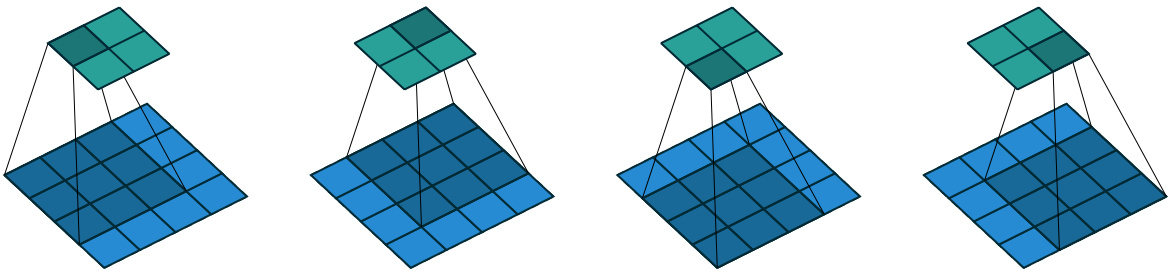


Figure 2.7: Visualization of a 2d convolutional operation for a single channel. The input (shown in blue) is filtered with a shared kernel (shown in dark blue) to produce the output (shown in green). The figure is taken from *Dumoulin and Visin* (2016).

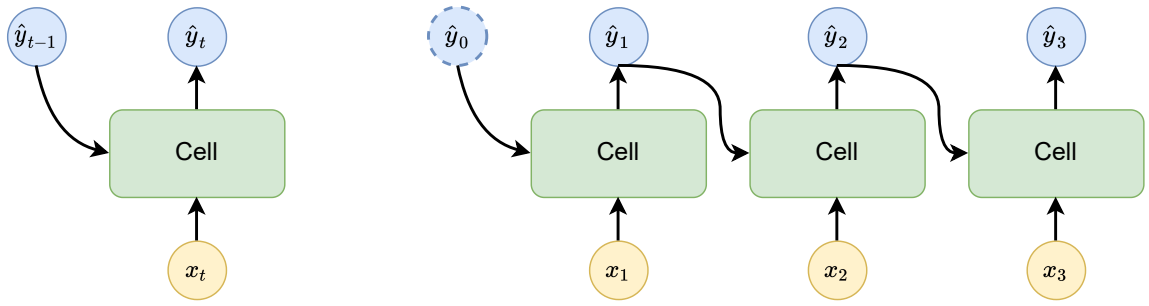


Figure 2.8: (a) Visualization of a Recurrent cell. (b) Visualization of sequence processing using a Recurrent cell.

in sequences, the vanilla formulation suffers from vanishing or exploding gradients during training (*Bengio et al.*, 1994). To address these issue, other forms of recurrent cells has been proposed which we will discuss below.

Long Short-Term Memory (LSTM) The Long Short-Term Memory (LSTM) recurrent cell was introduced by *Hochreiter and Schmidhuber* (1997) to address the vanishing gradient problem of the vanilla RNNs and make it more suitable for learning long-term dependencies. The main idea in LSTM is to have to maintain a *state* alongside the output. The other novelty in LSTM is the use of gating for reading from and writing to the outputs and states. The LSTM cell gets as input the previous output $\hat{y}_{t-1}$, the previous state c_{t-1}, and the current input x_t to produce the current output $\hat{y}_t$ and the current state c_t. LSTM uses the following formulas

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{yi}\hat{y}_{t-1} + W_{ci}c_{t-1} + b_i), \\
f_t &= \sigma(W_{xf}x_t + W_{yf}\hat{y}_{t-1} + W_{cf}c_{t-1} + b_f), \\
o_t &= \sigma(W_{xo}x_t + W_{yo}\hat{y}_{t-1} + W_{co}c_t + b_o), \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{yc}\hat{y}_{t-1} + b_c), \\
\hat{y}_t &= o_t \tanh(c_t),
\end{aligned}
\tag{2.6}
$$

where i_t, f_t, o_t are the input, forget and output gates respectively, W_{**} and b_* are the learnable biases and σ is the sigmoid activation function.

The value of each gate depends on the previous output, previous state, and current input. The value of the forget gate determines how much of the previous state is added to the current state. The value of the input gate determines how much of the previous output and the current input should be added to the current state. The value of the output gate determines the output of the state.

Gated Recurrent Unit (GRU) The Gated Recurrent Unit (GRU) introduced by *Cho et al.* (2014) is another recurrent cell that is inspired by LSTM and has fewer parameters and simpler internals. The GRU has only two cells. The GRU update rule is defined by the following formulas

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z \hat{y}_{t-1} + b_z), \\ r_t &= \sigma(W_r x_t + U_r \hat{y}_{t-1} + b_r), \\ c_t &= tanh(W_h x_t + U_h(r_t \odot \hat{y}_{t-1}) + b_h), \\ \hat{y}_t &= (1 - z_t) \odot \hat{y}_{t-1} + z_t \odot c_t, \end{aligned} \tag{2.7}$$

where z_t and r_t are the gates, W_*, U_*, and b_* are the learnable weights and biases, and $\odot$ is the element-wise multiplication. The c_t in GRU is known as the candidate state.

2.3.1.5 Softmax

Softmax is a layer used for classification networks to produce categorical distributions. It takes as input the unnormalized log probabilities which are produced by the network and transforms (projects) them into probabilities of a categorical distribution *i.e.* each element is between $[0, 1]$ and the sum of all elements is equal to 1.

Given the D dimensional input $x \in \mathbb{R}^D$, element i of the Softmax output is calculated using

$$\hat{y}[i] = \frac{e^{x[i]}}{\sum_{j=1}^{D} e^{x[j]}}. \tag{2.8}$$

2.3.1.6 Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU) is a parameter-less layer used to introduce non-linearity into the network. Each element of the output $\hat{y}$ of the ReLU layer is calculated using

$$\hat{y}[i] = max(0, x[i]) \tag{2.9}$$

from the input x. This simple operation sets the value into zero if the value is negative otherwise keeps it unchanged.

2.3.1.7 Cross Entropy Loss Function

In supervised learning problems that involve classification, the loss function used is the Cross Entropy. Cross Entropy loss function tries to maximize the probability of the correct class in a 1-vs-N classification problem *i.e.* where only 1 of the N classes is correct.

Given the estimated probability for each of the D classes $x \in [0,1]^D$, and the target class $y \in \{1,\dots,D\}$, the cross entropy loss function is calculated using the following formula

$$\mathcal{L} = -log(x[y]), \tag{2.10}$$

which is the negative log probability of the target class. Please note that the cross entropy is always a positive scalar value.

2.3.2 Network Training

In the previous sections, we discussed some of the building blocks of neural networks. In this section, we will describe how to optimize the parameters of the neural network model using the gradient descent optimization algorithm. To achieve this, we need to estimate the gradient of the loss function with respect to each parameter. Thus, we also describe the backpropagation algorithm for the gradient estimation.

Gradient descent is a well-known optimization algorithm for neural network models. Gradient descent is based on changing the parameters of the model in the negative direction of the gradients of the loss function with respect to that parameter.

At the beginning of training, the model parameters θ are initialized randomly (usually from a Gaussian distribution with a specific standard deviation). At each step of the optimization, the loss value $\mathcal{L}$ is calculated from a batch of the training data and using the current value of the parameters θ_t. Then the parameters are updated by calculating the gradient of the loss with respect to the parameters $\nabla_\theta \mathcal{L}$ and by taking the gradient descent step

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta \mathcal{L}, \tag{2.11}$$

where γ is the optimization step size or *learning rate*. This optimization technique is known as *Stochastic Gradient Descent* (SGD) since the gradients are estimated only from a batch of the training data. In SGD the choice of the learning rate is crucial and can have a great impact on the convergence of the network and its generalization ability (*Keskar and Socher*, 2017). Other variants of SGD like Adam (*Kingma and Ba*, 2015) are proposed to overcome this problem by adaptively changing the learning rate for each parameter depending on the variance of the gradient of that parameter.

The gradients of the loss with respect to the parameters are calculated using the *backpropagation* algorithm. Backpropagation uses the chain rule to estimate the gradient for the parameters of the intermediate layers. This is only possible if it is possible to estimate the gradients at each layer of the neural network. Usually, this is the case because the layers are differentiable functions (*e.g.* Softmax layer or fully connected layer). In non-differentiable cases (*e.g.* ReLu layer) usually, the layer is piece-wise differentiable.

2.4 Temporal Action Segmentation

In this section, we discuss the temporal action segmentation problem, and various types of supervision typically used for learning temporal action segmentation models and evaluation metrics.

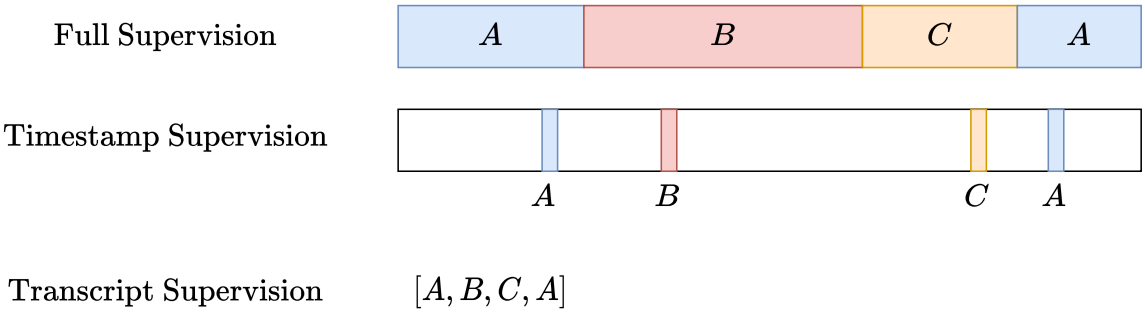


Figure 2.9: Visualization of the different types of temporal action segmentation supervision. The first row visualizes the full supervision, which is the case that the action label for each frame is known. The second row visualizes the timestamp weak supervision. In this case, only a single frame is annotated for the action segments. The final row visualizes the transcript weak supervision. In this setting, only the ordered list of actions that occur in the video is known and no information is available regarding the temporal boundary and size of the action segments.

2.4.1 Problem Statement

Temporal action segmentation is the task of predicting the action class for each frame of a video. More formally, given an input sequence of T D-dimensional frame-level features $X_{1:T} = [x_1, \ldots, x_T]$, $x_t \in \mathbb{R}^D$, the goal is to predict the output sequence of frame-level action labels $\hat{Y}_{1:T} = [\hat{y}_1, \ldots, \hat{y}_T]$, where $\hat{y}_t \in \mathcal{C}$ and $\mathcal{C}$ is the set of action classes. The feature representation of the input video is discussed in Section 2.2.1.

The frame-level action labels $\hat{Y}_{1:T}$ can also be represented as an ordered sequence of M segments $\hat{S}_{1:M}$ where each segment $\hat{s}_m$ is defined as an action label $\hat{a}_m \in \mathcal{C}$ and its corresponding length $\hat{\ell}_m \in \mathbb{R}_+$. The sum of the lengths is equal to the length of the video *i.e.* $\sum_{m=1}^{M} \hat{\ell}_m = T$.

2.4.2 Supervision

In this thesis, we utilize both full supervision and two types of weak supervision (transcript and timestamps) for temporal action segmentation (Figure 2.9).

Full supervision is the setting where the action label for each frame is known. Full supervision target provides the full information for calculating the loss during training, hence the approaches that utilize full supervision are the most straightforward. But it is considerably expensive to obtain full supervision for long untrimmed videos. Furthermore, the boundary of the actions are sometimes subjective and could result in annotation inconsistencies.

Transcript supervision is the setting where only the ordered list of actions is provided as the target and no information is known regarding the duration of the actions and their temporal position. As shown in Figure 2.10, transcript supervision is ambiguous regarding the temporal information, since all the action segmentations visualized in Figure 2.10 have the same transcript $[A, B, C, A]$. Transcript supervision is considerably less expensive to obtain compared to full supervision.

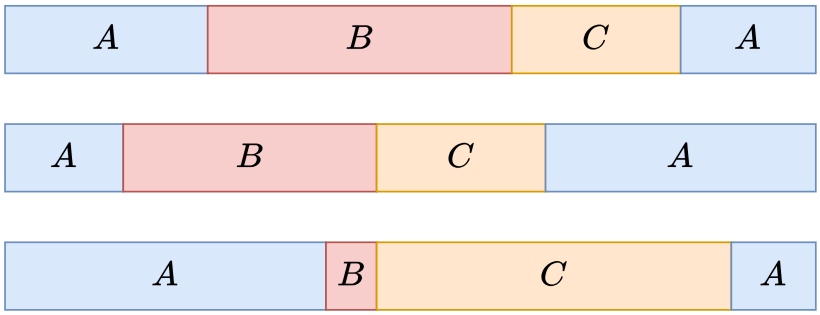


Figure 2.10: Visualization of the ambiguity in the transcript weak supervision for temporal action segmentation. All of the action segmentations visualized have the same transcript.

Timestamp supervision is the setting where a single frame is annotated for each action segment. In this setting, some information regarding the temporal location of the actions is available. Although timestamp supervision is only marginally more expensive to obtain compared to transcript supervision (*Ma et al.*, 2020), experiments have shown that it results in much more accurate temporal action segmentation models (*Li et al.*, 2021).

2.4.3 Evaluation Metrics

Below we define the details of how the evaluation metrics are calculated for temporal action segmentation in this dissertation.

2.4.3.1 Mean over Frames Accuracy (MoF) or Accuracy

The mean over frames accuracy (MoF) (*Abu Farha and Gall*, 2019; *Kuehne et al.*, 2016; *Richard et al.*, 2017, 2018b) is defined as the number of frames with correctly predicted labels divided by the total number of frames. Given a set of predictions $Y^{(j)}$ and ground truth labels $\hat{Y}^{(j)}$, where $j = 1, \ldots, V$ and V is the number of videos in the test set, the MoF is defined as

$$\text{MoF} = \frac{\sum_{j=1}^{V} \sum_{t=1}^{T^{(j)}} \mathbb{I}(Y_t^{(j)} = \hat{Y}_t^{(j)})}{\sum_{j=1}^{V} T^{(j)}} \tag{2.12}$$

where $T^{(j)}$ is the length of the jth video and $\mathbb{I}$ is the indicator function.

MoF-BG is similar to MoF, but it only considers the ground truth frames which are not annotated as background (BG).

2.4.3.2 Matching Score

The matching score (*Lambert*, 2019) measures the similarity of the predicted transcript compared to the ground truth transcript:

$$\text{Matching Score} = 2 \times \frac{\text{number of matches}}{|A| + |\hat{A}|} \tag{2.13}$$

where $|A|$ is the length of the predicted transcript, $|\hat{A}|$ is the length of the ground truth transcript, and *number of matches* refers to the number of matches found after applying the Levenshtine algorithm. The matching score penalizes over-segmentation errors but does not consider the temporal location of the segments.

2.4.3.3 Intersection Over Detection (IoD)

IoD is only defined for the alignment task since the transcript and number of segments are known for this task. This provides a one-to-one matching between the predicted and ground truth segments. The IoD is then computed by the average intersection of a ground truth segment with an associated predicted segment divided by the length of the predicted segment, i.e., $|S \cap \hat{S}|/|S|$. We use the same definition and code as (*Ding and Xu*, 2018; *Li et al.*, 2019).

2.4.3.4 F1 Score

F1 Score is similar to the matching score since it is calculated over segments, but it takes the temporal position of the predicted and ground truth segments into consideration. On the other hand, the F1 score is not sensitive to small deviations of the temporal boundaries. F1 Score is usually evaluated at an intersection over union (IoU) threshold k (*e.g.* $10\%, 25\%, 50\%$). Given a predicted segment and ground truth segment, the IoU is calculated as

$$IoU = \frac{predicted\ segment \cap ground\text{-}truth\ segment}{predicted\ segment \cup ground\text{-}truth\ segment}. \tag{2.14}$$

First, a mapping is found between the ground truth segments and predicted segments based on their IoUs. Each ground truth segment is mapped to the predicted segment with the highest IoU. If a predicted segment has the same label as the mapped ground-truth segment and the IoU ratio is higher than k, then it is considered a true positive (TP). Otherwise, it is considered a false positive (FP). If any ground-truth segment is not mapped to any predicted segment, it is considered a false negative (FN).

To calculate the F1 score for a threshold of k, first the precision and recall are calculated as

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN}, \end{aligned} \tag{2.15}$$

and finally the F1 score is calculated as

$$F1@k = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{2.16}$$

2.5 Datasets

2.5.0.1 Breakfast Dataset

The Breakfast dataset (*Kuehne et al.*, 2014) contains more than 1.7k videos of different cooking activities. The videos contain 10 different types of breakfast activities such as *prepare cereal* or *prepare*

coffee which consists of 48 different fine-grained actions. The dataset is recorded in a multi-view setting where the cocking activities are captured from multiple views using different synchronized cameras. The use of synchronized recordings reduced the cost of annotation considerably. The average length of the videos in the Breakfast dataset is 2.3 minutes while the longest video is over 10 minutes and the shortest video is just over 12 seconds. On average, each video in the dataset contains 6.9 segments. In our experiments, we follow the 4 train/test splits provided with the dataset and report the average.

2.5.0.2 50Salads Dataset

The 50Salads dataset (*Stein and McKenna*, 2013) consists of 50 videos of people preparing mixed salads. The average length of the videos is 6.4 minutes which is considerably longer than the Breakfast dataset. The dataset provides fine-granular annotations of 17 low level activities such as *peel cucumber* and *mix ingredients*. The dataset recordings provide accelerometer data and depth images which are not used in this thesis. For evaluation, we use five-fold cross-validation and report the average.

2.5.0.3 Hollywood Extended Dataset

The Hollywood extended dataset (*Bojanowski et al.*, 2014) contains 937 video sequences taken from Hollywood movies. The videos contain 16 different action classes such as *answer phone* or *drive car*. The non-background actions in this dataset are more sparse compared to the Breakfast dataset and the 50Salads datasets. In our experiments, we follow the train/test split strategy of (*Richard et al.*, 2017, 2018b; *Li et al.*, 2019).

2.5.0.4 Georgia Tech Egocentric Activities (GTEA)

The Georgia Tech Egocentric Activities (GTEA) dataset (*Fathi et al.*, 2011) contains 28 egocentric videos recorded in a laboratory setting by 4 subjects. The subjects perform 7 different activities such as preparing coffee or making a sandwich. The videos contain 11 action classes including the background class. We use this dataset for the task of action localization using timestamp supervision in Section 6.3.9. We use the timestamp annotations and splits provided by *Ma et al.* (2020).

2.5.0.5 Bristol Egocentric Object Interactions Dataset (BEOID)

The Bristol Egocentric Object Interactions Dataset (BEOID) (*Damen et al.*, 2014) contains 58 egocentric videos recorded in 6 different environments. The dataset was originally introduced for the task of object discovery. The dataset was later adapted for action recognition from timestamp supervision with 34 action classes. We use this dataset for the task of action localization using timestamp supervision in Section 6.3.9. We use the timestamp annotations and splits provided by *Ma et al.* (2020).

CHAPTER 3

Literature Review

In this chapter, we provide a literature review of the temporal action segmentation field. We first discuss action recognition approaches. Then we overview fully supervised, weakly supervised, and unsupervised approaches for temporal action segmentation. We further review the approaches that have utilized timestamp supervision in action recognition and temporal action localization and segmentation. Finally, we discuss some of the work on sequence-to-sequence learning which is related to MuCon (Chapter 4), and energy-based inference which is related to FIFA (Chapter 5).

Contents

3.1 Action Recognition 25
3.2 Temporal Action Segmentation 26
3.2.1 Fully Supervised Temporal Action Segmentation 26
3.2.2 Weakly Supervised Temporal Action Segmentation 29
3.3 Timestamp Supervision for Recognizing Activities 30
3.4 Sequence to Sequence Learning 31
3.5 Energy-Based Inference 31

3.1 Action Recognition

Action recognition involves classifying the action label for a given short video clip. However, real-world videos often tend to be long, untrimmed, and consist of multiple actions occurring over extended durations. Conversely, as discussed in Section 2.2.1.2, neural networks have proven to learn highly effective representations for video clips using action recognition datasets. Hence, in this section, we present an overview of the scientific literature of action recognition.

Early work in action recognition was focused on classifying actions performed by humans using hand-crafted features (*Schuldt et al.*, 2004; *Wang and Schmid*, 2013). The success of Deep Neural Networks resulted in a paradigm shift and the performance of action recognition approaches improved considerably by adopting the use of neural networks (*Simonyan and Zisserman*, 2014; *Wang et al.*, 2019; *Carreira and Zisserman*, 2017).

Deep learning based action recognition systems that utilize convolutional neural networks (CNN) can be grouped into 2D and 3D CNN approaches. 2D CNN approaches use a 2D convolutional neural network which utilized 2D convolutions (Section 2.3.1.2) and 3D CNN approaches use 3D convolutions. TSN (*Wang et al.*, 2019) samples a fixed number of snippets (frames) from the input video clip and processes each snippet using a 2D convolutional layer that estimates the log probabilities (logits) for action classes. Then the final prediction of the network is the average of the logits. Note that TSN does not take the temporal order of the frames into account. To address this issue, TRN (*Zhou*

et al., 2018a) and TSM (*Lin et al.*, 2019) were proposed to add temporal modeling capabilities for 2D CNN based action recognition approaches. TRN and TSM considerably outperformed TSN in action recognition tasks that require more temporal modeling, like the Something-Something dataset (*Goyal et al.*, 2017).

3D CNN based approaches on the other hand take as input multiple frames and process them using 3D convolutional layers. The 3D CNN networks are more computationally expensive as compared to 2D counterparts. *Carreira and Zisserman* (2017) proposed to initialize a 3D CNN by inflating a 2D CNN pretrained on image classification which resulted in the I3D approach which significantly outperformed other approaches. Further architectural advances such as Nonlocal Blocks (*Wang et al.*, 2018), Channel Separable Convolution (*Tran et al.*, 2019), and SlowFast (*Feichtenhofer et al.*, 2019) showed promising results for 3D CNN based action recognition.

Another approach for temporal modeling in convolutional neural network based approaches is using optical flow input modality instead of RGB frames. Action recognition approaches that utilize optical flow, typically train a second similar network that takes as input multiple consecutive 2 channel optical flow estimates (*Simonyan and Zisserman*, 2014; *Wang et al.*, 2019; *Carreira and Zisserman*, 2017). Since optical flow captures motions more explicitly it was shown to be beneficial in some approaches. But recent approaches have stopped using the optical flow for action recognition in favor of advanced architectures (*Feichtenhofer et al.*, 2019; *Arnab et al.*, 2021; *Feichtenhofer*, 2020) or more sophisticated pretraining (*Qian et al.*, 2021; *Feichtenhofer et al.*, 2021).

Recently with the success of Transformer (*Vaswani et al.*, 2017) based architectures in image classification (*Dosovitskiy et al.*, 2021; *Liu et al.*, 2021) action recognition approaches have achieved remarkable results using transformer based architectures (*Arnab et al.*, 2021; *Bertasius et al.*, 2021). The current state-of-the-art approach of *Yan et al.* (2022) achieves remarkable top-1 classification accuracy of 89% between 400 classes of the Kinetics dataset (*Kay et al.*, 2017). The success of transformers has resulted in unified architectures for both video and image classification which is a remarkable achievement (*Fan et al.*, 2021; *Li et al.*, 2022).

3.2 Temporal Action Segmentation

In the previous section, we discussed approaches for understanding trimmed video clips also known as action recognition. In this section, we review the literature related to temporal action segmentation in untrimmed videos. We organize this section according to the level of supervision used during training.

3.2.1 Fully Supervised Temporal Action Segmentation

Fully supervised approaches rely on frame-level annotations during training. Earlier methods in this setup applied a sliding window approach with non-maximum suppression (*Rohrbach et al.*, 2012; *Karaman et al.*, 2014). To achieve good performance, these models had to be evaluated at multiple different scales which made them computational expensive. These early approaches' performance was limited as they did not model the long-range temporal dependency between the video frames and actions.

There have been multiple attempts at long-range temporal modeling for untrimmed video understanding. Some approaches employed context-free grammars (*Vo and Bobick*, 2014; *Pirsiavash and*

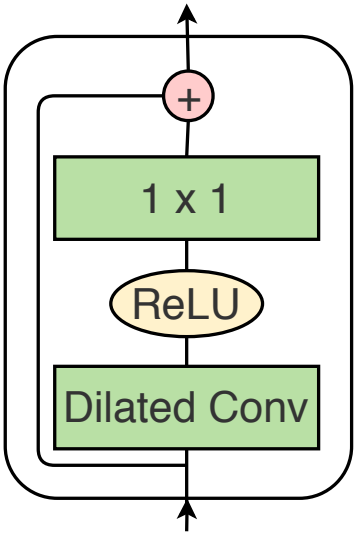


Figure 3.1: Overview of a dilated residual block in the MS-TCN architecture. The figure is taken from *Abu Farha and Gall* (2019).

Ramanan, 2014) while others used hidden markov models (HMMs) (*Lea et al.*, 2016; *Kuehne et al.*, 2016, 2020) to capture long-range dependencies. *Kuehne et al.* (2020) showed that incorporating a hybrid approach of RNNs and HMMs achieves strong results for temporal action segmentation. While the RNN was used to model short-term temporal dependencies inside each action segment, they used HMMs for long-term temporal dependencies between multiple action segments.

Temporal or 1D convolutional networks have been extensively used for temporal action segmentation. *Lea et al.* (2017) proposed the use of temporal (1D) convolutional networks for temporal action segmentation. To deal with the long-range nature of the data, they proposed to use an encoder-decoder architecture where the input's temporal size is reduced using pooling in the encoder and the output is produced in the decoder using upsampling layers. Built on top of the encoder-decoder architecture, *Lei and Todorovic* (2018) proposed to use deformable convolutions and *Ding and Xu* (2018) proposed to add lateral connections to the encode-decoder architecture at each temporal resolution. Both of these modifications demonstrated improved performance compared to *Lea et al.* (2017).

Abu Farha and Gall (2019) showed that convolutional networks with exponentially increasing dilations are exceptionally good at temporal modeling of the long-range dependencies, eliminating the need for encoder-decoder architectures. Furthermore, they proposed a multi-stage architecture (MS-TCN) that proved to be very effective in reducing over-segmentation errors that are typical for temporal convolutional networks employed for temporal action segmentation. This multi-stage architecture remains a cornerstone of most of the follow-up works on temporal action segmentation.

In Chapter 4 and 6 of our work, we have utilized the MS-TCN approach as a basis. To provide a comprehensive understanding of the MS-TCN backend, we present a detailed overview in this section. The network architecture of MS-TCN consists of multiple stages, and each stage comprises several dilated residual blocks (see Figure 3.1). The first layer within each block applies a non-causal temporal dilated convolution with a kernel size of 3, followed by a ReLU non-linearity. The output of the block is generated through a 1×1 temporal convolution with a residual connection. By employing exponentially increasing dilation factors, each stage incorporates several (typically 10) dilated

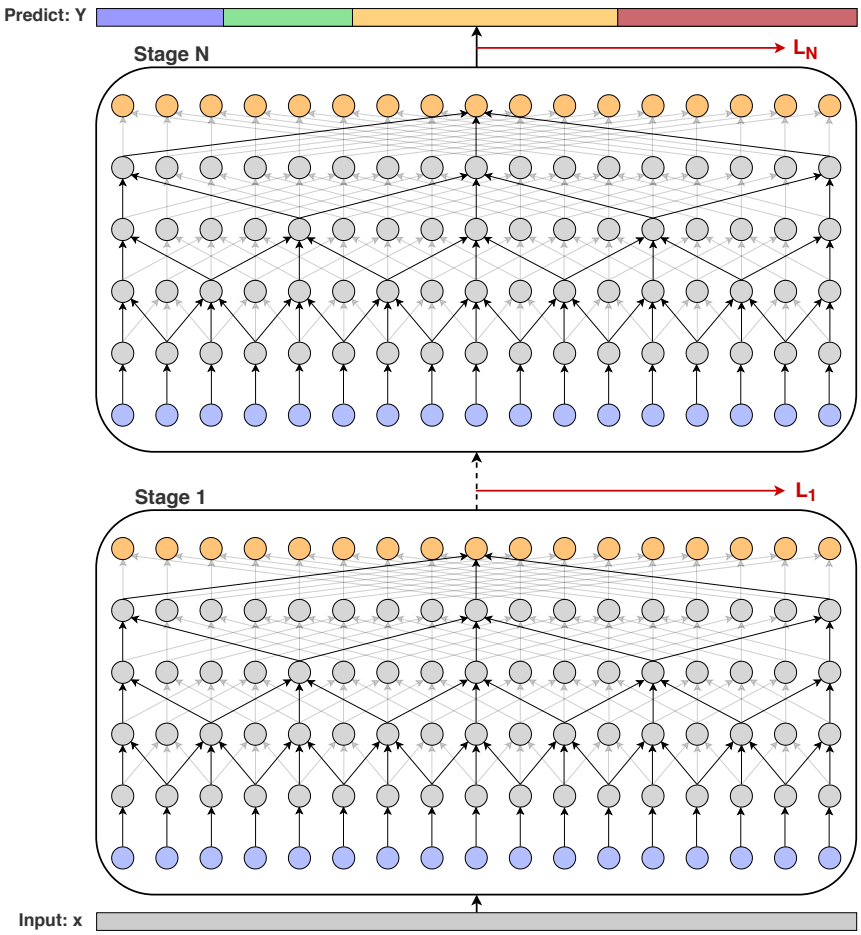


Figure 3.2: Overview of a the full MS-TCN network architecture, consisting of multiple stages of several dilated residual convolutional blocks. The figure is taken from *Abu Farha and Gall* (2019).

residual blocks, resulting in a significant receptive field for temporal modeling (see Figure 3.2). This architectural design helps to effectively capture temporal dependencies. To prevent over-segmented outputs, the complete MS-TCN architecture typically consists of multiple stages (usually 4).

Li et al. (2020) proposed an improvement on MS-TCN (*Abu Farha and Gall*, 2019) where the first stage includes both a dilation increasing and a dilation decreasing pathways. This two-pathway network architecture increases the receptive field of the initial layers of the first stage which results in improved performance. As mentioned, the temporal receptive field of each layer of MS-TCN (*Abu Farha and Gall*, 2019) is mainly controlled by the dilation factor of the convolutional layers. *Gao et al.* (2021) proposed that it is possible to find a combination of these receptive fields for different layers using architecture search and demonstrated significant performance improvements over MS-TCN.

Although MS-TCN reduced the amount of over-segmentation by using a multi-stage architecture, it could not guarantee that at test time the output would not be over-segmented. Some works tried to specifically address this issue. *Ishikawa et al.* (2021) introduced the Action Segment Refinement Framework (ASRF) which incorporates a boundary regression branch. The goal of the boundary regression branch is to predict the location of the boundary of action segments. Similarly, *Wang et al.* (2020) proposed a cascade network for action segmentation alongside a boundary prediction network. In this line of work, the boundaries are then used to refine the predictions of the temporal action segmentation network using pooling or majority voting approaches. In Chapter 5 we propose a fast inference approach which refines the action segmentation boundaries and prevents over-segmentation in a more systematic manner.

Recently multiple attempts were made to incorporate transformer (*Vaswani et al.*, 2017) based architectures for temporal action segmentation. *Yi et al.* (2021) proposed ASFormer which incorporates a multi-head attention (*Vaswani et al.*, 2017) into each block of the multi-stage network where each block is similar to the MS-TCN block and it includes the convolutional layers for efficiency and parameter sharing. *Wang et al.* (2022) proposed another architecture for temporal action segmentation using transformers.

The current state of the art approach for fully supervised temporal action segmentation is the work of *Singhania et al.* (2021). They propose multiple tricks to improve the performance of a base temporal action segmentation network. Specifically, they incorporate augmentation during training and testing alongside ensembling prediction from multiple input resolutions that are produced using various sampling sizes.

3.2.2 Weakly Supervised Temporal Action Segmentation

Despite the success of fully supervised action segmentation approaches, they rely on dense frame-level annotations. As acquiring such annotations is time-consuming and expensive, many approaches have been proposed to train action segmentation models using weaker forms of supervision. A popular form of weak supervision is transcripts which is the focus of our contribution in Chapter 4. In transcript supervision, only the ordered list of actions occurring in the videos is provided. Initial attempts in this setup used discriminative clustering (*Bojanowski et al.*, 2014) to localize action segments in Hollywood movies. *Huang et al.* (2016) extended the Connectionist Temporal Classification (CTC) loss function to enforce frame-to-frame similarity in CTC. *Chang et al.* (2019) proposed to use the soft-dtw (*Cuturi and Blondel*, 2017) loss function with positive and negative transcript mining to

learn the matching between the frames and the class labels. Our work in Chapter 4 introduces a new loss function (named MuCon loss) that is conceptually similar to soft-dtw, since it is differentiable and aligns the transcript to the predicted framewise probabilities.

One key idea in weakly supervised action segmentation is generating pseudo ground truth and then training the network using the generated pseudo ground truth as full-supervision. *Ding and Xu* (2018) proposed ISBA, an approach to initially generate the pseudo ground truth using equal length for each action segment and then refining the boundary of the action using the prediction of the network after each epoch of training. *Kuehne et al.* (2017) proposed to use an HMM framework to generate the pseudo ground truth after each epoch of training. *Richard et al.* (2017) further incorporated an RNN into the HMM framework to refine and extend the temporal modeling capabilities during pseudo ground truth generation.

While ISBA and the HMM frameworks for pseudo ground truth generation update the ground truths only after each epoch of training, *Richard et al.* (2018b) proposed Neural Network Viterbi (NNV) to allow for on-the-fly pseudo ground truth generation. NNV proposed to use an HMM framework with explicit length modeling. *Richard et al.* (2018b) showed that explicit length modeling is crucial in obtaining good performance. *Li et al.* (2019) proposed CDFL which extended the NNV approach to incorporate the use of hard negative mining and length sampling during pseudo ground truth generation. Our work in Chapter 4 can be seen as a differentiable way to generate pseudo ground truth for weakly supervised temporal action segmentation training. But unlike the HMM framework, NNV, or the CDFL approaches, our work does not require multiple temporal action segmentation alignments at test time and thus faster.

Another form of supervision that has recently emerged is based on action sets (*Richard et al.*, 2018a; *Fayyaz and Gall*, 2020; *Li and Todorovic*, 2020, 2021). In contrast to transcripts, set supervision provides only the set of actions that occur in the videos without any information regarding the order or how many times each action occurs. However, the performance of these approaches is still inferior compared to the transcript supervision.

3.3 Timestamp Supervision for Recognizing Activities

Timestamp supervision has just recently been used for the action segmentation task *Li et al.* (2021) and has shown competitive results compared to the full supervision. For timestamp supervision, *Li et al.* (2021) proposed to annotate only a single frame from each action segment. To train an action segmentation model with this level of supervision, the model output and the timestamps are used to generate pseudo ground truth labels. While this approach has been effective, it assumes that all the segments are annotated. In Chapter 6 we build on the work of *Li et al.* (2021) and proposed an improved training technique which results in a model that is more robust in realistic annotation scenarios.

Timestamp supervision has also been used recently in other activity understanding tasks. *Moltisanti et al.* (2019) proposed to sample frames around the annotated timestamps to train an action classifier based on a sampling function that is fitted to the classifier response. In the context of action localization, *Ma et al.* (2020) proposed a mining strategy to sample action frames and background frames for training. In *Lee and Byun* (2021), dense labels are generated to encourage complete prediction of the action segments.

3.4 Sequence to Sequence Learning

In Chapter 4, we propose MuCon, a novel approach for weakly supervised action segmentation. The proposed network architecture in MuCon, consists of two branches, framewise prediction branch and a segment-wise prediction branch. The segment-wise prediction branch, incorporates a sequence to sequence prediction network. There has been a lot of work in sequence to sequence learning, mostly in natural language processing (*Sutskever et al.*, 2014; *Bahdanau et al.*, 2015; *Gehring et al.*, 2017; *Vaswani et al.*, 2017). However, our problem is different in one major aspect: the size mismatch between the video features and transcripts is often large. In the Breakfast dataset (*Kuehne et al.*, 2014), for example, the average input video length is around 2100 frames and the longest video has around 9700 frames, while the average transcript length is 6.8 and the longest is 25. Because of this issue, we have specially designed our network to be able to temporally model such long sequences. *Wei et al.* (2018) proposed a sequence to sequence model for detecting temporal segments of interest, but not the action classes. We incorporate simple LSTM-based sequence-to-sequence network with attention similar to *Bahdanau et al.* (2015) for the segment-wise prediction branch of the MuCon network.

3.5 Energy-Based Inference

In Chapter 5, we propose FIFA, an approximate approach for temporal action segmentation inference that can be seen as a gradient based energy minimization. Similarly, in energy-based inference methods, gradient descent is used at inference time as described in *LeCun et al.* (2006). The goal is to minimize an energy function that measures the compatibility between the input variables and the predicted variables. This idea has been exploited for many structured prediction tasks such as image generation (*Gatys et al.*, 2015; *Johnson et al.*, 2016), machine translation (*Hoang et al.*, 2017) and structured prediction energy networks (*Belanger et al.*, 2017). *Belanger and McCallum* (2016) relaxed the discrete output space for multi-label classification tasks to a continuous space and used gradient descent to approximate the solution. Gradient-based methods have also been used for other applications such as generating adversarial examples (*Goodfellow et al.*, 2015) and learning text embeddings (*Le and Mikolov*, 2014).

In contrast to the previous works mentioned above, in FIFA (Chapter 5) the energy function is not learned but designed for the temporal action segmentation inference task following the proposed objective of NNV (*Richard et al.*, 2018b) in a differentiable manner.

CHAPTER 4

Mutual Consistency for Action Segmentation

This chapter is based on the following publication:
Fast Weakly Supervised Action Segmentation Using Mutual Consistency
Yaser Souri*, Mohsen Fayyaz*, Luca Minciullo, Gianpiero Francesca, Juergen Gall
IEEE Transactions on Machine Intelligence and Pattern Analysis (PAMI), 2021.

Below we will summarize the contributions of each author.

- **Yaser Souri and Mohsen Fayyaz**
  Both contributed equally to this work. Yaser proposed the two-branch network architecture and the consistency loss. Mohsen formulated the final version of the MuCon loss which performed better than the alternative possibilities. Yaser and Mohsen both contributed significantly to the implementation and experiments for the weakly supervised setting on the Breakfast dataset. Mohsen obtained the results on the Hollywood dataset. Yaser performed the fully supervised and mixed supervised implementation and experiments. Yaser performed the reproducibility studies of the previous works. Yaser produced most of the qualitative figures and visualization. Yaser and Mohsen both contributed considerably to the writing.
- **Luca Minciullo and Gianpiero Francesca**
  Both contributed with supervision, discussions, and writing. Gianpiero performed the statistical test.
- **Juergen Gall**
  Juergen supervised the project. He contributed with discussion and writing.

Contents

4.1 Introduction 34
4.2 Weakly Supervised Action Segmentation 36
4.3 Proposed Method 36
4.3.1 Temporal Backbone 38
4.3.2 Frame Classification Branch 38
4.3.3 Segment Generation Branch 38
4.3.4 Mutual Consistency Loss 40
4.3.5 Differentiable Mask Generation 41
4.3.6 Regularization 43

4.3.7 Fully Supervised and Mixed Training 44
4.3.8 Inference 44
4.4 Experiments 45
4.4.1 Evaluation Protocols and Datasets 45
4.4.2 Implementation Details 45
4.4.3 Ablation Experiments 46
4.4.4 Statistical Analysis 48
4.4.5 Training and Inference Time 49
4.4.6 Further Comparisons 50
4.4.7 Qualitative Evaluation 52
4.5 Conclusion 55

Action segmentation is the task of predicting the actions for each frame of a video. As obtaining the full annotation of videos for action segmentation is expensive, weakly supervised approaches that can learn only from transcripts are appealing. In this chapter, we propose a novel end-to-end approach for weakly supervised action segmentation based on a two-branch neural network. The two branches of our network predict two redundant but different representations for action segmentation and we propose a novel mutual consistency (MuCon) loss that enforces the consistency of the two redundant representations. Using the MuCon loss together with a loss for transcript prediction, our proposed approach achieves the accuracy of state-of-the-art approaches while being 14 times faster to train and 20 times faster during inference. The MuCon loss is even beneficial in the fully supervised setting.

4.1 Introduction

The production, availability, and consumption of video data is increasing every day at an exponential rate. With such growth comes the need to analyze, monitor, annotate, and learn from this huge amount of often publicly available data. The research community has therefore shown great interest in video data and approaches for action recognition on trimmed video clips have shown remarkable results in recent years (*Simonyan and Zisserman*, 2014; *Carreira and Zisserman*, 2017; *Diba et al.*, 2018b,a, 2020; *Wang et al.*, 2018; *Feichtenhofer et al.*, 2019). Although these results on trimmed video clips are important, in many real-world scenarios we are often faced with untrimmed videos that contain multiple actions with different lengths (*Kuehne et al.*, 2016; *Abu Farha and Gall*, 2019; *Lea et al.*, 2017; *Zhao et al.*, 2017).

Since annotating exact temporal boundaries of actions in long videos is cumbersome and costly, action segmentation approaches that can utilize weaker types of supervision are important. In particular ordered lists of actions such as *spoon powder - pour milk - stir milk*, which are termed transcripts and describe the order of the actions as they occur in a training video but not when they occur, are a popular type of weak supervision (*Huang et al.*, 2016; *Richard et al.*, 2017, 2018b; *Ding and Xu*, 2018; *Chang et al.*, 2019; *Li et al.*, 2019).

To learn from transcripts, previous approaches try to align the transcripts to the training videos, *i.e.*, they infer frame-wise labels of each training video based on the provided transcripts. This alignment is then used as the pseudo ground truth for training. For the transcript alignment, the Viterbi

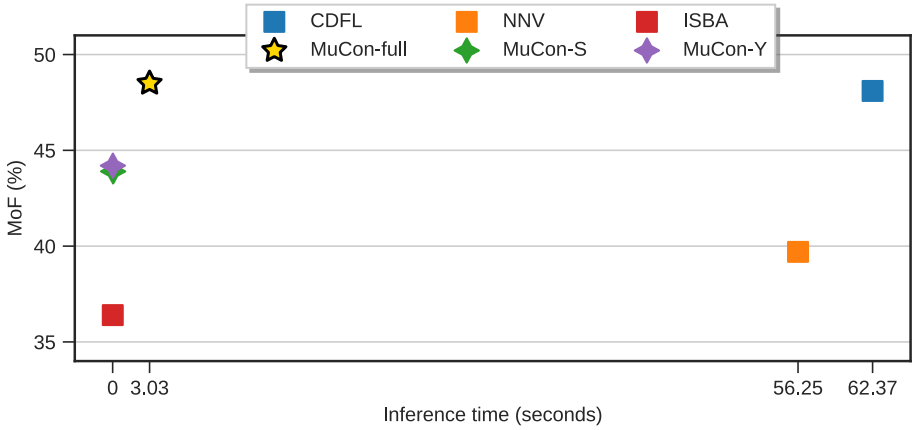


Figure 4.1: Average inference time per video (seconds) vs. average mean over frames (MoF) accuracy (%) for weakly supervised action segmentation approaches on the Breakfast dataset (*Kuehne et al.*, 2014). The average MoF is calculated over 5 training/inference iterations. The proposed approach (MuCon-full) provides the best trade-off between inference time and accuracy. More details are provided in Section 4.4.5.

algorithm is commonly used. It takes the estimated frame-wise class probabilities of a video and finds the best sequence of frame labels that does not violate the action order of the given transcript. While *Richard et al.* (2017); *Ding and Xu* (2018) perform the alignment after each epoch for all training videos, *Richard et al.* (2018b); *Li et al.* (2019) apply it at each iteration to a single video.

During inference, previous approaches except ISBA (*Ding and Xu*, 2018) rely on segmentation through alignment. This means that given an unseen video from the test set, the methods search over all transcripts of the training set and take the transcript that best aligns with the test video. This is highly undesirable since the inference time increases in this case as the number of different transcripts in the training set increases. As a result, these approaches are inefficient as shown in Figure 4.1.

In contrast to segmentation through alignment approaches, ISBA is fast, but it does not achieve the accuracy of state-of-the-art approaches. This means that at the moment one can only choose between accurate or fast approaches, but one cannot have both.

In this work, we therefore fill this gap and propose an approach that is nearly as fast as ISBA and nearly as accurate as the state-of-the-art as shown in Figure 4.1. Instead of optimizing over all possible transcripts during inference, our approach directly predicts the transcript for a video as well as the frame-wise class probabilities using two branches as illustrated in Figure 4.2. This means that the inference time does not depend on the number of transcripts used for training and that the best label sequence can be directly predicted from the estimated transcript and frame-wise class probabilities.

In addition, the proposed approach has also two major advantages during training. First, the branch, which predicts the transcripts, is directly trained with the type of supervision that is provided, namely transcripts. Second, the branch not only predicts the transcripts, but also the length of each action in the transcript. This means that the branch predicts already a full segmentation of the video without the need for the Viterbi algorithm, which also reduces the training time. Nevertheless, we need an additional loss to learn the action lengths. In contrast to previous works, we do not generate pseudo ground truth with hard labels during training, but we propose a novel differentiable mutual

consistency (MuCon) loss that enforces that the representations estimated by the two branches are mutually consistent and match each other. Furthermore, we show that the approach can be trained using weak supervision, full supervision, or a mixture of the two, where only few videos are fully annotated.

We provide a thorough analysis of the proposed approach including a detailed statistical analysis. We show that the proposed network with the mutual consistency loss achieves an accuracy that is either on par or better than existing approaches. At the same time, it is 20 times faster during inference compared to the most accurate approach (*Li et al.*, 2019) as shown in Figure 4.1.

4.2 Weakly Supervised Action Segmentation

Action segmentation is the task of predicting the action class for each frame of a video. More formally, given an input sequence of T D-dimensional frame-level features $X_{1:T} = [x_1, \dots, x_T]$, $x_t \in \mathbb{R}^D$, the goal is to predict the output sequence of frame-level action labels $\hat{Y}_{1:T} = [\hat{y}_1, \dots, \hat{y}_T]$, where $\hat{y}_t \in \mathcal{C}$ and $\mathcal{C}$ is the set of action classes. The frame-level action labels $\hat{Y}_{1:T}$ can also be represented as an ordered sequence of M segments $\hat{S}_{1:M}$ where each segment $\hat{s}_m$ is defined as an action label $\hat{a}_m \in \mathcal{C}$ and its corresponding length $\hat{\ell}_m \in \mathbb{R}_+$.

For fully supervised action segmentation, the target labels for every frame $\hat{y}_t$ are known during training. This means that the target lengths $\hat{\ell}_m$ are known. However, in weakly supervised action segmentation, the only supervisory signal is the ordered sequence of actions $\hat{A}_{1:M} = [\hat{a}_1, \dots, \hat{a}_m]$, often called *video transcript*, while the action lengths $\hat{L}_{1:M} = [\hat{\ell}_1, \dots, \hat{\ell}_M]$ are unknown.

In order to estimate $L_{1:M}$, we exploit the fact that the two unknown target representations $\hat{Y}_{1:T}$ and $\hat{S}_{1:M}$ for action segmentation are redundant and it is possible to generate one given the other. The main idea is therefore to predict both representations $Y_{1:T}$ and $S_{1:M}$ by different branches of the network such that they can supervise each other. In the ideal case, both predicted representations converge to the same solution during training. In order to train the model, however, we have to face the challenging problem that the mapping from one representation to the other needs to be differentiable, such that the loss that measures the mutual consistency of both representations is differentiable with respect to the predicted frame-wise class probabilities $Y_{1:T}$ and the predicted segment lengths $L_{1:M}$. Since the transcript $\hat{A}_{1:M}$ is given, it does not need to be differentiable with respect to $A_{1:M}$.

4.3 Proposed Method

During training, we have for each video the input video features $X_{1:T}$ and its corresponding transcript $\hat{A}_{1:M}$. Since only weak labels in form of the transcripts are provided for training, we propose a) to use the weak labels directly for training a transcript prediction subnetwork and b) exploit the two representations discussed in Section 4.2 as mutual supervision.

The proposed network is illustrated in Figure 4.2. The network consists of two branches that share the same backbone $f_t(X)$, which is a temporal convolutional network that maps the input video features $X \in \mathbb{R}^{T \times D}$ to a latent video representation $Z \in \mathbb{R}^{T' \times D'}$. The backbone architecture is described in Section 4.3.1. After the shared backbone, the network has two branches. The top branch $f_c(Z)$ is a standard frame classification branch that estimates the class probabilities of each

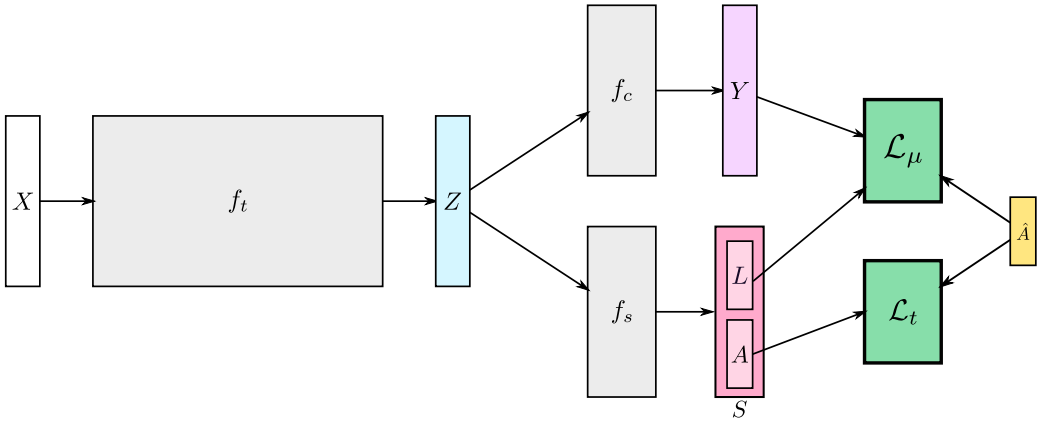


Figure 4.2: Our proposed network consists of three subnetworks (gray). The temporal backbone f_t embeds the input features in the hidden representation Z which is used for two branches. While the frame classification branch f_c predicts framewise class probabilities Y for action segmentation, the segment generation branch f_s predicts the segment representation S for action segmentation. We train our network using two loss functions. While the transcript prediction loss $\mathcal{L}_t$ enforces that the predicted transcript A matches the ground-truth transcript $\hat{A}$, our proposed mutual consistency (MuCon) loss $\mathcal{L}_\mu$ enforces that the two representations are consistent.

frame $Y \in \mathbb{R}^{T \times N}$, where N is the number of classes. The branch will be described in Section 4.3.2. The novelty of the network is the lower branch and the mutual consistency loss $\mathcal{L}_\mu$, which will be described in Sections 4.3.3 and 4.3.4, respectively.

In contrast to previous works that commonly use a network to predict Y and Viterbi decoding on top of it for computing the loss based on the given transcript $\hat{A}$, we use a second branch that predicts the segment representation S, *i.e.* A and L. This has two advantages during training. First, we can compare the predicted transcript A and the ground-truth transcript $\hat{A}$ for each video directly, *i.e.*, the network is directly trained with the type of supervision that is provided. The corresponding loss is denoted by $\mathcal{L}_t$. Second, the network has two branches that supervise each other and we do not need an additional Viterbi decoding step. Since the two branches predict different representations, the novel mutual consistency (MuCon) loss $\mathcal{L}_\mu$ requires a differentiable mask generator such that the loss is differentiable with respect to Y and L; otherwise we could not train the network.

After training, we need to infer the action class for each frame of an unseen test video. In case of action segmentation, no additional transcripts are given for the test videos. As shown in Figure 4.1, we can use three different approaches for inference, which are denoted by MuCon-Y, MuCon-S, and MuCon-full. In case of MuCon-Y and MuCon-S, we use the predicted Y or S representation, respectively. While these settings are as fast but more accurate than ISBA (*Ding and Xu*, 2018), the accuracy is lower compared to the state-of-the-art. For the model MuCon-full, we use therefore the predictions of both branches. To this end, we use the predicted framewise class probabilities Y and the predicted transcript A to find the best sequence of action labels using Viterbi decoding as described in Section 4.3.8. Note that the approach is still much faster than other methods except of ISBA since we do not need to optimize over all possible transcripts. We now provide more details for each part of the network.

4.3.1 Temporal Backbone

The temporal backbone $f_t(X)$, which is a temporal convolutional network (*van den Oord et al.*, 2016; *Abu Farha and Gall*, 2019; *Lea et al.*, 2017; *Li et al.*, 2020), outputs the latent video representation $Z \in \mathbb{R}^{T' \times D'}$ of the input video features $X \in \mathbb{R}^{T \times D}$. This hidden representation has a smaller temporal resolution due to temporal pooling. Similar to *Abu Farha and Gall* (2019), our temporal backbone consists of a set of 1-dimensional dilated convolutional layers with increasing dilation sizes. More specifically, we first apply a 1-d convolution with kernel size 1 to perform dimensionality reduction. Then a set of 11 layers with increasing dilation rates are applied, followed by a single 1-d convolution with kernel size 1 that generates the output. The amount of dilation for each layer is 2^i. We perform temporal max pooling with a kernel size of 2 after the layers 1, 2, 4, and 8. Additional details are given in Section 4.4.2. The design is similar to a single stage TCN (*Abu Farha and Gall*, 2019), but it includes additional pooling layers.

Empirically we show that existence and the exponentially increasing dilation factors in the WaveNet layers are crucial for temporal modeling and prediction of accurate transcripts during inference (Section 4.4.3).

4.3.2 Frame Classification Branch

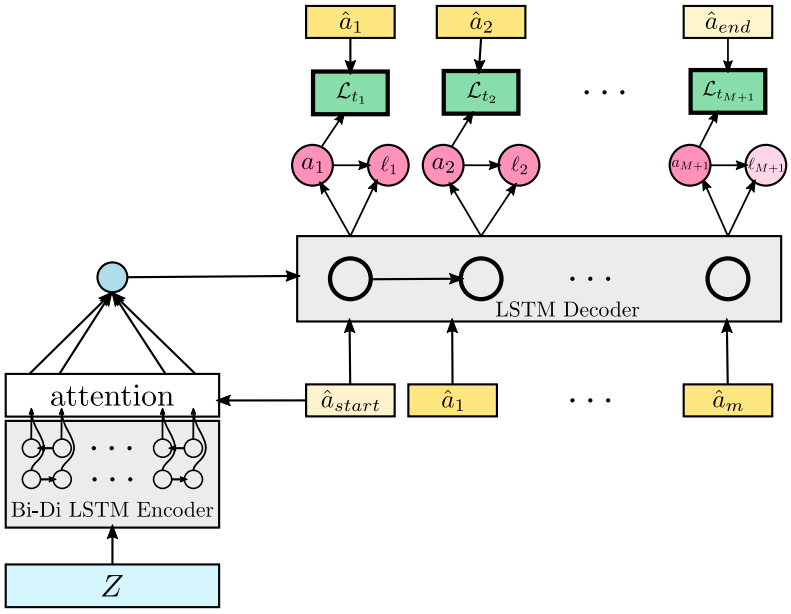
The classification branch takes the shared latent video representation $Z \in \mathbb{R}^{T' \times D'}$ as input and predicts the class probabilities $Y \in \mathbb{R}^{T \times N}$ where N is the number of actions in the dataset. Due to the temporal pooling of the temporal backbone, Z has a lower temporal resolution compared to the input features. To compensate for this, we first upsample Z using nearest-neighbor interpolation to the desired temporal size T. Then we use a single 1-d convolution with kernel size 1, which takes as input the upsampled $Z \in \mathbb{R}^{T \times D'}$ and outputs $Y \in \mathbb{R}^{T \times N}$.

4.3.3 Segment Generation Branch

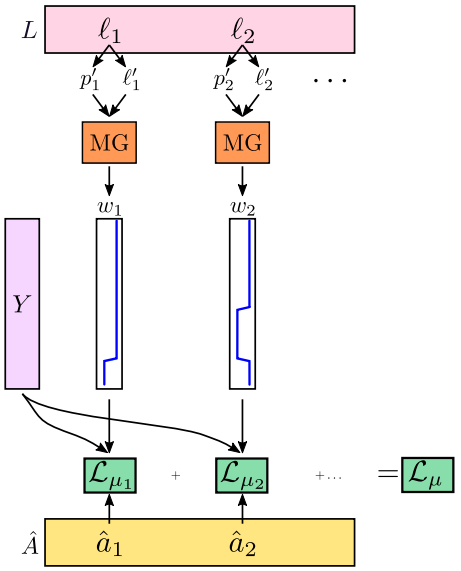
The second branch on top of Z is the segment generation subnetwork which predicts the segments S. Each segment s_m consists of predicted action probabilities a_m and the estimated relative log length ℓ_m of that segment. By predicting the log length, we implicitly enforce that the segment lengths are positive. We will discuss in Section 4.3.5.1 how the relative log length is mapped to the absolute length. The subnetwork and the transcript prediction loss $\mathcal{L}_t$ are illustrated in Figure 4.3a.

We employ a conventional sequence to sequence network with attention (*Bahdanau et al.*, 2015). Given the hidden video representation Z, we use a bidirectional LSTM encoder to encode it. Our decoder is an LSTM recurrent neural network with MLP attention. Although these networks on their own struggle to learn a temporal model for long input sequences (*Cho et al.*, 2014; *Singh et al.*, 2016), the temporal backbone, which encodes temporal relations at higher resolution, makes it easier for the segment generation subnetwork to learn temporal dependencies as shown by our ablation experiments.

As illustrated in Figure 4.3a, the decoding starts with the starting symbol $\hat{a}_{start}$. At each step of the decoding process, the ground truth action label $\hat{a}_{m-1}$ from the previous step is added to the encoded input sequence by concatenating it to the result of the attention. The concatenated vector is then given as input to the LSTM decoder cell, which estimates probability scores a_m using a fully connected MLP with two layers. Given these probability scores and the ground truth action label



(a)



(b)

Figure 4.3: Visualization of the segment generation branch and the loss functions. **(a)** Segment generation branch f_s and the transcript prediction loss $\mathcal{L}_t$. Given the hidden video representation Z, we use a sequence to sequence network with attention. The transcript prediction loss $\mathcal{L}_t$ compares the predicted action class probabilities a_m with the ground-truth action label $\hat{a}_m$. **(b)** Mutual consistency loss $\mathcal{L}_\mu$. Given the predicted lengths L, a set of masks w_m are generated using differentiable sampling by the mask generation module (MG). The loss then measures for each segment the consistency of the estimated framewise class probabilities Y with the ground-truth action $\hat{a}_m$.

$\hat{a}_m$, we compute the action prediction loss $\mathcal{L}_{t_m}$ per segment using cross-entropy. The final transcript prediction loss is defined as the sum of the action prediction losses, *i.e.*, $\mathcal{L}_t = \sum_{m=1}^{M+1} \mathcal{L}_{t_m}$. Note that we have $M+1$ terms since we add the end symbol $\hat{a}_{end}$ to the ground-truth transcripts, which needs to be predicted by the network as well. During training we use teacher forcing (*Williams and Zipser*, 1989), *i.e.*, we do not sample from the predicted action probabilities to feed the decoder, but we use the ground truth action labels $\hat{a}_m$.

Given the probability scores a_m and the hidden state of the decoder, we use another fully connected MLP with two layers to predict ℓ_m, which corresponds to the logarithm of the relative length of the segment. Notice that the parameters of this MLP are not updated based on the transcript prediction loss, but based on the mutual consistency loss $\mathcal{L}_\mu$.

4.3.4 Mutual Consistency Loss

Using the frame classification and the segment generation branches, two representations Y and S for the action segmentation are produced. However, so far we have only defined a loss for the predicted transcript A of the segment representation S, which we compare directly with the ground-truth transcript $\hat{A}$, but not for the predicted lengths of the actions L and the framewise class probabilities Y. We therefore propose the mutual consistency (MuCon) loss, which enforces that the two representations match each other and are mutually consistent. As shown in Figure 4.2, the mutual consistency loss takes the ground-truth transcript $\hat{A}$, the predicted segment lengths L, and framewise class probabilities Y as input. Since the loss is used to train both branches, it needs to be differentiable with respect to L and Y.

In principle, there are two choices: either we map Y to S or S to Y. The first approach, however, is not practical since it would need to detect consistent segments within Y. While this could be done using Viterbi decoding, it would make our network as expensive as segmentation through alignment approaches. Our goal, however, is to propose an approach that is fast and accurate. An explicit mapping from S to Y followed by the computation of a framewise loss, however, is also inefficient. We therefore combine the mapping and loss computation. Furthermore, we use $\hat{A}$ instead of A since we have already a loss that enforces that A is close to $\hat{A}$. Like teacher forcing, it also stabilizes the training since it guides the mutual consistency loss by $\hat{A}$ from the beginning of the training when A is still noisy. In our experiments, we show that this reduces the standard deviation over different runs.

The computation of the mutual consistency loss is illustrated in Figure 4.3b. We start with the estimated relative log length ℓ_m for each segment s_m. The relative log length ℓ_m is then converted into the absolute length ℓ'_m and for each segment we compute its absolute starting position p'_m within the video. This step is described in Section 4.3.5.1. Given ℓ'_m and p'_m, for each segment we generate a mask w_m using the differentiable mask generation module, which is described in Section 4.3.5. As mentioned before, for each segment we use the ground-truth action label $\hat{a}_m$ instead of the estimated class probabilities.

Finally, we can compare the consistency of the predicted frame-wise class probabilities Y with each segment defined by the mask w_m and the label $\hat{a}_m$. To this end, we first compute the average of Y for each segment based on the mask w_m:

$$g(Y, w_m) = \frac{\sum_{t=1}^{T} y_t w_m[t]}{\ell'_m} \tag{4.1}$$

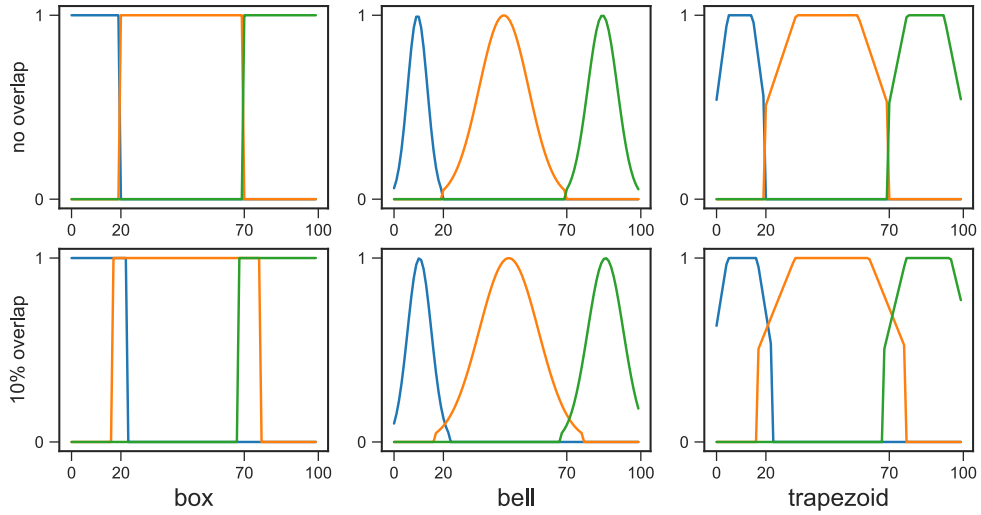


Figure 4.4: Examples of different masks for three consecutive action segments. The top row shows regular masks with different shapes while the bottom row shows masks generated with added 10% overlap. The left, middle and right figures depict box, bell, and trapezoid shaped masks.

where $g(Y, w_m) \in \mathbb{R}^N$, $w_m[t]$ is the value of the mask at frame t, and ℓ'_m is the absolute length of the segment. Then, we compute the cross-entropy for each segment:

$$\mathcal{L}_{\mu_m}(Y, w_m, \hat{a}_m) = -\log\left(\frac{e^{g(Y,w_m)[\hat{a}_m]}}{\sum_{n=1}^{N} e^{g(Y,w_m)[n]}}\right) \tag{4.2}$$

where we use the softmax function to normalize the class probabilities per segment and the ground-truth label $\hat{a}_m$. Since we normalize per segment using the softmax function, we use the estimates of Y before the softmax layer of the frame classification branch in (4.1). The final mutual consistency loss $\mathcal{L}_\mu$ is defined as the sum of all segment losses:

$$\mathcal{L}_\mu = \sum_{m=1}^{M} \mathcal{L}_{\mu_m}(Y, w_m, \hat{a}_m). \tag{4.3}$$

The mutual consistency loss (4.3) is a differentiable function of the masks w_m and the frame-wise class probabilities Y. Due to the definition of the differentiable mask generation, which we will describe in the following section, the masks are differentiable with respect to the estimated segment lengths L. This means that the gradients of the mutual consistency loss are backpropagated through both branches as it is required to train the network.

4.3.5 Differentiable Mask Generation

In order to compute the mutual consistency loss, we need to generate the masks w_m that act like gating functions that only allow information from the predicted temporal regions to pass through. The masks w_m are functions of the predicted absolute length ℓ'_m and predicted starting position p'_m

of each segment such that

$$w_m[t] \simeq \begin{cases} 1 & p'_m \leq t \leq p'_m + \ell'_m \\ 0 & otherwise \end{cases}, \quad t \in [1 \dots T]. \tag{4.4}$$

Examples of generated masks are shown in Figure 4.4.

4.3.5.1 Localization

In order to obtain the predicted absolute length ℓ'_m and the predicted starting position p'_m for each segment, we first calculate the absolute length values $L'_{1:M} = (\ell'_1, \dots, \ell'_M)$ for a video with T frames such that $\sum_{m=1}^{M} \ell'_m = T$, *i.e.*, the absolute lengths sum up to be equal to the length of the video. Having the absolute length ℓ'_m of each segment, we can also compute the absolute starting position p'_m for each segment:

$$\ell'_m = T \frac{e^{\ell_m}}{\sum_{k=1}^{M} e^{\ell_k}}, \quad p'_1 = 0, \quad p'_m = \sum_{k=1}^{m-1} \ell'_k. \tag{4.5}$$

4.3.5.2 Temporal Transformation

For generating the masks, we transform a reference template tensor $U \in [0,1]^J$ to $w_m \in [0,1]^T$ where J is a canonical value equal to 100. The reference template tensor can be of any shape and we evaluate three shapes, namely *box*, *bell*, and *trapezoid* which are depicted in Figure 4.4. By adjusting the absolute lengths ℓ'_m and starting positions p'_m, it is also possible to introduce an overlap for the masks.

We transform U to w_m by scaling and translating it using ℓ'_m and p'_m, respectively. Therefore, we use a 1D affine transformation matrix $\mathcal{T}_\theta$ such that

$$i_u[t] = \mathcal{T}_\theta(t) = [\theta_0 \ \theta_1] \begin{pmatrix} t \\ 1 \end{pmatrix} \quad \forall t \in [1, \dots, T] \tag{4.6}$$

where $i_u[t]$ are the sampling points over U. As a result of the affine transformation (4.6), the element indices of the sampling points in the template $i_u[t]$ can be outside the valid range of $[1, \dots, J]$. In such cases, the indices will be ignored in the sampling process (4.8). The affine transformation parameters are

$$\theta_0 = \frac{J}{\ell'_m}, \quad \theta_1 = \frac{-J p'_m}{\ell'_m} \tag{4.7}$$

where θ_0 scales the reference template U to the estimated length ℓ'_m and θ_1 translates it to the estimated position p'_m.

4.3.5.3 Temporal Sampling

To perform the aforementioned temporal transformation, we should sample from U using the sampling points $i_u[t]$ and produce the sampled mask w_m. Each index $i_u[t]$ refers to the element in U

where a sampling kernel must be applied to get the value at the corresponding element $w_m[t]$ in the output mask. Similar to *Jaderberg et al.* (2015), we perform this operation as follows

$$w_m[t] = \sum_{j=1}^{J} U[j]\Psi(i_u[t] - j) \quad \forall t \in [1, \dots, T] \tag{4.8}$$

where Ψ is the sampling kernel. Since we use a linear kernel, it can be written as

$$w_m[t] = \sum_{j=1}^{J} U[j] \max(0, 1 - |i_u[t] - j|) \quad \forall t \in [1, \dots, T]. \tag{4.9}$$

4.3.5.4 Backpropagation

To be able to backpropagate through the generated masks w_m, we define the gradients with respect to the sampling indices $i_u[t]$ as

$$\frac{\partial w_m[t]}{\partial i_u[t]} = \sum_{j=1}^{J} U_j \begin{cases} 0 & |i_u[t] - j| \geq 1 \\ 1 & j - 1 < i_u[t] \leq j \\ -1 & j < i_u[t] < j + 1 \end{cases}. \tag{4.10}$$

Since the sampling indices $i_u[t]$ are a function of the predicted lengths ℓ'_m, the loss gradients are backpropagated to the segment generation branch.

4.3.6 Regularization

To prevent degenerate solutions, *i.e.*, solutions where the lengths of some segments are large and the length of the other segments are almost zero, we add a regularization term for the predicted relative log lengths L:

$$\mathcal{L}_\ell = \sum_{m=1}^{M} \max(0, -\ell_m - w) + \max(0, \ell_m - w). \tag{4.11}$$

The proposed length regularizer adds a penalty if $\ell_m < -w$ or $\ell_m > w$. Note that the relative log lengths can be negative and are later converted into the absolute length as described in Section 4.3.5.1.

We also use the smoothing loss introduced in *Abu Farha and Gall* (2019) for the frame classification branch:

$$\mathcal{L}_s = \frac{1}{TN} \sum_{t,n} \hat{\Delta}_{t,n}^2, \tag{4.12}$$

$$\hat{\Delta}_{t,n} = \min(\tau, |\log y_{t,n} - \log y_{t-1,n}|), \tag{4.13}$$

with $\tau = 4$ as in *Abu Farha and Gall* (2019).

4.3.7 Fully Supervised and Mixed Training

Although our approach is designed for weakly supervised action segmentation, we can easily adapt it to a setting where all or some videos of the training set are fully annotated, *i.e.*, they are annotated not by the transcript $\hat{A}$ but by the framewise labels $\hat{Y}$.

In this case, we have two additional losses for our network shown in Figure 4.2. We use the cross-entropy loss for the frame classification branch to compare the predicted class probabilities Y with the framewise labels $\hat{Y}$, and we use the mean squared error loss for the segment generation branch to compare the estimated relative log lengths L with the ground truth segment lengths. To this end, we convert the absolute ground-truth lengths into relative log lengths.

Although the mutual consistency loss is only necessary in case of weakly supervised learning or in a setting where only a subset of the videos is fully annotated, we show that using the mutual consistency loss also improves the accuracy when the network is trained in a fully supervised way.

4.3.8 Inference

As discussed at the beginning of Section 4.3, we can either use the frame classification branch, which predicts the framewise class probabilities Y, or the segment generation branch, which predicts the segments S. In the later case, we start with the start symbol a_{start} and the decoder generates new segments until the special end symbol a_{end} is predicted as illustrated in Figure 4.3a. The estimated relative log segment lengths are then converted into absolute lengths as described in Section 4.3.5.1. The two approaches for inference are denoted by MuCon-Y and MuCon-S, respectively.

However, as it is shown in Figure 4.1, we achieve the highest accuracy at a small increase in inference time if we use the predictions of both branches. We denote this approach by MuCon-full. For simplicity, we use L instead of L' to denote the estimated absolute lengths of the segments in this section. In order to fuse both predictions, we keep the inferred transcript A, but reestimate the lengths of the segments L using Y:

$$L^* = \underset{\tilde{L}}{\operatorname{argmax}}\, p(\tilde{L}|Y, A, L). \tag{4.14}$$

Similar to *Richard and Gall* (2016); *Richard et al.* (2018b), we can factorize the term $p(\tilde{L}|Y, A, L)$ yielding

$$L^* = \underset{\tilde{L}}{\operatorname{argmax}} \prod_{t=1}^{T} y_t[a_{\alpha(t,\tilde{L})}] \prod_{m=1}^{M} P_{\ell_m}(\tilde{\ell}_m) \tag{4.15}$$

where $P_{\ell_m}(\tilde{\ell}_m)$ denotes a Poisson distribution with expected mean ℓ_m, which corresponds to the absolute segment length that has been estimated by the segment generation branch. Depending on $\tilde{L}$, the segment number changes for a frame t and it is denoted by $\alpha(t, L)$. Although L^* is obtained by dynamic programming as described in *Richard et al.* (2018b), we do not need to optimize over all possible transcripts as in *Li et al.* (2019); *Richard et al.* (2018b). While *Li et al.* (2019); *Richard et al.* (2018b) align each transcript of the training set to the test video and take the training transcript that best aligns to the test video, our approach infers the transcript A directly from the test video and only aligns A to the test video. MuCon-full is therefore still much faster than *Li et al.* (2019); *Richard et al.* (2018b) as shown in Figure 4.1.

4.4 Experiments

We evaluate our approach for two tasks, namely action segmentation as described in Section 4.2 and action alignment. In contrast to action segmentation, the transcripts are also given for the test videos in case of action alignment. Besides of the weakly supervised setting, we also evaluate the approach when it is trained fully supervised or in a mixed setting where some videos are fully annotated and the other videos are only weakly annotated by transcripts. Before we evaluate the approach, we discuss the evaluation protocols and further implementation details[1].

4.4.1 Evaluation Protocols and Datasets

We evaluate our method on two popular datasets, the Breakfast dataset (*Kuehne et al.*, 2014) and the Hollywood extended dataset (*Bojanowski et al.*, 2014). The Breakfast dataset contains more than 1.7k videos of different cooking activities. The videos contain 10 different types of breakfast activities such as *prepare cereal* or *prepare coffee* which consists of 48 different fine-grained actions. In our experiments, we follow the 4 train/test splits provided with the dataset and report the average. The Hollywood extended dataset contains 937 video sequences taken from Hollywood movies. The videos contain 16 different action classes. We follow the train/test split strategy of *Richard et al.* (2017, 2018b); *Li et al.* (2019).

The main performance metric used for the action segmentation task is the mean over frames (MoF) accuracy (*Abu Farha and Gall*, 2019; *Kuehne et al.*, 2016; *Richard et al.*, 2017, 2018b). We also report the F1 score that has been used by fully supervised approaches (*Abu Farha and Gall*, 2019; *Li et al.*, 2020). For action segmentation, we also directly evaluate the performance of the predicted transcripts, which is measured by the matching score or edit distance (*Lambert*, 2019). For action alignment, we use the intersection over detection (IoD) metric as in (*Bojanowski et al.*, 2014; *Richard et al.*, 2017, 2018b; *Ding and Xu*, 2018; *Chang et al.*, 2019; *Li et al.*, 2019).

If not otherwise specified, we report the average and standard deviation over 5 runs with different random seeds.

4.4.2 Implementation Details

We train the entire network end-to-end after initializing it with Gaussian random weights. In each iteration, we use only a single video, *i.e.*, the batch size is 1. We use a single group norm (*Wu and He*, 2018) layer with 32 groups after the final layer of the temporal backbone f_t. The dimensionality of the shared latent video representation Z is $D' = 128$. Unless otherwise stated, we apply temporal max pooling with kernel size 2 after the convolutional layers 1, 2, 4, and 8 for all experiments on the Breakfast dataset. For the experiments on the Hollywood Extended dataset, we omit the last temporal max pooling after the convolutional layer 8 since the videos in the Hollywood extended dataset are relatively short. The size of the hidden states of the bidirectional LSTM encoder and the LSTM decoder in our segment generation module are set to 128. We also employ an input embedding for the LSTM decoder of size 128 with 0.25 dropout.

In the weakly supervised setting, the training loss is defined as $\mathcal{L} = \mathcal{L}_\mu + \mathcal{L}_t + \alpha\mathcal{L}_\ell + \beta\mathcal{L}_s$. We use $\alpha = \beta = 0.1$ unless otherwise specified. Since most of the frames in the Hollywood extended

[1] Source code is available at: github.com/yassersouri/MuCon

f_t	$\mathcal{L}_\mu$	Matching Score
single 1-d conv	✗	$0.691_{\pm 0.010}$
fixed dilation	✗	$0.696_{\pm 0.021}$
increasing dilation, linear	✗	$0.727_{\pm 0.019}$
increasing dilation, exponential	✗	$0.729_{\pm 0.015}$
increasing dilation, exponential	✓	$0.774_{\pm 0.006}$

Table 4.1: Transcript prediction accuracy on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. The first four rows show the matching score of the segment generation branch for different dilation factors used for the temporal backbone. In these settings, the mutual consistency loss $\mathcal{L}_\mu$ is not used. The last row shows the proposed setting with the mutual consistency loss.

Shape	Overlap	MoF	Matching Score	F1@{10,25,50}		
box	✗	$49.0_{\pm 2.0}$	$0.774_{\pm 0.006}$	67.9	59.5	40.2
bell	✗	$43.6_{\pm 2.3}$	$0.777_{\pm 0.009}$	65.1	55.5	35.4
trapezoid	✗	$48.7_{\pm 1.8}$	$0.773_{\pm 0.006}$	67.3	59.0	39.4
box	10%	$48.8_{\pm 1.5}$	$0.787_{\pm 0.006}$	68.3	59.6	39.8
bell	10%	$43.5_{\pm 0.8}$	$0.779_{\pm 0.007}$	64.8	55.3	34.7
trapezoid	10%	$48.0_{\pm 2.7}$	$0.780_{\pm 0.009}$	67.7	59.1	39.5

Table 4.2: Impact of mask generation on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. As shown in Figure 4.4, different shapes with or without overlap can be used for the mask generation. We report the MoF accuracy, matching score, and F1 score.

dataset are annotated by background, which increases the possibility of degenerate solutions, we set α to 1 for the Hollywood extended dataset. We use $w = 2$ in (4.11). The initial learning rate is set to 0.01 and is lowered by a factor of 10 after 70 epochs for the Breakfast dataset and after 60 epochs for the Hollywood extended dataset. We train our network for 150 epochs for all weakly supervised experiments. In the fully supervised and mixed supervision experiments, we train for 110 epochs since convergence is faster with full supervision.

As input features for the Breakfast and Hollywood extended datasets, we use RGB+flow I3D (*Carreira and Zisserman*, 2017) features extracted from a network that was pretrained on the Kinetics400 dataset (*Kay et al.*, 2017). These features have been previously used for fully supervised approaches (*Abu Farha and Gall*, 2019; *Li et al.*, 2020). A recent study found that current weakly supervised approaches perform better with IDT features than I3D features (*Souri et al.*, 2020). We therefore report the results for the features where the corresponding methods perform best, *i.e.*, I3D for *Abu Farha and Gall* (2019); *Li et al.* (2020) and IDT for *Ding and Xu* (2018); *Richard et al.* (2018b); *Li et al.* (2019).

4.4.3 Ablation Experiments

In this section, we quantitatively examine different components in our method. For each metric, We report the average and standard deviation over 5 runs on split 1 of the Breakfast dataset (*Kuehne*

Regularizer	MoF	Matching Score	F1@{10,25,50}		
$\varnothing$	$47.4_{\pm 1.1}$	$0.787_{\pm 0.009}$	68.1	59.3	39.7
$\mathcal{L}_s$	$47.5_{\pm 1.7}$	$0.780_{\pm 0.005}$	67.8	59.0	39.5
$\mathcal{L}_\ell$	$48.3_{\pm 0.9}$	$0.777_{\pm 0.011}$	67.8	59.0	40.3
$\mathcal{L}_\ell + \mathcal{L}_s$	$49.0_{\pm 2.0}$	$0.774_{\pm 0.006}$	67.9	59.5	40.2

Table 4.3: Impact of regularizers on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. The first three rows denote settings where the length regularizer $\mathcal{L}_\ell$ for the segment generation branch, the smoothing loss $\mathcal{L}_s$ for the frame classification branch, or both are omitted.

Inference variant	MoF	F1@{10,25,50}		
MuCon-Y	$44.7_{\pm 1.4}$	28.1	22.6	13.3
MuCon-S	$43.6_{\pm 1.2}$	65.6	57.2	33.8
MuCon-full	$49.0_{\pm 2.0}$	67.9	59.5	40.2

Table 4.4: Impact of fusing both branches on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. While MuCon-Y and MuCon-S use either the frame classification branch or the segment generation branch for inference, MuCon-full fuses both branches for inference.

et al., 2014).

4.4.3.1 Transcript Prediction

We first analyze how well the proposed approach predicts the transcripts, which is measured by the matching score. If we only use the segment generation branch, the network achieves a matching score of 0.729 as shown in row 4 of Table 4.1. If we add the frame classification branch and the mutual consistency loss $\mathcal{L}_\mu$, the matching score increases to 0.774. This shows that the mutual consistency loss also improves the transcript prediction.

In the first four rows of Table 4.1, we furthermore evaluate the impact of the dilation factors for the temporal backbone f_t. If we replace the temporal backbone by a single 1-d convolution, the matching score decreases from 0.729 to 0.691. If we use as in the proposed network 11 layers with 1-d convolutions but with fixed dilation factor of 1, the matching score does not significantly increase since it does not substantially increase the receptive field. Only when we linearly or exponentially increase the dilation factors for each layer, we observe an improvement of the matching score.

4.4.3.2 Mask Generation Settings

As described in Section 4.3.5, different shapes of mask templates can be used for the mutual consistency loss. We evaluated three shapes, namely *box*, *bell*, and *trapezoid*, and added an optional overlap of 10% as depicted in Figure 4.4. The results reported in Table 4.2 show that the shape of the mask has little impact on the quality of the predicted transcript, but that the bell shape has the worst performance. In general, having hard boundaries for the masks without any overlap performs best. For all other experiments, we therefore use the *box* shape without any overlap.

Teacher forcing	MoF	Matching Score	F1@{10,25,50}		
None	$48.7_{\pm 2.9}$	$0.779_{\pm 0.009}$	68.2	59.5	40.3
70 epochs	$48.9_{\pm 2.1}$	$0.783_{\pm 0.009}$	68.2	59.5	39.8
All epochs	$49.0_{\pm 2.0}$	$0.774_{\pm 0.006}$	67.9	59.5	40.2

Table 4.5: Impact of teacher forcing on split 1 of the Breakfast dataset. We report the average and standard deviation over 5 runs. The first row denotes a setting where teacher forcing is not used for training. The second row denotes a setting where teacher forcing is used only for the first 70 epochs. The last row denotes a setting where teacher forcing is used for all epochs.

4.4.3.3 Effect of Regularizers

As described in Section 4.3.6, we also use two additional regularizers during training, namely the length loss $\mathcal{L}_\ell$ and the smoothing loss $\mathcal{L}_s$. We evaluate the impact of each term in Table 4.3 by removing $\mathcal{L}_\ell$, $\mathcal{L}_s$, or both. Without any regularizer, the MoF accuracy decreases from 49.0 to 47.4 and F1@50 from 40.2 to 39.7. While adding only the smoothing loss does not result in an improvement, the length regularizer and the combination of both improves MoF.

4.4.3.4 Different Variants for Inference

As described in Section 4.3.8, we have three options for inference. We can use the frame classification branch, the segment generation branch, or both. The three approaches are denoted by MuCon-Y, MuCon-S, and MuCon-full, respectively. The results in Table 4.4 show that fusing the two branches improves the MoF accuracy compared to MuCon-S by more than 5% and F1@50 by more than 6% at a small increase in inference time as shown in Figure 4.1. Since MuCon-Y predicts only frame-wise labels and not segments, the F1 scores are very low for MuCon-Y.

4.4.3.5 Effect of Teacher Forcing

As described in Sections 4.3.3 and 4.3.4, we use the ground truth transcript while predicting the transcript and calculating the MuCon loss in order to stabilize the training at the beginning. In Table 4.5, we evaluate the impact of teacher forcing where we also include a setting that uses teacher forcing only for the first 70 epochs of training. The results show that teacher forcing does not impact the average accuracy but it reduces the standard deviation, which is an indicator of a stable training procedure.

4.4.4 Statistical Analysis

As the standard deviations in Table 4.6 show, the results vary for different random seeds. In order to provide a thorough comparison with other weakly supervised action segmentation approaches, we use a statistical test to measure if the differences in accuracy are significant or not. To this end, we performed five training/inference iterations on the entire Breakfast dataset. Besides of our approach (MuCon), we included CDFL (*Li et al.*, 2019), NNV (*Richard et al.*, 2018b), and ISBA (*Ding and Xu*, 2018) for which the official open-source implementation is available. We use the hyperparameters as they were provided by the authors for this dataset. We use the Friedman statistical test (*Friedman*,

Approach	Avg MoF$_{\pm \text{Std}}$	Avg Mat Score$_{\pm \text{Std}}$	Training (hours)	Inference (seconds)
ISBA (*Ding and Xu*, 2018)	36.4$_{\pm 1.0}$	-	12.75	0.01
NNV (*Richard et al.*, 2018b)	39.7$_{\pm 2.4}$	0.686$_{\pm 0.009}$	11.23	56.25
CDFL (*Li et al.*, 2019)	48.1$_{\pm 2.5}$	0.712$_{\pm 0.009}$	66.73	62.37
MuCon-Y	44.2$_{\pm 1.4}$	-	4.57	0.02
MuCon-S	43.9$_{\pm 1.4}$	**0.785**$_{\pm 0.008}$	4.57	0.04
MuCon-full	**48.5**$_{\pm 1.8}$	**0.785**$_{\pm 0.008}$	4.57	3.03

Table 4.6: Comparison of accuracy, training, and inference time. We report the average and standard deviation of MoF and matching score for 5 runs on the entire Breakfast dataset (*Kuehne et al.*, 2014). For ISBA and MuCon-Y, the matching score is not calculated as they predict frame-wise labels and not the transcript. Training and inference time is measured as wall clock time. The training time is measured for training on split 1 of the Breakfast dataset (*Kuehne et al.*, 2014). The inference time is measured as the average inference time for a video from the test set of split 1.

1937) with the dataset split as the blocking factor. We selected the Friedman test as it does not make any restrictive assumption on the underlying distribution of the data.

While we report the average and standard deviation of the MoF accuracy and matching score in Table 4.6, the results of the Friedman test for the MoF accuracy and the matching score are shown in Tables 4.7a and 4.7b, respectively. We observe that MuCon and CDFL are both significantly better than ISBA and NNV. While the average MoF accuracy and the average matching score of MuCon are higher than the ones of CDFL, the difference is only statistically significant for the matching score. This shows that MuCon is on par with CDFL in terms of MoF accuracy, but significantly better in terms of matching score.

4.4.5 Training and Inference Time

Besides of the accuracy, we also compare the training and inference time of our approach with CDFL (*Li et al.*, 2019), NNV (*Richard et al.*, 2018b), and ISBA (*Ding and Xu*, 2018). For a fair comparison, we always used the same hardware. For training, we used a machine with an Nvidia GeForce GTX 1080Ti GPU, 188 GB of RAM, and an Intel(R) Xeon(R) Gold 5120 (2.20GHz) CPU. For testing, we used a machine with an Nvidia GeForce GTX Titan X GPU, 32 GB of RAM, and an Intel(R) Core(TM) i7-4930K (3.40GHz) CPU. We only calculate the wall time for training and deactivated for all methods any unnecessary operations like saving intermediate results. Since we used pre-computed features for all experiments, the time measurement includes the time to load the features but not the time to compute the features. In average, computing the features takes 92 seconds per video for the Breakfast dataset where 72.5 seconds are spent for calculating the optical flow.

The results are reported in Table 4.6. We observe that our approach is 14 times faster to train and 20 times faster during inference compared to the state-of-the-art approach CDFL (*Li et al.*, 2019). As mentioned before, our approach does not perform any Viterbi decoding during training, which makes it faster during training. Also and most importantly, our approach only performs one Viterbi decoding step for the estimated transcript during inference as compared to CDFL and NNV which need to optimize over all possible transcripts. This shows that our approach offers by far the best trade-off between accuracy and runtime as it is also illustrated in Figure 4.1.

p-value	ISBA (36.4)	NNV (39.7)	CDFL (48.1)	MuCon (48.5)
ISBA (36.4)		✓(0.001)	✓($1.77_{\times 10^{-7}}$)	✓($1.77_{\times 10^{-7}}$)
NNV (39.7)	✓(0.001)		✓($3.09_{\times 10^{-7}}$)	✓($5.35_{\times 10^{-7}}$)
CDFL (48.1)	✓($1.77_{\times 10^{-7}}$)	✓($3.09_{\times 10^{-7}}$)		✗(0.465)
MuCon (48.5)	✓($1.77_{\times 10^{-7}}$)	✓($5.35_{\times 10^{-7}}$)	✗(0.465)	

(a) Friedman statistical test for MoF.

p-value	NNV (0.686)	CDFL (0.712)	MuCon (0.785)
NNV (0.686)		✓($1.96_{\times 10^{-6}}$)	✓($1.77_{\times 10^{-7}}$)
CDFL (0.712)	✓($1.96_{\times 10^{-6}}$)		✓($1.77_{\times 10^{-7}}$)
MuCon (0.785)	✓($1.77_{\times 10^{-7}}$)	✓($1.77_{\times 10^{-7}}$)	

(b) Friedman statistical test for matching score.

Table 4.7: Friedman statistical test results. The top row and the leftmost column indicate different approaches and their MoF accuracy and matching score from Table 4.6. Note that the test is symmetrical. If the p-value is smaller than 0.05 the difference is considered as significant (indicated with ✓); otherwise the difference is not considered as significant (indicated with ✗).

In order to avoid the full alignment of all transcripts of the training set to a test video, beam search can be used. Beam search limits the maximum number of hypotheses and removes hypotheses with low probability at an early stage. The smaller the beam size, i.e., the number of hypotheses, is, the faster is the inference. This, however, comes at the cost of reducing the accuracy. In order to evaluate how much runtime reduction can be achieved by beam search, we evaluate CDFL with beam search. Figure 4.5 shows the MoF accuracy and inference time for different beam sizes. In contrast to Figure 4.1, we report the accuracy for only one run and only split 1 of the Breakfast dataset. We furthermore plot the inference time of the entire test set in log scale for better visualization. We observe that beam search reduces the inference time of CDFL by multiple orders of magnitude but also its accuracy. When we compare a setting where CDFL with beam search is as fast as MuCon, we observe that MuCon achieves a much higher accuracy.

4.4.6 Further Comparisons

Although a statistical test is the golden standard for comparing different methods, this is only possible if the results of multiple runs are available. We therefore also compare our method with state-of-the-art methods based on the reported numbers on the Breakfast dataset (*Kuehne et al.*, 2014) in Table 4.8a and on the Hollywood extended dataset (*Bojanowski et al.*, 2014) in Table 4.8b for weakly supervised action segmentation and alignment. Our approach outperforms all other methods except of CDFL for both tasks and datasets. As already discussed in Sections 4.4.4 and 4.4.5, our approach achieves either an accuracy that is comparable to CDFL or significantly better. Furthermore, our approach is 20 times faster during inference and provides thus a much better trade-off between accuracy and runtime.

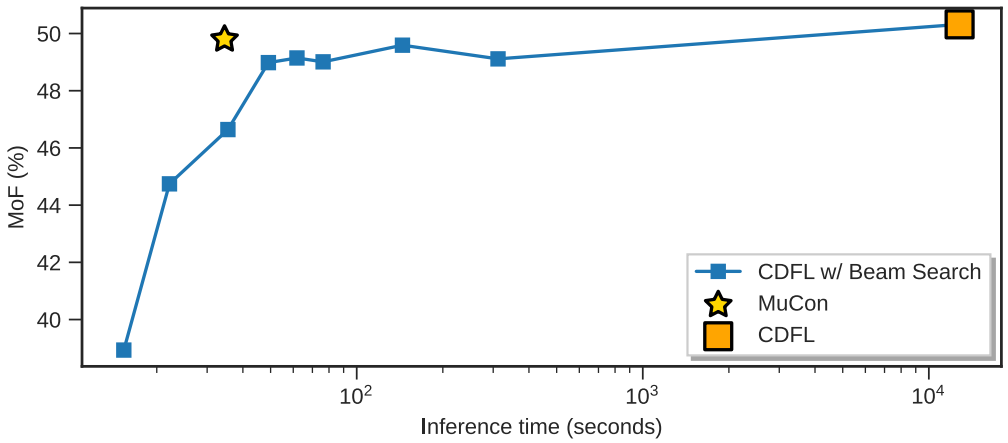


Figure 4.5: Using beam search to reduce the inference time of CDFL. The blue squares show the performance of CDFL with different values for the beam size. The x-axis is the total inference time for split 1 of the Breakfast dataset in log scale.

4.4.6.1 Fully Supervised

As mentioned in Section 4.3.7, we can apply our approach to the fully supervised setting as well. For comparison, we use the same metrics as in *Abu Farha and Gall* (2019); *Li et al.* (2020) namely MoF (termed accuracy in *Abu Farha and Gall* (2019)), Edit which measures predicted transcript similarity (similar to the matching score), and F1 score at different overlaps.

We first evaluate whether the mutual consistency loss $\mathcal{L}_\mu$ also improves the accuracy in a fully supervised setting. For this ablation experiment, we use split 1 of the Breakfast dataset. As it is shown in Table 4.9, the proposed mutual consistency loss improves the accuracy for all metrics. This shows that the mutual consistency loss is not only useful for weakly supervised learning, but also for fully supervised learning.

We furthermore compare our approach to other fully supervised action segmentation approaches in Table 4.10. Although our approach was designed for weakly supervised learning, it outperforms the state-of-the-art for most metrics. Only for the MoF accuracy, *Abu Farha and Gall* (2019); *Li et al.* (2020) perform better, but these networks use multiple stages and thus more layers.

4.4.6.2 Mixed Supervision

Since our network can be trained in a weakly as well in a fully supervised setting, we can train the network also in a mixed setting where a small percentage of the videos is fully annotated and the remaining videos are only weakly annotated by transcripts. As before, we report the average and standard deviation over 5 runs where we randomly sample the videos with frame-wise annotations for each run. The results for a varying percentage of fully annotated videos are reported in Table 4.11 and visualized in Figure 4.6. In case of 0%, the setting corresponds to weakly supervised learning and 100% corresponds to fully supervised learning. The difference in accuracy between the weakly and fully supervised cases is about 15%. While there is no significant improvement if only 1-2% of the videos are fully annotated, the accuracy increases when at least 5% of the videos are fully annotated. Having 10% of the videos fully annotated, the accuracy gap between the mixed setting

Approach	MoF - Action Segmentation	IoD - Action Alignment
ECTC (*Huang et al.*, 2016)	27.7	45.0
HMM/RNN (*Richard et al.*, 2017)	33.3	47.3
ISBA (*Ding and Xu*, 2018)	38.4	52.3
NNV (*Richard et al.*, 2018b)	43.0	-
D3TW (*Chang et al.*, 2019)	45.7	56.3
CDFL (*Li et al.*, 2019)	**50.2**	<u>63.9</u>
MuCon∗	<u>48.5</u>	**66.2**

(a) Comparison on the Breakfast dataset.

Approach	MoF-BG - Action Segmentation	IoD - Action Alignment
ECTC (*Huang et al.*, 2016)	-	41.0
HMM/RNN (*Richard et al.*, 2017)	-	46.3
ISBA (*Ding and Xu*, 2018)	34.5	39.6
NNV (*Richard et al.*, 2018b)	-	48.7
D3TW (*Chang et al.*, 2019)	33.6	50.9
CDFL (*Li et al.*, 2019)	<u>40.6</u>	**52.9**
MuCon∗	**41.6**	<u>52.3</u>

(b) Comparison on the Hollywood extended dataset.

Table 4.8: Comparison to the state-of-the-art. The highest score is indicated by a bold font and the second highest is underlined. ∗ indicates that the reported metric is averaged over 5 runs.

and the fully supervised setting is already reduced by about 50%.

4.4.7 Qualitative Evaluation

We provide some qualitative results for four different test videos of the Breakfast dataset in Figure 4.7. Figure 4.7a shows a video where NNV, CDFL, and MuCon perform well. Only NNV hallucinates the action *take cup*, which is not present in the video. The hallucination of actions occurs due to the alignment of the transcripts of the training set to the test video. In this example, there is a very high uncertainty among the class probabilities estimated by NNV at the beginning of the video. Since the probabilities for *take cup* are low but slightly higher than for *spoon powder* or *background*, the transcript *take cup - spoon powder - pour milk - stir milk* achieves a higher alignment score than the correct transcript *spoon powder - pour milk - stir milk*. Figures 4.7b and 4.7c show

Breakfast	F1@{10,25,50}			Edit	MoF
MuCon w/o $\mathcal{L}_\mu$	71.6$_{\pm 0.9}$	64.8$_{\pm 0.8}$	49.1$_{\pm 1.1}$	75.6$_{\pm 0.9}$	63.7$_{\pm 0.8}$
MuCon	**73.1**$_{\pm 0.4}$	**66.3**$_{\pm 0.6}$	**50.7**$_{\pm 1.2}$	**76.4**$_{\pm 0.6}$	**64.1**$_{\pm 1.1}$

Table 4.9: Effect of the MuCon loss on the accuracy in the fully supervised setting. The results are reported for split 1 of the Breakfast dataset.

Breakfast	F1@{10,25,50}			Edit	MoF
ED-TCN (*Lea et al.*, 2017)*	-	-	-	-	43.3
HTK (*Kuehne et al.*, 2017)	-	-	-	-	50.7
TCFPN (*Ding and Xu*, 2018)	-	-	-	-	52.0
HTK(64) (*Kuehne et al.*, 2016)	-	-	-	-	56.3
GRU (*Richard et al.*, 2017)*	-	-	-	-	60.6
GRU+length prior (*Kuehne et al.*, 2020)	-	-	-	-	61.3
MS-TCN (*Abu Farha and Gall*, 2019)	52.6	48.1	37.9	61.7	66.3
MS-TCN++ (*Li et al.*, 2020)	64.1	58.6	45.9	65.6	**67.6**
MuCon	**73.2**$\pm$0.4	**66.1**$\pm$0.5	**48.4**$\pm$0.6	**76.3**$\pm$0.5	62.8$\pm$1.0

Table 4.10: Comparison with the state-of-the-art for fully supervised action segmentation on the Breakfast dataset. (* obtained from *Ding and Xu* (2018)).

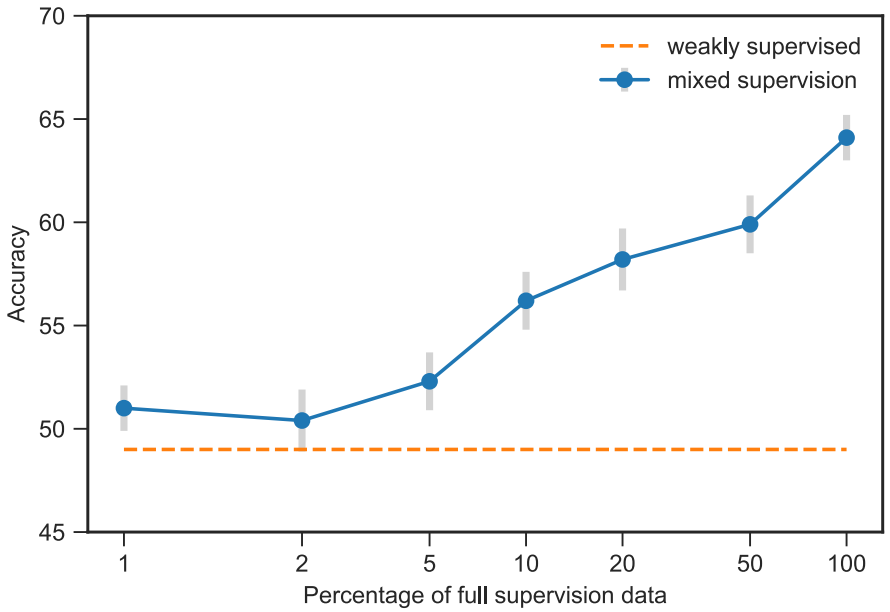


Figure 4.6: MoF accuracy (%) for training with mixed supervision. The x-axis denotes the percentage of videos that are fully annotated. The average accuracy for 5 runs on split 1 of the Breakfast dataset is reported.

Percentage of fully supervised data p%	0%	1%	2%	5%	10%	20%	50%	100%
Accuracy %	49.0$\pm$2.0	51.0$\pm$1.1	50.4$\pm$1.5	52.3$\pm$1.4	56.2$\pm$1.4	58.2$\pm$1.5	59.9$\pm$1.4	64.1$\pm$1.1

Table 4.11: Training with mixed supervision. The average MoF accuracy and standard deviation for 5 runs on split 1 of the Breakfast dataset is reported.



(a) *background* - *spoon powder* - *pour milk* - *stir milk* - *take cup*



(b) *background* - *pour cereals* - *pour milk* - *take bowl* - *stir cereals*



(c) *background* - *pour coffee* - *pour milk* - *take cup* - *add teabag* - *pour water* - *spoon sugar*



(d) *background* - *take plate* - *take knife* - *cut orange* - *squeeze orange* - *take glass* - *pour juice*

Figure 4.7: Qualitative examples for weakly supervised action segmentation on the Breakfast dataset. Each figure visualizes a different video from the test set of split 1. We compare the results from MuCon, NNV, and CDFL with the ground truth (GT). Each row shows the result for the entire duration of a video and the colored segments show when the actions occur in the video.

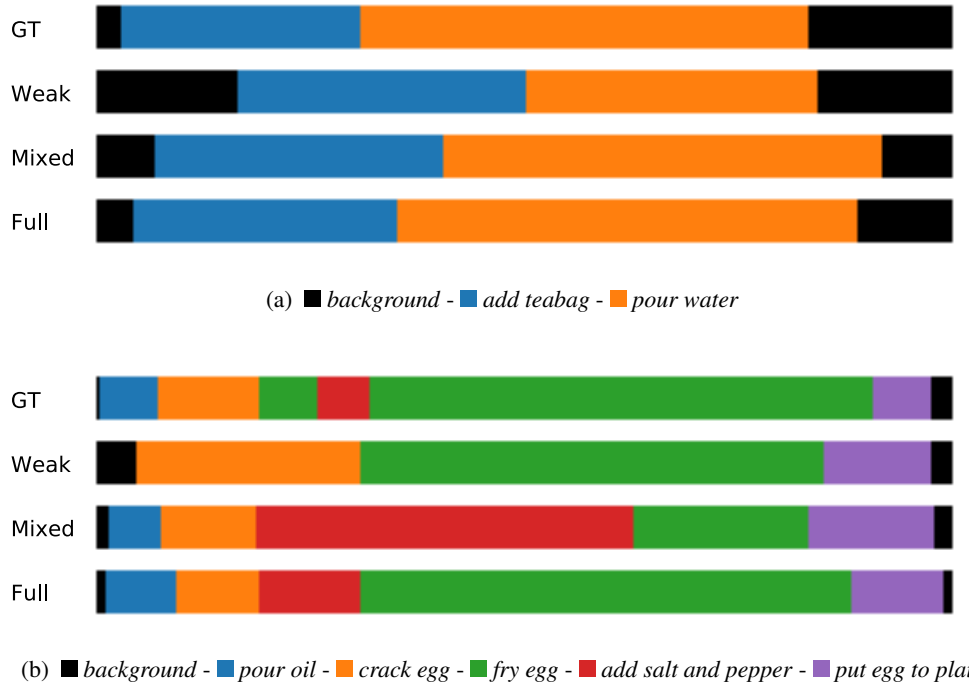


Figure 4.8: Qualitative examples for different levels of supervision on the Breakfast dataset. Each figure visualizes a different video from the test set of split 1. *GT* visualizes the ground truth segmentation and *Weak*, *Mixed*, and *Full* visualize the output of MuCon trained with weak, mixed (10%), or full supervision, respectively.

examples where also CDFL hallucinates actions that are plausible based on the transcripts but that do not occur in the video. In contrast, MuCon does not suffer from hallucinating actions since it infers the transcript directly from the test video and it does not search the training transcript that best aligns to the test video. Figure 4.7d visualizes a failure case where all approaches fail to infer the *take plate* action at the beginning of the video. In this example, CDFL provides a better estimate than MuCon.

Figure 4.8 shows qualitative examples for comparing the different types of supervision, namely weak, mixed (10%), and full supervision. Figure 4.8a shows the result for a test video where MuCon is able to infer the actions correctly using weak, mixed, or full supervision. However, the action boundaries are more accurately estimated if MuCon is trained with more supervision. Figure 4.8b shows a very difficult video where even the fully supervised approach makes a mistake.

4.5 Conclusion

We proposed a novel approach for weakly supervised action segmentation from transcripts. It consists of a two-branch network that predicts two representations for action segmentation. To train the network, we introduced a new mutual consistency loss (MuCon) that enforces these two representations to be consistent during training. Using MuCon and a transcript prediction loss, we can train our network end-to-end without the need for additional steps. We show that the proposed network with the mutual consistency loss achieves an accuracy that is either on par or better than the state-

of-the-art. At the same time, it is much faster during inference and offers the best trade-off between accuracy and inference time. Furthermore, we have shown that the mutual consistency loss increases the accuracy even for fully supervised learning and that the network can be applied to a mixed setting where a few videos are fully annotated and the other videos are only weakly annotated.

CHAPTER 5

Approximate Action Segmentation Inference

This chapter is based on the following publication:
FIFA: Fast Inference Approximation for Action Segmentation
Yaser Souri*, Yazan Abu Farha, Fabien Despinoy, Gianpiero Francesca, Juergen Gall
DAGM German Conference on Pattern Recognition (GCPR), 2021.

Below we will summarize the contributions of each author.

- **Yaser Souri**
  Yaser proposed the main idea of the paper. Yaser implemented the code, performed the experiments, made the qualitative figures and wrote most of the paper.
- **Yazan Abu Farha**
  Yazan contributed by sharing ideas, fixing bugs, experiment design, and analysis of results. Yazan also contributed to writing, creating figures and the fully supervised experiments.
- **Fabien Despinoy and Gianpiero Francesca**
  Both contributed with supervision, discussions, and writing.
- **Juergen Gall**
  Juergen supervised the project. He contributed with discussion and writing.

Contents

5.1 Introduction . . . 58
5.2 Background . . . 59
5.2.1 Action Segmentation . . . 59
5.2.2 Inference in Action Segmentation . . . 60
5.2.3 Exact Inference . . . 61
5.3 FIFA: Fast Inference Approximation . . . 61
5.3.1 Approximate Differentiable Energy E^* . . . 63
5.3.2 Time Complexity . . . 65
5.4 Experiments . . . 66
5.4.1 Evaluation Protocols and Datasets . . . 66
5.4.2 Implementation Details . . . 66
5.4.3 Results and Discussions . . . 66

5.4.4 Comparison to State of the Art 69
5.4.5 Qualitative Results and Visualization 72
5.5 Conclusion 79

In the previous chapter, we introduced a new architecture (MuCon) for weakly supervised and fully supervised temporal action segmentation. The proposed architecture makes it possible to perform action segmentation inference only for a single transcript that is predicted by the network, instead of iteratively performing the action segmentation for all possible transcripts. Although MuCon showed 20 times faster testing time compared to previous state-of-the-art approaches, the action segmentation inference stage in MuCon remains its main efficiency bottleneck. In this chapter, we focus on the inference algorithm itself. We introduce FIFA, a fast approximate inference method for action segmentation and alignment. Unlike previous approaches, FIFA does not rely on expensive dynamic programming for inference. Instead, it uses an approximate differentiable energy function that can be minimized using gradient-descent. FIFA is a general approach that can replace exact inference, improving its speed by more than 5 times while maintaining its performance. FIFA is an anytime inference algorithm that provides a better speed vs. accuracy trade-off compared to exact inference. We apply FIFA on top of state-of-the-art approaches for weakly supervised action segmentation and alignment as well as fully supervised action segmentation. FIFA achieves state-of-the-art results for most metrics on two action segmentation datasets.

5.1 Introduction

Action segmentation is the task of predicting the action label for each frame in the input video. Action segmentation is usually studied in the context of activities performed by a single person, where temporal smoothness of actions are assumed. Fully supervised approaches for action segmentation (*Lea et al.*, 2017; *Abu Farha and Gall*, 2019; *Li et al.*, 2020; *Wang et al.*, 2020) already achieve good performance on this task. Most approaches for fully supervised action segmentation make frame-wise predictions (*Lea et al.*, 2017; *Abu Farha and Gall*, 2019; *Li et al.*, 2020) while trying to model the temporal relationship between the action labels. These approaches usually suffer from over-segmentation. Recent works (*Wang et al.*, 2020; *Ishikawa et al.*, 2021) try to overcome the over-segmentation problem by finding the action boundaries and temporally smoothing the predictions inside each action segment. But these post-processing approaches still can not guarantee temporal smoothness.

Action segmentation inference is the problem of making segment-wise smooth predictions from frame-wise probabilities given a known grammar of the actions and their average lengths (*Richard et al.*, 2018b). The typical inference in action segmentation involves solving an expensive Viterbi-like dynamic programming problem that finds the best action sequence and its corresponding lengths. In the literature, weakly supervised action segmentation approaches (*Kuehne et al.*, 2017; *Richard et al.*, 2017, 2018b; *Li et al.*, 2019; *Souri et al.*, 2021b) usually use inference at test time. Despite being very useful for action segmentation, the inference problem remains the main computational bottleneck in the action segmentation pipeline (*Souri et al.*, 2021b).

In this chapter, we propose FIFA, a fast anytime approximate inference procedure that achieves comparable performance with respect to the dynamic programming based Viterbi decoding inference at a fraction of the computational time. Instead of relying on dynamic programming, we formulate

the energy function as an approximate differentiable function of segment lengths parameters and use gradient-descent-based methods to search for a configuration that minimizes the approximate energy function. Given a transcript of actions and the corresponding initial lengths configuration, we define the energy function as a sum over segment level energies. The segment level energy consists of two terms: a length energy term that penalizes the deviations from a global length model and an observation energy term that measures the compatibility between the current configuration and the predicted frame-wise probabilities. A naive approach to model the observation energy would be to sum up the negative log probabilities of the action labels that are defined based on the length configuration. Nevertheless, such an approach is not differentiable with respect to the segment lengths. In order to optimize the energy using gradient descent-based methods, the observation energy has to be differentiable with respect to the segment lengths. To this end, we construct a plateau-shaped mask for each segment which temporally locates the segment within the video. This mask is parameterized by the segment lengths, the position in the video, and a sharpness parameter. The observation energy is then defined as a product of a segment mask and the predicted frame-wise negative log probabilities, followed by a sum pooling operation. Finally, a gradient descent-based method is used to find a configuration for the segment lengths that minimizes the total energy.

FIFA is a general inference approach and can be applied at test time on top of different action segmentation approaches for fast inference. We evaluate our approach on top of the state-of-the-art methods for weakly supervised temporal action segmentation, weakly supervised action alignment, and fully supervised action segmentation. Results on the Breakfast (*Kuehne et al.*, 2014) and Hollywood extended (*Bojanowski et al.*, 2014) datasets show that FIFA achieves state-of-the-art results on most metrics. Compared to the exact inference using the Viterbi decoding, FIFA is at least 5 times faster. Furthermore, FIFA is an anytime algorithm which can be stopped after each step of the gradient-based optimization, therefore it provides a better speed vs. accuracy trade-off compared to exact inference.

5.2 Background

The following sections introduce all the concepts and notations required to understand the proposed FIFA methodology.

5.2.1 Action Segmentation

In action segmentation, we want to temporally localize all the action segments occurring in a video. In this chapter, we consider the case where the actions are from a predefined set of M classes (a background class is used to cover uninteresting parts of a video). The input video of length T is usually represented as a set of d dimensional features vectors $x_{1:T} = (x_1, \dots, x_T)$. These features are extracted offline and are assumed to be the input to the action segmentation model. The output of action segmentation can be represented in two ways:

- Frame-wise representation $y_{1:T} = (y_1, \dots, y_T)$ where y_t represents the action label at time t.
- Segment-wise representation $s_{1:N} = (s_1, \dots, s_N)$ where segment s_n is represented by both the action label of the segment c_n and its corresponding length ℓ_n, i.e., $s_n = (c_n, \ell_n)$. The ordered list of actions $c_{1:N}$ is usually referred to as the *transcript*.

These two representations are equal and redundant, i.e., it is possible to compute one from the other. We refer to the segment-wise representation as the *structured* representation, because it more explicitly captures the structure (segments and their lengths) in the output. In order to transfer from the segment-wise to the frame-wise representation, we introduce a mapping $\alpha(t; c_{1:N}, \ell_{1:N})$ which outputs the action label at frame t given the segment-wise labeling.

The target labels to train a segmentation model, depend on the level of supervision. In fully supervised action segmentation (*Abu Farha and Gall*, 2019; *Li et al.*, 2020; *Wang et al.*, 2020), the target label for each frame is provided. However, in weakly supervised approaches (*Richard et al.*, 2018b; *Li et al.*, 2019; *Souri et al.*, 2021b) only the ordered list of action labels are provided during training while their lengths are unknown.

Recent fully supervised approaches for action segmentation like MSTCN (*Abu Farha and Gall*, 2019) and its variants directly predict the frame-wise representation $y_{1:T}$ by choosing the action label with the highest probability for each frame independently. These approaches rely on auxiliary losses and the deep neural network to *implicitly* learn the structure in the output label. Although these approaches show promising results, there is no guarantee that the output won't be over segmented and noisy. This results in predictions that are sometimes oversegmented.

Conversely, recent weakly supervised action segmentation approaches like NNV (*Richard et al.*, 2018b) and follow-up work include an inference stage during testing where they explicitly predict the segment-wise representation. This inference stage involves a dynamic programming algorithm for solving an optimization problem which is a computational bottleneck for these approaches.

5.2.2 Inference in Action Segmentation

During testing, the inference stage involves an optimization problem to find the most likely segmentation for the input video, i.e.,

$$c_{1:N}, \ell_{1:N} = \underset{\hat{c}_{1:N}, \hat{\ell}_{1:N}}{\operatorname{argmax}} \Big\{ p(\hat{c}_{1:N}, \hat{\ell}_{1:N} | x_{1:T}) \Big\}. \tag{5.1}$$

Given the transcript $c_{1:N}$, the inference stage boils down to finding the segment lengths $\ell_{1:N}$ by aligning the transcript to the input video, i.e.,

$$\ell_{1:N} = \underset{\hat{\ell}_{1:N}}{\operatorname{argmax}} \Big\{ p(\hat{\ell}_{1:N} | x_{1:T}, c_{1:N}) \Big\}. \tag{5.2}$$

In approaches like NNV (*Richard et al.*, 2018b) and CDFL (*Li et al.*, 2019), the transcript is found by iterating over the transcripts seen during training and selecting the transcript that achieves the most likely alignment by optimizing (5.2). In MuCon (Chapter 4), the transcript is predicted by a sequence to sequence network.

The probability defined in (5.2) is broken down by making independences assumption between frames

$$p(\hat{\ell}_{1:N} | x_{1:T}, c_{1:N}) = \prod_{t=1}^{T} p\big(\alpha(t; c_{1:N}, \hat{\ell}_{1:N}) | x_t\big) \cdot \prod_{n=1}^{N} p\big(\hat{\ell}_n | c_n\big) \tag{5.3}$$

where $p\big(\alpha(t)|x_t\big)$ is referred to as the observation model and $p\big(\ell_n|c_n\big)$ as the length model. Here, $\alpha(t)$ is the mapping from time t to the action label given the segment-wise labeling. The observation model estimates the frame-wise action probabilities and is implemented using a neural network. The

length model is used to constrain the inference defined in (5.2) with the assumption that the lengths of segments for the same action follow a particular probability distribution. The segment length is usually modelled by a Poisson distribution with a class dependent mean parameter λ_{c_n}, i.e.,

$$p\big(\ell_n|c_n\big) = \frac{\lambda_{c_n}^{\ell_n}\exp(-\lambda_{c_n})}{\ell_n!}. \tag{5.4}$$

5.2.3 Exact Inference

The NNV approach (*Richard et al.*, 2018b) proposes an exact solution to the inference problem (5.2) using a Viterbi-like dynamic programming method. This dynamic programming approach was later adopted by CDFL (*Li et al.*, 2019) and MuCon (Chapter 4). First an auxiliary function $Q(t, \ell, n)$ is defined that yields the best probability score for a segmentation up to frame t satisfying the following conditions:

- the length of the last segment is ℓ,
- the last segment was the nth segment with label c_n.

The function Q can be computed recursively. The following two cases are distinguished. The first case defines when no new segment is hypothesized, i.e $\ell > 1$. Then,

$$Q(t, \ell, n) = Q(t-1, \ell-1, n) \cdot p(c_n|x_t), \tag{5.5}$$

with the current frame probability being multiplied with the value of the auxiliary function at the previous frame. The second case is a new segment being hypothesized at frame t, i.e. $\ell = 1$. Then,

$$\begin{aligned} &Q(t, \ell = 1, n) = \\ &\max_{\hat{\ell}}\Big\{Q(t-1, \hat{\ell}, n-1) \cdot p(c_n|x_t) \cdot p(\hat{\ell}|c_{n-1}))\Big\}, \end{aligned} \tag{5.6}$$

where the optimization being calculated over all possible previous segments with length $\hat{\ell}$ and label c_{n-1}. Here the probability of the previous segment having length $\hat{\ell}$ and label c_{n-1} is being multiplied to the previous value of the auxiliary function.

The most likely alignment is given by

$$max_{\ell}\Big\{Q(T, \ell, N) \cdot p(\ell|c_N)\Big\}. \tag{5.7}$$

The optimal lengths can be obtained by keeping track of the maximizing arguments $\hat{\ell}$ from (5.6).

Experiments have shown (*Souri et al.*, 2021b, 2020) that this exact inference stage is the main computational bottleneck of action segmentation approaches.

5.3 FIFA: Fast Inference Approximation

Our goal is to introduce a fast inference algorithm for action segmentation. We want the fast inference to be applicable in both weakly supervised and fully supervised action segmentation. We also want

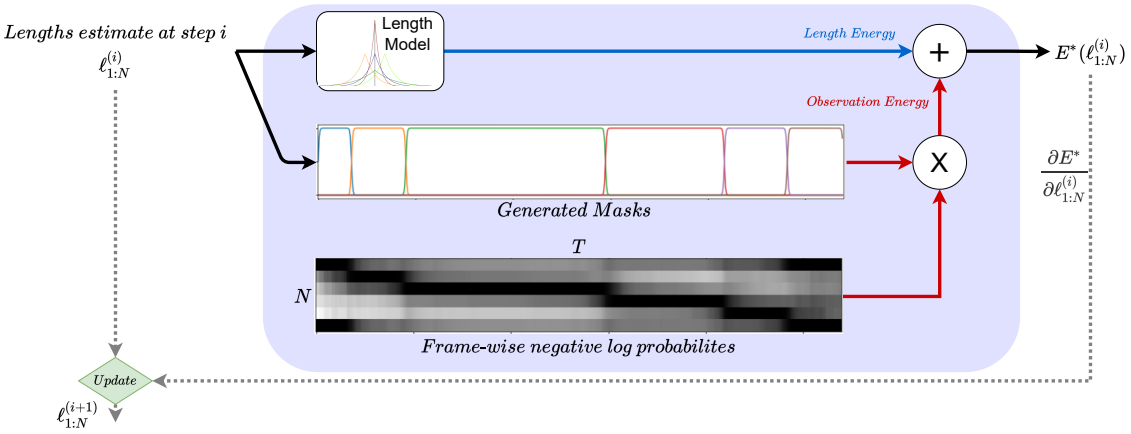


Figure 5.1: Overview of the FIFA optimization process. At each step in the optimization, a set of masks are generated using the current length estimates. Using the generated masks and the frame-wise negative log probabilities, the observation energy is calculated in an approximate but differentiable manner. The length energy is calculated from the current length estimate and added to the observation energy to calculate the total energy value. Taking the gradient of the total energy with respect to the length estimates we can update it using a gradient step.

the fast inference to be flexible enough to work with different action segmentation methods. To this end, we introduce FIFA, a novel approach for fast inference for action segmentation.

In the following, for brevity, we write the mapping $\alpha(t; c_{1:N}, \ell_{1:N})$ simply as $\alpha(t)$. Maximizing probability (5.2) can be rewritten as minimizing the negative logarithm of that probability

$$\operatorname{argmax}\Big\{p(\hat{\ell}_{1:N}|x_{1:T}, c_{1:N})\Big\} = \operatorname{argmin}\Big\{ -\log\big(p(\hat{\ell}_{1:N}|x_{1:T}, c_{1:N})\big)\Big\} \tag{5.8}$$

which we refer to as the energy $E(\ell_{1:N})$. Using (5.3) the energy is rewritten as

$$\begin{aligned} E(\ell_{1:N}) &= -\log\Big(p(\ell_{1:N}|x_{1:T}, c_{1:N})\Big) \\ &= -\log\Big(\prod_{t=1}^{T} p\big(\alpha(t)|x_t\big) \cdot \prod_{n=1}^{N} p\big(\ell_n|c_n\big)\Big) \\ &= \underbrace{\sum_{t=1}^{T} -\log p\big(\alpha(t)|x_t\big)}_{E_o} + \underbrace{\sum_{n=1}^{N} -\log p\big(\ell_n|c_n\big)}_{E_\ell}. \end{aligned} \tag{5.9}$$

The first term in (5.9), E_o is referred to as the observation energy. This term calculates the cost of assigning the labels for each frame and is calculated from the frame-wise probability estimates. The second term E_ℓ is referred to as the length energy. This term is the cost of each segment having a length given that we assume an average length for actions of a specific class.

We propose to optimize the energy defined in (5.9) using gradient based optimization in order to avoid the need for time-consuming dynamic programming. We start with an initial estimate of the lengths (obtained from the length model of each approach or calculated from training data when available) and update our estimate to minimize the energy function.

As the energy function $E(\ell_{1:N})$ is not differentiable with respect to the lengths, we have to calculate a relaxed and approximate energy function $E^*(\ell_{1:N})$ that is differentiable.

5.3.1 Approximate Differentiable Energy E^*

The energy function E as defined in (5.9) is not differentiable in two parts. First the observation energy term E_o is not differentiable because of the $\alpha(t)$ function. Second, the length energy term E_ℓ is not differentiable because it expects natural numbers as input and cannot be computed on real values which are dealt with in gradient-based optimization. Below we describe how we approximate and make each of the terms differentiable.

5.3.1.1 Approximate Differentiable Observation Energy

Consider a $N \times T$ matrix P containing negative log probabilities, i.e.,

$$P[n,t] = -\log p(c_n|x_t). \tag{5.10}$$

Furthermore, we define a mask matrix M with the same size $N \times T$ where

$$M[n,t] = \begin{cases} 0 & if\ \alpha(t) \neq c_n \\ 1 & if\ \alpha(t) = c_n \end{cases}. \tag{5.11}$$

Using the mask matrix we can rewrite the observation energy term as

$$E_o = \sum_{t=1}^{T} \sum_{n=1}^{N} M[n,t] \cdot P[n,t]. \tag{5.12}$$

In order to make the observation energy term differentiable with respect to the length, we propose to construct an approximate differentiable mask matrix M^*. We use the following smooth and parametric plateau function

$$f(t|\lambda^c, \lambda^w, \lambda^s) = \frac{1}{(e^{\lambda^s(t-\lambda^c-\lambda^w)}+1)(e^{\lambda^s(-t+\lambda^c-\lambda^w)}+1)} \tag{5.13}$$

from *Moltisanti et al.* (2019). This plateau function has three parameters and it is differentiable with respect to them: λ^c controls the center of the plateau, λ^w is the width and λ^s is the sharpness of the plateau function.

While the sharpness of the plateau functions λ^s is fixed as a hyper-parameter of our approach, the center λ^c and the width λ^w are computed from the lengths $\ell_{1:N}$. First we calculate the starting position of each plateau function b_n as

$$b_1 = 0, b_n = \sum_{n'=1}^{n-1} \ell_{n'}. \tag{5.14}$$

We can then define both the center and the width parameters of each plateau function as

$$\begin{aligned} \lambda_n^c &= b_n + \ell_n/2, \\ \lambda_n^w &= \ell_n/2 \end{aligned} \tag{5.15}$$

and define each row of the approximate mask as

$$M^*[n,t] = f(t|\lambda_n^c, \lambda_n^w, \lambda^s). \tag{5.16}$$

Now we can calculate a differentiable approximate observation energy similar to (5.12) as

$$E_o^* = \sum_{t=1}^{T}\sum_{n=1}^{N} M^*[n,t] \cdot P[n,t]. \tag{5.17}$$

5.3.1.2 Approximate Differentiable Length Energy

For the gradient-based optimization, we must relax the length values to be positive real values instead of natural numbers. As the Poisson distribution (5.4) is only defined on natural numbers, we propose to use a substitute distribution defined on real numbers. As a replacement, we experiment with a Laplace distribution and a Gaussian distribution. In both cases, the scale or the width parameter of the distribution is assumed to be fixed.

We can rewrite the length energy E_ℓ as the approximate length energy

$$E_\ell^*(\ell_{1:N}) = \sum_{n=1}^{N} -\log p(\ell_n|\lambda_{c_n}^\ell), \tag{5.18}$$

where $\lambda_{c_n}^\ell$ is the expected value for the length of a segment from the action c_n. In case of the Laplace distribution this length energy will be

$$E_\ell^*(\ell_{1:N}) = \frac{1}{Z}\sum_{n=1}^{N} |\ell_n - \lambda_{c_n}^\ell|, \tag{5.19}$$

where Z is the constant normalization factor. This means that the length energy will penalize any deviation from the expected average length linearly. Similarly, for the Gaussian distribution, the length energy will be

$$E_\ell^*(\ell_{1:N}) = \frac{1}{Z}\sum_{n=1}^{N} |\ell_n - \lambda_{c_n}^\ell|^2, \tag{5.20}$$

which means that the Gaussian length energy will penalize any deviation from the expected average length quadratically.

With the objective to maintain a positive value for the length during the optimization process, we estimate the length in log space and convert it to absolute space only in order to compute both the approximate mask matrix M^* and the approximate length energy E_ℓ^*.

5.3.1.3 Approximate Energy Optimization

The total approximate energy function is defined as a weighted sum of both the approximate observation and the approximate length energy functions

$$E^*(\ell_{1:N}) = E_o^*(\ell_{1:N}, Y) + \beta E_\ell^*(\ell_{1:N}) \tag{5.21}$$

where β is the multiplier for the length energy.

Given an initial length estimate $\ell^0_{1:N}$, we iteratively update this estimate to minimize the total energy. Figure 5.1 illustrates the optimization step for our approach. During each optimization step, we first calculate the energy E^* and then calculate the gradients of the energy with respect to the length values. Using the calculated gradients, we update the length estimate using a gradient descent update rule such as SGD or Adam. After a certain number of gradient steps (50 steps in our experiments) we will finally predict the segment length.

If the transcript for a test video is provided then it is used ,e.g,. using the MuCon approach (Chapter 4) or in a weakly supervised action alignment setting. However, if the latter is not known ,e.g., in a fully supervised approach or CDFL (*Li et al.*, 2019) for weakly supervised action segmentation, we perform the optimization for each of the transcripts seen during training and select the most likely one based on the final energy value at the end of the optimization.

The initial length estimates are calculated from the length model of each approach in case of a weakly supervised setting whereas in a fully supervised setting the average length of each action class is calculated from the training data and used as the initial length estimates. The initial length estimates are also used as the expected length parameters for the length energy calculations.

The hyper-parameters like the choice of the optimizer, number of steps, optimization step size, and the mask sharpness, remain as the hyper-parameters of our approach.

5.3.2 Time Complexity

5.3.2.1 Time Complexity of Exact Inference

The time complexity of the above exact inference is quadratic in the length of the video T and linear in the number of segments N. As input videos for action segmentation are usually long, it becomes computationally expensive to calculate. In practice, *Richard et al.* (2018b); *Li et al.* (2019); *Souri et al.* (2021b) limit the maximum size of each segment to a fixed value of $L = 2000$. The final time complexity of exact inference is $O(LNT)$. Furthermore, this optimization process is inherently not parallelizable. This is due to the $\max$ operation in (5.6).

5.3.2.2 Time Complexity of FIFA

At each optimization step, the time complexity is $O(NT)$, where N is the number of segments and T is the length of the video because we must create the M^* matrix and calculate the element-wise multiplication. Overall, the FIFA time complexity is $O(MNT)$, where M is the number of optimization steps. Compared to the exact inference which has a time complexity of $O(LNT)$, where L is the fixed value of 2000, our time complexity is lower since M is usually 50 steps and N is on average 10.

We also want to mention that the proposed approach is inherently a parallelizable optimization method (i.e. values of the mask, the element-wise multiplication, and the calculation of the gradient for each time step can be calculated in parallel) and is independent of any other time step values. This is in contrast to the dynamic programming approaches where the intermediate optimization values for each time step depend on the value of the previous time steps.

5.4 Experiments

5.4.1 Evaluation Protocols and Datasets

We evaluate FIFA on 3 different tasks: weakly supervised action segmentation, fully supervised action segmentation, and weakly supervised action alignment. We obtain the source code for the state-of-the-art approaches on each of these tasks and train a model using the standard training configuration of each model. Then we apply FIFA as a replacement for an existing inference stage or as an additional inference stage.

We evaluate our model using the Breakfast (*Kuehne et al.*, 2014) and Hollywood extended (*Bojanowski et al.*, 2014) datasets on the 3 different tasks. Details of the datasets are included in Section 2.5.

5.4.2 Implementation Details

We implement our approach using the PyTorch (*Paszke et al.*, 2019) library. For all experiments we set the number of FIFA's gradient-based optimization steps to 50 and we use the Adam (*Kingma and Ba*, 2015) optimization algorithm. Mask sharpness and the optimization step size is chosen depending on the approach that FIFA is applied on top of. When applying FIFA on top of MuCon (Chapter 4) we use 0.3 as the optimization step size and set the mask sharpness to 1.75. For CDFL (*Li et al.*, 2019), we set the mask sharpness to 0.1 and the optimization step size to 0.15. Looking at the visualization in Figure 5.12 it is clear that CDFL provides noisy framewise probability estimates. For this reason a lower mask sharpness is prefered. When applying FIFA on top of fully supervised approaches like MS-TCN (*Abu Farha and Gall*, 2019) we use mask sharpness value of 15 and optimization step size of 0.02. Looking at the visualization in Figure 5.18 we see that fully supervised approaches provide clean smooth framewise probabilities and having a sharp mask is recommented in these settings.

5.4.3 Results and Discussions

In this section, we study the speed-accuracy trade-off and the impact of the length model.

5.4.3.1 Speed vs. Accuracy Trade-off.

One of the major benefits of FIFA is that it is any-time. It provides the flexibility of choosing the number of optimization steps. The number of steps of the optimization can be a tool to trade-off speed vs. accuracy. In exact inference, we can use frame-sampling, i.e., lowering the resolution of the input features, or hypothesis pruning, i.e., beam search for speed vs. accuracy trade-off.

Figure 5.2 plots the speed vs. accuracy trade-off of exact inference compared to FIFA. We observe that FIFA provides a much better speed-accuracy trade-off as compared to frame-sampling for exact inference. The best performance is achieved after 50 steps with 5.9% improvement on the MoF accuracy compared to not performing any optimization (0 steps).

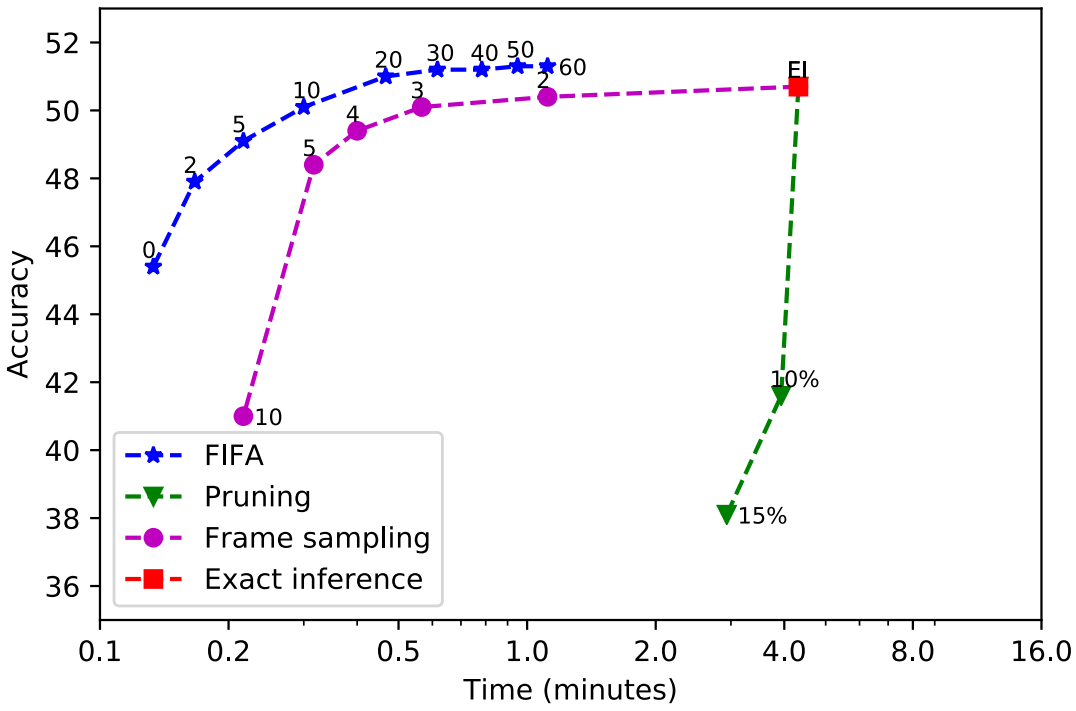


Figure 5.2: Speed vs. accuracy trade-off of different inference approaches applied to the MuCon method. Using FIFA we can achieve a better speed vs. accuracy trade-off compared to frame sampling or hypothesis pruning in exact inference.

5.4.3.2 Impact of the Length Energy Multiplier.

For the length energy, we assume that the segment lengths follow a Laplace distribution. Figure 5.3 shows the impact of the length energy multiplier on the weakly supervised action segmentation performance on the Breakfast dataset. The choice of this parameter depends on the dataset. While the best accuracy is achieved with a multiplier of 0.05, our approach is robust to the choice of these hyper-parameters on this dataset. We further experimented with a Gaussian length energy. However, as shown in the figure, the performance is much worse compared to the Laplace energy. This is due to the quadratic penalty that dominates the total energy, which makes the optimization biased towards the initial estimate and ignores the observation energy.

5.4.3.3 Impact of the Length Initialization.

Since FIFA starts with an initial estimate for the lengths, the choice of initialization might have an impact on the performance. Table 5.1 shows the effect of initializing the lengths with equal values compared to using the length model of MuCon (Chapter 4) for the weakly supervised action segmentation on the Breakfast dataset. As shown in the table, FIFA is more robust to initialization compared to the exact inference as the drop in performance is approximately half of the exact inference.

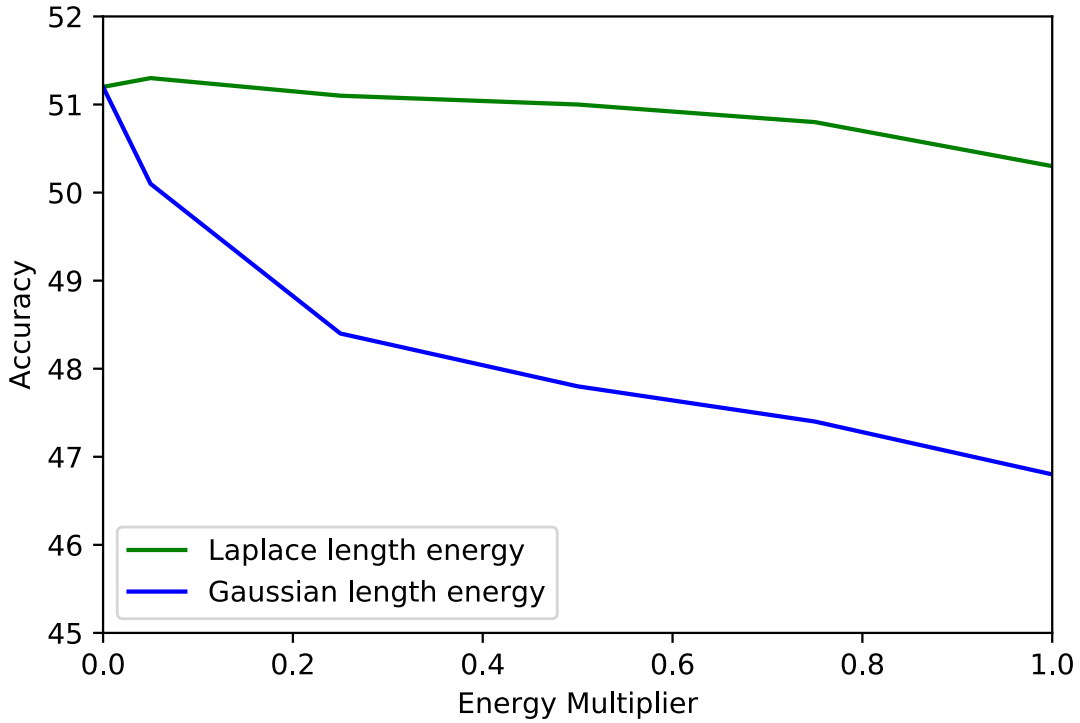


Figure 5.3: Effect of the length energy multiplier for Laplace and Gaussian length energy. Accuracy is calculated on the Breakfast dataset using FIFA applied to the MuCon approach trained in the weakly supervised action segmentation setting.

5.4.3.4 Optimization Algorithm and Its Step Sizer

The choice of the optimization algorithm used to update the length estimates using the calculated gradients is one of the hyper-parameters of our approach. We have experimented with two optimization algorithms, SGD and Adam. As shown in Figure 5.4, the best performing value for the step size hyper-parameter depends on the optimization algorithm used. For SGD a low value of 0.001 achieves the best performance with higher values causing major drops in performance. On the other hand, Adam optimizer works well with a range of step size values as it has an internal mechanism to adjust the step size. The best performance for Adam is observed at 0.3.

We further investigate and notice that the reason SGD performs so poorly for large values of the

Inference Method	Initialization	MoF
Exact	MuCon (Chapter 4)	**50.7**
	Equal	48.8 (-1.9)
FIFA	MuCon (Chapter 4)	**51.3**
	Equal	50.2 (-1.1)

Table 5.1: Impact of the length initialization for MuCon using exact inference and FIFA for weakly supervised action segmentation on the Breakfast dataset.

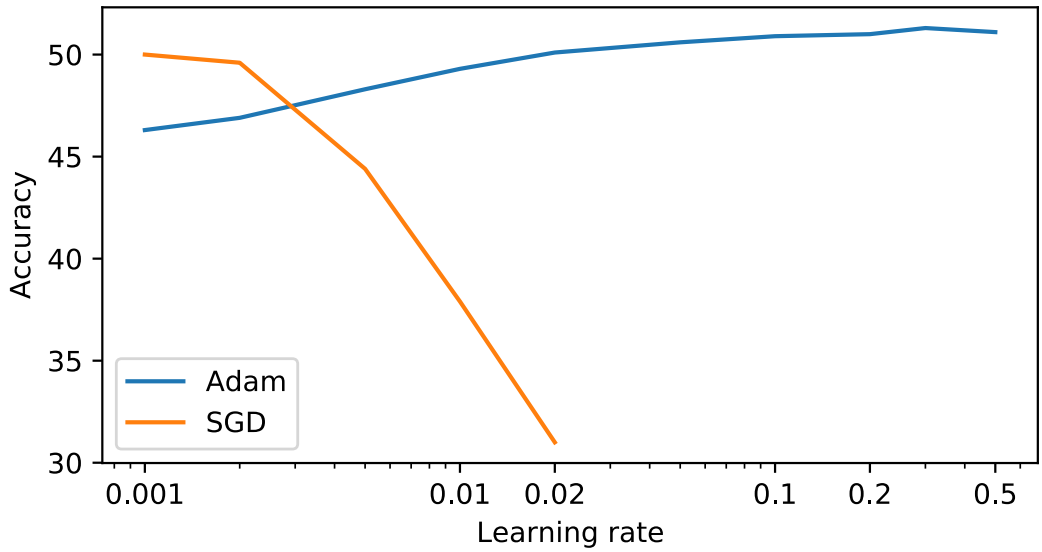


Figure 5.4: Effect of the optimization step size on the performance of weakly supervised action segmentation using FIFA applied on the MuCon approach. Accuracy is calculated on the Breakfast dataset.

step size is that it fluctuates and is not able to optimize the energy effectively. Figure 5.5 shows the value of the approximate energy during the optimization for Adam and SGD for the same inference. We observe that a large optimization step size causes SGD to fluctuate while Adam is stable and achieves a lower energy value at the end of the optimization.

5.4.4 Comparison to State of the Art

In this section, we compare FIFA to other state-of-the-art approaches.

5.4.4.1 Weakly Supervised Action Segmentation.

We apply FIFA on top of two state-of-the-art approaches for weakly supervised action segmentation namely MuCon (Chapter 4) and CDFL (*Li et al.*, 2019) on the Breakfast dataset (*Kuehne et al.*, 2014) and report the results in Table5.2. FIFA applied on CDFL achieves a 12 times faster inference speed while obtaining results comparable to exact inference. FIFA applied to MuCon achieves a 5 times faster inference speed and obtains a new state-of-the-art performance on the Breakfast dataset on most of the metrics.

Similarly for the Hollywood extended dataset (*Bojanowski et al.*, 2014) we apply FIFA to MuCon (Chapter 4) and report the results in Table 5.4. FIFA applied on MuCon achieves a 4 times faster inference speed while obtaining results comparable to exact inference.

5.4.4.2 Fully Supervised Action Segmentation.

In the fully supervised action segmentation we apply FIFA on top of MS-TCN (*Abu Farha and Gall*, 2019) and its variant MS-TCN++ (*Li et al.*, 2020) on the Breakfast dataset (*Kuehne et al.*, 2014) and report the results in Table 5.3. MS-TCN and MS-TCN++ are approaches that do not perform any

Method	MoF	MoF-BG	IoU	IoD	Time (min)
ISBA (*Ding and Xu*, 2018)	38.4	38.4	24.2	40.6	0.01
NNV (*Richard et al.*, 2018b)	43.0	-	-	-	234
D3TW (*Chang et al.*, 2019)	45.7	-	-	-	-
CDFL (*Li et al.*, 2019)	50.2	48.0	33.7	45.4	-
CDFL*	49.4	47.5	35.2	46.4	260
FIFA + CDFL*	47.9	46.3	34.7	48.0	20.4 ($\times$12.8)
MuCon (Chapter 4)	47.1	-	-	-	-
MuCon*	50.7	50.3	40.9	**54.0**	4.1
FIFA + MuCon*	**51.3**	**50.7**	**41.1**	53.3	0.8 ($\times$5.1)

Table 5.2: Results for weakly supervised action segmentation on the Breakfast dataset. * indicates results obtained by running the code on our machine.

Method	F1@{10, 25, 50}			Edit	MoF
BCN (*Wang et al.*, 2020)	68.7	65.5	55.0	66.2	**70.4**
ASRF (*Ishikawa et al.*, 2021)	74.3	68.9	**56.1**	72.4	67.6
MS-TCN++ (*Li et al.*, 2020)	64.1	58.6	45.9	65.6	67.6
FIFA + MS-TCN++	74.3	69.0	54.3	77.3	67.9
MS-TCN (*Abu Farha and Gall*, 2019)	52.6	48.1	37.9	61.7	66.3
FIFA + MS-TCN	**75.5**	**70.2**	54.8	**78.5**	68.6

Table 5.3: Results for fully supervised action segmentation setup on the Breakfast dataset.

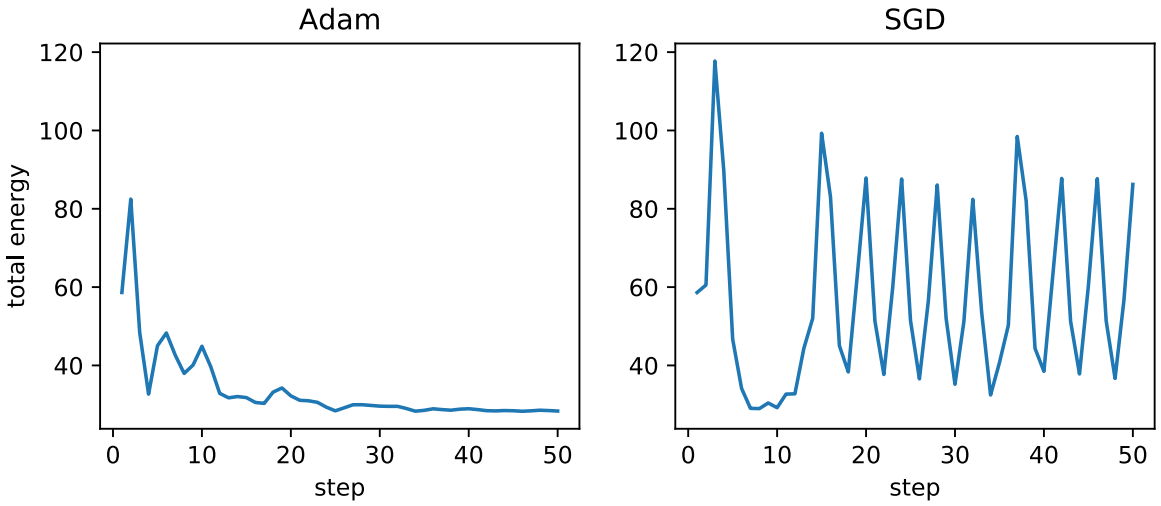


Figure 5.5: The value of the approximate energy during FIFA optimization for SGD and Adam optimizers for the same inference.

Method	MoF-BG	IoU	Time (speedup)
ISBA (*Ding and Xu*, 2018)	34.5	12.6	-
D3TW (*Chang et al.*, 2019)	33.6		-
CDFL (*Li et al.*, 2019)	40.6	19.5	-
MuCon (Chapter 4)	41.6		-
MuCon*	40.1	13.9	53
MuCon + FIFA*	41.2	13.7	13 (×4.1)

Table 5.4: Results for weakly supervised action segmentation on the Hollywood extended dataset. Time is reported in seconds. * indicates results obtained by running the code on our machine.

inference at test time. This usually results in over-segmentation and low F1 and Edit scores. Applying FIFA on top of these approaches improves the F1 and Edit scores significantly. FIFA applied on top of MS-TCN achieves state-of-the-art performance and sets new state-of-the-art performance on most metrics.

For the Hollywood extended dataset (*Bojanowski et al.*, 2014), we train MS-TCN (*Abu Farha and Gall*, 2019) and report results comparing exact inference (EI) compared to FIFA in Table 5.5. We observe that MS-TCN using an inference algorithm achieves new state-of-the-art results on this dataset. FIFA is comparable or better than exact inference on this dataset.

5.4.4.3 Weakly Supervised Action Alignment

Similar to weakly supervised action segmentation, we apply FIFA on top of CDFL and MuCon for weakly supervised action alignment task on the Breakfast dataset and report the results in Table 5.6. Our experiments show that FIFA applied on top of CDFL achieves state-of-the-art or better than state-of-the-art results on MoF and Mof-BG metrics, whereas FIFA applied on top of MuCon achieves

Method	MoF	MoF-BG	IoU	IoD
HTK (*Kuehne et al.*, 2017)	39.5		8.4	
ED-TCN (*Lea et al.*, 2017)	36.7	27.3	10.9	13.1
ISBA (*Ding and Xu*, 2018)	54.8	33.1	20.4	28.8
MSTCN (Chapter 4) (+ EI)	64.9	35.0	22.6	33.2
MSTCN + FIFA	66.2	34.8	23.9	35.8

Table 5.5: Results for fully supervised action segmentation on the Hollywood extended dataset. EI stands for Exact Inference.

Method	MoF	MoF-BG	IoU	IoD
ISBA (*Ding and Xu*, 2018)	53.5	51.7	35.3	52.3
D^3TW (*Chang et al.*, 2019)	57.0	-	-	56.3
CDFL (*Li et al.*, 2019)	63.0	61.4	45.8	63.9
ADP (*Ghoddoosian et al.*, 2021)	64.1	**65.5**	43.0	-
FIFA + CDFL*	**65.3**	64.3	46.3	61.3
FIFA + MuCon*	61.4	61.2	**48.4**	**64.1**

Table 5.6: Results for weakly supervised action alignment on the Breakfast dataset.

state-of-the-art results for IoD and IoU metrics.

5.4.5 Qualitative Results and Visualization

In this section we show various qualitative results of applying FIFA for action segmentation. In each figure on the right, the approximate total energy value is plotted as a function of number of steps. On the left, at the top, the framewise negative log probabilities (P) are visualized. The ground truth segmentation, optimization initialization, the generated masks and the segmentation obtained after approximate inference using FIFA is visualized in rows 2 to 5. The MoF metric is also calculated for a single video and reported for the optimization initialization and the approximate decoding.

Figures 5.6-5.10 show qualitative examples of applying FIFA on top of MuCon (Chapter 4) for weakly supervised action segmentation. Figures 5.12-5.16 show qualitative examples of applying FIFA on top of CDFL (*Li et al.*, 2019) for weakly supervised action segmentation. Figures 5.18-5.22 show qualitative examples of applying FIFA on top of MSTCN (*Abu Farha and Gall*, 2019) for fully supervised action segmentation.

5.4.5.1 Failure Cases

Figures 5.11, 5.17 and, 5.23 show failure cases for FIFA + MuCon, FIFA + CDFL and, FIFA + MSTCN respectively. We observe that the major failure case is when the optimization is initialized with an incorrect transcript (Figures 5.11 and 5.17). Another failure mode is when the predicted negative log probabilities are not correct (Figure 5.23) which causes the boundaries of actions to be in the wrong location.

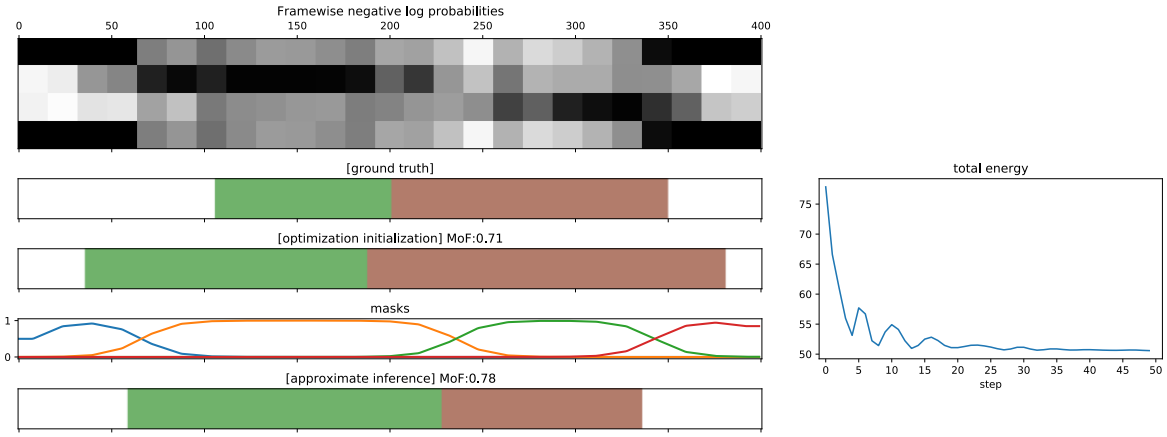


Figure 5.6: Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon

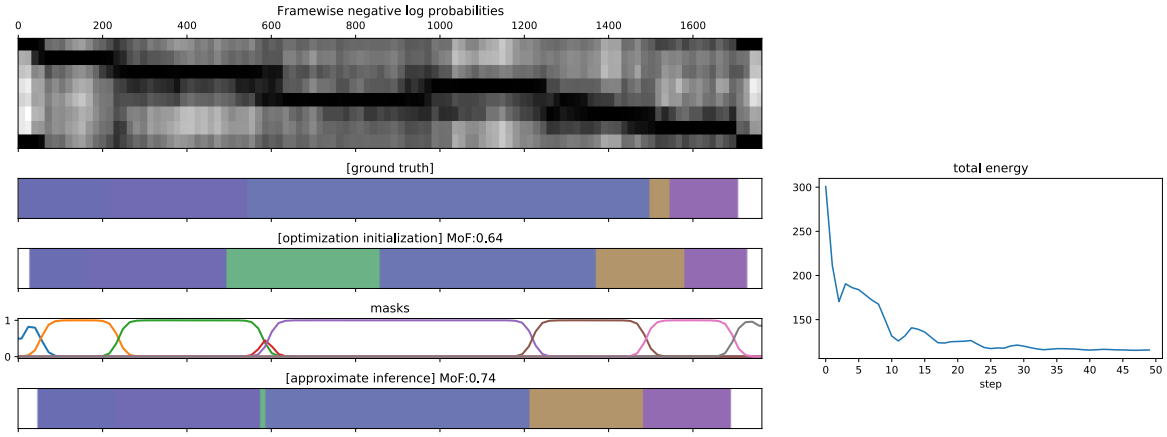


Figure 5.7: Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon

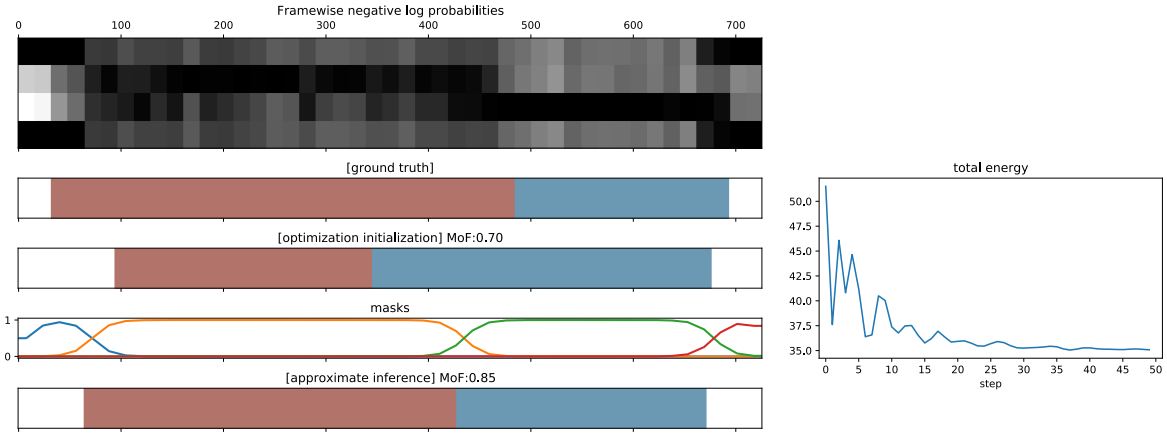


Figure 5.8: Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon

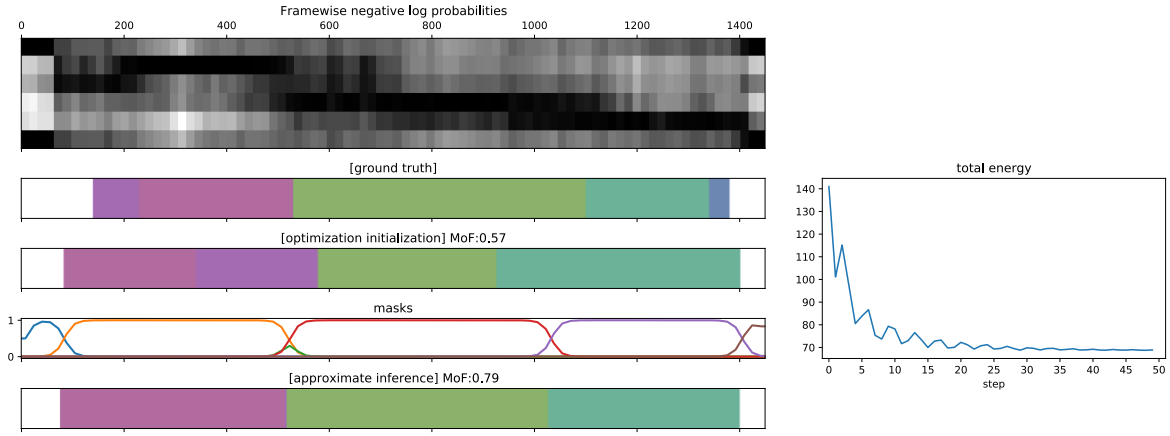


Figure 5.9: Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon

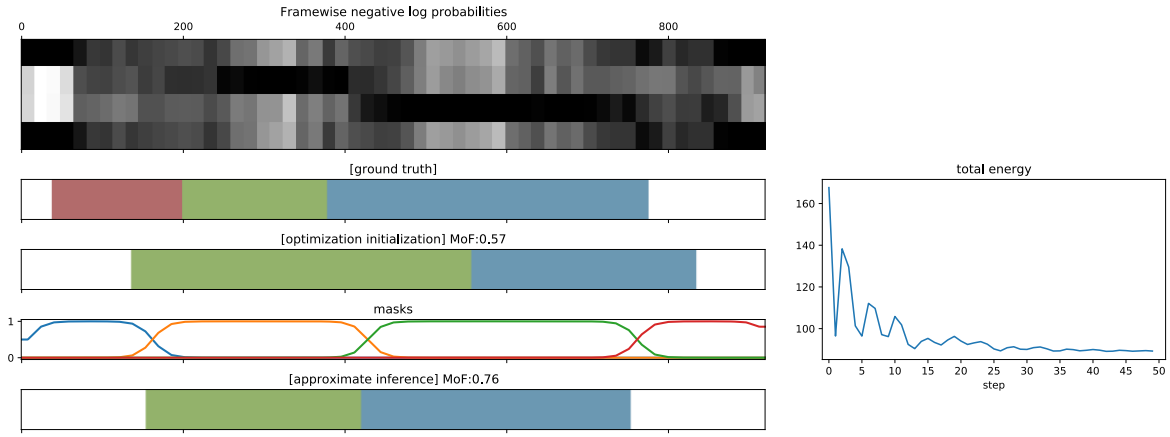


Figure 5.10: Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon

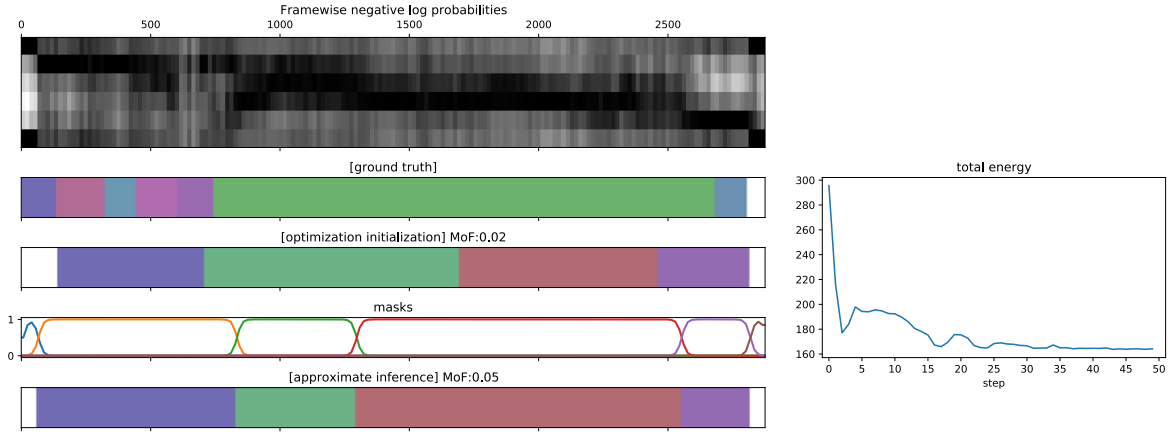


Figure 5.11: Qualitativ Result: Weakly supervised action segmentation, FIFA + MuCon, Failure Case

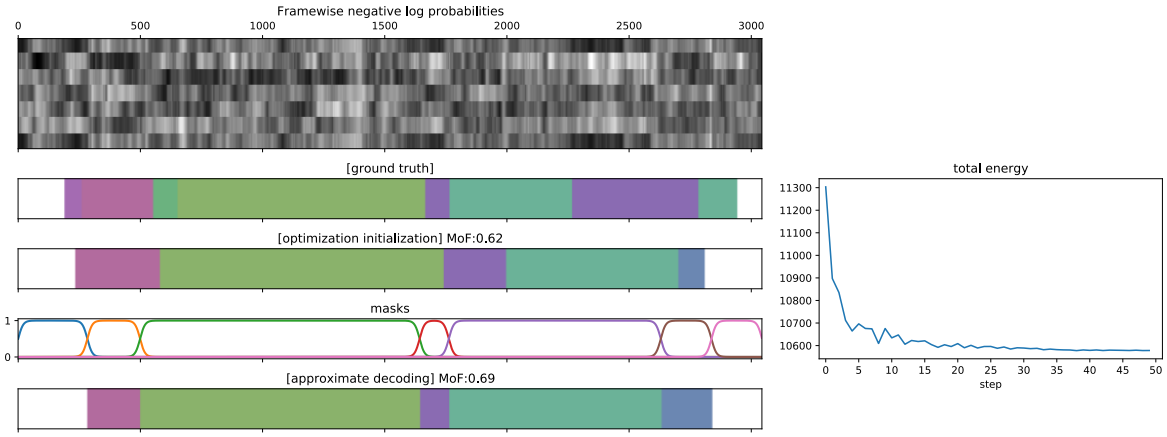


Figure 5.12: Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL

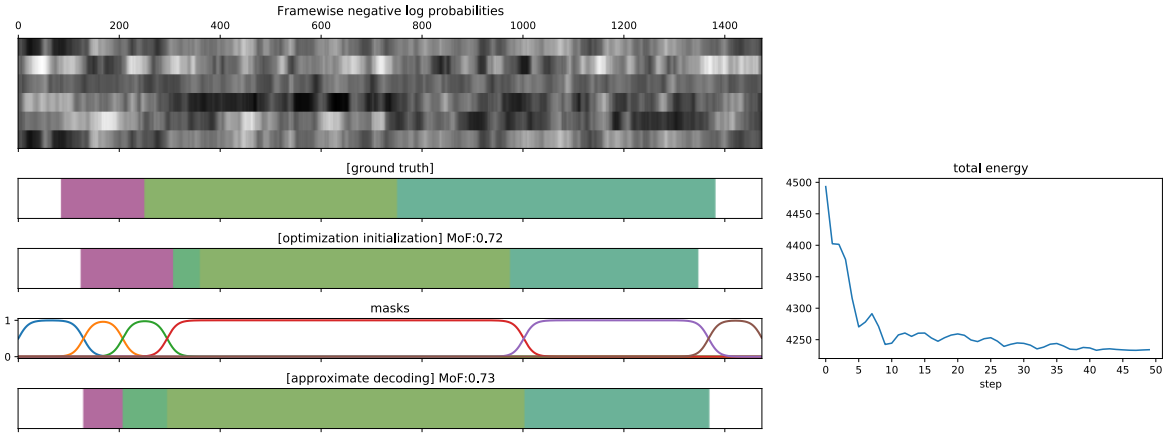


Figure 5.13: Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL

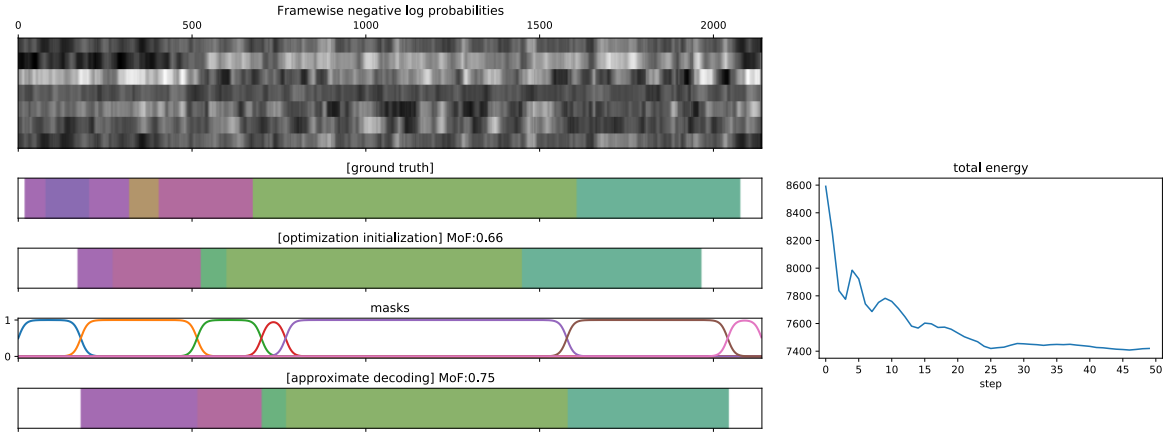


Figure 5.14: Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL

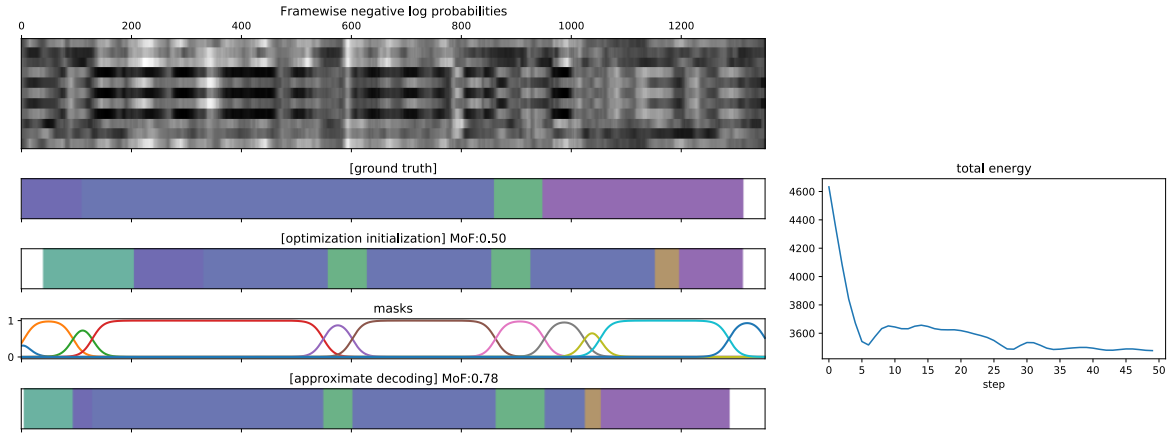


Figure 5.15: Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL

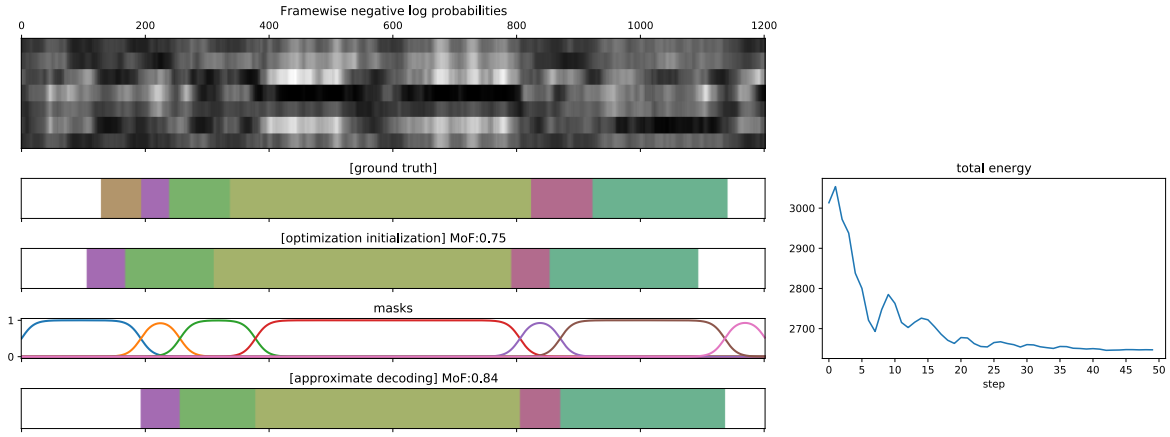


Figure 5.16: Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL

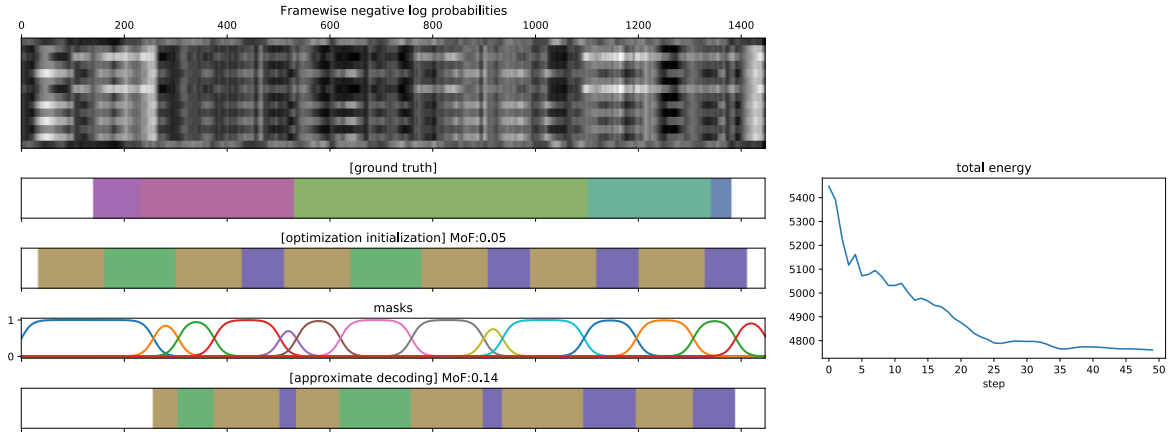


Figure 5.17: Qualitativ Result: Weakly supervised action segmentation, FIFA + CDFL, Failure Case

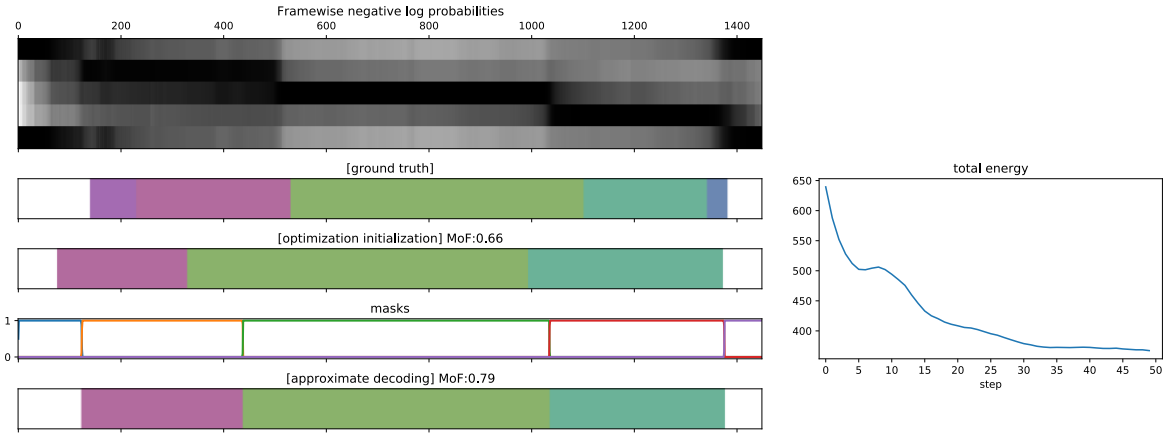


Figure 5.18: Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN

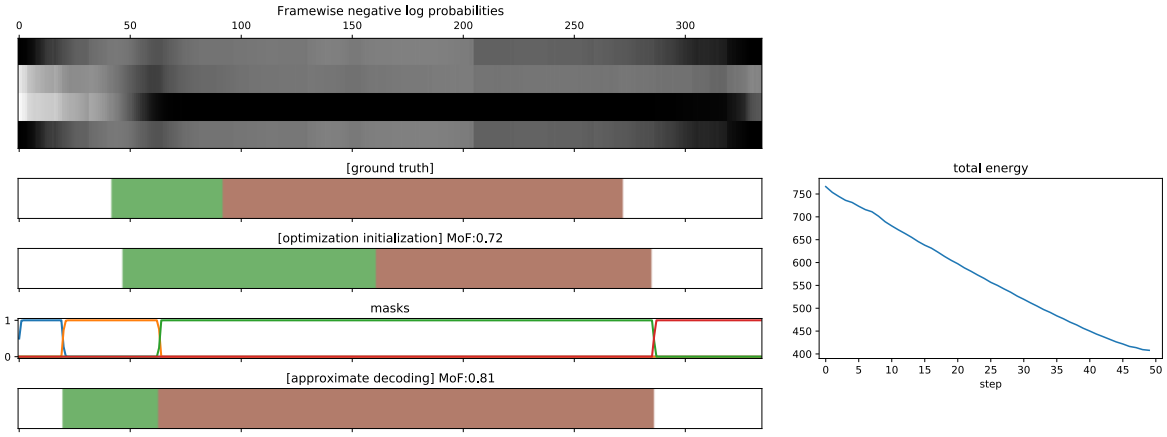


Figure 5.19: Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN

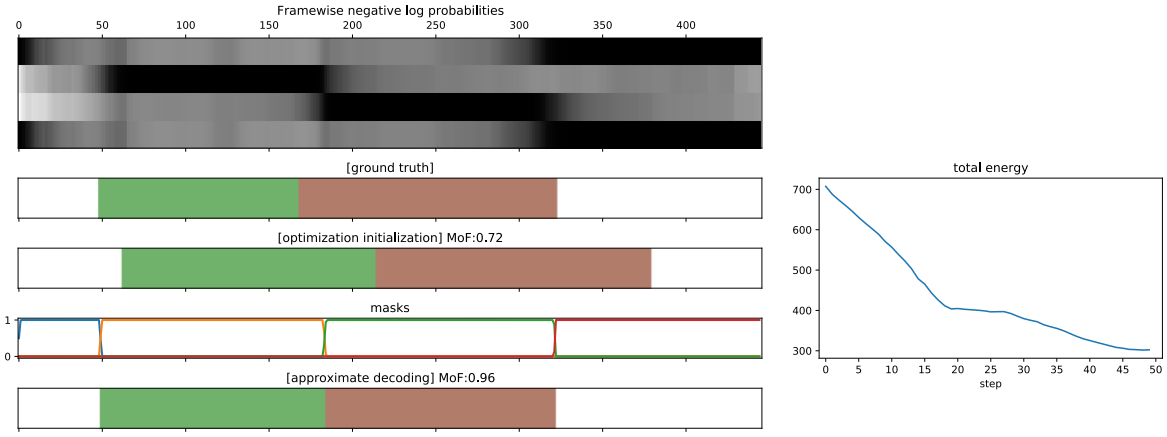


Figure 5.20: Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN

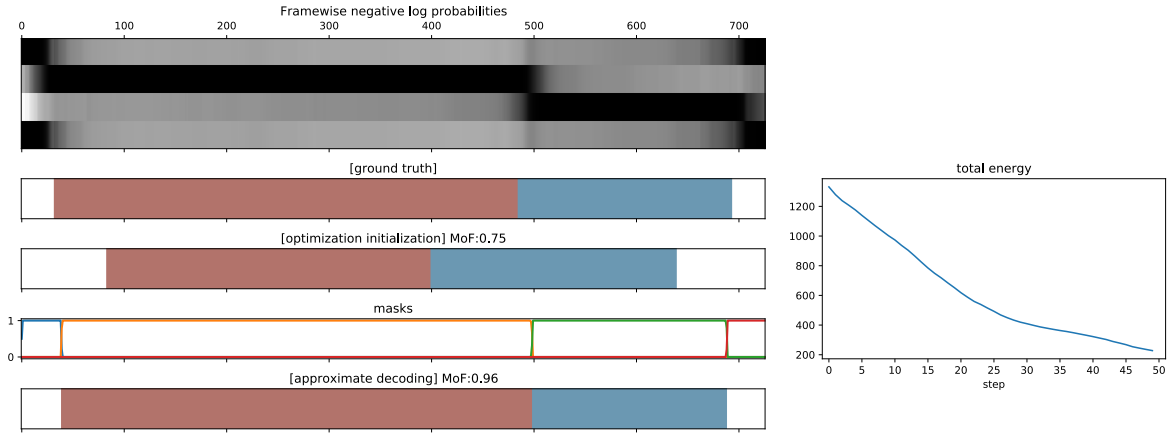


Figure 5.21: Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN

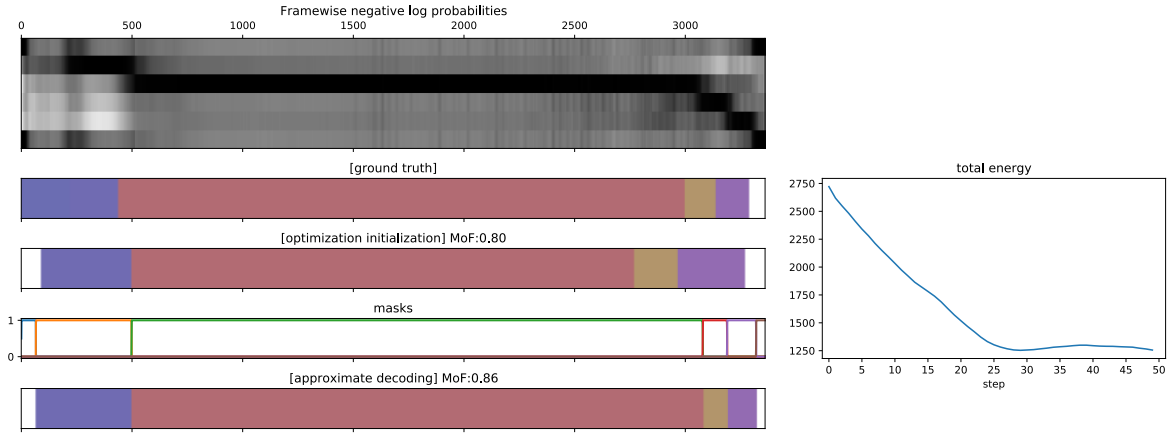


Figure 5.22: Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN

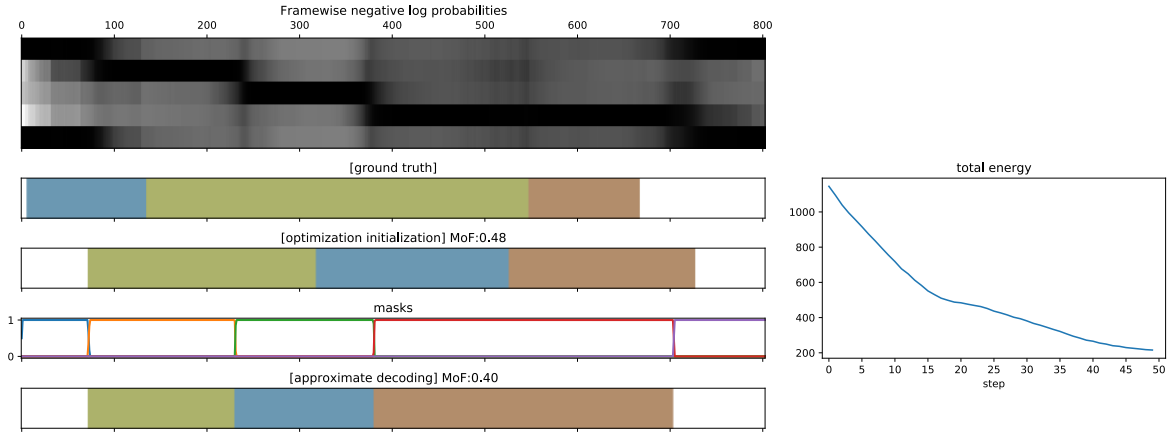


Figure 5.23: Qualitativ Result: Fully supervised action segmentation, FIFA + MSTCN, Failure Case

5.5 Conclusion

In this chapter, we proposed FIFA a fast approximate inference procedure for action segmentation and alignment. Unlike previous methods, our approach does not rely on dynamic programming-based Viterbi decoding for inference. Instead, FIFA optimizes a differentiable energy function that can be minimized using gradient-descent which allows for a fast and accurate inference during testing. We evaluated FIFA on top of fully and weakly supervised methods trained on the Breakfast and Hollywood extended datasets. The results show that FIFA is able to achieve comparable or better performance, while being at least 5 times faster than exact inference.

CHAPTER 6

Robust Action Segmentation from Timestamp Supervision

This chapter is based on the following publication:
Robust Action Segmentation From Timestamp Supervision
Yaser Souri*, Yazan Abu Farha*, Emad Bahrami*, Gianpiero Francesca, Juergen Gall
The British Machine Vision Conference (BMVC), 2022.

Below we will summarize the contributions of each author.

- **Yaser Souri and Yazan Abu Farha**
  Both contributed equally to this work. Yaser and Yazan equally contributed to the idea, implementation, experiments, hyper-parameter tuning and writing. Yazan created the simulated timestamp datasets with missing annotations. Yazan also performed the baseline experiments. Yaser implemented the initial version of the proposed pseudo ground truth generation. Yazan refined the implementation of the pseudo ground truth generation and formulated the final constraints for the optimization. Yaser performed most of the ablation experiments. Yaser and Yazan both performed the experiments for temporal action localization.

- **Gianpiero Francesca**
  Gianpiero contributed with supervision, discussions, and writing.

- **Juergen Gall**
  Juergen supervised the project. He proposed the idea of dealing with missing annotations in timestamp supervision. He contributed with discussion and writing.

Contents

6.1 Introduction 82
6.2 Proposed Method 84
6.2.1 Timestamp Supervision for Action Segmentation 85
6.2.2 Robust Pseudo Ground Truth Generation 85
6.3 Experiments 88
6.3.1 Datasets and Metrics 88
6.3.2 Implementation Details 89
6.3.3 Impact of Missing Annotations 89
6.3.4 Comparison with Baselines 90
6.3.5 Impact of Initialization 92

6.3.6 Evaluation of Different Timestamps Selection Strategies 92
6.3.7 Sensitivity to β 95
6.3.8 Performance on Timestamps without Missing Annotations 95
6.3.9 Localization Experiment 96
6.3.10 Speed of Pseudo Ground Truth Generation 96
6.4 Limitations 98
6.5 Conclusion 98

In the previous two chapters we focused on fast approaches for temporal action segmentation using full and weak supervision in the form of transcripts. We further proposed an approximate and fast temporal action segmentation inference. Recently weak supervision in the form of timestamps has seen a lot of attention from the video understanding community for action recognition, temporal action localization and temporal action segmentation. In temporal action segmentation timestamp supervision is a promising type of weak supervision as it is less expensive to obtain compared to full supervision and results in accurate action segmentation models. Previous works assume that every action segment is annotated with a timestamp which is a restrictive assumption. In this chapter we try to relax this restrictive assumption and take missing annotations for some segments into account. Robustness against missing annotations is of crucial importance as such mistakes occur during the annotation process. We show that our approach achieves much better action segmentation results compared to competitive approaches and various baselines.

6.1 Introduction

Understanding untrimmed videos has numerous applications in the real world such as home monitoring systems (*Dai et al.*, 2020), worker monitoring and guidance in the assembly line (*Kumar et al.*, 2021), tutorial generation from instructional videos (*Zhukov et al.*, 2019; *Tang et al.*, 2019) and more.

Action segmentation deals with the temporal understanding of actions that occur in an untrimmed video. A video is assumed to contain a number of non-overlapping consecutive action segments (Figure 6.1). The output of action segmentation is an action label for each frame of the input video. This would identify the resulting action segments and their labels.

Recently, fully supervised approaches for action segmentation (*Abu Farha and Gall*, 2019; *Li et al.*, 2020; *Souri et al.*, 2021a) have achieved encouraging results. Despite that, a major bottleneck that prevents the widespread adoption of action segmentation technologies is the cost of obtaining full supervision for large-scale datasets. Full supervision is a labor-intensive endeavor as it consists of annotating every instance of the actions of interest with their exact temporal boundaries. Furthermore, in many cases, the exact temporal boundary of actions is subjective and can lead to annotation inconsistencies. To address the issues of full supervision, a lot of effort has been allocated to weakly supervised approaches for action segmentation from transcripts (*Souri et al.*, 2021b; *Richard et al.*, 2018b; *Li et al.*, 2019) or action sets (*Li and Todorovic*, 2020; *Fayyaz and Gall*, 2020; *Richard et al.*, 2018a). Transcripts are the ordered set of actions that occur in a video without any temporal information about the start and end of each action instance. Transcript-level supervision is orders of magnitude cheaper to obtain compared to full supervision. Action sets are even a weaker form of supervision which contain only the set of actions that are present in the video without any order.

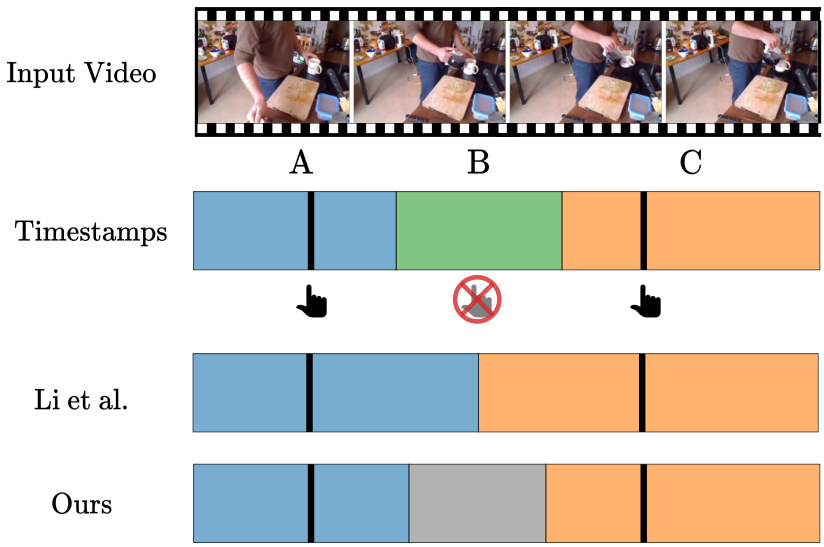


Figure 6.1: For an input video with 3 actions A, B, and C, the annotator has correctly annotated the actions A and C, but action B is not annotated. In this setting, *Li et al.* (2021) assumes that all actions are annotated and generates a pseudo ground truth that is incorrect for the entire region of action B. We propose an approach that takes into account the possibility of missing segments and correctly ignores the region corresponding to the missed action in the generated pseudo ground truth.

While weakly supervised approaches significantly reduced the annotation effort needed, they still perform considerably worse than fully supervised approaches. Recently, *Moltisanti et al.* (2019) introduced timestamp supervision for action recognition and *Li et al.* (2021) used this type of supervision for action segmentation. Timestamp supervision is a form of weak supervision where the annotator is required to annotate a single frame and its class for each action segment in the video. While only a very small fraction of the frames are annotated, this type of weak supervision provides a form of temporal information that is crucial for action segmentation which is missing from transcript supervision. While the cost of obtaining timestamp supervision is only marginally higher than transcript supervision, *Li et al.* (2021) showed that timestamp supervision results in action segmentation performance close to the fully supervised setting.

Despite the promising results for timestamp supervision, the current approaches assume that every action segment instance in the video is annotated. Based on this assumption the dense pseudo ground truth labels are generated and used for training. Nonetheless, an annotator could miss some action segments and as a result, not all action segments are necessarily annotated. Generating dense labels in this setting would produce incorrect pseudo ground truth (Figure 6.1) which has a severe negative impact on the performance (Figure 6.2).

In this chapter, we relax this assumption and propose an approach that is designed to deal with potential missing segments in the timestamp supervision framework. In contrast to *Li et al.* (2021), our approach can skip frames belonging to a missing segment during training while trying to generate labels for all frames belonging to annotated segments. To this end, we propose a pseudo ground truth generation process that groups frames between consecutive timestamps into three segments. While the first and last segments are assigned to labels based on the neighboring timestamp, the

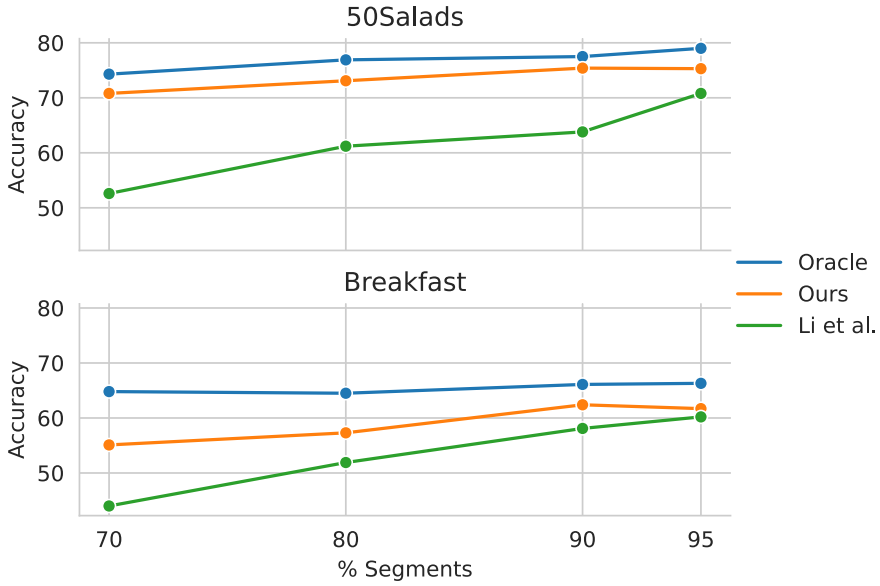


Figure 6.2: Accuracy of different action segmentation approaches trained using timestamp supervision with different percentages of the action segments being annotated with timestamps. The top figure depicts results on the 50Salads dataset (*Stein and McKenna*, 2013), while the bottom depicts results on the Breakfast dataset (*Kuehne et al.*, 2014). Our approach consistently achieves higher accuracy compared to the approach of *Li et al.* (2021), and the gap between our approach and the approach of *Li et al.* (2021) increases as the percentage of annotated segments decreases. In the "Oracle" setting the correct boundary is known for each annotated timestamp.

middle segment is ignored and not used for training (Figure 6.1). Our approach uses the predictions of a segmentation model to find the temporal boundaries of the ignore regions by finding a trade-off between the model's confidence in the frames with assigned labels and the size of the ignored regions.

Experiments on four action segmentation benchmarks show that our approach is considerably more robust to missing segments compared to *Li et al.* (2021) and other baselines. On 50Salads (*Stein and McKenna*, 2013), the accuracy of our approach drops by only 4.5% when the percentage of annotated segments is reduced from 95% to 70% compared to a drop of 18.2% for *Li et al.* (2021). We further evaluate our approach on the related task of temporal action localization from timestamp supervision. Our approach significantly outperforms *Li et al.* (2021) on two datasets. Although our approach is designed for action segmentation, it rivals the performance of the state-of-the-art approaches for temporal action localization from timestamp supervision on these datasets.

6.2 Proposed Method

Given an untrimmed video, action segmentation is the task of predicting the action label for each frame. The video is represented as a set of $D-$dimensional pre-extracted framewise features $X = \left(x_1, \cdots, x_T\right)$ with length T. We assume a fixed set of C action classes, and the ground truth label is represented by $Y = \left(y_1, \cdots, y_T\right)$. Action segmentation models are required to predict framewise

probability estimates $\tilde{Y} = \left(\tilde{y}_1, \cdots, \tilde{y}_T\right)$ where $\tilde{y}_t \in [0,1]^C$ represents the predicted probability for each class. If full supervision is available during training (Y is provided), a per-frame classification loss can be used to train the network *Abu Farha and Gall* (2019); *Li et al.* (2020).

6.2.1 Timestamp Supervision for Action Segmentation

Timestamps are a set of N positions in time $P = (p_1, \cdots, p_N)$ and their corresponding labels $y_{p_1}, \cdots, y_{p_N}$. It is assumed that the timestamps are ordered, *i.e.*

$$1 \leq p_1 \leq \cdots \leq p_N \leq T.$$

A naive approach for action segmentation using timestamps is to only use the timestamps for training the network. The network estimates the framewise probability for the entire video and the classification loss is calculated only using the frames with the timestamps.

Li et al. (2021) assumes that every action segment is annotated with one timestamp. Given this assumption, to uniquely produce the ground truth from the timestamps, one needs to only find the time location between consecutive timestamps where the action label changes from the previous timestamp to the next one. *Li et al.* (2021) proposes an approach to find the action boundary between the consecutive timestamps that minimizes the variance of framewise pre-classification features estimated by the action segmentation network. Given the boundary between the timestamps, the pseudo ground truth annotation $A = \left(a_1, \cdots, a_T\right)$ can be constructed and used for training the network on all temporal positions instead of only on the temporal positions of the timestamps as in the naive approach. *Li et al.* (2021) showed that the proposed approach for the generation of the labels for all frames is crucial to performance in timestamp supervision.

However, if some actions are not annotated by timestamps, the central assumption by *Li et al.* (2021) is not valid. In this setting, where some actions are skipped, for example, due to annotator mistake, the pseudo ground truth generated by *Li et al.* (2021) is incorrect for the entire region of the unannotated action (Figure 6.1). This incorrect pseudo ground truth generation severely affects the performance of their method (Figure 6.2). In the “Oracle" experiment where the correct boundary is known for each annotated timestamp, we observe that the accuracy of action segmentation only marginally drops when the number of annotated segments is reduced from 95% to 70%. In comparison, the approach by *Li et al.* (2021), suffers a large drop in accuracy (Figure 6.2).

6.2.2 Robust Pseudo Ground Truth Generation

We propose an approach for pseudo ground truth generation that is robust to unannotated segments. Our goal is to only extend the segment boundary from the timestamp position in each direction until a point where the network is still confident in the class of that segment while trying to minimize the size of ignored regions.

As shown in Figure 6.3, given two consecutive timestamps, we divide the region between the two timestamps into three parts. The first part with size r_i belongs to the first timestamp p_i, the middle part with size g_i is the ignored region and the third part with size l_{i+1} belongs to the second timestamp p_{i+1}. The beginning and ending parts of the video are also ignored segments g_0 and g_N, respectively. We want the network to be confident about the correct action class of the frames in each timestamp region $[p_i - l_i, p_i + r_i]$ (the colored regions in Figure 6.3). Simultaneously we want the size of the ignored regions g_i (the gray-colored regions in the Figure 6.3) to be as small as possible.

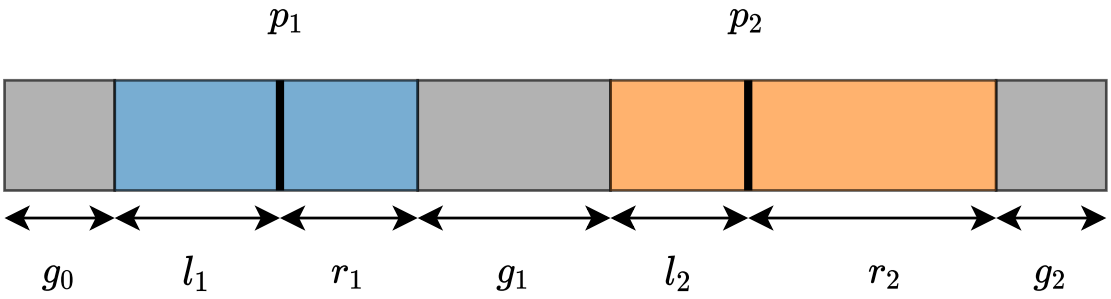


Figure 6.3: Given the timestamps, the temporal region of the video is divided into multiple sections. The region between the two consecutive timestamps p_1 and p_2 has been divided into 3 parts. The first part r_1 will take the same label as p_1 and the last part l_2 will take the same label as p_2. The gray colored regions g_i will be ignored and not used for training.

During training, our goal is to first find the appropriate size of the regions of the video l_i, g_i, r_i and then use them to construct a pseudo ground truth for the network training. In order to do so, we first estimate the framewise probabilities of the video $\tilde{Y}$ using the network. In order for the network to be confident about the correct class of the region for timestamp p_i, the sum of the negative log probabilities of the frames inside that region, *i.e.*

$$\sum_{t=p_i-l_i}^{p_i+r_i} -log\ \tilde{y}_t[y_{p_i}] \tag{6.1}$$

should be low. Furthermore we want the sum of the size of the ignore regions, *i.e.*

$$\sum_{i=0}^{N} g_i \tag{6.2}$$

to be as small as possible.

This goal can be formulated as the following constrained minimization problem

$$\begin{aligned} \{r_i^*, g_i^*, l_i^*\} = \\ \underset{r_i,g_i,l_i}{\text{argmin}} \sum_{i=1}^{N} \left(\sum_{t=p_i-l_i}^{p_i+r_i} -log\tilde{y}_t[y_{p_i}] \right) + \beta \sum_{i=0}^{N} g_i \\ \text{s.t. } p_{i+1} - p_i = r_i + g_i + l_{i+1} \\ r_i \geq 0 \\ g_i \geq 0 \\ l_i \geq 0. \end{aligned} \tag{6.3}$$

The hyper-parameter β, balances the sizes of ignored regions in the result. We study the effect of this hyper-parameter in Section 6.3.7. The first constraint ensures that the regions between two timestamps do not overlap and cover the entire temporal region. The last three constraints ensure the positivity of the size of the segments.

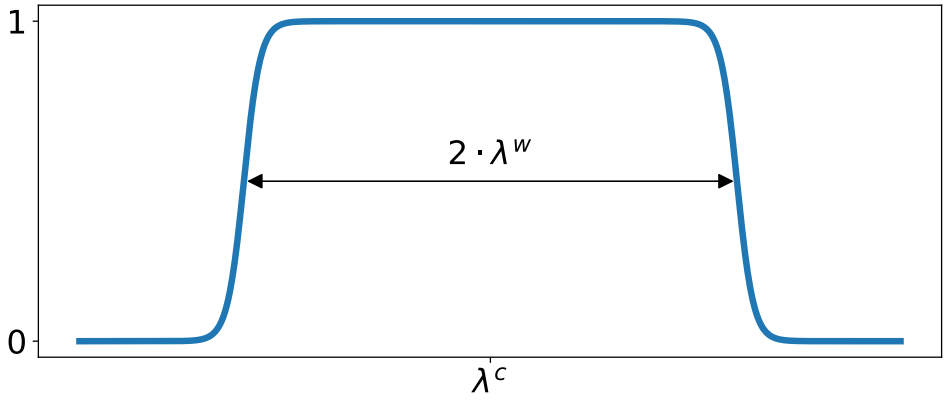


Figure 6.4: The plateau function (6.5) with center parameter λ^c and width parameter λ^w.

To solve this optimization problem using gradient descent similar to FIFA (*Souri et al.*, 2021a), we need to make a differentiable approximation of the objective and use reparameterization to remove the constraints from the optimization. Afterward, gradient descent can be used to approximately solve the optimization problem.

This optimization is solved for each video in the training batch separately. Please note that this optimization problem is only performed during training and to generate the pseudo ground truth. The objective specified in (6.3) is not used for updating the network weights but only for pseudo ground truth generation.

6.2.2.1 Differentiable Approximation

The objective function in (6.3) can be re-written as

$$\begin{aligned}\sum_{i=1}^{N}\left(\sum_{t=p_i-l_i}^{p_i+r_i} -log\tilde{y}_t[y_{p_i}]\right) + \beta\sum_{i=0}^{N} g_i = \\ \sum_{i=1}^{N}\left(\sum_{t=1}^{T} -log\tilde{y}_t[y_{p_i}]\mathcal{I}(t|p_i-l_i \leq t < p_i+r_i)\right) + \beta\sum_{i=0}^{N} g_i\end{aligned} \tag{6.4}$$

where $\mathcal{I}(t|p_i - l_i \leq t < p_i + r_i)$ is the indicator function with value 1 if t is within the left and right bounds of the timestamp and 0 otherwise.

As the $\mathcal{I}$ function is a non-differentiable function with value of 0 or 1 depending on its parameters, we replace it with the plateau function (*Moltisanti et al.*, 2019; *Souri et al.*, 2021a). The plateau function is defined as

$$\begin{aligned}f(t|\lambda^c, \lambda^w, \lambda^s) = \\ \frac{1}{(e^{\lambda^s(t-\lambda^c-\lambda^w)}+1)(e^{\lambda^s(-t+\lambda^c-\lambda^w)}+1)}\end{aligned} \tag{6.5}$$

and is a differentiable function that has a value between 0 everywhere except a window of size $2 \times \lambda^w$ around temporal position λ^c, where it takes value of 1 (Figure 6.4). The parameter λ^s of the plateau function controls the sharpness of the transition from 0 to 1.

We can replace the indicator function $\mathcal{I}$ with the plateau function f. Approximately we can say

$$\mathcal{I}(t|p_i - l_i \leq t < p_i + r_i) \simeq f(t|\lambda^{c_i}, \lambda^{w_i}, \lambda^s) \tag{6.6}$$

where $\lambda^{c_i} = p_i + \frac{r_i - l_i}{2}$, $\lambda^{w_i} = \frac{r_i + l_i}{2}$, and s is the sharpness of the plateau function which is fixed. Equation (6.4) can be re-written as

$$\sum_{i=1}^{N} \left(\sum_{t=1}^{T} -log\tilde{y}_t[y_{p_i}] f(t|\lambda^{c_i}, \lambda^{w_i}, \lambda^s) \right) + \beta \sum_{i=0}^{N} g_i. \tag{6.7}$$

The second term in (6.7) is penalizing the ignore areas which can be approximately calculated as the regions not covered by the plateau functions, *i.e.*

$$\sum_{i=0}^{N} g_i \simeq \sum_{t=1}^{T} \left(1 - \sum_{i=1}^{N} f(t|\lambda^{c_i}, \lambda^{w_i}, \lambda^s) \right). \tag{6.8}$$

By using (6.8) we can write the final differentiable approximate objective as

$$\begin{aligned} &\sum_{i=1}^{N} \left(\sum_{t=1}^{T} -log\tilde{y}_t[y_{p_i}] f(t|\lambda^{c_i}, \lambda^{w_i}, \lambda^s) \right) + \\ &\beta \sum_{t=1}^{T} \left(1 - \sum_{i=1}^{N} f(t|\lambda^{c_i}, \lambda^{w_i}, \lambda^s) \right). \end{aligned} \tag{6.9}$$

6.2.2.2 Reparameterization

The original parameters of the optimization r_i, g_i, and l_i are constrained, *i.e.*each one of them has to be positive and the sum of the lengths of the three sections denoted by r_i, g_i, and l_i should be equal to the distance between the timestamps p_i and p_{i+1}.

To relax these constraints, we introduce new parameters r'_i, g'_i, and l'_i which are the log of the length of the lengths relative to each other. These parameters can take any real value. The original parameters r_i, g_i, and l_i can be calculated from the new parameters by using the softmax function, *i.e.*

$$r_i, g_i, l_{i+1} = (p_{i+1} - p_i) \cdot \text{softmax}(r'_i, g'_i, l'_{i+1}). \tag{6.10}$$

The projection (6.10) makes sure that the parameters r_i, g_i, and l_i are satisfying the original constraints.

After solving the optimization problem, we generate the pseudo ground truth that includes the ignored regions. This pseudo ground truth is used for training the action segmentation network as in *Li et al.* (2021).

6.3 Experiments

6.3.1 Datasets and Metrics

Datasets. We evaluate our approach on four datasets: 50Salads (*Stein and McKenna*, 2013), Breakfast (*Kuehne et al.*, 2014), BEOID (*Damen et al.*, 2014), and Georgia Tech Egocentric Activities (GTEA) (*Fathi et al.*, 2011).

The **50Salads** dataset consists of 50 videos of people preparing mixed salads. The average length of the videos is around 6.4 minutes. The dataset provides fine-granular annotations of 17 low level activities such as *peel cucumber* and *mix ingredients*. For evaluation, we use five-fold cross-validation and report the average.

The **Breakfast** dataset contains 1712 videos of breakfast-related activities. The average length of the videos is around 2.3 minutes. The frames are annotated with 48 fine-grained action classes such as *take bowl* and *pour cereals*. For evaluation, we use the standard four splits and report the average.

The **BEOID** dataset contains 58 egocentric videos, where the frames are annotated with 34 actions classes. The videos were recorded in six locations: kitchen, workspace, laser printer area, corridor with a locked door, cardiac gym, and weight-lifting machine. For evaluation, we use the same training-testing split as in *Ma et al.* (2020).

The **GTEA** dataset contains egocentric videos recorded by a camera that is mounted on the actor's head. In total, there are 28 videos with roughly 32K frames. The frames are annotated with 11 action classes. For evaluation, we report the average of four splits.

For 50Salads and the Breakfast dataset, we use the timestamps annotations provided by *Li et al.* (2021) and simulate missing annotations by randomly removing timestamps. Whereas for GTEA and BEOID datasets, we use the human annotations provided by *Ma et al.* (2020).

Metrics. Following the state of the art approaches for action segmentation, we report frame-wise accuracy (Acc), segmental edit distance (Edit), and segmental F1 scores at overlapping thresholds 10%, 25% and 50%.

6.3.2 Implementation Details

We follow the setup of *Li et al.* (2021) and use the same backbone network and train it with the same loss function for a fair comparison. All models are trained for 50 epochs, where only the timestamps annotations are used for the first 30 epochs and the generated labels are used afterward. If not otherwise mentioned, we set the value of β to 0.7. For the gradient descent based optimization of (6.3), we initialized r_i, l_i, g_i uniformly using equal lengths and optimize for 30 iterations using the Adam optimizer with a step size of 0.03.

6.3.3 Impact of Missing Annotations

In this section, we compare our approach with *Li et al.* (2021) on different percentages of annotated segments. Results on 50Salads and the Breakfast dataset are shown in Table 6.1 and Table 6.2, respectively. Our approach is robust to missing annotations and outperforms the approach of *Li et al.* (2021) on all the percentages and both datasets. As shown in the tables, the framewise accuracy of our approach drops by only 4.5% on 50Salads and by 6.6% on the Breakfast dataset when the percentage of annotated segments is reduced from 95% to 70%. On the contrary, the accuracy of *Li et al.* (2021) drops considerably by 18.2% on 50Salads and 16.2% on the Breakfast dataset. Using 70% of the timestamps for training our approach outperforms *Li et al.* (2021) by 15.2% and 11.5% for the F1@25 metric on the 50Salad and the Breakfast respectively. This is expected as the previous approach generates dense labels, which results in many wrong labels in the case of many missing segments. By contrast, our approach explicitly handles missing annotations by ignoring some frames during training. This effect is also visible in the qualitative results shown in Figure 6.5. While the

% Segments	Method	F1@{10, 25, 50}			Edit	Acc
95%	*Li et al.* (2021)	70.9	67.4	53.4	63.8	70.8
	Ours	**72.9**	**69.6**	**57.5**	**64.2**	**75.3**
	Oracle PGT	74.8	72.0	64.1	67.9	79.0
90%	*Li et al.* (2021)	63.9	59.6	44.3	57.6	63.8
	Ours	**70.0**	**65.1**	**55.2**	**62.1**	**75.4**
	Oracle PGT	73.9	71.6	62.5	66.9	77.5
80%	*Li et al.* (2021)	62.7	56.9	40.3	54.2	61.2
	Ours	**70.9**	**67.8**	**53.7**	**61.4**	**73.1**
	Oracle PGT	73.3	70.2	61.0	65.8	76.9
70%	*Li et al.* (2021)	50.2	44.0	29.5	44.7	52.6
	Ours	**64.1**	**59.2**	**44.8**	**56.9**	**70.8**
	Oracle PGT	67.4	63.3	53.2	59.0	74.3

Table 6.1: Impact of missing segments on the 50Salads dataset.

approach of *Li et al.* (2021) assigned wrong labels to the missing segments, our approach successfully assigned these frames to the ignored region.

We also report the results for an oracle baseline “Oracle PGT" in Table 6.1 and Table 6.2. In this baseline, we generate the pseudo labels by using the ground truth boundaries for the annotated segments, whereas the missing segments are ignored. In the example shown in Figure 6.1, for instance, we use the ground truth segments A and C for training, but we ignore segment B. This baseline serves as an upper bound for our approach. An interesting observation is that the accuracy of the oracle baseline drops only slightly when the percentage of annotated segments is reduced from 95% to 70%.

6.3.4 Comparison with Baselines

Our approach generates pseudo ground truth labels by grouping frames between consecutive timestamps into three segments. The boundaries of these segments are determined by finding a trade-off between the model confidence on the annotated segments and the size of the ignored segments. In this section, we compare our approach with different baseline alternatives to generate pseudo labels. The first baseline uses only the annotated timestamps for training and ignores all the frames in between, which is denoted by “Timestamps only". The second baseline “Uniform-2" divides the frame between the timestamps equally into two segments and assign labels to each frame based on the label of the nearest timestamp. Whereas in the last baseline “Uniform-3", the frames between timestamps are divided into three equal-sized segments. In this baseline, only the first and last segments are assigned to labels and the middle segments are ignored during training. Results for our approach and the baselines on the 50Salads dataset are shown in Table 6.3. While the Uniform-3 baseline outperforms Uniform-2, using only the annotated timestamps still achieves much higher results, especially for lower percentages of annotated segments. This indicates that generating wrong labels for the missing segments has a huge impact on the performance. On the other hand, our approach outperforms all the baselines on the different percentages of annotated segments. This indicates that

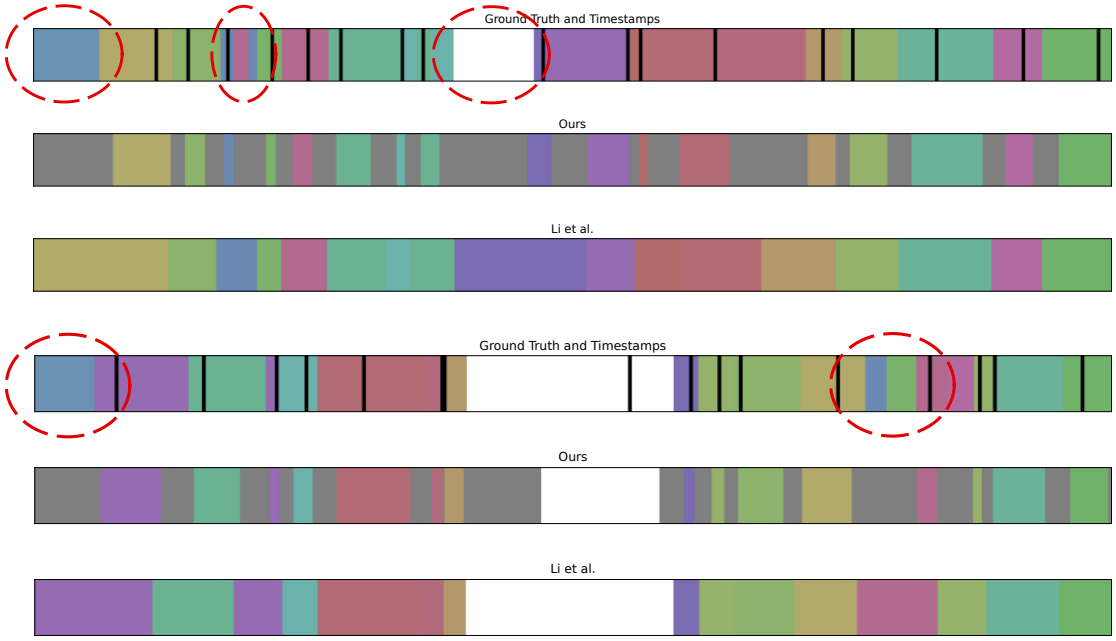


Figure 6.5: Qualitative results for comparing the generated labels by our approach and *Li et al.* (2021) on two videos from the 50Salads dataset. The top row shows the ground truth frame-wise labels and the annotated timestamps in black. Missing annotated segments are highlighted with red dashed ellipses. The pseudo labels generated by our approach are shown in the second row, where the ignored regions are colored in gray. The pseudo labels generated by *Li et al.* (2021) are shown in the third row. For the missing segments, our approach successfully assigned these frames to the ignored region. On the contrary, the approach of *Li et al.* (2021) densely generates labels that result in wrong pseudo labels for these segments.

% Segments	Method	F1@{10, 25, 50}			Edit	Acc
95%	*Li et al.* (2021)	67.3	59.7	42.7	68.2	60.2
	Ours	**70.2**	**62.4**	**44.8**	**69.7**	**61.7**
	Oracle PGT	71.0	65.3	51.4	70.2	66.3
90%	*Li et al.* (2021)	65.0	56.5	39.7	66.8	58.1
	Ours	**69.8**	**62.2**	**44.5**	**69.7**	**62.4**
	Oracle PGT	69.0	63.0	49.1	68.6	66.1
80%	*Li et al.* (2021)	59.8	50.4	33.0	62.8	51.9
	Ours	**67.3**	**58.9**	**40.8**	**68.5**	**57.3**
	Oracle PGT	69.2	62.8	48.7	68.4	64.5
70%	*Li et al.* (2021)	53.9	43.5	26.8	59.2	44.0
	Ours	**65.0**	**55.0**	**35.7**	**66.5**	**55.1**
	Oracle PGT	69.5	63.2	49	69	64.8

Table 6.2: Impact of missing segments on the Breakfast dataset.

our approach is robust to missing annotations and can generate more accurate labels to train the segmentation model.

6.3.5 Impact of Initialization

To generate the pseudo labels, we initialize our approach with an equal size of three segments between consecutive timestamps. Table 6.4 shows the results of the uniform initialization compared to initialization based on a fixed duration. In the fixed duration initialization, we assign frames within a 3-second range in both sides of the annotated timestamps with the same label as the corresponding timestamp, and all other frames are initialized as ignore regions. As shown in the table, the impact of initialization is small and the uniform initialization scheme performs better.

6.3.6 Evaluation of Different Timestamps Selection Strategies

Table 6.5 shows the results of our approach and the approach of *Li et al.* (2021) on different timestamps selection strategies. “Start frame" and “Center frame" indicate scenarios in which always the *start* or *center* frames are selected as timestamps respectively. “Gaussian" and “Uniform" indicate scenarios in which the location of the timestamps are randomly selected inside each action segment using *Gaussian* or *Uniform* distributions. Our approach outperforms *Li et al.* (2021) regardless of how the location of the annotated timestamps are provided. Apart from the starting frame, the performance of other approaches are comparable. On the contrary, annotating the starting frame of the segments results in a considerable drop in performance. This is expected as the start frame is not discriminative enough to represent the action segment.

% Segments	Method	F1@{10, 25, 50}			Edit	Acc
95%	Uniform-2	63.1	56.4	37.8	58.8	59.5
	Uniform-3	63.4	58.5	40.8	56.9	63.5
	Timestamps only	59.9	55.2	45.6	49.6	71.5
	Ours	**72.9**	**69.6**	**57.5**	**64.2**	**75.3**
90%	Uniform-2	60.8	53.0	34.7	56.0	56.1
	Uniform-3	62.0	56.3	39.2	56.1	61.5
	Timestamps only	55.4	51.4	40.2	46.0	69.6
	Ours	**70.0**	**65.1**	**55.2**	**62.1**	**75.4**
80%	Uniform-2	56.2	49.3	32.1	51.1	56.3
	Uniform-3	59.6	52.5	35.3	54.4	59.7
	Timestamps only	55.1	50.8	39.6	44.8	66.2
	Ours	**70.9**	**67.8**	**53.7**	**61.4**	**73.1**
70%	Uniform-2	42.2	36.0	19.0	40.1	45.8
	Uniform-3	48.8	43.2	28.5	46.0	54.1
	Timestamps only	46.6	41.4	30.2	39.3	60.0
	Ours	**64.1**	**59.2**	**44.8**	**56.9**	**70.8**

Table 6.3: Comparison with different baselines on the 50Salads dataset.

% Segments	Initialization	F1@{10, 25, 50}			Edit	Acc
95%	Fixed (3 sec)	69.7	66.9	55.3	62.4	73.3
	Uniform	**72.9**	**69.6**	**57.5**	**64.2**	**75.3**
90%	Fixed (3 sec)	68.4	**65.7**	**55.3**	58.5	72.9
	Uniform	**70.0**	65.1	55.2	**62.1**	**75.4**
80%	Fixed (3 sec)	66.2	63.1	50.7	57.6	71.1
	Uniform	**70.9**	**67.8**	**53.7**	**61.4**	**73.1**
70%	Fixed (3 sec)	62.0	58.5	44.3	53.5	67.0
	Uniform	**64.1**	**59.2**	**44.8**	**56.9**	**70.8**

Table 6.4: Impact of initialization on the 50Salads dataset.

Method	Timestamps	F1@{10, 25, 50}			Edit	Acc
Li et al. (2021)	Start frame	49.7	36.8	14.8	49.8	41.5
	Center frame	69.5	65.6	48.5	61.8	66.6
	Gaussian	67.1	62.5	45.4	58.0	66.3
	Uniform	63.9	59.6	44.3	57.6	63.8
Ours	Start frame	54.1	40.7	17.3	52.8	44.8
	Center frame	71.5	68.9	56.9	63.2	72.4
	Gaussian	70.8	67.2	55.4	62.3	71.9
	Uniform	70.0	65.1	55.2	62.1	75.4

Table 6.5: Comparing different selection strategies of timestamps using 90% of the segments on the 50Salads dataset. Our approach outperforms *Li et al.* (2021) using every strategy on every metric.

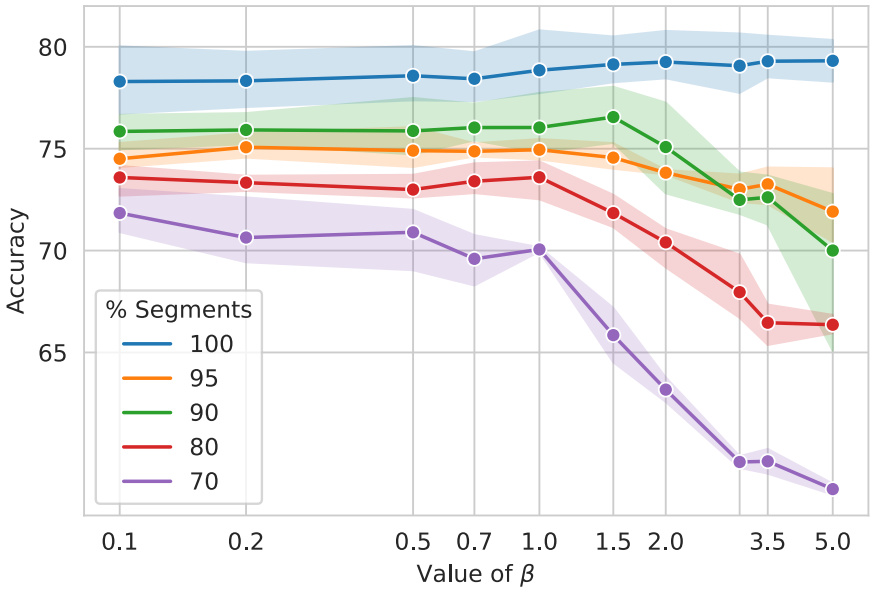


Figure 6.6: Impact of β on different percentages of the annotated segments on the 50Salads dataset. For each experiment, the mean and standard deviation of three runs are shown.

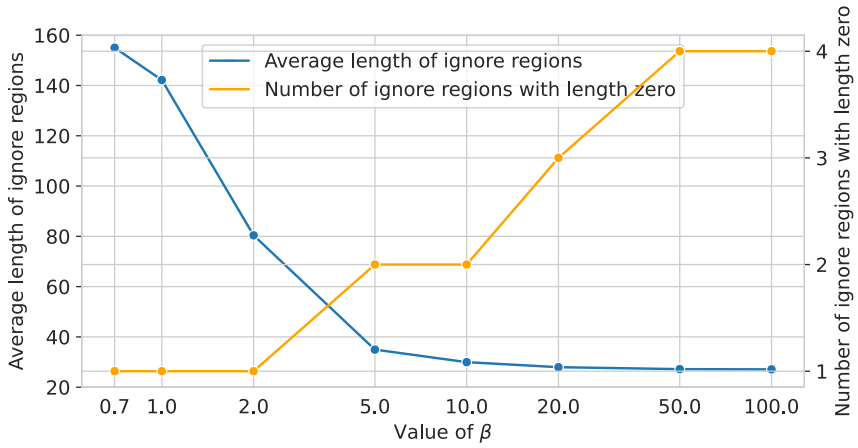


Figure 6.7: Average length of the ignore regions and the number of the ignore regions with length 0 for different values of β. The numbers are reported for the training set of split 1 of the 50Salads dataset after the initial 30 epochs.

6.3.7 Sensitivity to β

In our approach, the size of the ignore region is controlled by β. This is a hyper-parameter for our approach and is set to 0.7 in all experiments. In this section, we study the impact of β on performance. Figure 6.6 shows the results of our approach using different values for β on different percentages of annotated segments on the 50Salads dataset. We plot the mean and standard deviation of three runs for each setting. As shown in the figure, lower values for β tend to achieve better results, with decreasing the percentage of annotated segments. This is expected since with a lower percentage of annotations more frames should be assigned to the ignore region, which is achieved by lowering the value of β. When there are no missing segments, no frame should be assigned to the ignore region, and higher values of β are better.

We also perform an analysis of the size of the ignore regions in the generated pseudo ground truth and the effect of the parameter β on it. Figure 6.7 shows the average size of the ignore regions and the number of completely empty ignore regions with different values of β calculated on the training set of split 1 of the 50Salads dataset after the initial 30 epochs. We observe that as expected the average size of the ignore region decreases as the value of β increases. Furthermore in practice we see a few ignore regions become completely empty. An empty ignore region is theoretically possible given the optimization problem in (6.3) and we see that in practice it occurs as well.

6.3.8 Performance on Timestamps without Missing Annotations

In all the experiments so far, we only considered timestamps annotations with missing segments. In this section, we evaluate our approach on the case where all the segments are annotated. This is the same setup as in *Li et al.* (2021). Table 6.6 and Table 6.7 show the results on both 50Salads and the Breakfast dataset. While our approach is designed to handle missing annotations, it also works well when there are no missing segments. Our approach outperforms previous approaches on 50Salads and achieves competitive results on the Breakfast dataset. As discussed in Section 6.3.7, if we know that there are no missing segments, then it is possible to achieve better results by using a higher value

Supervision	Method	F1@{10, 25, 50}			Edit	Acc
Full	MS-TCN (*Abu Farha and Gall*, 2019)	76.3	74.0	64.5	67.9	80.7
	MS-TCN++ (*Li et al.*, 2020)	80.7	78.5	70.1	74.3	83.7
	BCN (*Wang et al.*, 2020)	82.3	81.3	74.0	74.3	84.4
	ASRF (*Ishikawa et al.*, 2021)	84.9	83.5	77.3	79.3	84.5
	ASFormer (*Yi et al.*, 2021)	85.1	83.4	76.0	79.6	85.6
Timestamps	Plateau (*Moltisanti et al.*, 2019)*	71.2	68.2	56.1	62.6	73.9
	Li et al. (2021)	73.9	70.9	60.1	66.8	75.6
	Ours	75.2	72.1	61.5	67.7	78.4
	Ours ($\beta = 2$)	**77.0**	**74.2**	**62.2**	**69.8**	**79.3**
Transcripts	CDFL (*Li et al.*, 2019)	-	-	-	-	54.7
	NN-Viterbi (*Richard et al.*, 2018b)	-	-	-	-	49.4
	HMM-RNN (*Richard et al.*, 2017)	-	-	-	-	45.5

Table 6.6: Results on timestamps without missing segments on the 50Salads dataset. * results obtained from *Li et al.* (2021).

of β (for instance $\beta = 2$). By increasing β we encourage our approach to further decrease the number of frames assigned to the ignore region.

6.3.9 Localization Experiment

While our approach is designed for action segmentation, it can also be applied to the action localization task. Following *Li et al.* (2021), we use our label generation approach to train an action localization model using the human-annotated timestamps on the GTEA and BEOID datasets provided by *Ma et al.* (2020). Results on both datasets are shown in Table 6.8. Our approach outperforms *Li et al.* (2021) by a large margin of 7% average mAP on the GTEA and 18.5% mAP on the BEOID dataset. Furthermore, our approach achieves competitive results to *Lee and Byun* (2021) which is specific for action localization.

6.3.10 Speed of Pseudo Ground Truth Generation

The approach we have chosen for pseudo ground truth generation is different compared to the pseudo ground truth generation of *Li et al.* (2021). Our approach uses gradient descent to optimize the objective while *Li et al.* (2021) uses an exhaustive search. We measured the wall clock time for generating the pseudo ground truth for the whole training set of split 1 of the 50Salads dataset for our approach and *Li et al.* (2021) on the same hardware. While *Li et al.* (2021) takes 116 seconds to generate the pseudo ground truth, our approach takes 1.7 seconds which shows a 68 times faster pseudo ground truth generation.

Supervision	Method	F1@{10, 25, 50}			Edit	Acc
Full	MS-TCN (*Abu Farha and Gall*, 2019)	52.6	48.1	37.9	61.7	66.3
	MS-TCN++ (*Li et al.*, 2020)	64.1	58.6	45.9	65.6	67.6
	BCN (*Wang et al.*, 2020)	68.7	65.5	55.0	66.2	70.4
	ASRF (*Ishikawa et al.*, 2021)	74.3	68.9	56.1	72.4	67.6
	FIFA (Chapter 5)	75.5	70.2	54.8	78.5	68.6
	MuCon (Chapter 4)	73.2	66.1	48.4	76.3	62.8
	ASFormer (*Yi et al.*, 2021)	76.0	70.6	57.4	75.0	73.5
Timestamps	Plateau (*Moltisanti et al.*, 2019)*	65.5	59.1	43.2	65.9	63.5
	Li et al. (2021)	70.5	63.6	**47.4**	69.9	**64.1**
	Ours	67.0	60.0	43.8	66.7	60.8
	Ours ($\beta = 2$)	**71.5**	**64.3**	47.3	**70.9**	62.9
Transcripts	FIFA (Chapter 5)	-	-	-	-	51.3
	CDFL (*Li et al.*, 2019)	-	-	-	-	50.2
	MuCon (Chapter 4)	-	-	-	-	48.5
	D^{3}TW (*Chang et al.*, 2019)	-	-	-	-	45.7
	NN-Viterbi (*Richard et al.*, 2018b)	-	-	-	-	43.0
	TCFPN (*Ding and Xu*, 2018)	-	-	-	-	38.4
Sets	SCT (*Fayyaz and Gall*, 2020)	-	-	-	-	30.4
	SCV (*Li and Todorovic*, 2020)	-	-	-	-	30.2
	Action Sets (*Richard et al.*, 2018a)	-	-	-	-	23.3

Table 6.7: Results on timestamps without missing segments on the Breakfast dataset. * results obtained from *Li et al.* (2021).

mAP@IoU	0.1	0.3	0.5	0.7	Avg
GTEA					
SF-Net (*Ma et al.*, 2020)	58.0	37.9	19.3	11.9	31.0
Li et al. (2021)	60.2	44.7	28.8	12.2	36.4
Lee and Byun (2021)	**63.9**	**55.7**	33.9	**20.8**	**43.5**
Ours	63.6	54.1	**36.4**	20.3	43.4
BEOID					
SF-Net (*Ma et al.*, 2020)	62.9	40.6	16.7	3.5	30.1
Li et al. (2021)	71.5	40.3	20.3	5.5	34.4
Lee and Byun (2021)	76.9	61.4	**42.7**	**25.1**	51.8
Ours	**79.0**	**68.3**	42.1	17.2	**52.9**

Table 6.8: Comparison for action localization with timestamp supervision on the GTEA and BEOID datasets.

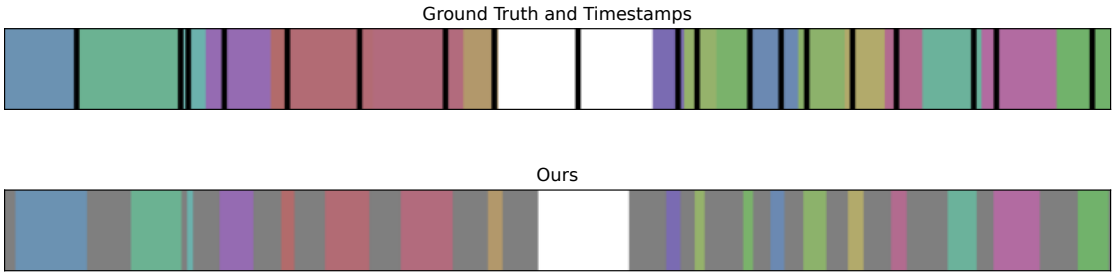


Figure 6.8: Qualitative results showing a limitation of our proposed pseudo ground truth generation approach for small action segments. In this example a large portion of frames were ignored around small action segments.

6.4 Limitations

For a fair comparison with *Li et al.* (2021), we have chosen to use the same backbone action segmentation network, the same type of feature extractor for the videos, and the same training hyperparameters as *Li et al.* (2021). For this reason, some limitations are shared between *Li et al.* (2021) and our approach since the accuracy of these approaches is not perfect. These incorrect predictions are usually because the features used are not discriminative enough or the network is not powerful enough. A better feature extractor, base network, or training strategy, could improve this aspect of our approach.

Some limitations are a consequence of our proposed pseudo ground truth generation approach. Figure 6.8 depicts an example where many frames are ignored and therefore not used for training. This is in particular a problem for short action segments. We think addressing these limitations is an interesting direction for future work.

6.5 Conclusion

In this chapter, we proposed a robust action segmentation approach for timestamp supervision. In contrast to the previous method *Li et al.* (2021) that assumes that all action segments are annotated, we proposed a label generation method that handles missing segment annotations and is thus more suitable for real-world applications. Our approach achieves competitive results when all segments are annotated and shows a relatively small drop in performance with increasing percentage of missing segments. Using 70% of annotated segments, our approach achieves considerably higher accuracy, edit score and f1 score compared to *Li et al.* (2021) on two action segmentation datasets.

CHAPTER 7

What Object Should I Use? - Task Driven Object Detection

This chapter is based on the following publication:
What Object Should I Use? - Task Driven Object Detection
Johann Sawatzky*, Yaser Souri*, Christian Grund, Juergen Gall
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

Below we will summarize the contributions of each author.

- **Yaser Souri and Johann Sawatzky**
  Both contributed equally to this work and are ordered based on alphabetical order. Yaser implemented the tool to annotate the dataset. Johann managed the annotators. Yaser created the final dataset from the annotations and released the dataset. Yaser and Johann proposed the proposed approach. Yaser implemented the proposed approach, performed most of the experiments and reported the results. Johann implemented most of the baselines and performed various analytics. Yaser and Johann contributed most of the paper text.

- **Christian Grund**
  Christian helped with optimizing the annotation tool.

- **Johann Sawatzky and Juergen Gall**
  Johann and Juergen proposed the novel task of finding the most suitable object to perform a task.

- **Juergen Gall**
  Juergen supervised the project. He contributed with discussions and writing.

Contents

7.1 Introduction . . . 100
7.2 Related Work . . . 101
7.3 COCO-Tasks Dataset . . . 102
7.4 Task Driven Object Detection . . . 109
7.4.1 Proposed Method . . . 110
7.4.2 Implementation Details . . . 111
7.5 Experiments . . . 112
7.5.1 Comparison to Baselines . . . 112

7.5.2 Ablation Experiments 114
7.5.3 Class Bias Analysis 115
7.5.4 Task Wise Difficulty Analysis 116
7.5.5 Scene Context Learned by GGNN 116
7.6 Conclusion 123

In the previous three chapters we focused on recognizing actions. In robotics applications after recognizing the actions, the goal is to perform an action and complete a task. When humans have to solve everyday tasks, they simply pick the objects that are most suitable. While the question which object should one use for a specific task sounds trivial for humans, it is very difficult to answer for robots or other autonomous systems. This issue, however, is not addressed by current benchmarks for object detection that focus on detecting object categories. In this chapter we therefore introduce the COCO-Tasks dataset which comprises about 40,000 images where the most suitable objects for 14 tasks have been annotated. We furthermore propose an approach that detects the most suitable objects for a given task. The approach builds on a Gated Graph Neural Network to exploit the appearance of each object as well as the global context of all present objects in the scene. In our experiments, we show that the proposed approach outperforms other approaches that are evaluated on the dataset like classification or ranking approaches.

7.1 Introduction

The task of object detection in images has been widely studied and the community achieved impressive progress on datasets like COCO (*Lin et al.*, 2014) or Pascal VOC (*Everingham et al.*, 2014). For many applications like assistive or autonomous systems, however, it is insufficient to detect all instances of a set of object categories. Similar to humans, the systems interact with the environment to solve certain tasks. For instance, if a service robot is asked to serve a glass of wine, detecting all glasses in an image does not answer the question which of them it should use. Taking a beer glass is definitely the wrong choice if a wine glass is available, but if no other glasses are available it might be the best option for the task. Even if there are several wine glasses, not all of them are necessary suitable since some of the glasses might be already used by someone else or need to be cleaned. If no glasses are available, some alternatives have to be considered. For instance, wine can be drunk from a cup or jug as well. This shows that answering the question, which object should be used for a task is very difficult since it depends on the present object categories in an image and the properties of the objects.

In this work, we address the problem of *task driven object detection*. It requires to detect all objects in an image which serve a given task best. To this end, we propose the *task driven object detection* (COCO-Tasks) dataset, which is based on the images and annotated objects of the COCO dataset (*Lin et al.*, 2014). For evaluation, we define 14 tasks and asked humans to mark all objects in an image which they favor to solve a given task. If none of the objects in an image is suitable, the annotators were allowed to select none of the objects. The dataset comprises about 40,000 annotated images and for each task between 1,100 and 9,900 objects have been marked by the annotators, where the number of different object categories varies between 6 and 30 for the different tasks. Figures 7.1 and 7.3 show a few examples.

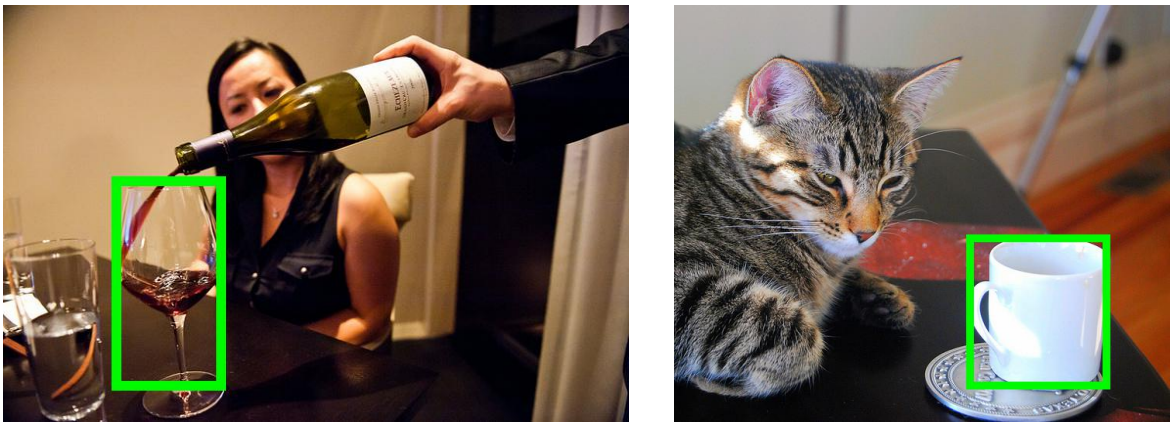


Figure 7.1: What object in the scene would a human choose to serve wine? In the left image, the wine glass is preferred to other drinking glasses. In the right image, neither a wine glass nor other drinking glasses are present. The cup is therefore chosen by the human.

In our experimental evaluation, we show that *task driven object detection* cannot be treated as a standard object detection task. If a standard object detector is trained for each task using the human annotations as ground-truth, the predictions are not very accurate since the favored objects strongly depend on the presence of other objects and their properties. We therefore propose a method based on Gated Graph Neural Networks (GGNN) (*Li et al.*, 2016) that explicitly incorporates all detection hypotheses in an image to infer which objects are preferred for a task. Our experimental results show that our proposed method outperforms various ranking and classification based baselines and a thorough ablation study analyzes the design choices of our proposed approach. COCO-Tasks dataset and the code for reproducing our experiments are publicly available online[1].

7.2 Related Work

Due to public benchmarks like Pascal VOC (*Everingham et al.*, 2014) and COCO (*Lin et al.*, 2014), there was a tremendous advancement in the area of object detection. State-of-the-art object detectors (*Dai et al.*, 2017b; *Bodla et al.*, 2017; *Dai et al.*, 2017b; *Chollet*, 2017; *Xie et al.*, 2017; *Mao et al.*, 2017; *Peng et al.*, 2017; *Zhou et al.*, 2018b) rely exclusively on convolutional neural networks where in particular Faster R-CNN (*Ren et al.*, 2015b) has been widely used. For applications where runtime is critical, other detectors like (*Redmon and Farhadi*, 2017; *Liu et al.*, 2016) provide a very good trade-off between efficiency and accuracy.

In contrast to standard object detection, task driven object detection requires an understanding of the entire scene. This relates it to the task of visual question answering which takes as input a question regarding the content of an image and returns an answer in text form, whereas for task driven object detection the input is a task and the output are bounding boxes around objects that are best suitable for solving the task.

While *Antol et al.* (2015); *Gao et al.* (2015); *Yu et al.* (2015); *Ren et al.* (2015a); *Malinowski and Fritz* (2014) pioneered in visual question answering, *Teney et al.* (2018); *Teney and van den Hengel*

[1] coco-tasks.github.io

(2018); *Anderson et al.* (2018); *Nguyen and Okatani* (2018); *Zhang et al.* (2018) are examples of current state-of-the-art methods.

Choosing the best object among the available requires not only recognizing its class but judging its functional attributes, *i.e.*its affordances. Detecting and segmenting affordances in images has therefore received an increased interest (*Nguyen et al.*, 2016, 2017; *Kumra and Kanan*, 2017; *Sawatzky et al.*, 2017; *Do et al.*, 2018). In the work *Zhu et al.* (2015), learning functional and physical properties together with the handling of objects as tools is investigated. The model is learned from human demonstration and relies on 3d models of objects. The model is then used to recognize tools and affordance regions for 3D objects. *Fang et al.* (2018) propose to learn to detect affordances from demo videos.

Applying deep neural network on graph structured data has seen a lot of attention from the community recently (*Henaff et al.*, 2015; *Duvenaud et al.*, 2015; *Kipf and Welling*, 2017; *Li et al.*, 2016). Many computer vision problems including scene context can naturally be represented as a graph. *Wang and Gupta* (2018) use a Graph Convolutional Network (*Kipf and Welling*, 2017) to represent a video and achieve very good results on video classification. *Qi et al.* (2017) have used graph neural networks for semantic segmentation. *Chuang et al.* (2018) used Gated Graph Neural Networks (*Li et al.*, 2016) to model affordances in context. While our work compares the objects to each other, *Chuang et al.* (2018) focuses on the interaction of objects with their environment.

The task of scene graph generation proposed by *Johnson et al.* (2015) requires the detection of objects and relationships between pairs of them. These relationships are typically prepositions indicating relative geometric position and physical interactions. While earlier approaches (*Lu et al.*, 2016; *Zhuang et al.*, 2017; *Peyre et al.*, 2017; *Zhang et al.*, 2017b,a; *Xu et al.*, 2017; *Li et al.*, 2017a; *Liang et al.*, 2017; *Dai et al.*, 2017a; *Li et al.*, 2017b; *Newell and Deng*, 2017; *Zellers et al.*, 2018) avoid the search over the exhaustive number of relations by heuristics, more recently *Yang et al.* (2018) propose a method which learns to prune unlikely object relationships. While *Li et al.* (2018) rely on modeling subgraphs for scene graph generation, *Zellers et al.* (2018) focus on correlations between objects and higher order graph structure statistics.

7.3 COCO-Tasks Dataset

Detecting the objects, which are favored for a given task, is very difficult. It requires localizing objects as for a standard object detection task, but the preferred objects in an image vary among image and task. Figure 7.3 shows a few examples for the first task (`step on something to reach top of a shelf`) that requires to move an object to a shelf in order to step on it and take something from the top of the shelf, which cannot be reached otherwise. The first image shows a table which is selected by the annotator since it serves the task. In the second image, however, the table is not selected, since a chair which is much handier is also present. This constitutes the additional difficulty of task driven object detection compared to object detection: the validity of a detection also depends on the presence of better options which need to be detected and assessed. One needs to understand the scene in order to judge a particular object. The third image shows a task specific preference of instances within an object category: The neglected bed on the left hand side looks heavier than the bed on the right hand side. The height of the bed on the right hand side is also sufficient to reach the top of the shelf. In this case, the choice is not anymore at the object category level, but on a finer level where attributes of the instances need to be compared. In summary,

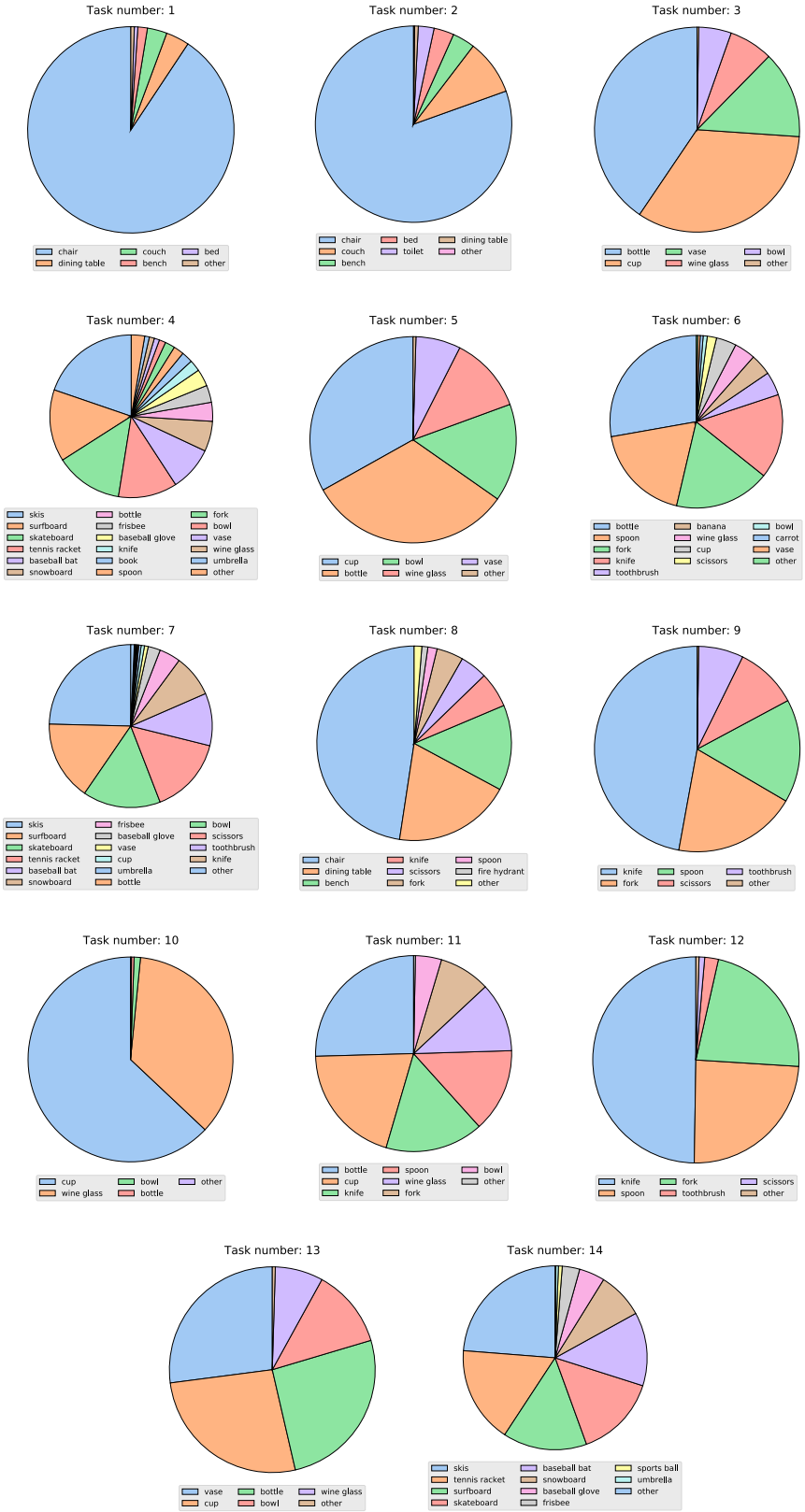


Figure 7.2: Distribution of chosen objects for all tasks across COCO categories.

	Task	Selected object categories	Objects of all selected categories	Objects chosen by humans	Intra class differentiations	Annotation consistency
1	step on something	12	30214	5783	964	0.927
2	sit comfortably	12	31392	9870	1004	0.938
3	place flowers	10	14732	3737	734	0.925
4	get potatoes out of fire	30	32775	6889	525	0.921
5	water plant	13	19050	4043	760	0.918
6	get lemon out of tea	15	22386	4707	661	0.873
7	dig hole	29	34015	6857	402	0.922
8	open bottle of beer	12	18177	1105	373	0.921
9	open parcel	7	7172	1759	160	0.921
10	serve wine	6	19209	3778	566	0.963
11	pour sugar	11	20596	5739	944	0.863
12	smear butter	9	17489	1819	270	0.896
13	extinguish fire	8	14821	2535	272	0.940
14	pound carpet	14	34160	7176	432	0.941

Table 7.1: List of the 14 tasks in the COCO-Tasks dataset and some statistics. Selected object categories (column 3) are COCO object categories for which there exists at least one instance chosen by the majority of the annotators for a given task. Column 4 reports how many instances of each of the selected categories are in the images. Column 5 provides the numbers of object instances that are chosen for each task. Column 6 counts the number of instances of categories in an image where at least one instance but not all instances of the same category are selected. Examples of such cases are shown in the last column of Figure 7.3. The last column reports the probability that two annotators agree if an object is preferred or not. Overall, we have a very high annotation consistency.

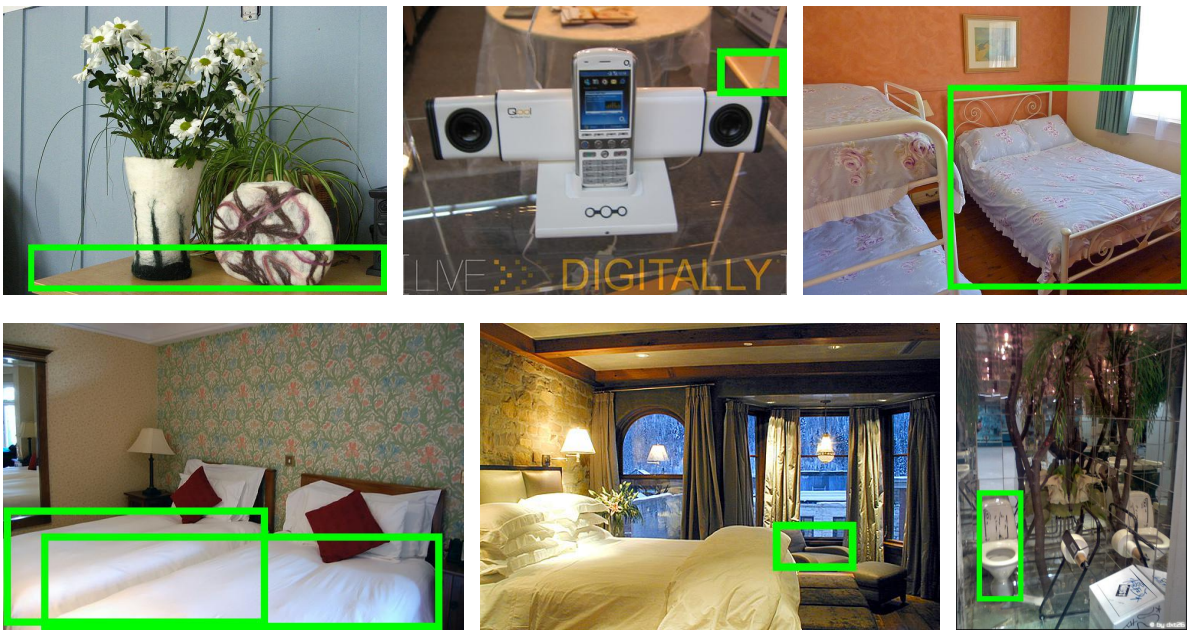


Figure 7.3: Whether an object should be chosen for a task depends on the object properties as well as the presence of better alternatives. The image in the first row shows the objects (green bounding boxes) that have been chosen by the majority of the annotators for the task `step on something to reach top of a shelf`. While in the first image the table is chosen, the chair is preferred instead of the table in the second image. In the last image, one of the two beds is chosen. The second row shows examples for the task `sit comfortably`. In the first image, both beds are selected. In the second image, the comfortable chair is preferred over the bed and and the stool. In the third image, the real toilet is selected. Note that the reflection of the toilet in the mirror is annotated as object in COCO as object, but it is not selected since one cannot sit on it.

Task #	Task	COCO supercategories
1	step on something	furniture
2	sit comfortably	furniture
3	place flowers	kitchen, outdoor
4	get potatoes out of fire	sports, kitchen, outdoor
5	water plant	kitchen, indoor
6	get lemon out of tea	kitchen
7	dig hole	sports, kitchen, indoor
8	open bottle of beer	furniture, kitchen, indoor
9	open parcel	kitchen, indoor
10	serve wine	kitchen
11	pour sugar	kitchen
12	smear butter	kitchen
13	extinguish fire	kitchen, indoor
14	pound carpet	sports

Table 7.2: We ensured that for each task images with certain COCO supercategories are overrepresented in the dataset. This table shows the respective supercategories.

task driven object detection requires a detailed understanding of an image, *i.e.*, it needs to be known what objects are in the image and what are the attributes or properties of an instance relative to other objects in an image.

In order to address the problem of task driven object detection, we introduce the COCO-Tasks dataset and we propose a first approach for task driven object detection which will be described in Section 7.4. The COCO-Tasks dataset is based on the COCO dataset (*Lin et al.*, 2014), which is the standard benchmark for object detection. We have defined 14 tasks which are listed in Table 7.1 together with some statistics. The tasks are quite diverse and include tasks that prefer a specific object shape and material like `serve wine` or `place flowers` and tasks that are related but require different attributes of the objects like `step on something to reach top of the shelf` or `sit comfortably`. For each of these tasks, we sample 3600 train images from the COCO train2014 split and 900 test images from the COCO val2014 split. To focus on more complex scenes with multiple objects to choose from, we bias the sampling procedure. For each task, we define which COCO supercategories are most useful (Table 7.2). Then we make sure that 40% of the images contain multiple categories from these supercategories, 40% contain exactly one category from these supercategories but multiple instances of it, and 10% of the images contain exactly one instance from one category. The remaining 10% are randomly sampled. In total, our train set contains 30,229 images and our test set contains 9,495 images.

In order to annotate the preferred objects in each of the 4,500 images for each task, we use the available COCO segmentation masks. We highlight the segmentation masks of all objects annotated in the COCO dataset for the annotator. To specify the requirements for the task on a more intuitive level, we visualize all tasks besides of providing a textual description of the task. For instance, we show an image of a shelf for the task `step on something to reach top of the shelf`. The annotators could choose any object, multiple objects or none of them if none of the objects is considered as suitable for this task. The annotators neither knew the procedure of sampling

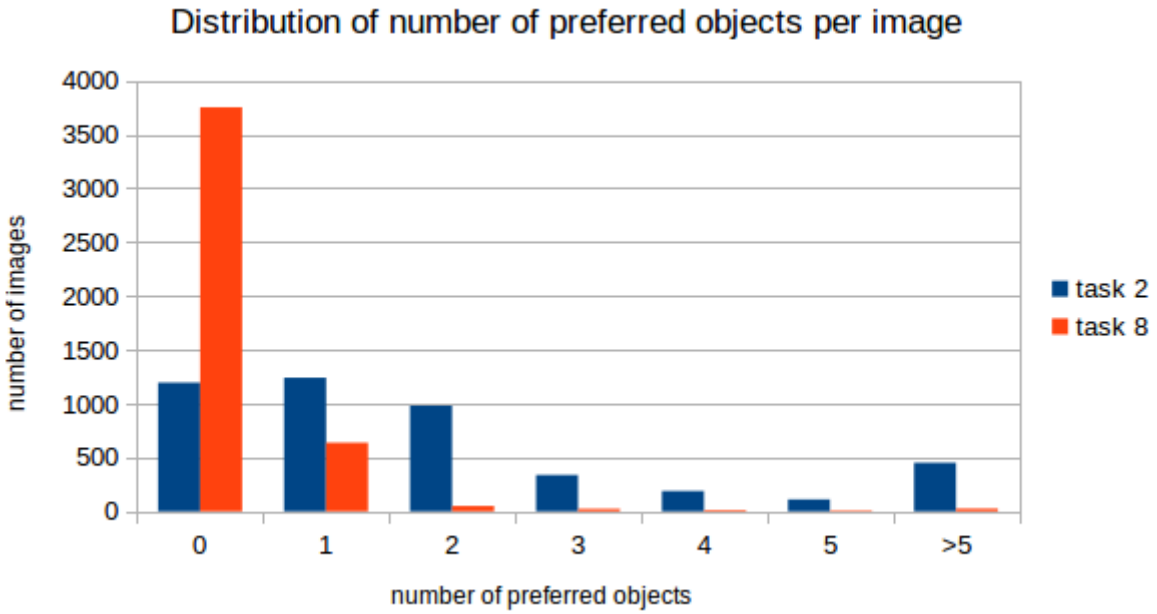


Figure 7.4: Distribution of the number of preferred objects per image for tasks 2 and 8, which are the tasks with highest and lowest number of selected instances per image, respectively.

the images nor the supercategories, i.e., they could choose from all 80 COCO categories for each task. Each task was annotated by 5 trained annotators. An object is considered to be preferred if it was chosen by the majority of the annotators. Some example annotations are shown in Figure 7.3. A screen shot of the annotation tool is provided in Figure 7.5.

Table 7.1 provides some statistics of our dataset. We measured how diverse the selected objects with respect to COCO categories are by counting all categories where at least one instance was selected by the majority of the annotators. The datasets shows a high variation in terms of categories per tasks and the number of selected object categories varies between 6 and 30 depending on the task. From the 80 COCO object class categories of the object detection challenge 2014, instances of 49 classes have been selected for at least one of the 14 tasks. Note that COCO classes also include animals, which are not relevant for the tasks in our dataset. We then measured how many instances of all selected categories for each task are present in our datasets, which also largely varies between 7,172 and 34,160 instances. This shows that just reducing the number of categories to a small set that could be relevant for a task would still leave many instances to choose from. The number of instances that have been selected for each task varies between 1,105 and 9,870. We finally provide the number of instances where the annotators differentiate between instances of the same category as it is shown in the last column of Figure 7.3. In such cases, the properties or attributes of the instances are relevant to make the decision which object should be used. In Figure 7.2, we show the distribution of selected object categories for all tasks in the dataset. While for task 10 (`serving wine`) instances from the categories wine glass and cup are mostly selected, there is a large diversity of categories that have been selected for task 4 (`getting potatoes out of the fire`). Additionally, we report the distribution of the number of selected instances per image in Figure 7.4.

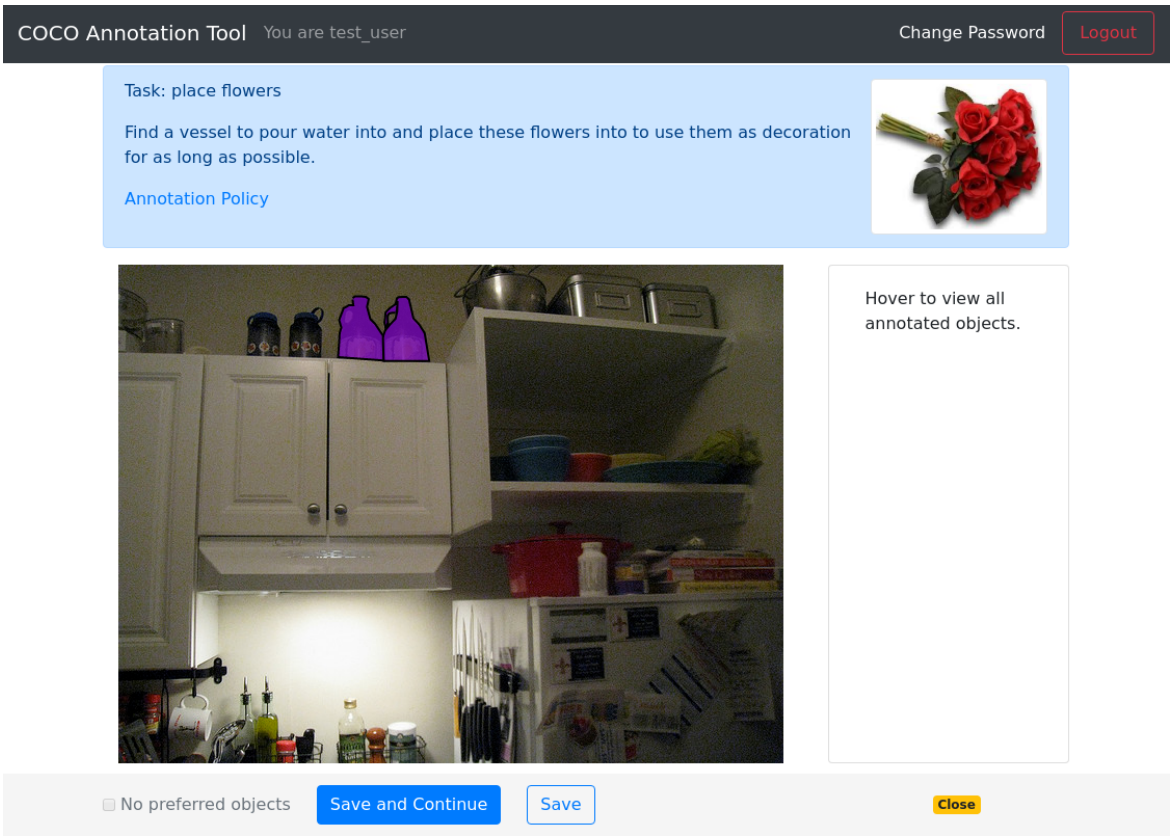


Figure 7.5: Overview of the annotation tool used by the annotators. At the top (blue section) the user can see the description of the task with an image which clarifies the intention of the task. Below is the image with already selected preferred objects (as purple). The annotator should select any object that is suitable by clicking on it. If the annotator hovers on an COCO annotated instance in the image, it will become highlighted with yellow color. No COCO class information is provided to the annotator. The user is able to click on an already selected instance to deselect it. On the right hand side of the image is an area over that the annotator can hover with the mouse to view all COCO annotated instances in the image (all with the same color). If there are no preferred instances for the task in the image, the user is able to mark a check box at the bottom and save the annotation result for the image.

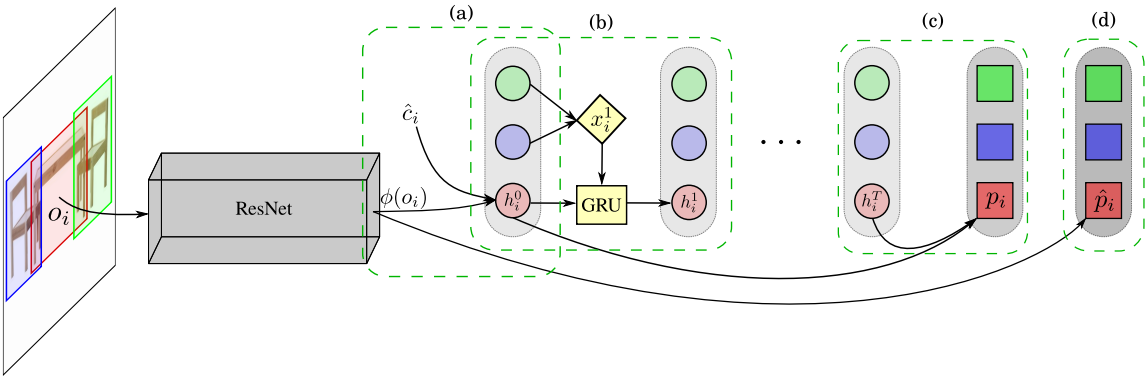


Figure 7.6: Overview of the proposed method. Given an image containing a number of objects (3 in this example shown with colors green, blue and red), our method first extracts ResNet features from each bounding box containing the object. (a) Using the extracted features and one-hot encoding of the detected class of the object ($\hat{c}_i$), we compute the initial hidden state of the graph node corresponding to that object using (7.1). (b) Using the hidden states of all of the other graph nodes, we aggregate the scene information using (7.2) and update the node's hidden state as in (7.3). (c) After T iterations of the GGNN, we combine each node's initial and final hidden states using (7.4) to compute the probability of that object being favored for the task. (d) Finally to make the features learned by the ResNet discriminative we also force the network to estimate suitability scores only from visual features of a single object. At test time, we average the two estimated probabilities.

While for `open bottle of beer` the number of suitable objects is low, there is large diversity in the number of selected instances per image for `sitting comfortable`. The examples show the large variety of category and instance distributions among the tasks. Furthermore, we evaluated the consistency of the annotations. For each task and each object, we calculated the probability that two annotators agree if this object is preferred or not. As can be seen from Table 7.1, the consistency between annotators is very high.

As evaluation metric, we use the AP@0.5 object detection evaluation metric of the COCO detection challenge (*Lin et al.*, 2014) where the preferred objects for a particular task are the ground truth instances to calculate average precision on. Taking the mean over the tasks yields mAP@0.5.

7.4 Task Driven Object Detection

In order to identify the most suitable objects in an image for a task, it is required to understand what objects are in the scene and why is an object preferred to other present objects. While the objects in an image can be detected by an off-the-shelf object detector, we have to model the relations of all present objects in an image to select the preferred objects among all detected objects. To this end, we will use a Gated Graph Neural Network (GGNN) (*Li et al.*, 2016) to model the global information of all objects in an image.

7.4.1 Proposed Method

Our model consists of a ResNet101 (*He et al.*, 2016) network without the final fully connected layer with the weights initialized from ILSVRC. On top of the ResNet features, we construct a Gated Graph Neural Network (*Li et al.*, 2016) where each node is an object in the image and each node is connected to all of the other nodes to gather the information from all of the objects present in the scene. On top of the GGNN, we have a fully connected layer which predicts the probability of each object being suitable for each task. We train the whole network end-to-end using binary cross entropy loss. Below we will describe the model in more detail.

An overview of our method is shown in Figure 7.6. Given an input image I and a collection of N detected objects in that image $o_i, i = 1, ..., N$ specified with their corresponding bounding boxes b_i, detection scores d_i and predicted category c_i, our method predicts p_i the probability of the object o_i being selected for a task.

We first preprocess the bounding boxes by making them square and 10% larger in each dimension, and then crop the image with the preprocessed bounding boxes. We then extract the features from each cropped bounding box arriving at $\phi(o_i)$.

We create a GGNN with one node for each object in the image. We set the initial hidden value of each node based on the one-hot encoding of the category of that object $\hat{c}_i$ and the ResNet features $\phi(o_i)$ such that

$$h_i^0 = g(W_c\hat{c}_i) \odot g(W_\phi\phi(o_i)) \tag{7.1}$$

where $g(.)$ is the ReLU activation, $\odot$ is the element-wise multiplication and W_c and W_ϕ are parameters of the model.

At each step of the GGNN, we first aggregate the information from all other nodes in the graph:

$$x_i^t = \sum_{j,j\neq i} W_p d_j h_j^{t-1} + b_p \tag{7.2}$$

where W_p and b_p are the parameters of the learned linear mapping in the aggregation step. This corresponds to a graph where each node is connected to all other nodes. We call the multiplication of d_j in (7.2) *weighted aggregation*. It gives the possibility to our method to account for misinformation in bad detections with low detection scores. Using the aggregated x_i^t and the previous hidden state of the node h_i^{t-1} we arrive at the new hidden state of each node in the graph using the GRU (*Cho et al.*, 2014) update rule

$$\begin{aligned}
z_i^t &= \sigma(W_z x_i^t + U_z h_i^{t-1} + b_z) \\
r_i^t &= \sigma(W_r x_i^t + U_r h_i^{t-1} + b_r) \\
\hat{h}_i^t &= tanh(W_h x_i^t + U_h(r_i^t \odot h_i^{t-1}) + b_h) \\
h_i^t &= (1 - z_i^t) \odot h_i^{t-1} + z_i^t \odot \hat{h}_i^t
\end{aligned} \tag{7.3}$$

where σ is the sigmoid activation and the GRU weights (W_z, W_r, W_h, U_z, U_r, U_h, b_z, b_r, b_h) are learned end-to-end and are shared between all tasks just like the ResNet backbone network. This update rule is applied T times. In our experiments T is set to 3. We observed that increasing T does not improve our results.

At the end of the T iterations the model calculates the probability estimate from the concatenation of the initial and final hidden state of each node

$$p_i = \sigma(f([h_i^0; h_i^T])) \tag{7.4}$$

while learning the weights. $f(.)$ corresponds to a 2 layer fully connected MLP with ReLU activations for the hidden layer where the final layer has a single output. We can modify this output model to generate one probability value for each task using a final layer with M outputs where M is the number of tasks and train a single model for all tasks jointly. In order to make the features learned by the ResNet discriminative, we also directly compute suitability estimates

$$\hat{p}_i = \sigma(\hat{f}(\phi(o_i))) \tag{7.5}$$

from only ResNet features $\phi(o_i)$ as shown in Figure 7.6 (d). We use two binary cross entropy losses during training for p_i and $\hat{p}_i$. At test time, we use average fusion of p_i and $\hat{p}_i$ to estimate the final probability.

To train our model, we construct each minibatch from objects inside a single image from our training set. All COCO annotated objects are included in the batch, the ones which are specified by our dataset as being preferred for a task are considered as positive examples for that task and the others are considered as negative. Since we use the COCO annotated bounding boxes during training, we set all d_is to 1. During testing, we first perform standard object detection on the test image and get a set of object bounding boxes and their corresponding detection scores and categories. We then perform testing by constructing a batch from all of the detected objects and estimate the probability of each object being preferred for each task as described above. The final confidence for mAP evaluation is obtained by multiplying the detection score with the estimated probability. Implementation details are provided in the supplementary material.

7.4.2 Implementation Details

We train our models using standard SGD for 3 epochs. We use an initial learning rate of 10^{-2} and decay it by a factor of 10 every epoch. During training, we shuffle the order of images before each epoch starts. After calculating the gradients, we clip them if the ℓ_2 norm of the gradients are larger than 15. For regularization, we use Dropout with 0.25 probability. We place dropout layers before final layers in all our methods. We also apply dropout to the x_i^t input to the GRU at each step. The input and hidden size of the GRU is 128. The source code to reproduce our results is available online[2].

We weight by 10 and 1 the cross entropy losses on top of p_i and $\hat{p}_i$ respectively. To compute the ranking for mAP calculations, we first compute the final probability estimate of each object and then multiply it with the detection score.

Our vanilla object detector is a derivative of Faster-RCNN (*Ren et al.*, 2015b) provided by *Yang et al.* (2017). As backbone model we take ResNet101 (*He et al.*, 2016) and the weights are updated with SGD. We use 120 region proposals, a batch size of 2, start learning rate of 0.005 and train for 10 epochs. The learning rate decays by 0.1 every 5 epochs. We use the large scale option to upscale the shorter side of the image to 800 pixels. We train on all images of the COCO train2014 split and all images of val2014 split which do not belong to our test set. The detector achieves an object detection mAP of 25.0% and a mAP@0.5 of 39.4% on our test set.

[2] coco-tasks.github.io

Comparison to Baselines mAP@0.5			
	gt bbox	Faster-RCNN detections	Yolo detections
object detector	-	0.206	-
pick best class	0.386	0.141	-
detection stats	0.675	0.141	-
ranker	0.564	0.091	-
classification	0.616	0.288	0.291
proposed + fusion	**0.742**	**0.326**	**0.332**

Table 7.3: Comparison of the proposed method to several baselines on ground truth bounding boxes as well as Faster-RCNN (*Ren et al.*, 2015b) detections. The classification baseline is the strongest one but achieves 12.6% lower mAP on ground truth bounding boxes and 3.8% lower mAP on detections compared for our proposed approach.

7.5 Experiments

In this section, we first evaluate the performance of several baselines as well as our proposed method on COCO-Tasks. After that, we demonstrate both qualitatively and quantitatively that our proposed method learns useful information about the scene context. Furthermore with ablation experiments, we show the benefits of each component of our proposed method. For all of our experiments except the object detection baseline we train and test the models three times and report the average performance numbers.

7.5.1 Comparison to Baselines

For the object detection baseline, we train a separate object detector for each task on our train set and infer on the test set. For all other baselines as well as the proposed method, we train the respective method on ground truth bounding boxes of all COCO objects in the train set. We then evaluate all algorithms on (a) ground truth bounding boxes of COCO objects and (b) COCO object detections of a Faster-RCNN object detector (*Ren et al.*, 2015b). While the latter evaluates the performance in a realistic scenario, the former demonstrates the potential of our method that can be reached with a perfect object detector. As metric, we use mAP@0.5 for all experiments and report the numbers in Table 7.3.

Object Detector Baseline. The most straightforward approach for task driven object detection is to treat it as a standard object detection task. To this end, for each of the 14 tasks, we train a 1-class object detector. All objects preferred for the respective task constitute the object class to detect. As detector, we use the same Faster-RCNN implementation. Apart from changing the number of classes from 80 to 1, we reduce the learning rate from 0.005 to 0.0001, all other hyperparameters stay identical. As reported in Table 7.3, this yields an mAP@0.5 of 20.6%, which is more than 10% lower than the proposed approach. This verifies that task driven object detection can not be treated as a standard object detection task because of the necessity to look for scene context and all present objects.

Pick Best Class Baseline. COCO classes differ significantly in their suitability for household tasks. To analyse this effect, we first rank the classes for each task by the fraction of all instances

of this class to be preferred on the train set. Then for each task and each image of the test set, we omit all detections with detection confidence lower than 0.1. Among the remaining detections, we determine the highest ranked class and only keep the detections belonging to this class with their detection confidence as final confidence. The result is 14.1% on detections which is significantly worse than the *object detector* baseline. On ground truth bounding boxes, this baseline yields only 38.6% mAP@0.5. This shows that the task driven object detection problem is not solvable by the category information alone, but visual information from the objects and image context are required.

Detection Stats Baseline. We want to test if visual input is necessary or if the statistics of the present objects alone is sufficient to solve task driven object detection. To this end, we trained a smaller 2-layer MLP to predict from the normalized histogram of present object classes as well as the average normalized bounding box coordinates for each class, the preferred object class. As for the proposed method, we obtained the final confidence by multiplying the MLP output and the detector confidence. As for the *pick best class* baseline, we prefilter the detections by a detection confidence of 0.1. While this baseline (MLP on detection stats) performs well for ground-truth bounding boxes, it is still worse than the proposed approach. For detected bounding boxes, MLP performs poorly.

Ranker Baseline. For the *ranker* baseline, we train a model similar to Deep Relative Attributes (*Souri et al.*, 2016) to rank COCO objects in terms of their suitability for a task. We exchange the original VGG16 backbone (*Simonyan and Zisserman*, 2015) of the ranker for a ResNet101 (*He et al.*, 2016) backbone to make the method comparable to other baselines. We train a model for each task separately using the Adam optimizer with 10^{-4} learning rate for 3 epochs to remain as close as possible to *Souri et al.* (2016). As for the *pick best class* baseline, we prefilter the detections by a detection confidence threshold of 0.1. Then for each image and each task, we rank all n detections and assign each detection i of rank r_i the confidence $c_i = 1 - \frac{r_i - 1}{n}$. Although on ground truth bounding boxes this method performs better than *pick best class*, it is the worst baseline on detections giving only 9.1% mAP@0.5 as can be seen from Table 7.3. The reason is that a single detection ranked erroneously highly affects all other detections.

Classification Baseline. To investigate if a global analysis of all objects present in the scene is necessary, we train a binary classifier on top of the ResNet features for each task and apply it on detections and ground truth bounding boxes. As for the proposed method, we obtained the final confidence by multiplying the classifier output and the detector confidence. This baseline model is equivalent to our method, without the class information input, the context modeling using a graph and joint training for all tasks simultaneously. This is the strongest baseline as can be seen from Table 7.3. It gives 61.6% on ground truth bounding boxes and 28.8% on detections. However, this is still substantially below our proposed method which takes the scene context into account.

Proposed Method. The proposed method with fusion where we average p_i and $\hat{p}_i$ for final estimate, yields 32.6% on Faster-RCNN (*Ren et al.*, 2015b) detections and 74.2% on ground truth bounding boxes outperforming our baselines by a large margin. Various ablation experiments showing the effect of different components of our method will follow.

Other Detector Our method outperforms the strongest classifier baseline even if we use the Yolov2 detector (*Redmon and Farhadi*, 2017) as can be seen from Table 7.3.

Ablation experiment results, mAP@0.5		
	gt bbox	Faster-RCNN detections
classifier	0.616	0.288
(a) joint classifier	0.647	0.302
(b) joint classifier + class	0.719	0.301
(c) joint GGNN + class	0.763	0.293
(d) joint GGNN + class + w. aggreg.	-	0.303
(e) proposed	**0.771**	0.318
(f) proposed + fusion	0.742	**0.326**
(g) no visual input	0.589	0.237
(h) no visual input + bounding box	0.412	0.152

Table 7.4: Evaluation of the components of our proposed method. We start with a task wise classifier, (a) then add joint training, (b) add COCO classes as input, (c) introduce the GGNN, (d) add weighted aggregation, (e) add the discriminatory loss and (f) perform fusion. Further ablation experiments (g) and (h) reveal the impact of the visual information.

7.5.2 Ablation Experiments

We observed that the classification baseline was lacking in performance compared to our proposed method. This is due to the differences between the classification baseline and our proposed method. These differences are: (a) joint training of all tasks together, (b) direct class information input, and (c) GGNN for scene context modeling. We will add these 3 components one by one to the classification baseline and show the effect of each of them. Furthermore, in our GGNN we show the effect of (d) *weighted aggregation*, (e) the direct discriminatory loss on top of the ResNet features and (f) fusion of p_i and $\hat{p}_i$ for the final probability estimate. The results for these ablation experiments are reported in Table 7.4.

a) Joint Training. While for the classification baseline we train a separate classifier for each of the tasks, a first improvement can be easily obtained by training a classifier jointly for all tasks, *i.e.*using shared features. This is done by replacing the final single output fully connected layer that estimates p_i into a layer with M outputs, where M is the number of tasks. If a task is annotated for an image during training, we calculate the binary cross entropy loss and skip that task otherwise. Training the classifier jointly increases the performance on ground truth bounding boxes from 61.6% to 64.7% and on detections from 28.8% to 30.2%. We think this is due to the higher number of training images and better features that are learned by ResNet.

b) Direct Class Information Input. The object's class as a direct input provides additional valuable information that might be harder for the network to learn from ResNet features. Given this insight we use (7.1) to combine the ResNet features ($\phi(o_i)$) and the one-hot encoding of the classes ($\hat{c}_i$) as it is done in our proposed method. We then use the hidden representations h_i^0 as input to the final classification layer. During training we use the ground truth class, during inference we use the detected classes which might be noisy. On ground truth, the results get boosted from 64.7% to 71.9%. However, on detections, the performance stays almost the same. We reckon that this is due to the difference between reliable ground truth classes during training and erroneous classes as predicted by the detector during inference. In our proposed method, this problem is addressed by our *weighted*

aggregation mechanism.

c) GGNN for Scene Context Modeling. We now add the GGNNs as described in Section 7.4.1 to see the effect of scene context modeling. For this ablation experiment, the *weighted aggregation* (by setting all d_is in (7.2) to 1) and the discriminator loss are not used. This is equivalent to a simplified GGNN. On ground truth bounding boxes we get an improvement of 4.4% arriving at 76.3% as a result of scene context modeling, but on detections the performance slightly drops to 29.3%. The detection confidence problems encountered by the classifier are amplified, since the GGNN takes all detections into account when judging a single one. Thus the final result for each detection is affected by low confidence detections during inference. Typically these are wrong detections, thus the GGNN is confronted with visual input not seen during training. To solve this issue, we have incorporated the *weighted aggregation*.

d) Weighted Aggregation. By the *weighted aggregation*, we take the confidences of the detections d_is into account (7.2). We observe that addition of such weighting improves our results considerably on detections. This thwarts the propagation of visual features of low confidence detections through the GGNN resulting in an improvement from 29.3% to 30.3%. Note that the *weighted aggregation* does not change the result on ground truth bounding boxes since the d_is are equal to 1 in this case.

e) Direct Discriminator Loss. We also impose intermediate supervision on the visual features fed into the initializer. We add a fully connected layer mapping these features onto probabilities for each task and apply a task wise binary cross entropy loss to these probabilities. This loss makes the visual features more discriminative for the final goal. The features give a better backup in case the class information is not correct. In general such a loss improves the performance of our model to 77.1% and 31.8% on ground truth bounding boxes and detections, respectively.

f) Probability Fusion. Average fusion of the probabilities p_i from (7.4) and $\hat{p}_i$ from (7.5) further improves the results on detections. We observe that this does cause some performance decrease for the case of a perfect detector. The fusion is therefore only relevant if the detections are noisy.

g) Removing Visual Input $\phi(o_i)$. Since class information improves the results on ground truth bounding boxes significantly, the question comes to mind if visual information inside the bounding boxes is necessary at all. To test this, we do not use the visual features $\phi(o_i)$ for GGNN and only keep the class information as input. As a result, the mAP significantly drops, showing that the appearance of the objects is very important for the task and that GGNN takes it into account.

h) Removing Visual Input $\phi(o_i)$ and Adding Bounding Box Geometry. We then used the coordinates of the bounding boxes normalized by image width and height $b(o_i)$ instead of the visual features $\phi(o_i)$ for GGNN (proposed no vis. input + bbox). This leads to even worse results since the model overfits to the coordinates of the bounding boxes of the objects inside the training images.

7.5.3 Class Bias Analysis

In Table 7.5, we report the results only for the subset of the test images where an instance of the most suitable class is present, but an instance of another suitable class has been selected by the users. We conducted the experiment for our proposed method as well as the classification baseline which is the strongest baseline on Faster-RCNN detections and the detection stats baseline which is the strongest on ground truth bounding boxes. As expected, these cases are much more difficult and mAP@0.5 is much lower compared to Table 7.3. Nevertheless, our approach outperforms the baselines. This

Comparison to Baselines mAP@0.5		
	gt bbox	Faster-RCNN detections
detection stats	0.162	0.039
classification	0.197	0.090
proposed + fusion	**0.247**	**0.115**

Table 7.5: Results on the subset of the test images where an object belonging to the most frequently chosen category was neglected in favor of an object from a less frequently chosen category.

shows that the proposed model takes the appearance of the object and the scene context into account and is able to make decisions against the class bias, but there is a large room for improvement even for ground truth bounding boxes.

7.5.4 Task Wise Difficulty Analysis

In Table 7.6 we report the mAP@0.5 for each task of the proposed method on ground truth bounding boxes and the proposed+fusion method for Faster-RCNN (*Ren et al.*, 2015b) detections and Yolov2 (*Redmon and Farhadi*, 2017) detections. As expected, our method generalizes well to another detector. The performance on the individual tasks ranges from 37.0% to 96.1% on ground truth bounding boxes. The different difficulty of the tasks can be explained by the typical class of the preferred object.

Consider the pie charts in Figure 7.2. The three tasks with mAP@0.5 higher than 90% on ground truth detections can typically be solved with sports tools and these occur in sports images. In these images, only a small number of objects of one selected category occurs with little functional intra class differentiation. Therefore selecting all objects in this category is a good choice. Contrary to that, the five tasks where our proposed method gives less than 70% on ground truth detections, have to be solved with vessels or cutlery. These objects typically occur on kitchen or party images, showing a high number of selected categories and multiple small objects of each category. Especially for the task "open bottle of beer", different objects of the same category might have a suitable ridge or not, i.e. exhibit a high functional intra class variance.

7.5.5 Scene Context Learned by GGNN

The aim of introducing the GGNN was to consider scene context in our model. Intuitively, the GGNN aggregates the information about all objects relevant for the task which are present in the image and stores them in the final hidden node representation h_i^T. To prove this intuition quantitatively, we retrieve the 5 most similar objects for each task and each object of the test set. Then we use the categories present in the scene of the query object as a prediction for the categories present in the scene of the retrieved objects and measure the prediction accuracy. We compute the similarity based on h_i^T, which should contain scene information and compare it to the similarity computed based on h_i^0.

The prediction accuracies are high in both cases, which is primarily due to the fact that most COCO categories are absent in any image. However as can be seen from Figure 7.7, when retrieving based on similarity of h_i^0 instead of h_i^T, the accuracy of this prediction is significantly lower for all tasks. This verifies our intuition.

Task wise results, mAP@0.5															
task #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	**avg.**
gt bboxes	0.831	0.853	0.732	0.934	0.699	0.690	0.961	0.370	0.804	0.810	0.638	0.803	0.706	0.960	0.771
Faster-RCNN dets	0.366	0.298	0.405	0.376	0.410	0.172	0.436	0.179	0.210	0.406	0.223	0.284	0.391	0.407	0.326
Yolov2 dets	0.368	0.319	0.391	0.380	0.416	0.165	0.444	0.187	0.230	0.390	0.223	0.269	0.440	0.420	0.332

Table 7.6: We present the task wise numbers of the proposed method on Faster-RCNN (*Ren et al.*, 2015b) detections, the Yolov2 (*Redmon and Farhadi*, 2017) detection and on ground truth bounding boxes.

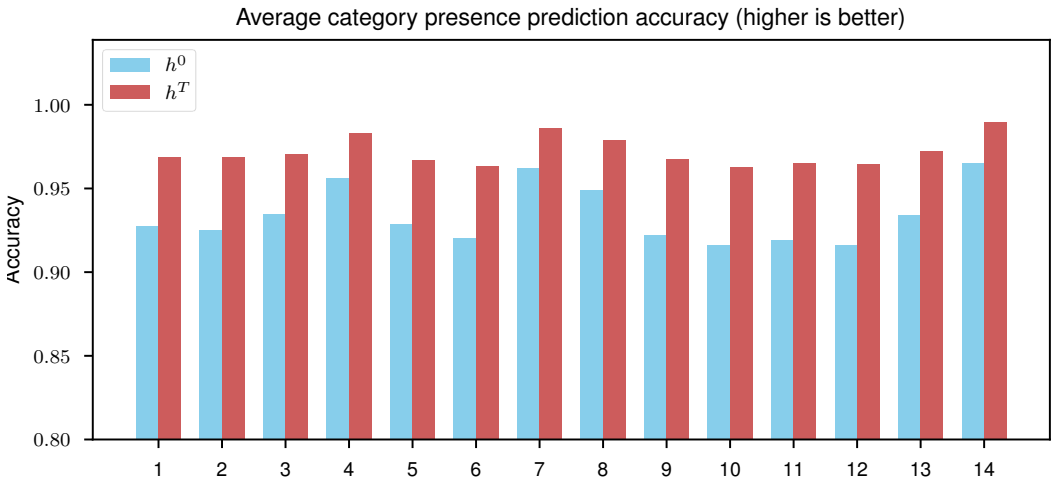


Figure 7.7: Accuracy of the predicted categories when retrieving nearest neighbors based on h_i^0 vs. retrieving based on h_i^T.

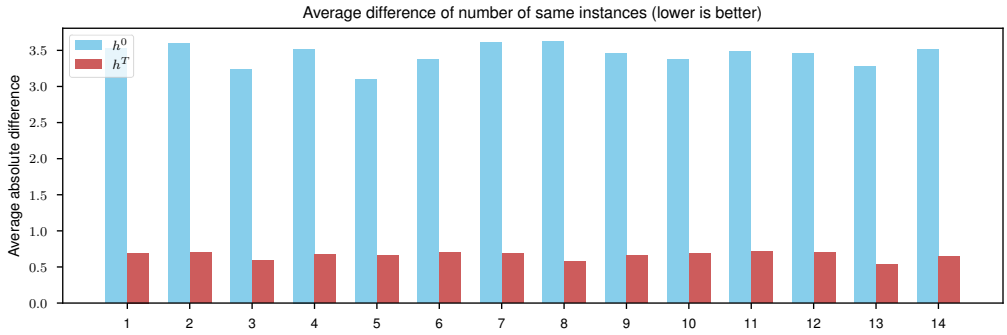


Figure 7.8: Average difference of the number of same instances between the query image and the nearest neighbor images based on h_i^0 vs. h_i^T.



Figure 7.9: A qualitative example from Task 1 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.



Figure 7.10: A qualitative example from Task 2 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

We confirm this observation by another experiment. We measure the absolute difference between the number of instances belonging to the COCO category of the query object in the example image and the retrieved images. When using nearest neighbors according to the ℓ_2 distance between h_i^0, the difference is far higher than when using h_i^T (Figure 7.8). This implies that the GGNN also improves the ability of the model to implicitly count the objects.

We also provide some qualitative nearest neighbor comparisons. In Figures 7.9 to 7.22, we show the query object with a red bounding box on the left and the top 5 nearest neighbor objects from the test set of each task based on h_i^0 and h_i^T on the top row and the bottom row, respectively. In general, we can see that the retrieved images based on the h_i^T distance show a more similar scene configuration. We would like to mention that although the displayed examples show better



Figure 7.11: A qualitative example from Task 3 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

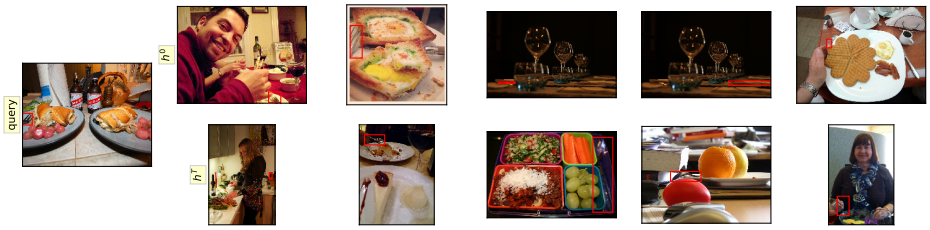


Figure 7.12: A qualitative example from Task 4 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

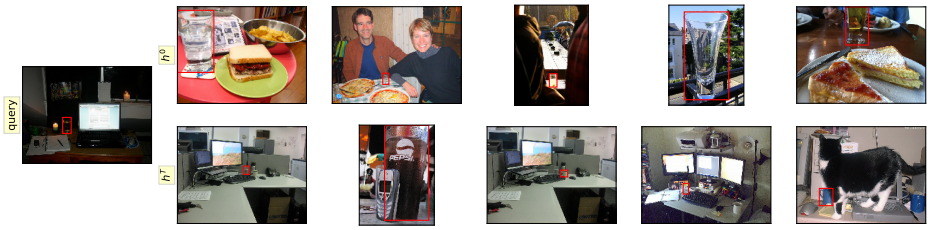


Figure 7.13: A qualitative example from Task 5 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.



Figure 7.14: A qualitative example from Task 6 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.



Figure 7.15: A qualitative example from Task 7 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

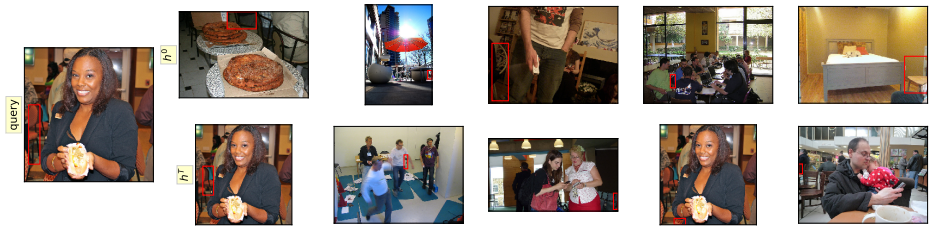


Figure 7.16: A qualitative example from Task 8 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

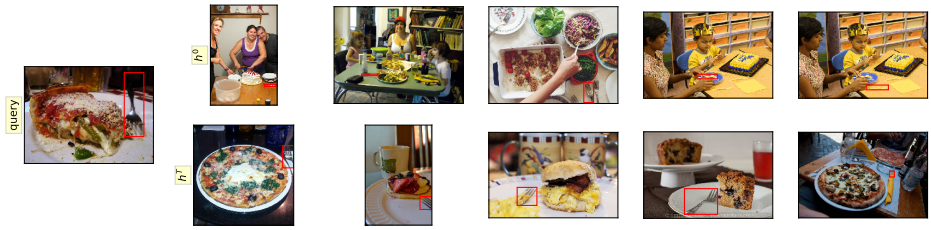


Figure 7.17: A qualitative example from Task 9 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.



Figure 7.18: A qualitative example from Task 10 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

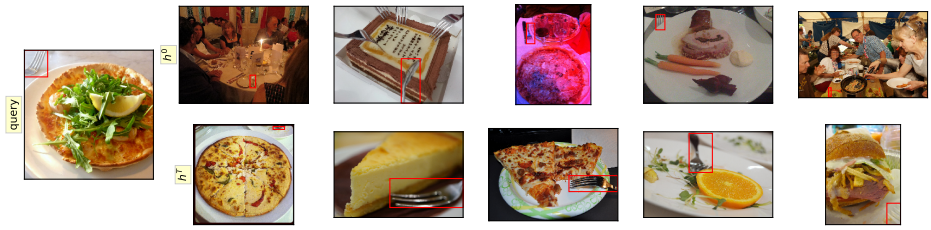


Figure 7.19: A qualitative example from Task 11 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

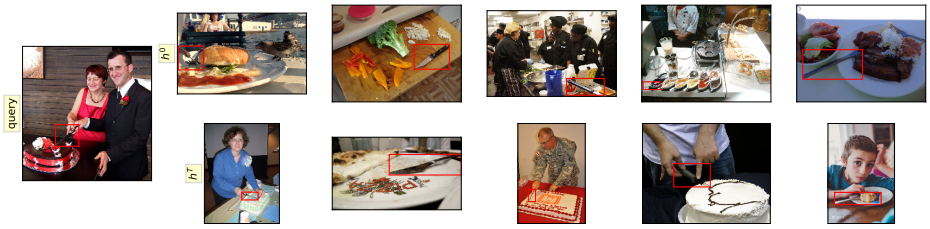


Figure 7.20: A qualitative example from Task 12 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

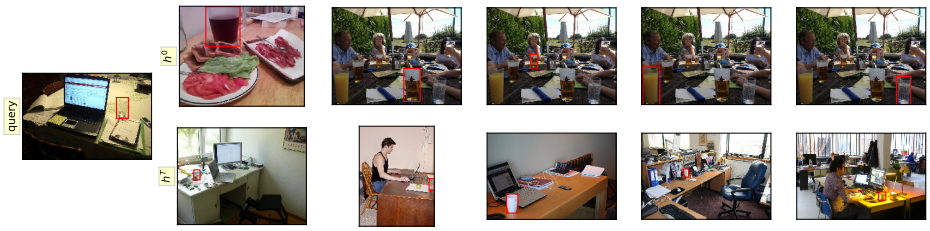


Figure 7.21: A qualitative example from Task 13 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

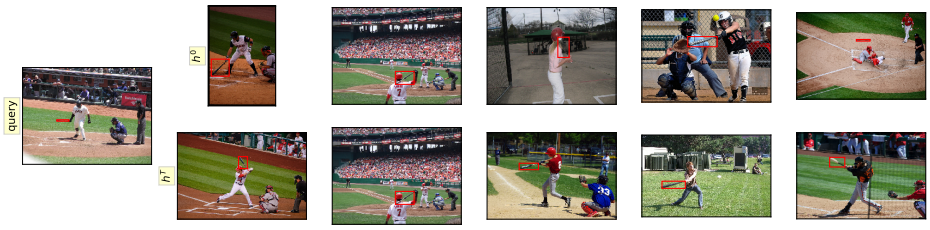


Figure 7.22: A qualitative example from Task 14 test set comparing nearest neighbors according to h_i^0 vs. h_i^T.

scene configuration similarity for the retrieved images based on h_i^T, this is not always the case and qualitative examples are sometimes hard to interpret.

7.6 Conclusion

In this chapter, we have addressed the problem of task driven object detection. In contrast to standard object detection, it requires to detect and select the best objects for solving a given task. To study this problem, we created a dataset based on the COCO dataset (*Lin et al.*, 2014). It comprises about 40k images with annotations for 14 tasks. We evaluated several baselines based on ranking or classification approaches on this dataset. We furthermore introduced a novel approach for this task that takes as input all detected objects in an image and uses a Gated Graph Neural Network to model the relations of the object hypotheses in order to infer the objects that are preferred for a given task.

CHAPTER 8

Conclusion

In this chapter, we summarize the contributions of this thesis and discuss some promising directions for future work in temporal action segmentation.

Contents

8.1 Summary 125
8.2 Future Work 126
8.2.1 Larger Datasets 126
8.2.2 Incorporating FIFA Inference in MuCon Training 126
8.2.3 Temporal Action Segmentation From Multiple Levels of Supervision . . . 127
8.2.4 Feature Learning for Temporal Action Segmentation 127
8.2.5 Moving Beyond Fixed Tasks for Task Driven Object Detection 127

8.1 Summary

In this thesis, we focused on the efficiency and weak supervision for temporal action segmentation. We proposed a novel approach for weakly supervised action segmentation from transcripts. It consists of a two-branch network that predicts two representations for action segmentation. To train the network, we introduced a new mutual consistency loss (MuCon) that enforces these two representations to be consistent during training. Using MuCon and a transcript prediction loss, we can train our network end-to-end without the need for additional steps. We show that the proposed network with the mutual consistency loss achieved an accuracy that is either on par or better than the state-of-the-art. At the same time, it is much faster during inference and offers the best trade-off between accuracy and inference time. Furthermore, we have shown that the mutual consistency loss increases the accuracy even for fully supervised learning and that the network can be applied to a mixed setting where a few videos are fully annotated and the other videos are only weakly annotated.

By noticing the important role inference plays in temporal action segmentation, we then focus on the efficiency of the inference algorithm. We proposed FIFA a fast approximate inference procedure for action segmentation and alignment. Unlike previous methods, our approach does not rely on dynamic programming-based Viterbi decoding for inference. Instead, FIFA optimizes a differentiable energy function that can be minimized using gradient-descent which allows for a fast and accurate inference during testing. We evaluated FIFA on top of fully and weakly supervised methods trained on the Breakfast and Hollywood extended datasets. The results show that FIFA can achieve comparable or better performance while being at least 5 times faster than exact inference.

Our final contribution to temporal action segmentation was weakly supervised learning from timestamps. We proposed a robust action segmentation approach for timestamp supervision. In

contrast to previous works that assume all action segments are annotated, we proposed a label generation method that handles missing segment annotations and is thus more suitable for real-world applications. Our approach achieves competitive results when all segments are annotated and shows a relatively small drop in performance with increasing percentage of missing segments.

Furthermore, we investigated the problem of learning what object can be used to execute an action. Taking actions can be seen as the step after recognizing actions. In our final contribution, we have addressed the problem of task driven object detection or finding the most suitable object to execute an action. In contrast to standard object detection, it requires to detect and select the best objects for solving a given task. To study this problem, we created a dataset. It comprises about 40k images with annotations for 14 tasks. We evaluated several baselines based on ranking or classification approaches on this dataset. We furthermore introduced a novel approach for this task that takes as input all detected objects in an image and uses a Gated Graph Neural Network to model the relations of the object hypotheses to infer the objects that are preferred for a given task.

8.2 Future Work

We believe there are still promising directions for future research in temporal action segmentation. Below we discuss some of these directions.

8.2.1 Larger Datasets

The datasets (Section 2.5) used in this thesis and the current approaches in temporal action segmentation are relatively small in size and limited in diversity. Breakfast (*Kuehne et al.*, 2014) which is the largest dataset currently in temporal action segmentation is recorded in a limited environment and it only contains around 1700 videos. Given such small datasets, approaches like MuCon (Chapter 4) struggle to learn to predict the transcripts without overfitting. Furthermore, the current models trained on the Breakfast dataset, have very limited use in the real world, since these models do not generalize to other environments. We think similar to other fields in computer vision, the production, use, and analysis of a much larger dataset can be of great value. Recently some new untrimmed video understanding datasets have been proposed. Mainly COIN (*Tang et al.*, 2019), CrossTask (*Zhukov et al.*, 2019), EpicKitchens (*Damen et al.*, 2018), Ego4D (*Grauman et al.*, 2022). Although these datasets are not gathered and annotated similar to the typical temporal action segmentation datasets (Breakfast or 50Salads), their largescale nature and diversity is of great value. Moving towards utilizing and learning from larger scale and more diverse datasets is a promising direction for temporal action segmentation models.

8.2.2 Incorporating FIFA Inference in MuCon Training

Our proposed approach for fast weakly supervised temporal action segmentation MuCon (Chapter 4), does not include temporal action segmentation inference during its training. The role of temporal action segmentation inference during training which is used by other approaches such as CDFL (*Li et al.*, 2019) and NNV (*Richard et al.*, 2018b) is to use a better pseudo ground truth for weakly supervised training. The issue with the exact temporal action segmentation inference used in CDFL and NNV are that it is slow and non-differentiable which made it not applicable for use in MuCon.

Since the proposed approximate inference FIFA (Chapter 5) is fast and differentiable, incorporating FIFA as part of the MuCon training is an interesting future direction.

8.2.3 Temporal Action Segmentation From Multiple Levels of Supervision

The current approaches for temporal action segmentation from full supervision differ significantly from the approaches for weakly supervised temporal action segmentation. For example, the MS-TCN (*Abu Farha and Gall*, 2019) approach for fully supervised temporal action segmentation is different from the CDFL approach (*Li et al.*, 2019) in backbone network design, loss function, and inference. It can be observed that the fully and weakly supervised approaches have diverged. With MuCon(Chapter 4) we introduced an approach that unifies fully supervised and transcript supervised temporal action segmentation, allowing for mixed supervision training. But the fully supervised MuCon's performance is not as good as other state-of-the-art approaches. Designing an approach for temporal action segmentation from any level of supervision (full, timestamp, transcript, and set) is an exciting future direction.

8.2.4 Feature Learning for Temporal Action Segmentation

Current approaches for temporal action segmentation and untrimmed video understanding in general, utilize a pretrained model as feature extractor for the input video (Section 2.2.1). During the training of the model weights, the feature extractor network's weights are fixed. Since the feature extraction network is trained on a different task (usually action recognition), the features are suboptimal for temporal action segmentation. We think approaches that can learn better features for temporal action segmentation either using end-to-end training or by defining some auxiliary task could be very important in the advancement of the temporal action segmentation field.

8.2.5 Moving Beyond Fixed Tasks for Task Driven Object Detection

The proposed approach in Chapter 7 for task driven object detection trains a model for each pre-defined tasks. In realistic settings, it is hard to pre-defined a set of tasks and train a model for each task. In the current proposed approach for a novel task, even if it is semantically similar, a training set and retraining of the model is required. A promising future direction is to utilize large scale pretrained language models such as GPT (*Radford et al.*, 2018) and BERT (*Devlin et al.*, 2019) and extend the task driven object detector to utilize the embedding of the task definition. By utilizing the embedding of a task definition instead of training a detector for each task separately, the model could be applied to new and related tasks in a zero-shot setting.

Bibliography

Abu Farha, Yazan and Gall, Juergen. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on pages viii, 2, 3, 4, 13, 21, 27, 28, 29, 34, 38, 43, 45, 46, 51, 53, 58, 60, 66, 69, 70, 71, 72, 82, 85, 96, 97 and 127.)

Anderson, Peter; He, Xiaodong; Buehler, Chris; Teney, Damien; Johnson, Mark; Gould, Stephen, and Zhang, Lei. Bottom-up and top-down attention for image captioning and visual question answering. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 102.)

Antol, Stanislaw; Agrawal, Aishwarya; Lu, Jiasen; Mitchell, Margaret; Batra, Dhruv; Lawrence Zitnick, C, and Parikh, Devi. VQA: Visual question answering. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. (Cited on page 101.)

Arnab, Anurag; Dehghani, Mostafa; Heigold, Georg; Sun, Chen; Lučić, Mario, and Schmid, Cordelia. Vivit: A video vision transformer. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. (Cited on pages 1 and 26.)

Bahdanau, Dzmitry; Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on pages 31 and 38.)

Belanger, David and McCallum, Andrew. Structured prediction energy networks. In *International Conference on Machine Learning (ICML)*, 2016. (Cited on page 31.)

Belanger, David; Yang, Bishan, and McCallum, Andrew. End-to-end learning for structured prediction energy networks. In *International Conference on Machine Learning (ICML)*, 2017. (Cited on page 31.)

Bengio, Yoshua; Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 1994. (Cited on page 17.)

Bertasius, Gedas; Wang, Heng, and Torresani, Lorenzo. Is space-time attention all you need for video understanding? In *International Conference on Machine Learning (ICML)*, 2021. (Cited on page 26.)

Bodla, Navaneeth; Singh, Bharat; Chellappa, Rama, and Davis, Larry S. Soft-NMS – Improving object detection with one line of code. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. (Cited on page 101.)

Bojanowski, Piotr; Lajugie, Rémi; Bach, Francis; Laptev, Ivan; Ponce, Jean; Schmid, Cordelia, and Sivic, Josef. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision (ECCV)*, 2014. (Cited on pages 23, 29, 45, 50, 59, 66, 69 and 71.)

Carreira, Joao and Zisserman, Andrew. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages vii, 1, 12, 13, 25, 26, 34 and 46.)

Carreira, Joao; Noland, Eric; Hillier, Chloe, and Zisserman, Andrew. A short note on the kinetics-700 human action dataset, 2019. (Cited on page 1.)

Chang, Chien-Yi; Huang, De-An; Sui, Yanan; Fei-Fei, Li, and Niebles, Juan Carlos. D^3TW: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on pages 29, 34, 45, 52, 70, 71, 72 and 97.)

Cho, Kyunghyun; Van Merriënboer, Bart; Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014. (Cited on pages 18, 38 and 110.)

Chollet, François. Xception: Deep learning with depthwise separable convolutions. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 101.)

Chuang, Ching-Yao; Li, Jiaman; Torralba, Antonio, and Fidler, Sanja. Learning to act properly: Predicting and explaining affordances from images. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 102.)

Cuturi, Marco and Blondel, Mathieu. Soft-dtw: A differentiable loss function for time-series. In *International Conference on Machine Learning*, 2017. (Cited on page 29.)

Dai, Bo; Zhang, Yuqi, and Lin, Dahua. Detecting visual relationships with deep relational networks. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017a. (Cited on page 102.)

Dai, Jifeng; Qi, Haozhi; Xiong, Yuwen; Li, Yi; Zhang, Guodong; Hu, Han, and Wei, Yichen. Deformable convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017b. (Cited on page 101.)

Dai, Rui; Das, Srijan; Sharma, Saurav; Minciullo, Luca; Garattoni, Lorenzo; Bremond, Francois, and Francesca, Gianpiero. Toyota smarthome untrimmed: Real-world untrimmed videos for activity detection, 2020. (Cited on page 82.)

Damen, Dima; Leelasawassuk, Teesid; Haines, Osian; Calway, Andrew, and Mayol-Cuevas, Walterio. You-do, I-learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. In *British Machine Vision Conference (BMVC)*, 2014. (Cited on pages 23 and 88.)

Damen, Dima; Doughty, Hazel; Farinella, Giovanni Maria; Fidler, Sanja; Furnari, Antonino; Kazakos, Evangelos; Moltisanti, Davide; Munro, Jonathan; Perrett, Toby; Price, Will, and Wray, Michael. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on page 126.)

Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton, and Toutanova, Kristina. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019. (Cited on page 127.)

Diba, Ali; Fayyaz, Mohsen; Sharma, Vivek; Hossein Karami, A.; Mahdi Arzani, M.; Yousefzadeh, Rahman, and Van Gool, Luc. Temporal 3d convnets using temporal transition layer. In *IEEE Computer Vision and Pattern Recognition Workshops*, 2018a. (Cited on page 34.)

Diba, Ali; Fayyaz, Mohsen; Sharma, Vivek; Mahdi Arzani, M.; Yousefzadeh, Rahman; Gall, Juergen, and Van Gool, Luc. Spatio-temporal channel correlation networks for action classification. In *European Conference on Computer Vision (ECCV)*, 2018b. (Cited on page 34.)

Diba, Ali; Fayyaz, Mohsen; Sharma, Vivek; Manohar, Paluri.; Gall, Juergen; Stiefelhagen, Rainer, and Van Gool, Luc. Large scale holistic video understanding. In *European Conference on Computer Vision (ECCV)*, 2020. (Cited on pages 1 and 34.)

Ding, Li and Xu, Chenliang. Weakly-supervised action segmentation with iterative soft boundary assignment. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on pages xiv, 22, 27, 30, 34, 35, 37, 45, 46, 48, 49, 52, 53, 70, 71, 72 and 97.)

Do, Thanh-Toan; Nguyen, Anh; Reid, Ian; Caldwell, Darwin G., and Tsagarakis, Nikos G. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018. (Cited on page 102.)

Dosovitskiy, Alexey; Beyer, Lucas; Kolesnikov, Alexander; Weissenborn, Dirk; Zhai, Xiaohua; Unterthiner, Thomas; Dehghani, Mostafa; Minderer, Matthias; Heigold, Georg; Gelly, Sylvain; Uszkoreit, Jakob, and Houlsby, Neil. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. (Cited on page 26.)

Dumoulin, Vincent and Visin, Francesco. A guide to convolution arithmetic for deep learning, 2016. (Cited on pages viii and 17.)

Duvenaud, David K; Maclaurin, Dougal; Iparraguirre, Jorge; Bombarell, Rafael; Hirzel, Timothy; Aspuru-Guzik, Alan, and Adams, Ryan P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. (Cited on page 102.)

Everingham, Mark; Eslami, SM Ali; Van Gool, Luc; Williams, Christopher K.I.; Winn, John, and Zisserman, Andrew. The PASCAL Visual Object Classes Challenge 2012: A Retrospective. *International Journal of Computer Vision (IJCV)*, 2014. (Cited on pages 100 and 101.)

Fan, Haoqi; Xiong, Bo; Mangalam, Karttikeya; Li, Yanghao; Yan, Zhicheng; Malik, Jitendra, and Feichtenhofer, Christoph. Multiscale vision transformers. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. (Cited on page 26.)

Fang, Kuan; Wu, Te-Lin; Yang, Daniel; Savarese, Silvio, and Lim, Joseph J. Demo2Vec: Reasoning object affordances from online videos. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 102.)

Fathi, Alireza; Ren, Xiaofeng, and Rehg, James M. Learning to recognize objects in egocentric activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. (Cited on pages 23 and 88.)

Fayyaz, Mohsen and Gall, Jurgen. SCT: Set constrained temporal transformer for set supervised action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on pages 2, 30, 82 and 97.)

Feichtenhofer, Christoph. X3d: Expanding architectures for efficient video recognition. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on pages 1 and 26.)

Feichtenhofer, Christoph; Fan, Haoqi; Malik, Jitendra, and He, Kaiming. Slowfast networks for video recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. (Cited on pages 1, 26 and 34.)

Feichtenhofer, Christoph; Fan, Haoqi; Xiong, Bo; Girshick, Ross, and He, Kaiming. A large-scale study on unsupervised spatiotemporal representation learning. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2021. (Cited on page 26.)

Friedman, Milton. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 1937. (Cited on page 48.)

Gao, Haoyuan; Mao, Junhua; Zhou, Jie; Huang, Zhiheng; Wang, Lei, and Xu, Wei. Are you talking to a machine? Dataset and methods for multilingual image question. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. (Cited on page 101.)

Gao, Shang-Hua; Han, Qi; Li, Zhong-Yu; Peng, Pai; Wang, Liang, and Cheng, Ming-Ming. Global2local: Efficient structure search for video action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (Cited on page 29.)

Gatys, Leon A; Ecker, Alexander S, and Bethge, Matthias. A neural algorithm of artistic style. *arXiv*, 2015. (Cited on page 31.)

Gehring, Jonas; Auli, Michael; Grangier, David; Yarats, Denis, and Dauphin, Yann N. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*, 2017. (Cited on page 31.)

Ghoddoosian, Reza; Sayed, Saif, and Athitsos, Vassilis. Action duration prediction for segment-level alignment of weakly-labeled videos. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021. (Cited on page 72.)

Goodfellow, Ian J; Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on page 31.)

Goyal, Raghav; Ebrahimi Kahou, Samira; Michalski, Vincent; Materzynska, Joanna; Westphal, Susanne; Kim, Heuna; Haenel, Valentin; Fruend, Ingo; Yianilos, Peter; Mueller-Freitag, Moritz, and others, . The" something something" video database for learning and evaluating visual common sense. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. (Cited on page 26.)

Grauman, Kristen; Westbury, Andrew; Byrne, Eugene; Chavis, Zachary; Furnari, Antonino; Girdhar, Rohit; Hamburger, Jackson; Jiang, Hao; Liu, Miao; Liu, Xingyu; Martin, Miguel; Nagarajan, Tushar; Radosavovic, Ilija; Ramakrishnan, Santhosh Kumar; Ryan, Fiona; Sharma, Jayant; Wray, Michael; Xu, Mengmeng; Xu, Eric Zhongcong; Zhao, Chen; Bansal, Siddhant; Batra, Dhruv; Cartillier, Vincent; Crane, Sean; Do, Tien; Doulaty, Morrie; Erapalli, Akshay; Feichtenhofer, Christoph; Fragomeni, Adriano; Fu, Qichen; Fuegen, Christian; Gebreselasie, Abrham;

Gonzalez, Cristina; Hillis, James; Huang, Xuhua; Huang, Yifei; Jia, Wenqi; Khoo, Weslie; Kolar, Jachym; Kottur, Satwik; Kumar, Anurag; Landini, Federico; Li, Chao; Li, Yanghao; Li, Zhenqiang; Mangalam, Karttikeya; Modhugu, Raghava; Munro, Jonathan; Murrell, Tullie; Nishiyasu, Takumi; Price, Will; Puentes, Paola Ruiz; Ramazanova, Merey; Sari, Leda; Somasundaram, Kiran; Southerland, Audrey; Sugano, Yusuke; Tao, Ruijie; Vo, Minh; Wang, Yuchen; Wu, Xindi; Yagi, Takuma; Zhu, Yunyi; Arbelaez, Pablo; Crandall, David; Damen, Dima; Farinella, Giovanni Maria; Ghanem, Bernard; Ithapu, Vamsi Krishna; Jawahar, C. V.; Joo, Hanbyul; Kitani, Kris; Li, Haizhou; Newcombe, Richard; Oliva, Aude; Park, Hyun Soo; Rehg, James M.; Sato, Yoichi; Shi, Jianbo; Shou, Mike Zheng; Torralba, Antonio; Torresani, Lorenzo; Yan, Mingfei, and Malik, Jitendra. Ego4d: Around the World in 3,000 Hours of Egocentric Video. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2022. (Cited on page 126.)

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on pages 110, 111 and 113.)

Henaff, Mikael; Bruna, Joan, and LeCun, Yann. Deep convolutional networks on graph-structured data. *arXiv*, 2015. (Cited on page 102.)

Hoang, Cong Duy Vu; Haffari, Gholamreza, and Cohn, Trevor. Towards decoding as continuous optimization in neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017. (Cited on page 31.)

Hochreiter, Sepp and Schmidhuber, Juergen. Long short-term memory. *Neural computation*, 1997. (Cited on page 17.)

Huang, De-An; Fei-Fei, Li, and Niebles, Juan Carlos. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on pages 29, 34 and 52.)

Ishikawa, Yuchi; Kasai, Seito; Aoki, Yoshimitsu, and Kataoka, Hirokatsu. Alleviating oversegmentation errors by detecting action boundaries. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021. (Cited on pages 29, 58, 70, 96 and 97.)

Jaderberg, Max; Simonyan, Karen; Zisserman, Andrew, and others, . Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. (Cited on page 43.)

Johnson, Justin; Krishna, Ranjay; Stark, Michael; Li, Li-Jia; Shamma, David; Bernstein, Michael, and Fei-Fei, Li. Image retrieval using scene graphs. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on page 102.)

Johnson, Justin; Alahi, Alexandre, and Fei-Fei, Li. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on page 31.)

Karaman, Svebor; Seidenari, Lorenzo, and Del Bimbo, Alberto. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, 2014. (Cited on page 26.)

Kay, Will; Carreira, Joao; Simonyan, Karen; Zhang, Brian; Hillier, Chloe; Vijayanarasimhan, Sudheendra; Viola, Fabio; Green, Tim; Back, Trevor; Natsev, Paul, and others, . The kinetics human action video dataset. *arXiv*, 2017. (Cited on pages 1, 13, 26 and 46.)

Keskar, Nitish Shirish and Socher, Richard. Improving generalization performance by switching from adam to sgd, 2017. (Cited on page 19.)

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on pages 19 and 66.)

Kipf, Thomas N and Welling, Max. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017. (Cited on page 102.)

Kuehne, Hilde; Arslan, Ali, and Serre, Thomas. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2014. (Cited on pages viii, x, xiii, 4, 22, 31, 35, 45, 46, 49, 50, 59, 66, 69, 84, 88 and 126.)

Kuehne, Hilde; Gall, Juergen, and Serre, Thomas. An end-to-end generative framework for video segmentation and recognition. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016. (Cited on pages 2, 3, 12, 21, 27, 34, 45 and 53.)

Kuehne, Hilde; Richard, Alexander, and Gall, Juergen. Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding (CVIU)*, 2017. (Cited on pages 30, 53, 58 and 72.)

Kuehne, Hilde; Richard, Alexander, and Gall, Juergen. A Hybrid RNN-HMM approach for weakly supervised temporal action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020. (Cited on pages 27 and 53.)

Kumar, Sateesh; Haresh, Sanjay; Ahmed, Awais; Konin, Andrey; Zia, M Zeeshan, and Tran, Quoc-Huy. Unsupervised activity segmentation by joint representation learning and online clustering. *arXiv*, 2021. (Cited on pages 2 and 82.)

Kumra, Sulabh and Kanan, Christopher. Robotic grasp detection using deep convolutional neural networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. (Cited on page 102.)

Lambert, Ben. edit-distance. `https://github.com/belambert/edit-distance`, 2019. (Cited on pages 21 and 45.)

Le, Quoc and Mikolov, Tomas. Distributed representations of sentences and documents. In *International Conference on Machine Learning (ICML)*, 2014. (Cited on page 31.)

Lea, Colin; Reiter, Austin; Vidal, René, and Hager, Gregory D. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on pages 2 and 27.)

Lea, Colin; Flynn, Michael D; Vidal, Rene; Reiter, Austin, and Hager, Gregory D. Temporal convolutional networks for action segmentation and detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 2, 27, 34, 38, 53, 58 and 72.)

LeCun, Yann; Chopra, Sumit; Hadsell, Raia; Ranzato, M, and Huang, F. A tutorial on energy-based learning. *Predicting Structured Data*, 2006. (Cited on page 31.)

Lee, Pilhyeon and Byun, Hyeran. Learning action completeness from points for weakly-supervised temporal action localization. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. (Cited on pages 30, 96 and 97.)

Lei, Peng and Todorovic, Sinisa. Temporal deformable residual networks for action segmentation in videos. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 27.)

Li, Jun and Todorovic, Sinisa. Set-constrained viterbi for set-supervised action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on pages 30, 82 and 97.)

Li, Jun and Todorovic, Sinisa. Anchor-constrained viterbi for set-supervised action segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (Cited on page 30.)

Li, Jun; Lei, Peng, and Todorovic, Sinisa. Weakly supervised energy-based learning for action segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. (Cited on pages 2, 4, 22, 23, 30, 34, 35, 36, 44, 45, 46, 48, 49, 52, 58, 60, 61, 65, 66, 69, 70, 71, 72, 82, 96, 97, 126 and 127.)

Li, Shi-Jie; AbuFarha, Yazan; Liu, Yun; Cheng, Ming-Ming, and Gall, Juergen. MS-TCN++: Multi-Stage Temporal Convolutional Network for Action Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020. (Cited on pages 3, 13, 29, 38, 45, 46, 51, 53, 58, 60, 69, 70, 82, 85, 96 and 97.)

Li, Yanghao; Wu, Chao-Yuan; Fan, Haoqi; Mangalam, Karttikeya; Xiong, Bo; Malik, Jitendra, and Feichtenhofer, Christoph. Mvitv2: Improved multiscale vision transformers for classification and detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2022. (Cited on page 26.)

Li, Yikang; Ouyang, Wanli, and Wang, Xiaogang. ViP-CNN: A visual phrase reasoning convolutional neural network for visual relationship detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017a. (Cited on page 102.)

Li, Yikang; Ouyang, Wanli; Zhou, Bolei; Wang, Kun, and Wang, Xiaogang. Scene graph generation from objects, phrases and region captions. In *IEEE International Conference on Computer Vision (ICCV)*, 2017b. (Cited on page 102.)

Li, Yikang; Ouyang, Wanli; Zhou, Bolei; Shi, Jianping; Zhang, Chao, and Wang, Xiaogang. Factorizable net: An efficient subgraph-based framework for scene graph generation. In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on page 102.)

Li, Yujia; Tarlow, Daniel; Brockschmidt, Marc, and Zemel, Richard. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016. (Cited on pages 101, 102, 109 and 110.)

Li, Zhe; Abu Farha, Yazan, and Gall, Juergen. Temporal action segmentation from timestamp supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (Cited on pages x, xiv, 2, 3, 4, 21, 30, 83, 84, 85, 88, 89, 90, 91, 92, 94, 95, 96, 97 and 98.)

Liang, Xiaodan; Lee, Lisa, and Xing, Eric P. Deep variation-structured reinforcement learning for visual relationship and attribute detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 102.)

Lin, Ji; Gan, Chuang, and Han, Song. Tsm: Temporal shift module for efficient video understanding. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. (Cited on page 26.)

Lin, Tsung-Yi; Maire, Michael; Belongie, Serge; Hays, James; Perona, Pietro; Ramanan, Deva; Dollár, Piotr, and Zitnick, C Lawrence. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014. (Cited on pages 100, 101, 106, 109 and 123.)

Liu, Wei; Anguelov, Dragomir; Erhan, Dumitru; Szegedy, Christian; Reed, Scott; Fu, Cheng-Yang, and Berg, Alexander C. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on page 101.)

Liu, Ze; Lin, Yutong; Cao, Yue; Hu, Han; Wei, Yixuan; Zhang, Zheng; Lin, Stephen, and Guo, Baining. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. (Cited on page 26.)

Lu, Cewu; Krishna, Ranjay; Bernstein, Michael, and Fei-Fei, Li. Visual relationship detection with language priors. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on page 102.)

Ma, Fan; Zhu, Linchao; Yang, Yi; Zha, Shengxin; Kundu, Gourab; Feiszli, Matt, and Shou, Zheng. SF-Net: Single-frame supervision for temporal action localization. In *European Conference on Computer Vision (ECCV)*, 2020. (Cited on pages 3, 4, 21, 23, 30, 89, 96 and 97.)

Malinowski, Mateusz and Fritz, Mario. A multiworld approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. (Cited on page 101.)

Mao, Jiayuan; Xiao, Tete; Jiang, Yuning, and Cao, Zhimin. What can help pedestrian detection? In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 101.)

Moltisanti, Davide; Fidler, Sanja, and Damen, Dima. Action recognition from single timestamp supervision in untrimmed videos. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on pages 3, 4, 30, 63, 83, 87, 96 and 97.)

Newell, Alejandro and Deng, Jia. Pixels to graphs by associative embedding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. (Cited on page 102.)

Nguyen, Anh; Kanoulas, Dimitrios; Caldwell, Darwin G., and Tsagarakis, Nikos G. Detecting object affordances with convolutional neural networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016. (Cited on page 102.)

Nguyen, Anh; Kanoulas, Dimitrios; Caldwell, Darwin G, and Tsagarakis, Nikos G. Object-based affordances detection with convolutional neural networks and dense conditional random fields. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. (Cited on page 102.)

Nguyen, Duy-Kien and Okatani, Takayuki. Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 102.)

Paszke, Adam; Gross, Sam; Massa, Francisco; Lerer, Adam; Bradbury, James; Chanan, Gregory; Killeen, Trevor; Lin, Zeming; Gimelshein, Natalia; Antiga, Luca; Desmaison, Alban; Kopf, Andreas; Yang, Edward; DeVito, Zachary; Raison, Martin; Tejani, Alykhan; Chilamkurthy, Sasank; Steiner, Benoit; Fang, Lu; Bai, Junjie, and Chintala, Soumith. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. (Cited on page 66.)

Peng, Chao; Zhang, Xuangyu; Yu, Gang; Luo, Guiming, and Sun, Jian. Large kernel matters–Improve semantic segmentation by global convolutional network. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 101.)

Perronnin, Florent; Sánchez, Jorge, and Mensink, Thomas. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision (ECCV)*, 2010. (Cited on page 12.)

Peyre, Julia; Laptev, Ivan; Schmid, Cordelia, and Sivic, Josef. Weakly-supervised learning of visual relations. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. (Cited on page 102.)

Pirsiavash, Hamed and Ramanan, Deva. Parsing videos of actions with segmental grammars. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2014. (Cited on page 26.)

Poppe, Ronald. A survey on vision-based human action recognition. *Image and Vision Computing*, 2010. (Cited on page 1.)

Qi, Xiaojuan; Liao, Renjie; Jia, Jiaya; Fidler, Sanja, and Urtasun, Raquel. 3D graph neural networks for RGBD semantic segmentation. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 102.)

Qian, Rui; Meng, Tianjian; Gong, Boqing; Yang, Ming-Hsuan; Wang, Huisheng; Belongie, Serge, and Cui, Yin. Spatiotemporal contrastive video representation learning. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2021. (Cited on page 26.)

Radford, Alec; Narasimhan, Karthik; Salimans, Tim; Sutskever, Ilya, and others, . Improving language understanding by generative pre-training. 2018. (Cited on page 127.)

Redmon, Joseph and Farhadi, Ali. YOLO 9000: Better, faster, stronger. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages xv, 101, 113, 116 and 117.)

Ren, Mengye; Kiros, Ryan, and Zemel, Richard. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015a. (Cited on page 101.)

Ren, Shaoqing; He, Kaiming; Girshik, Ross, and Sun, Jian. Faster R-CNN: Towards real-time object detection with region proposal networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015b. (Cited on pages xv, 101, 111, 112, 113, 116 and 117.)

Richard, Alexander and Gall, Juergen. Temporal action detection using a statistical language model. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on page 44.)

Richard, Alexander; Kuehne, Hilde, and Gall, Juergen. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 3, 21, 23, 30, 34, 35, 45, 52, 53, 58 and 96.)

Richard, Alexander; Kuehne, Hilde, and Gall, Juergen. Action sets: Weakly supervised action segmentation without ordering constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018a. (Cited on pages 2, 30, 82 and 97.)

Richard, Alexander; Kuehne, Hilde; Iqbal, Ahsan, and Gall, Juergen. Neuralnetwork-viterbi: A framework for weakly supervised video learning. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018b. (Cited on pages 2, 4, 21, 23, 30, 31, 34, 35, 44, 45, 46, 48, 49, 52, 58, 60, 61, 65, 70, 82, 96, 97 and 126.)

Rohrbach, Marcus; Amin, Sikandar; Andriluka, Mykhaylo, and Schiele, Bernt. A database for fine grained activity detection of cooking activities. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2012. (Cited on pages 2 and 26.)

Russakovsky, Olga; Deng, Jia; Su, Hao; Krause, Jonathan; Satheesh, Sanjeev; Ma, Sean; Huang, Zhiheng; Karpathy, Andrej; Khosla, Aditya; Bernstein, Michael, and others, . Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015. (Cited on page 12.)

Sánchez, Jorge; Perronnin, Florent; Mensink, Thomas, and Verbeek, Jakob. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision (IJCV)*, 2013. (Cited on page 12.)

Sawatzky, Johann; Srikantha, Abhilash, and Gall, Juergen. Weakly supervised affordance detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 102.)

Sawatzky, Johann; Souri, Yaser; Grund, Christian, and Gall, Juergen. What Object Should I Use? - Task Driven Object Detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on page 7.)

Schuldt, C.; Laptev, I., and Caputo, B. Recognizing human actions: a local svm approach. In *International Conference on Pattern Recognition (ICPR)*, 2004. (Cited on pages 1 and 25.)

Sharif Razavian, Ali; Azizpour, Hossein; Sullivan, Josephine, and Carlsson, Stefan. Cnn features off-the-shelf: An astounding baseline for recognition. In *IEEE Computer Vision and Pattern Recognition Workshops*, 2014. (Cited on page 12.)

Simonyan, Karen and Zisserman, Andrew. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. (Cited on pages 1, 25, 26 and 34.)

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on page 113.)

Singh, Bharat; Marks, Tim K.; Jones, Michael; Tuzel, Oncel, and Shao, Ming. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on page 38.)

Singhania, Dipika; Rahaman, Rahul, and Yao, Angela. Coarse to fine multi-resolution temporal convolutional network. *arXiv*, 2021. (Cited on page 29.)

Souri, Yaser; Nouri, Erfan, and Adeli, Ehsan. Deep relative attributes. In *Asian Conference on Computer Vision (ACCV)*, 2016. (Cited on page 113.)

Souri, Yaser; Richard, Alexander; Minciullo, Luca, and Gall, Juergen. On Evaluating Weakly Supervised Action Segmentation Methods. In *arXiv*, 2020. (Cited on pages 4, 46 and 61.)

Souri, Yaser; Abu Farha, Yazan; Despinoy, Fabien; Francesca, Gianpiero, and Gall, Juergen. FIFA: Fast Inference Approximation for Action Segmentation. In *German Conference on Pattern Recognition (GCPR)*, 2021a. (Cited on pages 4, 7, 82 and 87.)

Souri, Yaser; Fayyaz, Mohsen; Minciullo, Luca; Francesca, Gianpiero, and Gall, Juergen. Fast Weakly Supervised Action Segmentation Using Mutual Consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2021b. (Cited on pages 2, 4, 7, 13, 58, 60, 61, 65 and 82.)

Spriggs, Ekaterina H; De La Torre, Fernando, and Hebert, Martial. Temporal segmentation and activity classification from first-person sensing. In *IEEE Computer Vision and Pattern Recognition Workshops*, 2009. (Cited on page 2.)

Statista, . Youtube: Hours of video uploaded every minute 2019, 2022. URL `https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/`. (Cited on page 1.)

Stein, Sebastian and McKenna, Stephen J. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013. (Cited on pages x, 23, 84 and 88.)

Sutskever, Ilya; Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. (Cited on page 31.)

Tang, Yansong; Ding, Dajun; Rao, Yongming; Zheng, Yu; Zhang, Danyang; Zhao, Lili; Lu, Jiwen, and Zhou, Jie. Coin: A large-scale dataset for comprehensive instructional video analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on pages 82 and 126.)

Tasty, . How to make the fluffiest pancakes recipe by tasty, 2019. URL `https://tasty.co/recipe/tasty-101-buttermilk-pancakes`. (Cited on pages vii and 2.)

Teney, Damien and van den Hengel, Anton. Visual question answering as a meta learning task. In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on page 101.)

Teney, Damien; Anderson, Peter; He, Xiaodong, and van den Hengel, Anton. Tips and tricks for visual question answering: Learnings from the 2017 challenge. In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on page 101.)

Tran, Du; Wang, Heng; Torresani, Lorenzo, and Feiszli, Matt. Video classification with channel-separated convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. (Cited on page 26.)

van den Oord, Aäron; Dieleman, Sander; Zen, Heiga; Simonyan, Karen; Vinyals, Oriol; Graves, Alex; Kalchbrenner, Nal; Senior, Andrew W., and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. In *SSW*, 2016. (Cited on page 38.)

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. (Cited on pages 26, 29 and 31.)

Vo, Nam N and Bobick, Aaron F. From stochastic grammar to bayes network: Probabilistic parsing of complex activity. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2014. (Cited on page 26.)

Wang, Heng and Schmid, Cordelia. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. (Cited on pages vii, 1, 11, 12 and 25.)

Wang, Jiahui; Wang, Zhenyou; Zhuang, Shanna, and Wang, Hui. Cross-enhancement transformer for action segmentation. *arXiv*, 2022. (Cited on page 29.)

Wang, Limin; Xiong, Yuanjun; Wang, Zhe; Qiao, Yu; Lin, Dahua; Tang, Xiaoou, and Van Gool, Luc. Temporal segment networks for action recognition in videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2019. (Cited on pages 1, 25 and 26.)

Wang, Xiaolong and Gupta, Abhinav. Videos as space-time region graphs. In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on page 102.)

Wang, Xiaolong; Girshick, Ross; Gupta, Abhinav, and He, Kaiming. Non-local neural networks. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on pages 26 and 34.)

Wang, Zhenzhi; Gao, Ziteng; Wang, Limin; Li, Zhifeng, and Wu, Gangshan. Boundary-aware cascade networks for temporal action segmentation. In *European Conference on Computer Vision (ECCV)*, 2020. (Cited on pages 29, 58, 60, 70, 96 and 97.)

Wei, Zijun; Wang, Boyu; Nguyen, Minh Hoai; Zhang, Jianming; Lin, Zhe; Shen, Xiaohui; Mech, Radomir, and Samaras, Dimitris. Sequence-to-segment networks for segment detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. (Cited on page 31.)

Williams, R. J. and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989. (Cited on page 40.)

Wu, Yuxin and He, Kaiming. Group normalization. In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on page 45.)

Xie, Saining; Girshick, Ross; DollÃ¡r, Piotr; Tu, Zhuowen, and He, Kaiming. Aggregated residual transformations for deep neural networks. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 101.)

Xu, Danfei; Zhu, Yuke; Choy, Christopher B., and Fei-Fei, Li. Scene graph generation by iterative message passing. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 102.)

Yan, Shen; Xiong, Xuehan; Arnab, Anurag; Lu, Zhichao; Zhang, Mi; Sun, Chen, and Schmid, Cordelia. Multiview transformers for video recognition. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2022. (Cited on pages 1 and 26.)

Yang, Jianwei; Lu, Jiasen; Batra, Dhruv, and Parikh, Devi. A faster pytorch implementation of faster R-CNN. *https://github.com/jwyang/faster-rcnn.pytorch*, 2017. (Cited on page 111.)

Yang, Jianwei; Lu, Jiasen; Lee, Stefan; Batra, Dhruv, and Parikh, Devi. Graph R-CNN for scene graph generation. In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on page 102.)

Yi, Fangqiu; Wen, Hongyu, and Jiang, Tingting. Asformer: Transformer for action segmentation. In *British Machine Vision Conference (BMVC)*, 2021. (Cited on pages 3, 13, 29, 96 and 97.)

Yu, Licheng; Park, Eunbyung; Berg, Alexander C., and Berg, Tamara L. Visual madlibs: Fill in the blank description generation and question answering. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. (Cited on page 101.)

Zellers, Rowan; Yatskar, Mark; Thomson, Sam, and Choi, Yejin. Neural motifs: Scene graph parsing with global context. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 102.)

Zhang, Hanwang; Kyaw, Zawlin; Chang, Shih-Fu, and Chua, Tat-Seng. Visual translation embedding network for visual relation detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017a. (Cited on page 102.)

Zhang, Hanwang; Kyaw, Zawlin; Yu, Jinyang, and Chang, Shih-Fu. PPR-FCN: Weakly supervised visual relation detection via parallel pairwise R-FCN. In *IEEE International Conference on Computer Vision (ICCV)*, 2017b. (Cited on page 102.)

Zhang, Yan; Hare, Jonathon, and Prugel-Bennett, Adam. Learning to count objects in natural images for visual question answering. In *International Conference on Learning Representations (ICLR)*, 2018. (Cited on page 102.)

Zhao, Yue; Xiong, Yuanjun; Wang, Limin; Wu, Zhirong; Tang, Xiaoou, and Lin, Dahua. Temporal action detection with structured segment networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. (Cited on page 34.)

Zhou, Bolei; Andonian, Alex; Oliva, Aude, and Torralba, Antonio. Temporal relational reasoning in videos. In *European Conference on Computer Vision (ECCV)*, 2018a. (Cited on page 25.)

Zhou, Peng; Ni, Bingbing; Geng, Cong; Hu, Jianguo, and Xu, Yi. Scale-transferrable object detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018b. (Cited on page 101.)

Zhu, Yi; Li, Xinyu; Liu, Chunhui; Zolfaghari, Mohammadreza; Xiong, Yuanjun; Wu, Chongruo; Zhang, Zhi; Tighe, Joseph; Manmatha, R, and Li, Mu. A comprehensive study of deep video action recognition. *arXiv*, 2020. (Cited on page 1.)

Zhu, Yixin; Zhao, Yibiao, and Chun Zhu, Song. Understanding tools: Task-oriented object modeling, learning and recognition. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on page 102.)

Zhuang, Bohan; Liu, Lingqiao; Shen, Chunhua, and Reid, Ian. Towards context-aware interaction recognition for visual relationship detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. (Cited on page 102.)

Zhukov, Dimitri; Alayrac, Jean-Baptiste; Cinbis, Ramazan Gokberk; Fouhey, David; Laptev, Ivan, and Sivic, Josef. Cross-task weakly supervised learning from instructional videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on pages 82 and 126.)