Guilherme de Azevedo Pereira Marques

A Cluster-Based Method for Action Segmentation Using Spatio-Temporal and Positional Encoded Embeddings

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Sérgio Colcher

Rio de Janeiro
March 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Guilherme de Azevedo Pereira Marques

A Cluster-Based Method for Action Segmentation Using Spatio-Temporal and Positional Encoded Embeddings

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee.

Prof. Sérgio Colcher
Advisor
Departamento de Informática – PUC-Rio

Prof. Hélio Côrtes Vieira Lopes
Departamento de Informática – PUC-Rio

Dr. Julio Cesar Duarte
IME

Rio de Janeiro, March 18th, 2022

Guilherme de Azevedo Pereira Marques

Bachelor's degree in Computer Science at Pontifical Catholic University of Rio de Janeiro (PUC-Rio) in 2018.

Bibliographic data

Marques, Guilherme de Azevedo Pereira

A Cluster-Based Method for Action Segmentation Using Spatio-Temporal and Positional Encoded Embeddings / Guilherme de Azevedo Pereira Marques; advisor: Sérgio Colcher. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2022.

v., 59 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Clusterização. 2. Segmentação de Ações. 3. Aprendizado profundo. 4. Codificação Posicional.
I. Colcher, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

Thanks to my advisor Prof. Sérgio Colcher for his guidance and support in this journey. Thanks to my family for the endless support. Thanks to my friends from TeleMídia Lab, for their friendship and support. To all colleagues, faculty and staff of the PUC Rio Department of Informatics for the fellowship, learning and support. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Marques, Guilherme de Azevedo Pereira; Colcher, Sérgio (Advisor). **A Cluster-Based Method for Action Segmentation Using Spatio-Temporal and Positional Encoded Embeddings**. Rio de Janeiro, 2022. 59p. Dissertação de mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The rise of video content as the main media for communication has been creating massive volumes of video data every second. The ability of understanding this huge quantities of data automatically has become increasingly important, therefore better *video understanding* methods are needed. A crucial task to overall video understanding is the recognition and localisation in time of different actions. To address this problem, *action segmentation* must be achieved. Action segmentation consists of temporally segmenting a video by labeling each frame with a specific action. In this work, we propose a novel action segmentation method that requires no prior video analysis and no annotated data. Our method involves extracting spatio-temporal features from videos using a pre-trained deep network. Data is then transformed using a positional encoder, and finally a clustering algorithm is applied where each cluster presumably corresponds to a different single and distinguishable action. In experiments, we show that our method produces competitive results on the *Breakfast* and *Inria Instructional Videos* dataset benchmarks.

Keywords

Clustering; Action Segmentation; Deep learning; Positional Encoding.

Resumo

Marques, Guilherme de Azevedo Pereira; Colcher, Sérgio. **Método baseado em agrupamento para a segmentação de ações utilizando embeddings espaço-temporais e com codificação posicional**. Rio de Janeiro, 2022. 59p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Vídeos se tornaram a principal mídia para a comunicação, com um volume massivo de dados criado a cada segundo. Conseguir entender essa quantidade de dados de forma automática se tornou importante e, por conseguinte, métodos de *video understanding* são cada vez mais necessários. Uma tarefa crucial para o entendimento de vídeos é a classificação e localização no tempo de diferentes ações. Para isso, a *segmentação de ações* precisa ser realizada. Segmentação de ações é a tarefa que consiste em segmentar temporalmente um vídeo, classificando cada quadro com alguma ação. Neste trabalho, é proposto um método de segmentação de ações que não requer análise prévia do vídeo e nenhum dado anotado. O método envolve a extração de *embeddings* espaço-temporais dos vídeos com redes de aprendizado profundo pré-treinadas, seguida por uma transformação realizada por um codificador posicional e pela aplicação de um algoritmo de grupamento em que cada *cluster* gerado corresponde a uma ação diferente. Os experimentos realizados demonstram que o método produz resultados competitivos nos conjuntos de dados *Breakfast* e *Inria Instructional Videos*.

Palavras-chave

Clusterização; Segmentação de Ações; Aprendizado profundo; Codificação Posicional.

Table of contents

List of figures

[2]https://anhreynolds.com/blogs/cnn.html

List of tables

List of abreviations

BLSTM	Bidirectional long short-term memory
CDFL	Constrained Discriminative Forward Loss
CNN	Convolutional Neural Network
FINCH	First Integer Neighbor Clustering Hierarchy
FLOPs	Floating Point Operations per second
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GT	Ground Truth
HMM	Hidden Markov Model
HOG3D	Histogram of Oriented Gradients 3D
I3D	Two-Stream Inflated 3D Convolutional Neural Network
iDT	Improved Dense Trajectories
IoT	Internet of Things
IoU	Intersection Over Union
LSTM	Long short-term memory
MLP	Multilayer Perceptron
MoF	Mean Over Frames
MS	Motion Squeeze
MuCon	Mutual Consistency Loss
NLP	Natural Language Processing
PE	Positional Encoding
PWC-Net	Pyramid Warping Cost Volume Network
ResNet	Residual Neural Network
RNN	Recurrent Neural Network
SS	Silhoutte Score
seq2seq	Sequence to Sequence
TSM	Temporal Shift Module
TW-FINCH	Temporally Weighted First Integer Neighbor Clustering Hierarchy
X3D	Expand 3D

1
Introduction

In recent years, video streaming platforms and services have been growing immensely. This growth has led to the production and consumption of a massive volume of video data. For instance, in 2019 more than one billion hours of YouTube videos were watched per day.[1] Therefore, to effectively extract information from this data, better *video understanding* methods are needed.

Video understanding methods aims to extract high-level semantic information from videos such as activities, actions, objects and scenes. In particular, a crucial task of video understanding is the segmentation in time of different actions that are present along a video (1, 2). In other words, this task aims to provide answer for the following questions: (i) What are the actions and (ii) when do the actions happen?

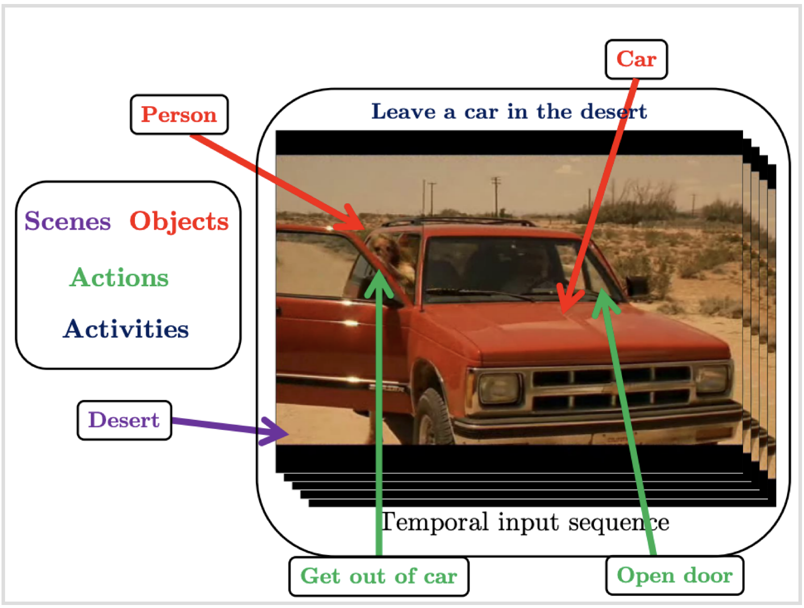

Figure 1.1: Video understanding goals and examples in the same color.[2]

To properly answer both questions we ought to define what is an action. A few works in the literature have proposed definitions for actions (3, 4, 5, 6, 7, 8), but the one we think is the most complete and in accordance to our goals is the following from Herath et al. (8): “Action is the most elementary human-

[1]https://kinsta.com/blog/youtube-stats/
[2]https://feichtenhofer.github.io/pubs/teaching/IVU$_Convolutional_Networks_and_Video_Representations.pdf$

surrounding interaction with a meaning". Where we can define the category (e.g., take cup, pour coffee) of an action by the meaning of its interaction. Moreover, the surrounding in this definition might be a specific object that enables a proper meaning for a given interaction. An important aspect in this definition in our interpretation is that when thinking of actions as a hierarchical breakdown of motions, one can derive meaning for an interaction as it better suits the context of its task or objective. For instance, in Figure 1.2 one could say that multiple actions are happening. First the throwing of the ball, followed by the striking of the ball by the other player, until it stops. As a matter of fact, there are even more sub-actions that form these actions described that it may make sense for someone to call it an action as we have defined it. However, this sequence of frames might also be regarded as a cricket shot, which in this case is a single action, as it is in the Kinetics dataset (9). Therefore, even with a definition, actions are context-dependent and subjective.

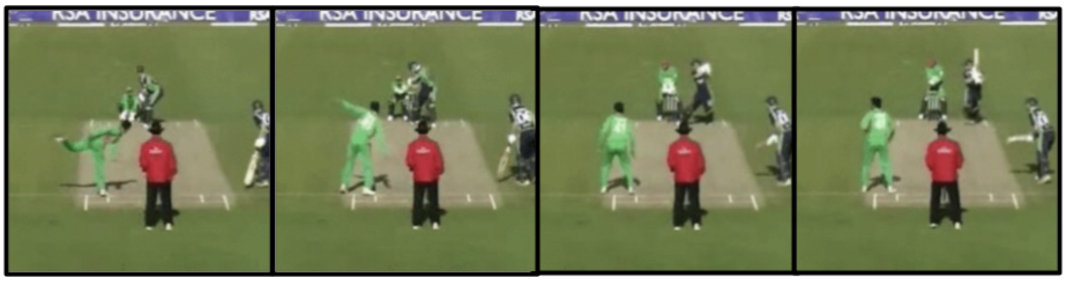

Figure 1.2: Sequence of actions or one action?

Initial efforts focused on answering (i), the so called *action recognition task*, which aims to classify trimmed videos with a single action (10, 11, 12, 13). Early methods were primarily based on the extraction of hand-crafted features (10), but more recently, deep learning methods became end-to-end learning models with automatic feature extraction (11, 12, 13), achieving state-of-the-art results.

Since in real-life situations videos are not always trimmed and may have multiple actions, the research community started to address the complex problem of *action segmentation*, which encompasses both questions (i) and (ii). Action segmentation consists of temporally segmenting a video by labeling each frame with a specific action. Such task might be of particular interest for applications of surveillance, human robots interaction, medical diagnosis, sports analytics, video recommendation and video summarisation (8, 4).

Although the performance achieved by fully supervised methods for this task is encouraging, whenever we propose a fully-supervised approach, there is a necessity for labelled data, which is a very challenging task. In a supervised setting, solutions may require frame-level annotations that are incredibly laborious and time-consuming. For instance, a short video of 5 seconds with a frame rate of 30 frames per second would have 150 frames to be labelled. Furthermore, actions might be subjective and there might not be a clear definition about the exact temporal span of an action (14, 15), which may be an indication that discovering action boundaries should be done directly from data.

For this reason, researchers started focusing on methods with less supervision, such as weakly-supervised (16, 17, 18, 19) and unsupervised methods (20, 21, 22). Most of these methods (17, 18, 19, 23, 24) are based on the idea of generating pseudo-labels that are used to train supervised models. However, the feature embeddings of these approaches are usually not effective due to the high level of uncertainty introduced by these pseudo-labels. Therefore, in this dissertation, we propose a *cluster-based method for the temporal segmentation of actions in videos.*

Our method takes advantage of spatio-temporal feature spaces learned from state-of-the-art action recognition models trained on large-scale datasets to generate highly discriminative visual and motion representation of videos. We break a video into short video clips and feed them to an action recognition model, generating for each video clip a feature descriptor $f \in \mathbb{R}^d$, where d is the descriptor's dimensionality. Thus for a video with t video clips it generates a video representation $V \in \mathbb{R}^{t \times d}$. We can then use V as the input for a clustering algorithm, which already produces good results. However, common clustering algorithms do not use the temporal position of these embeddings. Moreover, the video snippets embeddings we generate are only capturing visual and motion patterns within that temporal window and are ignoring long-range relations. To mitigate this problem, we propose the usage of a *positional encoding* module to inject positional information into the feature space and make it available for the clustering procedures.

In order to demonstrate the effectiveness of our proposal, we follow a quantitative methodology, in which we experiment our method with two different challenging benchmarks in the literature, reporting three different metrics. Moreover, our experiments help to shed light on how positional encoding affects action segmentation and how different video snippet lengths influence the quality of segmentations.

The remainder of this dissertation is structured as follows. In chapter 2, we discuss works related to ours such as action recognition, temporal action segmentation and positional encoding. In Chapter 3 we define the core of this dissertation: our *cluster-based method for action segmentation using spatio-temporal feature extraction and positional encoding.* It is composed of: *video snippets sampling*, *embeddings generation*, *positional encoding* and *clustering*. Chapter 4 is devoted to detail the datasets and metrics, and to present the results with our findings. Finally, Chapter 5 points out the overall contributions, conclusions and future work.

2
Related Work

In this chapter we make a review of the related works. In section 2.1, we present works with focus on the action recognition task, followed by section 2.2, where we concentrate on works about temporal action segmentation. Finally, in section 2.3, we make a review about positional encoding.

2.1
Action recognition

Action recognition on trimmed videos has been widely studied. Early methods were based on hand-crafted features such as the work proposed by Wang and Schmid (10), in which they present one of the most used feature descriptors, the *Improved Dense Trajectories* (IDT). However, following the impressive results of deep convolutional neural networks proposed by Krizhevsky *et al.* (25) at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (26), deep learning methods started to flourish for video classification. For instance, Karpathy et al. (27) proposed a 2D CNN, basically by repurposing an image classification network for video by extracting features of individual frames and then pooling predictions. The problem with this approach is that there is no temporal learning (e.g this model can not differentiate the opening and closing of a window). To overcome this limitation, 2D CNN combined with LSTM were introduced by Donahue et al. (28) and Ng et al.(29). There was also propositions such as Simonyan *et al.* (11) in which they applied 2D CNN with two-way streams for action recognition, where one stream aims to incorporate spatial information and the other temporal information with optical flow (30) as inputs. Optical flow or motion-estimation algorithms as defined by Horn and Brian G. Schunck (30) as the distribution of apparent velocities of movement of brightness pattern in an image, in other words, these algorithms compute estimates of the motion of image intensities over time in a video.

As conventional 2D CNNs are not very good in capturing temporal relations, the community started to explore 3D CNNs. Carreira *et al.* (12) proposed a two-stream architecture, but now with 3D ConvNets. By inflating 2D ConvNets into 3D, they were able to extract better spatio-temporal features. Besides that, they also bootstrapped the parameters of the 3D filters and layers with inflated 2D parameters from pre-trained ImageNet models. Moreover, they also applied the strategy of feeding RGB frames to one stream and optical flow to the other, which greatly improved results. Feichtenhofer *et al.* (13) also proposed a two-pathway model called SlowFast, where one pathway is designed to capture semantic and spatial information by operating at low frame rates and a second pathway operating at a high temporal rate designed to capture motion information. They proposed this different temporal rates for each path based on the intuition that spatial semantics often evolve slowly, and on the other hand, motion may evolve much faster. For instance, a person is still a person even if she is running or walking, meanwhile its motion information might be lost if we use a low temporal rate, since walking and running have evolve at different rates over time. By handling input video with different temporal rates, their method with two pathways, each with their own expertise on video modeling, achieved state-of-the-art results with no extra data. Another important contribution is that they do not have to compute optical flow from videos, which is computationally expensive, and all features extracted are learned end-to-end from the raw data.

In contrast to 2D CNNs, 3D CNNs are good in modelling temporal relations, but their computational cost is large. The following works focused on tackling accuracy and computation trade-off, a crucial consideration for on-device applications with a constrained computational budget, such as mobile cameras and some IoT devices. Li *et al.* (31) proposed a novel module called *Temporal Shift Module (TSM)* that is adds no computational cost on top of a 2D convolution, but is able to capture and model temporal relations. TSM works by shifting the feature map along the temporal dimension. Authors showed that their approach significantly improved pure 2D CNNs baselines. Moreover, at the time, they reached state-of-the-art results, outperforming 3D convolution based methods on both Something-Something V1&V2 (32).

A similar proposal to extend 2D CNNs capabilities by Kwon *et al.* (33) introduced a generic learnable motion feature extractor that can be inserted in any neural network called *MotionSqueeze (MS)*. The MS module, given two adjacent frames learns to extract motion features in three steps: (i) correlation computation, (ii) displacement estimation, and (iii) feature transformation. Another contribution from their work is the *MSNet* which is a network architecture with the TSM ResNet as its backbone combined with MS modules, at the time, with a marginal increase of computation, they achieved state-of-the-art results in the Something-Something V1&V2 (32) datasets. Feichtenhofer (34) on the other hand proposed a novel network called

Expand 3D (X3D). Their core idea is to progressively "expand" a lightweight 2D image architecture into an architecture with spatio-temporal capabilities by expanding different axes, while searching for the best computation/accuracy trade-off. They propose a neural architecture search method that searches the hyper-parameter space defined by the candidate axes of temporal duration, frame rate, spatial resolution, network width, bottleneck width, and depth. X3D reaches competitive performance in the Kinetics-400 (9), Kinetics-600 (35) and Charades (36) datasets while needing less computational resources. A recent proposal by Kondratyuk *et al.* (37) called *Mobile Video Networks (MoViNets)* in similar spirit to X3D introduces a neural architecture search method for MoViNets, creating a vast search space that encompass a family of versatile networks. Nevertheless, one of the search dimensions is the number of input frames that increases linearly with the memory footprint, which hamper handling long videos on mobile devices. To tackle this problem they introduced a Stream Buffer that reduces this consumption increase from linear to constant in video length, but with an accuracy drop of 1% on the dataset Kinetics-600. So to mitigate this drop, their final contribution is Temporal Ensembles, where they introduced a simple ensembling strategy that maintains the same FLOPs as a single model, but achieving higher accuracy. MoViNets achieved remarkable accuracy with significantly less FLOPs and lower memory usage, reaching better computational and accuracy trade-off compared to X3D in all main benchmarks.

The proposed method in this dissertation clusters video snippets, and these snippets are very similar to the trimmed videos that action recognition models are trained for. Therefore we utilized models of this task for feature extraction. For this end we picked the two 3D CNNs models, I3D (12) and SlowFast (13), introduced here due to their remarkable results in the main benchmarks and since this type of architectures are more mature and commonly used (19) for feature extraction. However, our method works as a framework for action segmentation, and other models may be utilized for extracting features depending on your requirements. For instance, an IoT scenario with edge devices might need a model that is computationally efficient, and works such as (31, 32, 33, 34, 37) could be used.

2.2 Temporal Action Segmentation

Temporal action segmentation has been getting increasing attention in recent years. Fully supervised methods have achieved encouraging results, but at the cost of widely annotated data that is prohibitive for many realcase scenarios. Therefore, our focus is on weakly supervised and unsupervised setups.

Methods for weakly supervised action segmentation use the actions ordering, termed as *transcripts*, and the video-level activity as weak supervision. One of the first works from Bojanowski *et al.* (16) proposed a method where they use the information on the ordering of actions to train a discriminative clustering approach for action alignment to perform segmentation. Koller and Ney (38) proposed an iterative learning setup with a CNN-BLSTM embedded with a HMM. Their iterative training consists of the CNN-BLSTM predicting actions for each frame and the HMM aligning these predictions with the transcripts, where the output of each is the input of the other. Similarly to Koller and Ney (38), Richard et al. (39) proposed a pipeline also composed of a RNN that classifies frames with actions and a HMM that aligns them based on the transcript. The main difference between their works is that Richard et al. (39) used iDT and Fisher Vectors features as input for their RNN, while Koller and Ney (38) used the video frames directly as input. Moreover, Richard et al. (39) also proposed to model each action as a sequential combination of subactions treated as latent variables that are learned by the model, they showed that this strategy of fine to coarse learning is beneficial. The last two works introduced are based on a similar strategy, which in order to learn action segmentation from weakly annotated data, they use an iterative training strategy of pseudo ground-truth generation and model learning that is not ideal. This approach has the following drawbacks, first they are sensitive to the initialization of the pseudo ground-truth (39). Secondly, since pseudo-labels are noisy the training tends to be unstable and oscillate between iterations (23), and finally, both need to process the entire dataset in each step, preventing incremental learning. So another work from Richard *et al.* (23) proposed a new learning algorithm that draws a random sequence of frames, which then is forwarded through a neural network (it can be any architecture) and a Viterbi decoding is used to predict the final segmentation. With a novel Viterbi-based loss that directly leverages transcripts they allowed online and incremental learning, while also discarding the need for pseudo-labels. A recent proposal by Li *et al.* (18) also used the approach of a RNN combined with a HMM, but to mitigate the problem of training with pseudo-labels, they developed an efficient recursive estimation of energy score of all valid and invalid paths in the HMM, where each path is a candidate segmentation. Then with the accumulated energy difference they compute a novel loss called constrained discriminative forward loss (CDFL), hence minimizing the total energy of valid paths in the segmentation graph, instead of optimizing for label probabilities of each frame.

Kuehne *et al.* (40) contributed with a method very similar to Richard et al. (39) where an RNN is used to recognize and classify small temporal clips, this way learning local temporal information. By classifying these small clips, they model complex action classes with subactions. These subactions allow their model to learn fine-grained movements but still capture mid and long-range temporal information frames. Moreover, to enhance even further the quality of segmentations, they proposed the usage of a length prior to act as a regularizer for the states of the HMM. Souri et al. (19) pointed out that during inference, most approaches that rely on segmentation through alignment with HMM or the Viterbi algorithm have to iterate over all of the training transcripts and choose the transcript that best aligns with an unseen video. This characteristic results in two major drawbacks. First, since it has to perform alignment over all training transcripts with the unseen video, the inference is really slow. Moreover, this approaches do not generalize to transcripts that are not seen during training. So to tackle these issues, Souri *et al.* (19), proposed a two-branch network where one branch predicts the transcripts and lengths of each action, and the other branch predicts the frame-wise class probability. Since both branches are predicting redundant representations of action segmentation, they exploit this characteristic by proposing a new mutual consistency (MuCon) loss that makes both representations consistent with each other.

To avoid the necessity of any labeled data, unsupervised methods have started to get more attention. Sener and Yao (24) proposed a linear model that maps the visual features of every frame into a latent embedding space. This embedding space is learned through an iterative process of clustering frames of same actions into anchor points. This anchor points represent actions present in the videos, however this is an unsupervised learning process, therefore they proposed a generative model to learn the temporal structure of actions in videos and the outputs of this temporal model are used to update the visual model. Kukleva et al. (41) proposed a pipeline that first learns temporal embedded frame-wise features, them cluster all features into K clusters (or actions). For each cluster they calculate the mean over time of each frame to create a cluster ordering. Finally by applying a frame-wise decoding with the Viterbi Algorithm they temporally segment each video by maximizing the probability of a sequence of frames that follows the cluster ordering, this way helping to ensure consistency among the labels for each frame. VidalMatal et al. (21) presented an unsupervised approach that segments actions based on visual-temporal embeddings. To achieve this, they do a two stage training method. In the first stage they train two disjoint models, one for visual and another for temporal embeddings, then in the second stage, they train a joint visual-temporal model that learns a feature space that accounts for visual and temporal appearance. With this joint visual-temporal embeddings they cluster all frames with a Gaussian Mixture Model and use a Viterbi decoding to generate the final segmentation of each video. Li et al.(20) presented a new self-supervised learning approach to generate embeddings that, instead of focusing on capturing the temporal structure at the frame level or video level, they focus on learning the temporal structure at the action level. They train this model with a HMM in an iterative fashion, where the HMM generates the action segmentations followed by a Viterbi decoding and this outputs are used to update the action embeddings which are then fed to the HMM. To train this pipeline they used a expectation-maximization algorithm. Sarfraz et al.(22) proposed action segmentation as a grouping problem. With this in mind they developed a new hierarchical clustering algorithm designed for action segmentation. This clustering algorithm works by creating a spatio-temporal graph representation of a video of nearest neighbours based on their visual features proximity and position in time. With this graph they recursively cluster each sample to generate the segmentation. Their approach achieved state-of-the-art results in all main benchmarks.

Almost all methods, weakly supervised or unsupervised, are based on the usage of pseudo-labels, which as explained before have some major drawbacks. Moreover, many of them perform segmentation through an alignment strategy with the Viterbi algorithm or HMM, that also introduces important limitations. Additionally, most methods require training in order to generate meaningful embeddings, either through the iterative training with pseudo-labels or conventional training setups. In comparison, our method utilizes features of powerful pre-trained action recognition models that are highly discriminative and are able to capture short-range spation-temporal features. Furthermore, instead of embedding these features in a new latent space to encode long-range temporal positioning information and perform training in a new objective, we utilize a common strategy in NLP tasks (42, 43, 44) to encode positioning in sequential data, the so called positional encoding that requires no training at all. Also, in line with Sarfraz el al. (22) we see action segmentation purely as a grouping problem, rather than, for example, a task that requires modeling actions as states with a HMM. A brief review of these differences can bee seen in 2.1 below.

Table 2.1: Comparison on temporal action segmentation methods

Method	Supervision	Pseudo-labels?	Actions as states?	Training
Bojanowski et al. (16)	Weakly	Yes	Yes	Yes
Koller and Ney (38)	Weakly	Yes	Yes	Yes
Richard et al. (39)	Weakly	Yes	Yes	Yes
Richard et al. (23)	Weakly	No	Yes	Yes
Li et al. (18)	Weakly	Yes	Yes	Yes
Kuehne et al. (40)	Weakly	Yes	Yes	Yes
Souri et al. (19)	Weakly	No	No	Yes
Sener and Yao (24)	Unsupervised	Yes	Yes	Yes
Kukleva (41)	Unsupervised	No	Yes	Yes
VidalMata et al. (41)	Unsupervised	No	Yes	Yes
Li et al. (20)	Unsupervised	Yes	Yes	Yes
Sarfraz et al. (22)	Unsupervised	No	No	No
Ours	Unsupervised	No	No	No

2.3 Positional Encoding

The concept of using positional encoding to inject positional information was first proposed in the context of NLP tasks. Long short-term memory (45) and gated recurrent neural networks (46) are common approaches for sequence modeling. However, LSTM and GRU are very inefficient in modern GPU computation due to their sequential nature that inhibits parallelization. To overcome this problem works such as convolutional seq2seq (42) used convolutional neural networks to enable the usage of GPUs. Nevertheless, convolutions, outside of their kernel size, are position-insensitive and for this reason they proposed the first usage of a positional encoding as a learnable position embedding layer. Soon after them, in 2017 (43) proposed the Transformer model architecture based on the self-attention module that employs no usage of positional information. To overcome that they used a deterministic positional encoding based on sinusoidal functions, the one we used in this dissertation. Finally, there is also a relative positional encoding proposed by (44) that works directly with the mechanisms of the self-attention module.

3
A Cluster-Based Method For Temporal Action Segmentation

This chapter describes the core of this dissertation, which is a method for *Temporal Action Segmentation* based on clustering.

We consider a video as a sequence $V = \{f_i\}_{i=1}^{N}$ of N samples, where each sample f_i is a video snippet with the same number of frames. Since our method relies on clustering, we have to generate discriminative descriptors for each sample. In order to do this, we extract spatio-temporal embeddings for each sample $f_i \in V$, transforming each video snippet into a one-dimensional vector representation that captures spatial and motion information. This produces a matrix $M_{N\times d_{model}}$ where d_{model} is the embeddings's dimensionality.

Since this clustering approach lacks any frame ordering information, following the approach proposed by Vaswani *et al.* (43), we use a positional encoding technique to inject positional information into the video's embedding. Then, we cluster the embeddings with either FINCH or, when using Kmeans, applying the cluster-based heuristic proposed by Mendes *et al.* (47) to find the optimal number of clusters in which each cluster is expected to represent a different action. Figure 3.1 illustrates the method.

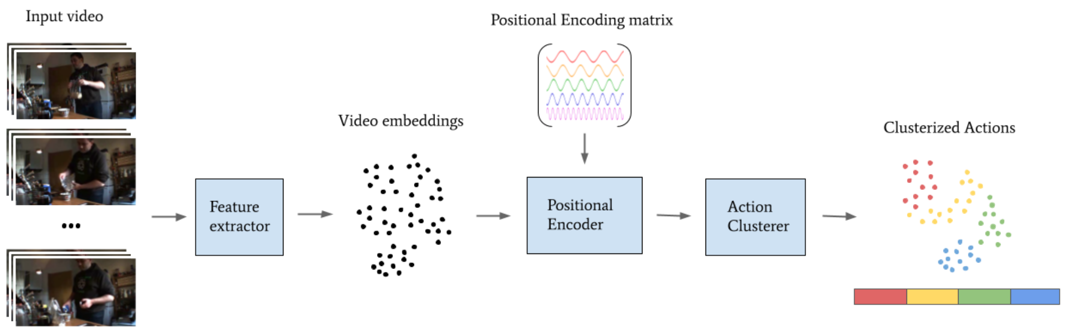

Figure 3.1: Temporal action segmentation clustering process.

In the remainder of this section, we detail each step involved in our proposal.

3.1 Video snippets sampling

The first step in our method is to break a video into snippets as illustrated in figure 3.2. In order to do this, first we have to define the temporal window

t we are using to generate each video snippet. The temporal window defines how many seconds this video snippet is going to last, and therefore, depending on the video's frame rate, how many frames each video clip is going to have.

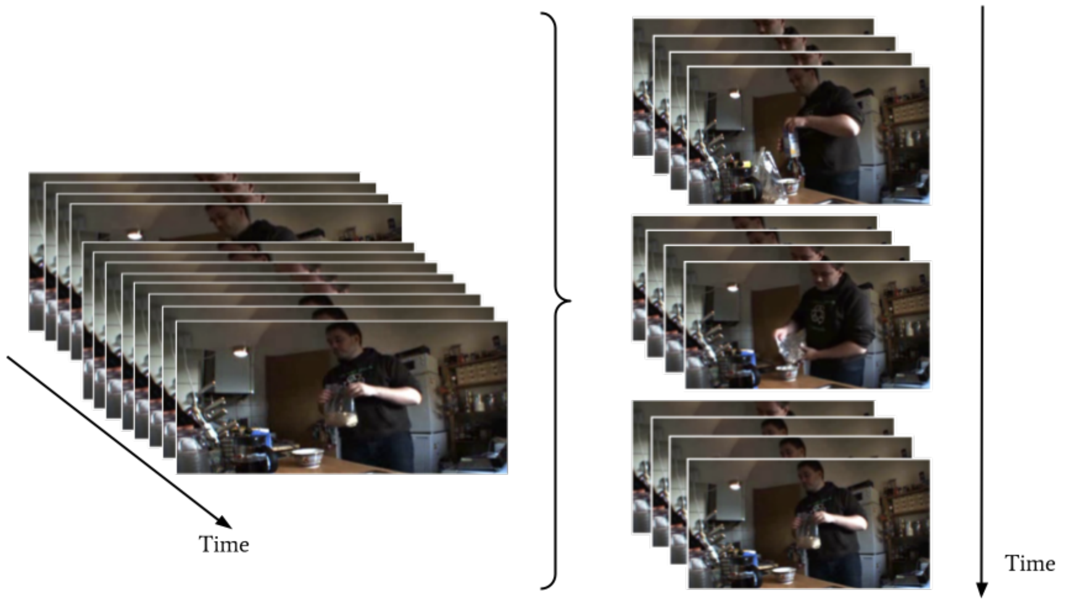

Figure 3.2: In the left we have the whole video. The first four frames are higher than the rest, indicating a four frames length per video snippet. In the right is the original video splitted into 3 video snippets each with 4 frames.

A video V with T seconds can be represented as $V = [v_1, \cdots, v_k]$ where v_i is a video clip with t seconds and $k = \left\lceil \frac{T}{t} \right\rceil$. We can also define the length of v_i as the number of frames we want it to have as n, this way $k = \left\lceil \frac{N}{n} \right\rceil$, where N is the total number of frames in V. In theory, each video snippet length n (measured in frames) should be constrained by $2 \leq n \leq N$. We limit the minimum length for a video snippet to be 2 frames since a feature descriptor of a single frame would actually be generating a representation with only spatial information and no temporal information at all.

The length of video snippets is also an important hyperparameter to be chosen in our method, since it has important implications related to computational requirements and to the quality of the embeddings. Regarding the computational requirements, this step has some important consequences. Firstly, the memory footprint, m, for each video snippet increases linearly with the temporal window size, $m = n \times H \times W \times C$, which might be prohibitive for systems without large memories, specially in the feature extraction process, since it is necessary to have the full clip in memory. On the other hand, an increase in the value of n, represents a linear decrease in the value of k, which might be helpful for the rest of the steps in our method's pipeline. For instance, the generation of the positional encoding matrix has a complexity of $\mathcal{O}(kd)$

where d is the dimensionality of each feature vector, which would result in a linear decrease in computational complexity and memory in this step. The value of n also greatly influences the quality of our method, as n increases, more information is packed in a single video snippet. However, to properly extract this information, the feature extractor's algorithm has to be able to capture long-range dependencies, otherwise the process of feature extraction is only more computationally intensive. Also, a greater n means we have a greater probability of capturing more than one action per video snippet, and since we are clustering each video snippet, this might cause an under-segmentation and the lost of fine-grained actions.

3.2 Video Embeddings Extraction

With video snippets in hand, we have to define the feature-descriptor generation. We look for a function f that maps each video snippet in $V = [v_1, \cdots, v_k]$ to a point in a feature space $\mathbb{R}^d$, that is

$$f : \mathbb{R}^{n \times k \times H \times W \times C} \to \mathbb{R}^{k \times d}$$

where $v_i \in \mathbb{R}^{n \times H \times W \times C}$, n is the number of frames per video snippet, H and W are the height and width of each frame, C is the number of channels and d is the dimensionality of the feature space.

To extract spatio-temporal features there are two main options in the literature: (i) hand-crafted features such as Improved Dense Trajectories (IDT) (48) and HOG3D (49), or, (ii) deep learning models. Recently, the state-of-the-art results for the main benchmarks of tasks that require spatio-temporal features are based on deep learning models. So to extract our features, we used action recognition models, since these expect inputs very similar to our video snippets. Action recognition models are trained on datasets with short duration, trimmed videos with a single action, which in theory, are exactly the same as our video snippets. So for the function f, we used two different action recognition models to extract spatio-temporal features: the I3D model (12) and the SlowFast model (13).

The I3D model has a two-pathway architecture, where the inputs for one pathway are the RGB frames of the input video and for the other pathway are the optical flow (30) frames extracted from the input video. Extracting optical flow is well known to have a high computational cost. The algorithm used in the I3D paper is the TV-L1 (50), that requires heavy computation. For this reason, we used the *PWC-Net* (pyramid, warping, and cost volume network) (51) in our method, which is a time efficient and accurate CNN for optical flow extraction. Moreover, the I3D model generates embeddings in the $\mathbb{R}^{2048}$ feature space, so for a video with k video snippets we would generate a video embedding representation $E \in \mathbb{R}^{k\times 2048}$. We evaluate our method with video snippets embeddings from the I3D model with the following number of frames, $n \in \{10, 16, 24, 32, 64\}$, with $n = 10$ being the shortest possible video snippet input for this architecture.

The SlowFast model is a generic two-pathway architecture, each pathway focused on a different aspect of video modeling. The *slow pathway* is focused on the spatial domain, so its inputs are low frame rate RGB frames of a video snippet, while the *fast pathway* is responsible for capturing motion and temporal information from the high frame rate RGB frames. The model can be instantiated with different backbones. We used the 3D ResNet-50 8x8 model, where the number of frames for each pathway input is 4 and 32 for the slow and fast pathways respectively. The embeddings generated by this model are in the $\mathbb{R}^{2304}$ feature space; therefore, a video embedding representation with the SlowFast can be described as $E \in \mathbb{R}^{k\times 2304}$, and we varied the video snippets length by using $n \in \{32, 40, 48, 64, 72, 128\}$, with $n = 32$ being the shortest possible video snippet input for this architecture.

For both models the number of frames we experimented are based on the minimum number of frames each model is able to receive and the number of frames used in test time for each model in the original papers, which is 64 frames. So the idea here is to progress from the shortest possible video snippet to the length of the video snippet in test time with a fine-grained granularity of frames, so we chose to increase 8 frames in order to get to 64 frames (with exception from 10 to 16 in the I3D case).

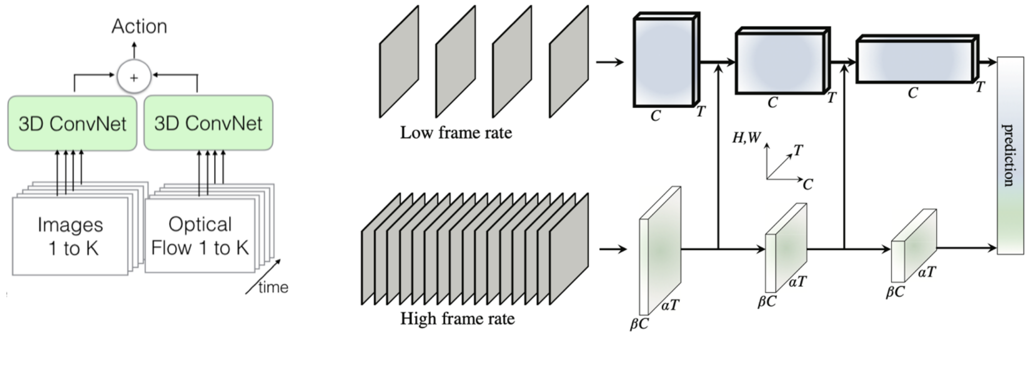

Figure 3.3: Diagrams of both architectures, in the left the I3D and in the right the Slowfast.

Both models are very good when it comes to extracting spatio-temporal features, although they have significant differences. First, even though both have a two-pathway architecture, their strategy to learn video representations are quite different. I3D and SlowFast rely on the fact that 3D CNNs can directly learn temporal patterns, and on top of that, both have a specialized strategy to go further when extracting temporal information. The I3D model have two identical streams, but one of them, as mentioned earlier, utilizes optical flow as the input to one of the streams. They argue that 3D CNNs are pure feed-forward networks and that optical flow algorithms are recurrent, and to feed a flow representation to the model is beneficial. SlowFast, on the other hand, utilizes two different streams to process data and extract specialized information. The SlowFast spatial stream receives low frame rate inputs, in other words, its inputs have a large temporal stride because their focus here is not to capture displacement and motion, but static features. Moreover, in order to focus on visual appearance, this network utilizes a 2D convolutional kernel in most layers because they argue that using temporal convolution in all layers is detrimental to accuracy. Inversely, the temporal stream has a high frame rate, this high temporal resolution is processed by a stream that has no temporal down-sampling until the last layers, this way maintaining temporal fidelity through almost all the network. Furthermore, the fast pathway has a lower channel capacity which translates into a lower ability to model spatial features. Their experiments shows that this is a desired trade-off, to weaken the spatial modeling ability while strengthening its temporal counterpart and having a lower computational requirement. Another important difference is in their backbones, the I3D utilizes inflated (from 2D to 3D) inception modules and the SlowFast a 3D ResNet-50. Finally, their training strategy and prediction differs greatly. The I3D trains both streams separately and average their predictions. While the SlowFast trains both streams simultaneously and has lateral connections between streams, so no pathway is unaware of the representation learned by the other. At the end of the network both feature vectors are pooled and then concatenated to be the input of a fully-connected classifier to output the final predictions. The SlowFast network has a more sophisticated strategy and better results in all benchmarks when compared to the I3D. However, the I3D was one of the first 3D CNN with incredible results in most benchmarks and was important to the community get to the point of developing the SlowFast model. Even though the SlowFast architecture produces better models for action recognition, features from the I3D might be more discriminative depending on the snippets size, hence we tested both.

Finally, when feeding the video snippets to these models is also necessary to define the frames sampling strategy. Frames sampling strategy determines how we are picking frames for each video. For instance, if we have a video snippet composed of 32 frames, we could use all 32 frames to generate its feature vector and that would be a dense temporal sampling. Here we used this exact strategy, however as pointed out by Wang *et al.* (52), consecutive frames are highly redundant and depending on the task may be unnecessary. For this reason they proposed a sparse sampling strategy that might be beneficial for our case, but we leave it as future work.

3.3 Positional Encoding

In our method, we aim to segment a video by clustering video snippets feature vectors. However, common clustering algorithms make no use of positional information of the frames within the flow, which is important for temporally segmenting a video. This positional information is crucial to establish the sub-actions that form more complex actions, which of course are close to each other in the time dimension. To address this problem, most works utilize a strategy where feature vectors of frames (or directly the frames) are used to train a new model that learns the timestamp or some temporal related objective function. This function generates an embedded space that captures the temporal characteristics for each frame. For instance, VidalMata et al.(21) trained a temporal embedding model as a Multilayer Perceptron with the learning goal of predicting the relative timestamp t of a given frame.

We have chosen a simpler way that requires no training to store temporal information in the video embeddings. We opted to use the *positional encoding method*, proposed by Vaswani et al.(43), in which we produce an encoding matrix, $PE_{k \times d_{model}}$, where k is the number of video snippets feature vectors and d_{model} is the dimensionality of each video feature descriptor. Likewise, we use the positional encoding with sinusoidal functions:

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{model}}})$$
$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i+1/d_{model}}})$$

Where each position in the PE matrix corresponds to either a sine or cosine function. Positions with even columns are sine functions, and odd columns with a cosine function. Both sine and cosine are parameterized by the position's row and column, and also by the feature's descriptor dimensionality.

After constructing the PE, we sum it to the video representation in the feature space resulting in a representation with time-related positional information. In other words, we are equipping each video snippet feature descriptor with information about its position in that video timeline.

A possible drawback of this approach is that video snippets of same actions occurring in distant moments in the timeline might be assigned to different clusters due to the increasing difference of their positional encodings, predicting more clusters (actions) than necessary — causing the so called over-segmentation problem. Therefore, datasets with actions that occur in different places in the timeline might have deterioration in accuracy. In fact, we did observe this behavior in our experiments, and we will detail it in section 4.

3.4 Action Clustering

For this step, following (22) we considered two representative clustering algorithms: (i) Kmeans (53) representing centroid-based methods and one of the most used unsupervised learning algorithms, and (ii) FINCH (54) a state-of-the-art hierarchical agglomerative clustering method.

Combined with Kmeans, we apply the method proposed by Mendes *et al.* (see Algorithm 1), that uses the *Silhouette Score* to find the optimal number of clusters. The *Silhouette Score* corresponds to the mean of the *Silhouette Coefficient* of all samples, which is calculated by the following equation:

$$S = \frac{b-a}{max(a,b)} \tag{3-1}$$

where a is the mean distance from a sample to all other samples in the same cluster, and b is the mean distance from a sample to all other samples in the closest cluster to that sample.

This way, the best value is 1, and the worst is -1. Values close to 0 indicate overlapping clusters, whereas negative values usually indicate that a sample has been assigned to the wrong cluster since a different cluster is more similar. In this strategy, we try to increase the number of clusters until the maximum Silhouette Score does not increase for more than t times in a row or until it reaches the maximum number of clusters, which is a hyperparameter to be defined. When it stops, we return the clustering configuration with the highest Silhouette Score, where each cluster corresponds to a different action.

While with Kmeans, we used the *Silhouette Score* to propose an automatic way of finding the optimal number of clusters, FINCH does not need that. Most clustering methods are based on the direct distance between samples. However, in high dimensional spaces, distances are less informative. For this reason, Sarfraz et al.(54) proposed FINCH, a clustering method based on the intuition that semantic relations are indirect relations that are not sensitive for high dimensional spaces. They observed that the first neighbor of each data point is sufficient to discover linking chains in the data. So given the indices

Algorithm 1 Iteratively finding the best clustering configuration for unknown number of clusters.

```
1: procedure BlindClustering($e, t, \omega$)
2:     $n_K \leftarrow 1$
3:     $s_{max} \leftarrow -1$
4:     $t_{cur} \leftarrow 0$
5:     while $t_{cur} \leq t$ & $n_K < |e|$ do
6:         $n_K \leftarrow n_K + 1$
7:         $K_{cur} \leftarrow Clustering(e, n_K)$
8:         $s \leftarrow SilhouetteScore(K_{cur})$
9:         if $s < s_{max}$ then
10:            $t_{cur} \leftarrow t_{cur} + 1$
11:        else
12:            $K \leftarrow K_{cur}$
13:            $t_{cur} \leftarrow 0$
14:            if $s > s_{max}$ then
15:                $s_{max} \leftarrow s$
16:            end if
17:        end if
18:    end while
19:    return $K$
20: end procedure
```

of the first neighbor of each data point, they define the first neighbor graph as an adjacency link matrix with equation 3-2, where k_i^1 is the first neighbor of point i.

$$A(i,j) = \begin{cases} 1, & \text{if } j = k_i^1 \text{ or } k_j^1 = i \text{ or } k_i^1 = k_j^1 \\ 0, & \text{otherwise} \end{cases} \tag{3-2}$$

So with a recursive approach, they generate a first neighbors adjacency matrix with equation 3-2, representing the clusters for the first partition. For each new partition, they use mean vectors of the previous partitions to generate the next one, applying again equation 3-2. At the end of this process, they generate a hierarchical structure where each successive partition is a superset of all previous partitions. This algorithm achieved state-of-the-art results in the main clustering benchmarks with the complexity of $\mathcal{O}(n \log n)$.

By the end of this process, we expect to have the temporal segmentation of the actions present in a video. The following chapter describe the experiments we investigate in this dissertation. We dive in the datasets and setup used, the evaluation method and metrics, the experiments conducted, the results and findings.

4 Experimentation

In this chapter, we describe the experiments to evaluate the effectiveness of our method by comparing it with state-of-the-art models in two benchmark datasets. Additionally, we compare the two models for spatio-temporal features extraction: SlowFast (13) and I3D (12). Moreover, we also present the experiments varying the temporal window size to understand how the duration of video snippets affects the segmentation quality. Since our proposal consists of an unsupervised method, we restrict the model list in both benchmarks only to models of the unsupervised or weakly supervised type for a fair comparison.

The remainder of this chapter is structured as follows. Section 4.1 describes the two datasets used in the experiments. Section 4.2 presents our experimental setup. Our evaluation protocol is defined in section 4.3, and the experiments carried out are detailed in section 4.4. Results are registered in section 4.5 followed by section 4.6 inn which we discuss these results and comment on our empirical findings.

4.1 Datasets

We evaluate our method using two of the most well-known benchmark datasets in the literature: the Breakfast Actions Dataset (55), and Inria Instructional Videos (56).

The Breakfast dataset is a large-scale dataset with 1,712 videos of common cooking activities performed in a breakfast situation. The videos comprise ten different complex cooking activities, such as *prepare scrambled eggs* or *prepare coffee*, resulting in a total of 48 different actions. Each video has on average six actions, the videos duration can vary significantly, ranging from 30 seconds to a few minutes, and, the whole dataset contains only 6% of background frames. Figure 4.1 illustrates examples from this dataset.

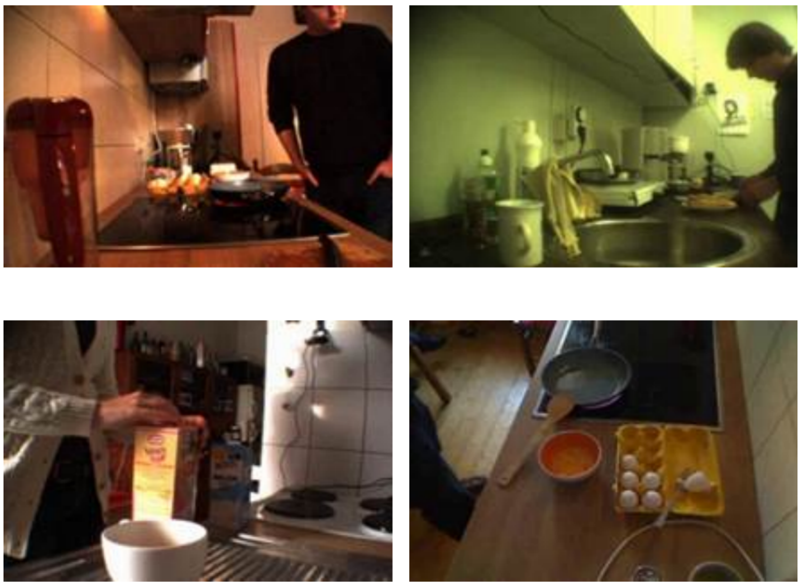

Figure 4.1: Samples from the *Prepare tea* and *Prepare coffee* activities from the Breakfast Dataset

The Inria Instructional Videos (INRIA) dataset has 150 videos of different activities such as *Making a coffee*, *Changing car tire*, *Performing cardiopulmonary resuscitation (CPR)*, *Jumping a car and Repotting a plant*, with a total number of actions equal to 47. Each video has 2 minutes of duration on average, containing nine actions on average. This dataset also has a very high ratio of background frames of 65%. Figure 4.2 illustrate examples of the INRIA dataset.

Figure 4.2: Samples from “Change car tire“ action in INRIA Dataset

Background frames are frames that contain no actions relevant for the activity that a video depicts. They present a major challenge in the context of action segmentation since they are randomly distributed in the time dimension, as well as in the spatial dimensions, meaning that they may appear at any point in the timeline and a single video may have background frames of completely different motion and visual appearance. For this reason, the INRIA dataset is more challenging, since it has a very high rate of background frames. A more detailed analysis of its results are discussed in section 4.6.

4.2 Setup

Our method was tested with a 6 cores i7 2.60 GHz CPU and a RTX-2070 Max-Q Design GPU. We set 3 hyperparameters: the number of times, t, that the Silhoutte Score does not increase which we used $t = 5$; the maximum number of clusters, C that the Silhoute Score heuristic could reach we set it to $C = 98$ which is twice the total number of actions in all activities combined, to make sure our experiments wouldn't be biased to a number that in a real-life situation we wouldn't know.

4.3 Evaluation Protocol

In this section we detail the metrics used to evaluate each dataset, how we compute them and describe the mapping algorithm between clusters and ground-truth labels necessary to evaluate the unsupervised method.

For both datasets, we report the mean over frames (MoF) metric, which is calculated the same way as the accuracy metric as can be seen in equation 4-1. Moreover, for the Breakfast dataset we also compute the intersection over union metric, since it is less sensitive for class imbalance, because it is calculated as the average of the IoU of each class. The computing of IoU for a single class is described in equation 4-2. For the INRIA dataset we also report the F1-Score, also known as the harmonic mean between precision and recall, shown in equation 4-3.

$$MoF = \frac{TP + TN}{TP + TN + FP + FN} \tag{4-1}$$

$$IoU = \frac{TP}{TP + FP + FN} \tag{4-2}$$

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{4-3}$$

In order to compute these metrics and evaluate the temporal segmentation predicted by our unsupervised approach, we need some means of mapping our method's output clusters of actions to a ground-truth annotated base.

The output of our method, represented by cluster labels for each video snippet, do not necessarily match the boundaries nor the action's labels used in the dataset. For this reason we need to make a one-to-one mapping between the predicted segments and the ground truth labels. For example, in figure 4.3 there are 5 actions and 4 predicted clusters and we need to find the mapping that maximizes the similarity between the ground truth and the predictions.

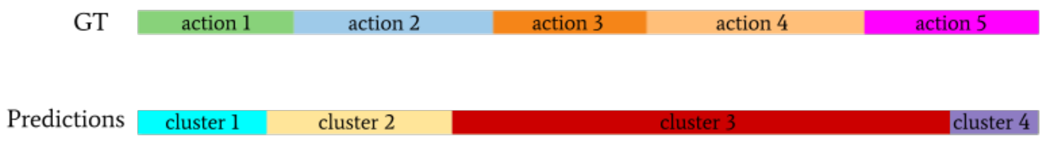

Figure 4.3: The problem of finding the mapping that maximizes the similarity between the ground truth actions and the predicted clusters

So to generate such a mapping, following (22, 24, 41), we use the Hungarian method. This method is an algorithm that solves the assignment problem, in our case, the goal is the optimum assignment that maximizes the similarity between the predicted clusters and the ground-truth actions. Figure 4.4 illustrates the problem posed in figure 4.3 as a bipartite graph where each edge is weighted by the number of matching frames each ground-truth action has with each cluster, in this case is easy to visually understand the mapping that maximises the frame similarity.

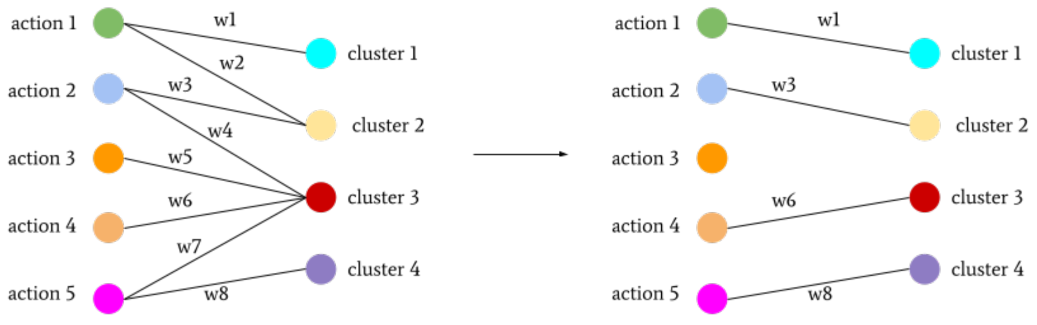

Figure 4.4: The assignment problem represented as a bipartite graph where each edge is weighted by the number of matching frames each ground-truth action has with each cluster.

So the Hungarian method assigns for each action a predicted cluster that maximizes its similarity, where the similarity is the number of intersecting frames. This method tries to find a *perfect matching*, which is a matching that matches all vertices of the graph while maximizing its weights. To properly understand this algorithm first we have to define the concept of *labeling* and an *equality subgraph*.

1. A *labeling* for graph $G = (V, E)$ is a function $l : \mathbb{V} \to \mathbb{R}$, such that:
$$\forall (u, v) \in E : l(u) + l(v) \geq weight((u, v)) \tag{4-4}$$
2. An *Equality Subgraph* is a subgraph $G_l = (V, E_l) \subset G(V, E)$, fixed on a labeling l, such that:
$$E_l = (u, v) \in E : l(u) + l(v) weight((u, v)) : \tag{4-5}$$

Now that we have these two definitions we have to state the theorem that is the basis of the Hungarian Algorithm, the Kuhn-Munkres theorem:

Teorema 4.1 *Given labeling l, if M is a perfect matching on G_l, then M is maximal-weighting of G.*

1. Let $M^{'}$ be any perfect matching in G. By definition of a labeling function and since $M^{'}$ is perfect,
$$weight(M^{'}) = \sum_{(u,v) \in M^{'}} weight((u, v)) \leq \sum_{(u,v) \in M^{'}} l(u) + l(v) = \sum_{v \in V} l(v) \tag{4-6}$$
2. This means: $\sum_{v \in V}$ is an upper bound for any perfect matching $M^{'}$ of G.
3. Now the weight of matching M:
$$weight(M) = \sum_{(u,v) \in M} weight((u, v)) = \sum_{(u,v) \in M} l(u)+l(v) = \sum_{v \in V} l(v) \tag{4-7}$$
4. By equations 4-6 and 4-7, we have that for all perfect matchings $M^{'}$ of G: $weight(M) \geq weight(M^{'})$

Therefore, by the Kuhn-Munkres theorem the problem of finding a maximum weight assignment is reduced to finding the right labeling function and any perfect matching on the corresponding equality subgraph. Explained next in a step-by-step manner.

The Kuhn-Munkres Algorithm (57)
Given a bipartite graph $G = (X, Y, E)$. Start with an arbitrary feasible vertex labeling l, determine G_l, and choose an arbitraty matching M in G_l

1. If M is complete for G, then M is optimal and stop. Otherwise, there is some umatched $x \in X$. Set $S = x$ and $T =$

2. if $J_{G_l}(S) \neq T$, go to step 3. Otherwise, find

$$\alpha_l = \min_{x \in S, y \in T^C} l(x) + l(y) - w(xy) \tag{4-8}$$

Where T^C denotes the complement of T in Y, and construct a new labeling $l^{'}$ by

$$l^{'}(v) = \begin{cases} l(v) - \alpha_l & \text{for v} \in S \\ l(v) + \alpha_l & \text{for v} \in T \\ l(v) & otherwise \end{cases} \tag{4-9}$$

Note that $\alpha_l \geq 0$ and $J_{G_l}(S) \neq T$. Replace l by $l^{'}$ and G_l by $G_{l^{'}}$

3. Choose a vertex y in $J_{G_l}(S)$. If y is matched in M, say with $z \in X$, replace S by $S \cup z$ and T by $T \cup y$, and go to step 2. Otherwise, there will be an M-alternating path from x to y, and we may use this path to find a larger matching $M^{'}$ in G_l. Replace M by $M^{'}$ and go to step 1.

4.4 Experiments

In this section we go through the experiments performed, detailing the goal and configuration of each. To properly evaluate our method from different perspectives, we devised experiments changing its main components. For all of the experiments defined in the next paragraph, we implemented them for both datasets described in section 4.1. The main components of our method are the video snippet temporal window and its feature extractor, the positional encoding and the clustering algorithm. The video snippet and feature extractor are considered one component because the temporal window size is dependent on the model we are using, as mentioned in section 3.2, the I3D model has a minimum video input length of 10 frames, while the Slowfast of 32 frames.

The first experiment devised we fixed the temporal window of video snippets at the minimum length the feature extractor used accepted, what would be the *vanilla* version of our method, since this minimum snippet values are the default inputs length for both architectures. In this test we compared how the different combinations of feature extractor and clustering algorithm affected the performance of the method. For this experiment, we also performed an ablation test, where we utilize the positional encoding component or not.

The other experiment performed consists in changing the temporal window of video snippets. However, for this experiment, we only tested it with the FINCH algorithm. The computing time for KMeans with Silhouette Score, depending on the video was taking too long, this could be mitigated by testing different t and C values, but we leave it as future work. For the SlowFast model we tried values of $n \in \{32, 40, 48, 64, 72, 128\}$ and for the I3D $n \in \{10, 16, 32, 40, 48, 64\}$. We stopped at 128 frames for SlowFast and 64 frames for I3D due to memory issues. We could go a bit further with SlowFast since it does not use optical flow and has a more computational efficient architecture. Here we also performed an ablation experiment with the positional encoder component. This experiment's goal is to shed light into how the representations learned by the models of longer temporal windows affects our method. Moreover, how the positional encoder alters the geometry of the embeddings of different temporal window's length.

4.5 Results

Next we present the main results of our method for both datasets. First we present the best overall results of our experiments, than go through the results for the experiments described in the previous section. A more in-depth discussion is detailed in 4.6.

The main results for the Breakfast Actions Dataset can be seen in table 4.1, where we compare our best results for both MoF and IoU metrics, for the pair of feature extractors with or without positional encoding compared to other state-of-the-art works.

Table 4.1: Comparison on the Breakfast dataset. Here we present the best results for both metrics with and without positional encoding, while comparing it with other approaches.

#	Method	Type	MoF	IoU
01	TW-FINCH (22)	Unsupervised	62.7	42.3
02	**I3D-64+FINCH (Ours)**	**Unsupervised**	**57.99**	**29.61**
03	**I3D-64+FINCH+PE (Ours)**	**Unsupervised**	**57.75**	**35.52**
04	**Slowfast-72+FINCH+PE (Ours)**	**Unsupervised**	**57.38**	**32.17**
05	**Slowfast-32+KMeans+SS (Ours)**	**Unsupervised**	**56.50**	**33.80**
06	CDFL (18)	Weakly Sup.	50.2	33.7
07	MuCon (19)	Weakly Sup.	49.7	-
08	VTE-UNET (21)	Unsupervised	48.08	-
09	D3TW (58)	Weakly Sup.	45.7	-
10	**Slowfast-32+KMeans+SS+PE (Ours)**	**Unsupervised**	**45.10**	**43.40**
11	NN-vit (23)	Weakly Sup.	43.0	-
12	LSTM+AL (59)	Unsupervised	42.9	46.9
13	CTE (41)	Unsupervised	41.8	-
14	TCFPN (60)	Weakly Sup.	38.4	24.2
15	RNN+HMM (40)	Weakly Sup.	36.7	-
16	Mallow (24)	Unsupervised	34.6	47.1
17	RNN-FC (17)	Weakly Sup.	33.3	-
18	SCT (61)	Weakly Sup.	30.4	-
19	GMM+CNN (62)	Weakly Sup.	28.2	12.9

Our method's best results (weather using the positional encoder or not) outperforms almost all other unsupervised and weakly supervised methods. In the case of positional encoded features extracted with the I3D model, the best performance for both metrics was produced with the configuration in which the video snippets had a length of 64 frames clustered with FINCH (I3D-64+FINCH+PE), achieving 57.75% and 35.53% for the MoF and IoU metrics respectively.

The best results for both metrics without positional encoding with the I3D model are also with 64 frames videos snippets clustered with FINCH (I3D-64+FINCH), scoring 57.99% for MoF and 29.61% for IoU.

For positional encoded Slowfast embeddings, the best results for the MoF were found with video snippets of 72 frames clustered with FINCH (Slowfast-72+FINCH+PE), achieving a MoF of 57.38% . For the IoU, the best value of 43.4% was achieved with 32 frames long video snippets clustered with KMeans (KM) and the Silhoutte score method (Slowfast-32+KM+SS+PE). Finally, without positional encoding, the best performing configuration with Slowfast features for both metrics was the 32 frames video snippets clustered with KMeans and the Silhoutte score (Slowfast-32+KM+SS+PE), scoring 56.5% for MoF and 33.8% for IoU.

The main results for the INRIA Instructional Videos dataset are presented in table 4.2.

Table 4.2: Comparison on the Inria Instructional Videos dataset. Here we present the best results for both metrics with and without positional encoding, while comparing it with other approaches.

#	Method	Type	MoF	F1-Score
01	TW-FINCH (22)	Unsupervised	58.6	51.9
02	**Slowfast-128+FINCH+PE (Ours)**	**Unsupervised**	**53.89**	**46.32**
03	**I3D-64+FINCH+PE (Ours)**	**Unsupervised**	**53.35**	**45.97**
04	**I3D-64+FINCH (Ours)**	**Unsupervised**	**52.74**	**44.56**
05	**Slowfast-48+FINCH (Ours)**	**Unsupervised**	**51.34**	**44.17**
06	LSTM + AL (59)	Unsupervised	-	39.7
07	CTE (41)	Unsupervised	39.0	28.3
08	Mallow (24)	Unsupervised	27.8	27.0

Once again, the top results of our method surpass almost all other approaches. Our top achieving result with Slowfast positional encoded embeddings for both metrics is the setup combined with FINCH, with 128 frames video snippets (Slowfast-128+FINCH+PE), achieving 53.80% and 46.32% for MoF and F1-Score respectively. Without positional encoding, the best Slowfast configuration for both metrics was with snippets of 48 frames clustered with FINCH (Slowfast-48+FINCH), with a MoF of 51.34% and an F1-Score of 44.17%.

For I3D features with positional encoding, our top score for both metrics was achieved with 64 frames clustered with FINCH, scoring 53.35% on MoF and 45.97% on the F1-Score. Removing the positional encoder component, our top achieving result with the MoF metric is the I3D combined with FINCH with snippets of 48 frames, attaining a MoF score of 53.08%; for the F1-Score, the best result was also with FINCH, but now with 64 frame-snippets, scoring 44.56%.

Vanilla experiment's results can be seen in table 4.3 and 4.4 for the Breakfast and INRIA datasets respectively. The best results for both datasets are without positional encoding, but with different feature extractors and clustering procedure. For the Breakfast our best result in the *vanilla* setup is the SlowFast with KMeans, meanwhile for INRIA is I3D with FINCH. Furthermore, we point it out that, on average, the MoF metric is 51.72% with positional encoding, and 54.98% without it. For the IoU metric is 35.72%, and

29.63% with positional encoding and without, respectively. Looking at the INRIA results, on average, the MoF metric is 44.72% for positional encoded embeddings, and without is 46.44%. Moreover, the F1-Score is on average 39.95% with positional encoding, and without is 40.70%.

Table 4.3: Our method's results from the *vanilla* experiment with the Breakfast dataset

#	Method	MoF	IoU
01	**Slowfast-32+KMeans**	**56.5**	**33.8**
02	**I3D-10+FINCH**	**55.33**	**27.83**
03	**I3D-10+KMeans+PE**	**54.7**	**29.4**
04	**I3D-10+KMeans**	**54.2**	**29.4**
05	**Slowfast-32+FINCH**	**53.9**	**27.5**
06	**I3D-10+FINCH+PE**	**53.89**	**29.68**
07	**Slowfast-32+FINCH+PE**	**53.2**	**40.4**
08	**Slowfast-32+KMeans+PE**	**45.1**	**43.4**

Table 4.4: Our method's results from the *vanilla* experiment with the INRIA dataset

#	Method	MoF	F1-Score
01	**I3D-10+FINCH**	**49.85**	**43.42**
02	**I3D-10+FINCH+PE**	**47.25**	**43.22**
03	**Slowfast-32+FINCH**	**45.83**	**40.27**
04	**Slowfast-32+FINCH+PE**	**45.47**	**40.17**
05	**Slowfast-32+KMeans**	**45.4**	**39.89**
06	**Slowfast-32+KMeans+PE**	**44.93**	**39.71**
07	**I3D-10+KMeans**	**44.69**	**39.23**
08	**I3D-10+KMeans+PE**	**41.22**	**36.72**

The temporal window length variation experiment results are displayed in table 4.5 for the Breakfast dataset, and in table 4.6 for the INRIA dataset. This time we separate the results also by feature extractor, since increasing the temporal window length helps us measure and understand how well the representations learned by both models capture larger and larger spatio-temporal information from videos.

For the Breakfast Dataset while using the Slowfast feature extractor, the top-5 results are only formed by the largest temporal window embeddings combined with positional encoding. For the I3D, almost all are positional encoded, but the first position, which is the 64 frames temporal window.

Table 4.5: Our method's results from the temporal window length variation experiment with the Breakfast dataset

#	Method	MoF	IoU
01	**Slowfast-72+FINCH+PE**	**57.38**	**32.17**
02	**Slowfast-64+FINCH+PE**	**57.11**	**32.75**
03	**Slowfast-48+FINCH+PE**	**56.72**	**36.03**
04	**Slowfast-128+FINCH+PE**	**55.86**	**25.21**
05	**Slowfast-40+FINCH+PE**	**54.75**	**36.94**
06	**Slowfast-48+FINCH**	**54.07**	**27.05**
07	**Slowfast-72+FINCH**	**53.92**	**25.24**
08	**Slowfast-32+FINCH**	**53.90**	**27.5**
09	**Slowfast-32+FINCH+PE**	**53.20**	**40.40**
10	**Slowfast-128+FINCH**	**51.2**	**20.42**
11	**Slowfast-40+FINCH**	**50.71**	**24.84**
01	**I3D-64+FINCH**	**57.99**	**29.61**
02	**I3D-48+FINCH+PE**	**57.75**	**34.3**
03	**I3D-40+FINCH+PE**	**57.73**	**33.32**
04	**I3D-64+FINCH+PE**	**57.49**	**35.52**
05	**I3D-32+FINCH+PE**	**57.47**	**32.98**
06	**I3D-48+FINCH**	**57.42**	**28.87**
07	**I3D-24+FINCH+PE**	**56.33**	**32.09**
08	**I3D-40+FINCH**	**56.07**	**29.25**
09	**I3D-16+FINCH**	**55.45**	**30.49**
10	**I3D-32+FINCH**	**55.4**	**28.6**
11	**I3D-10+FINCH**	**55.33**	**27.83**
12	**I3D-24+FINCH**	**54.49**	**27.8**
13	**I3D-16+FINCH**	**54.28**	**27.67**
14	**I3D-10+FINCH+PE**	**53.89**	**29.68**

The INRIA results for the Slowfast features behave very similarly to the Breakfast dataset. The top-5 results contain only one combination without positional encoding embeddings in fifth place, while the top-4 is all formed by the largest temporal window features with positional encoding. On the other hand, the I3D results are not regular regarding this analysis. The top-5 is formed by the largest temporal window embeddings as well, but even though the first place is the 64 frames video snippets positional encoded features, there are three variations with no positional encoding in the top-5.

Table 4.6: Our method's results from the temporal window length variation experiment with the INRIA dataset

#	Method	MoF	F1-Score
01	**Slowfast-128+FINCH+PE**	**53.89**	**46.32**
02	**Slowfast-64+FINCH+PE**	**52.91**	**44.81**
03	**Slowfast-48+FINCH+PE**	**51.56**	**44.48**
04	**Slowfast-40+FINCH+PE**	**51.45**	**45.91**
05	**Slowfast-48+FINCH**	**51.34**	**44.17**
06	**Slowfast-64+FINCH**	**50.09**	**43.75**
07	**Slowfast-40+FINCH**	**49.53**	**43.74**
08	**Slowfast-128+FINCH**	**48.47**	**42.26**
09	**Slowfast-32+FINCH**	**45.83**	**40.27**
10	**Slowfast-32+FINCH+PE**	**45.47**	**40.17**
01	**I3D-64+FINCH+PE**	**53.35**	**45.97**
02	**I3D-48+FINCH**	**53.08**	**44.37**
03	**I3D-40+FINCH**	**52.86**	**44.42**
04	**I3D-64+FINCH**	**52.74**	**44.56**
05	**I3D-48+FINCH+PE**	**52.51**	**45.79**
06	**I3D-16+FINCH**	**50.91**	**43.95**
07	**I3D-40+FINCH+PE**	**50.77**	**44.54**
08	**I3D-10+FINCH**	**49.85**	**43.42**
09	**I3D-16+FINCH+PE**	**49.24**	**44.56**
10	**I3D-32+FINCH**	**48.88**	**42.27**
11	**I3D-32+FINCH+PE**	**48.49**	**42.07**
12	**I3D-24+FINCH**	**48.31**	**41.9**
13	**I3D-10+FINCH+PE**	**47.25**	**43.22**
14	**I3D-24+FINCH+PE**	**46.76**	**41.5**

4.6 Discussion

In this section we go through the results as well, but now discussing findings and analysis of each experiment.

4.6.1 Comparison with the state-of-the art

Our method, independently of the configuration, reached very good results when looking at the current landscape of weakly supervised and unsupervised approaches. Our worst performing test for the Breakfast dataset when looking at the MoF metric, the Slowfast with 32 frames long positional encoded embeddings, clustered with KMeans and the Silhouette Score, is in the top-10 results with a MoF of 45.1%. Furthermore, this same configuration reaches top-4 results when looking at the IoU metric, achieving a 43.4% score. The average results of our method with positional encoding are 55.39% and 33.64% for the MoF and the IoU metric respectively, reaching a second place overall when comparing with the best results of other approaches. All of this without the need for any kind of annotation, nor training, very common in many of other approaches as shown in table 2.1 in chapter 2.

Following a similar analysis done for the Breakfast dataset, our worst performing configuration, I3D with 10 frames long positional encoded embeddings clustered with KMeans and the Silhouttte Score, reached the second place overall for the INRIA dataset. When looking at the MoF metric with 41.22%, and a third place when comparing the F1-Score with a result of 36.72%. Additionally, on average our method reached improvements of over 10% for the MoF metric, and of almost 4% for the F1-Score.

However, even though our method performed much better than almost all other approaches, the top scoring approach is still TW-FINCH (22) by a good margin for both datasets. We own this difference for two main reasons, first the way TW-FINCH utilizes positional information to build its clusters. FINCH utilizes a one nearest neighbor graph to generate clusters recursively, and TW-FINCH is very similar. The difference is that when TW-FINCH builds its graph, it does by computing the spatio-temporal feature space distance and modulating the frame's features with their respective temporal position. The second reason is due to the over-segmentation problem, where same actions occurring in temporally distant moments are assigned to different clusters. This issue can be amplified in our method as a side effect of the positional encoding component. For instance, two frames are injected with positional information that is increasingly different as their temporal distance increases. This effect of over-segmentation may be especially detrimental for datasets that have actions which appear multiple times in distant temporal instants, which is the case of INRIA. The INRIA dataset has a very high percentage of background frames, so multiple segments that should be assigned to the same cluster are instead assigned to different ones, such as in the Figure 4.5. In this example, the high percentage of background frames creates a lot of noise among the target actions happening in the video. In this case, the ground truth has 9 actions in total, where the majority of the video is composed by only one of them, which is the randomly distributed background class. This makes the target actions interspersed by background frames, fooling our method into misclassifying same actions in different ones, specially background frames, causing the over-segmentation. Here our method predicted 15 classes, instead of 9.

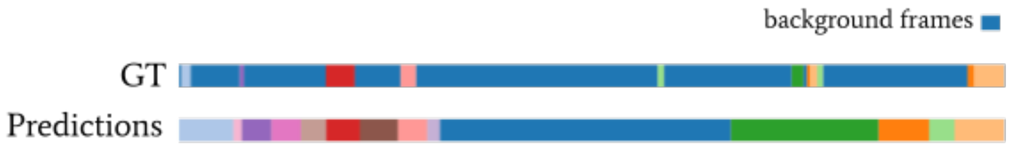

Figure 4.5: Action predictions produced by the SlowFast 64 frames positional encoded embeddings clustered with FINCH of the video coffee_0018 from the INRIA dataset. Segments with equal colours belong to the same class.

One way to mitigate this issue is to add one more step to our method, where after generating the clusters with the positional encoding, and finding continuous segments of different actions, apply a time-series cluster method. The idea here is to interpret each cluster of video snippets as a time-series sample, and cluster them. In theory, a clustering approach such Dynamic Time Warping (63) should be able to match the actions that are classified as different clusters because of their similarities, we leave this as future work.

4.6.2 Positional encoding analysis

For both datasets the positional encoder has proven to be helpful. On average, the MoF metric is 0.78% higher, and the IoU has an impressive 6.07% improvement for the Breakfast dataset. Even though, our best result considering the MoF metric is the configuration with I3D 64 frames without positional encoding clustered with FINCH (I3D-64+FINCH), the difference of 0.24 % to our second best configuration to the MoF metric is statistically insignificant, which is the same set up but with positional encoding. Now, when comparing both IoU results, the positional encoded version presents significant gains of 5.91%. This high gains in the IoU metric are very relevant, since its calculation is an average of the IoU per class, showing that the positional encoder is helpful to achieve better class-wise segmentations.

When looking at the INRIA dataset, the MoF metric for configurations without the positional encoder is slightly higher with a difference of 0.28%. Meanwhile, the F1-Score has an average performance better with positional encoding, 43.27% versus 42.72%, a gain of 0.55%. In spite of the statistical equivalence when averaging all the results, our top result is a configuration with positional encoding. The SlowFast 128 frames positional encoding embeddings clustered with FINCH is our best result, with improvements of 0.81% for the

MoF metric, and 1.95% for the F1-Score when compared with the best result of our method without positional encoding.

The results are far better for the Breakfast dataset since its videos have actions that are contiguous and appear only once, furthermore, they also have a much lower percentage of background frames as can be seen in the example displayed in Figure 4.6. This characteristics are very good for positional encoding embeddings because video snippets that are closer to each other in the timeline will be injected with similar positional encoding information, while farther video snippets in the timeline have increasingly different positional encodings.

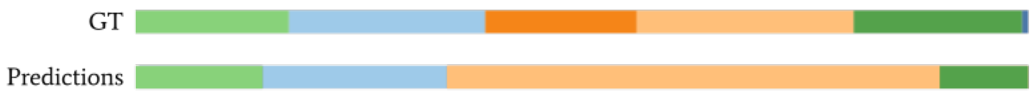

Figure 4.6: The timelines for the same video, the first are the ground truth segments and in the second are the predictions made by our I3D with 48 frames positional encoded embeddings. Even though it misses two classes, there is a nice smooth characteristic for the predicted segments.

This trait is very desirable for a clustering algorithm based on first neighbors to find the linking chains of each action such as FINCH. In Figure 4.7 we can see the clear path the video snippets make with positional encoding, and how dispersed they are without it. In contrast with the affinity of positional encoding and FINCH, the same can be said for KMeans and the absence of positional encoding. Even so, that most results for both datasets of our method with KMeans are better without positional encoding. For instance, SlowFast 32 frames embeddings with KMeans and no positional encoding is 11.4% better in the MoF metric comparing to its positional encoded version. While for INRIA, the I3D 10 frames embeddings with KMeans and no positional encoding configuration has a 3.47% advantage when comparing the MoF metric with its positional encoded counterpart.

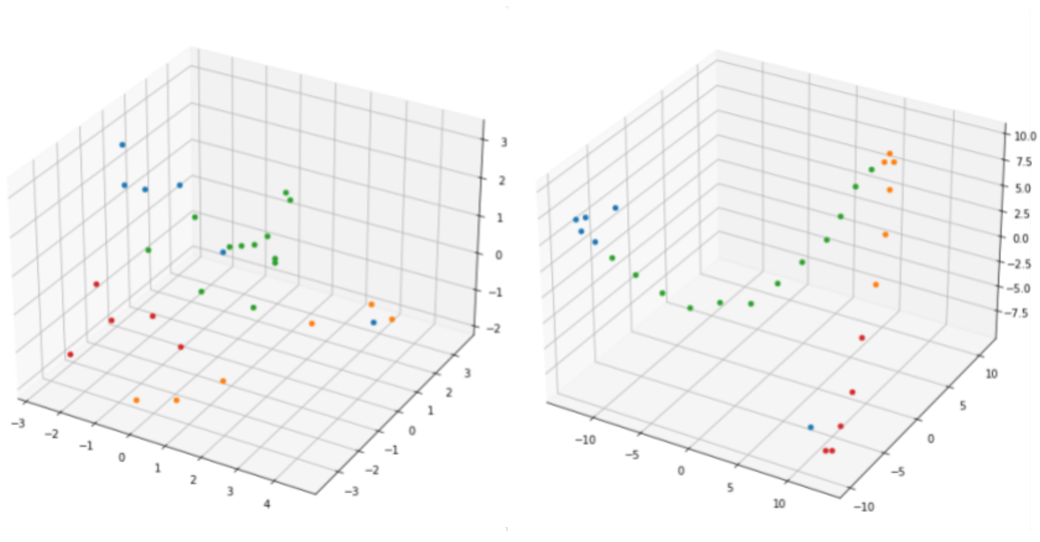

Figure 4.7: Projections of the Slowfast 32 frames video snippets embeddings of the video *webcam01/P03_coffee.avi*. On the left, embeddings without positional encoding, and on the right, with positional encoding.

4.6.3 Temporal window size analysis

The temporal window size has a very important role in our method, since its length changes completely the quality of the embeddings, and therefore in the behavior of the predicted clusters. Logically, the amount of spatio-temporal information contained in a video snippet is proportional to its length. However, to be able to translate and capture this information into an embedding we are dependent on the capacity of the feature extractor to do so.

There are two main types of dependencies to be captured in the spatial and temporal domain, short and long. To understand this concept, in figure 4.8, a 2D kernel of a convolutional layer is processing an image. In each step its algorithm is processing spatial data within the kernel dimensions, therefore capturing a short-range dependency. However, to properly understand an image, algorithms must be able to generate long-range dependencies. To capture a long-range dependency in the image example an algorithm would have to be able to compute the dependencies between distant pixels in the space dimensions. This is true also for the time domain, but instead of extracting the dependencies only in the two dimensions of space in a frame, there is also the time dimension. Capturing long-range dependencies is well-known to be a harder task than short-range (64). In our experiments, we tried to explore that by testing our embeddings with the shortest possible temporal window per model, to the longest that our computers permitted.

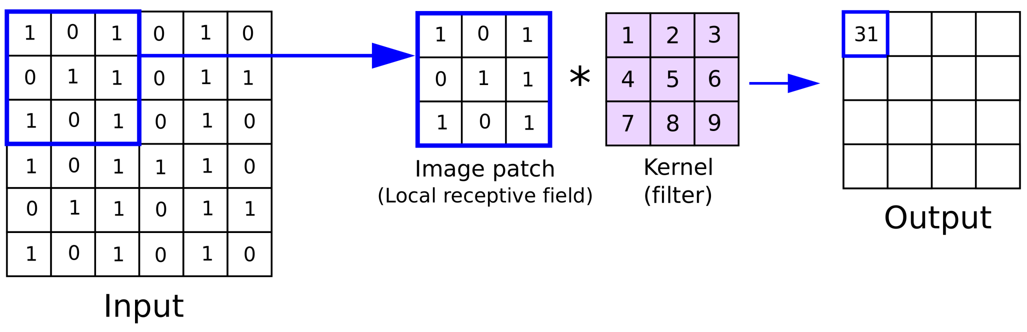

Figure 4.8: Visualization of how a 2D convolutional kernel works. Here it is easy to see why a convolutional operation captures only short-range dependencies, more precisely inside the kernel dimensions. [1]

The Slowfast model in the Breakfast dataset shows that larger temporal window sizes are good until a certain point for the MoF metric. As can be seen in Figure 4.9, the chart in the left shows that the MoF metric increases until the temporal window of 70 frames, but then drops for 128 frames. This makes sense since larger and larger temporal windows will end up merging multiple actions in one video snippet, thus degrading the performance, such as in the example in Figure 4.10. Another important observation is that the increase of temporal window size only benefits positional encoded embeddings. The pure embeddings seem to have a non-linear behavior as the temporal window increases with a downwards trend.

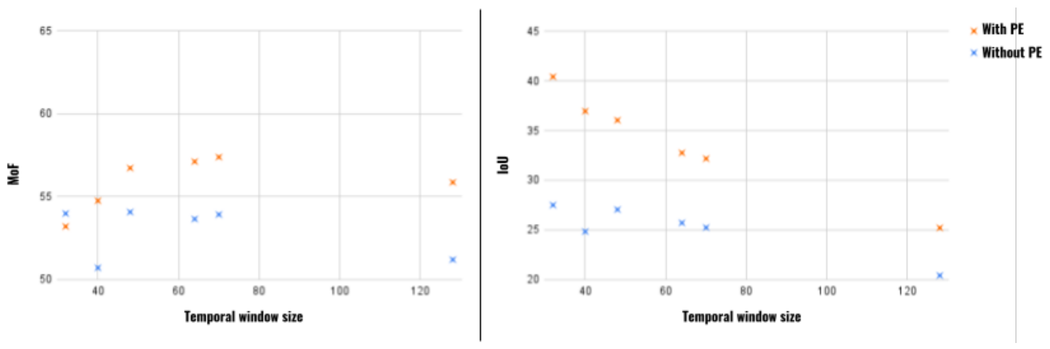

Figure 4.9: Slowfast results for the Breakfast dataset, on the left the chart of MoF vs Temporal window Size, and on the right of IoU vs Temporal window Size.

For the IoU the increase of the temporal window only led to losses, in the right chart of Figure 4.9, it is clear the negative correlation that this two variable present. This decrease may also be explained by the effect shown in Figure 4.10, since the IoU is calculated as the average IoU of each class, therefore, this under-segmentation we see for larger temporal windows may be leading to these losses.

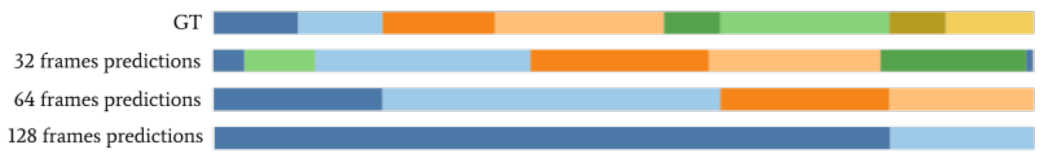

Figure 4.10: Each timeline represents a segmentation of the video *cam02/P42_coffee.avi* with Slowfast positional encoded features and clustered with FINCH. The first row is the ground truth, and the following ones are the predictions with increasingly larger temporal window sizes. In this example, same colours in different timelines do not necessarily correspond to the same action. The main point here is to show an example that a larger temporal window yields less clusters.

The I3D showed results that are very different from the Slowfast, for both metrics, as shown in Figure 4.11. First, the only similarity is that when looking at the MoF metric for positional encoded embeddings, is safe to say that in spite the positive correlation between MoF and the temporal window size, we can see that the increase in the MoF score starts to get dimmer and dimmer as the temporal window increases, even so that the 48 frames embeddings reach better MoF than the 64 frames. We couldn't test this model with 128 frames video snippets due to computational constraints, but a similar trend can be seen in the Slowfast before the 128 frames test. A important point to notice here is that the results with pure embeddings extracted by the I3D are also increasing with the temporal window. The same is true for the IoU, but now for both positional encoded and pure embeddings. This may be an indication that the I3D features are better suited for cluster-based methods, even though the Slowfast reaches better results in action recognition benchmarks.

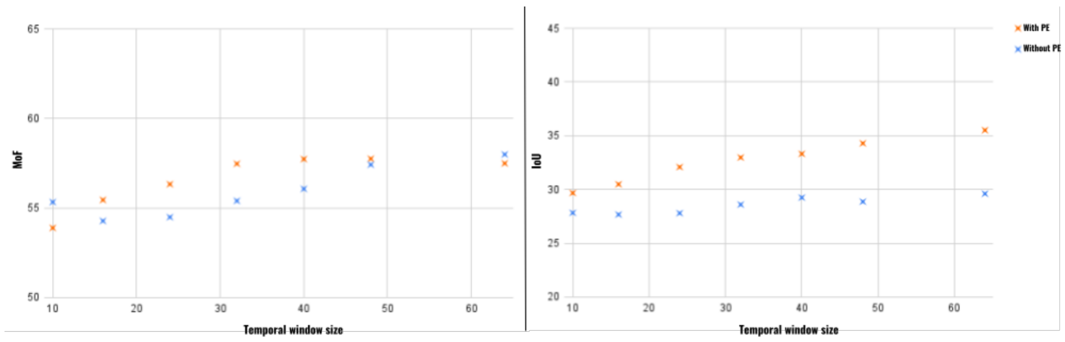

Figure 4.11: I3D results for the Breakfast dataset, on the left the chart of MoF vs Temporal window size, and on the right of IoU vs Temporal window size.

The INRIA dataset results for the Slowfast model are displayed in the chart in Figure 4.12. There we can see that there is an overall positive correlation for positional encoded embeddings temporal window size and both metrics. We believe this is happening because as explained before, larger temporal windows yield less clusters, which may be mitigating the over-segmentation problem caused by background frames and interspersed actions. Meanwhile, without the positional encoding, the behavior is totally non-linear for the MoF and IoU metrics. This behavior was also seen for Breakfast dataset, which might be an indication that the capabilities of capturing long-range dependencies for the Slowfast model are not so good. Moreover, the positional encoding seems to be helping to produce better segmentations, even when the embeddings are not very discriminative.

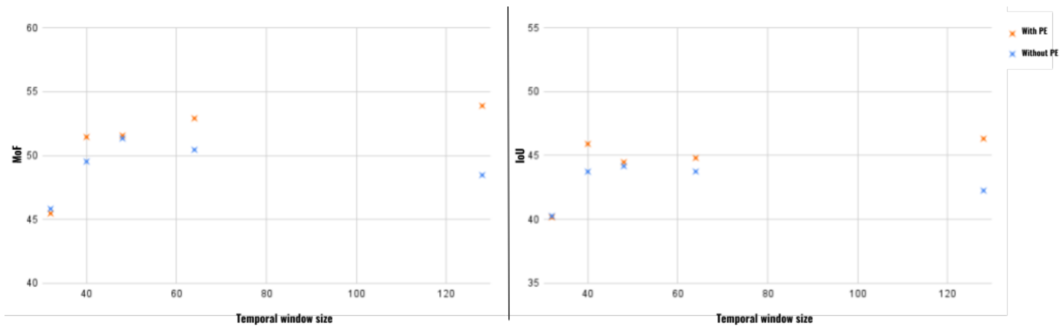

Figure 4.12: Slowfast results for the INRIA dataset, on the left the chart of MoF vs Temporal window size, and on the right of IoU vs Temporal window size.

Now for the INRIA dataset analysis with the I3D model, its results are displayed in the charts of Figure 4.13. For both metrics and positional encoded or not, the results increase until 16 frames, drop and then go back to increase. We believe this a mix of the under-segmentation caused by longer temporal windows mitigating the over-segmentation problem present in the INRIA, and also that the I3D embeddings seem to be highly discriminative, independently of the temporal window size.

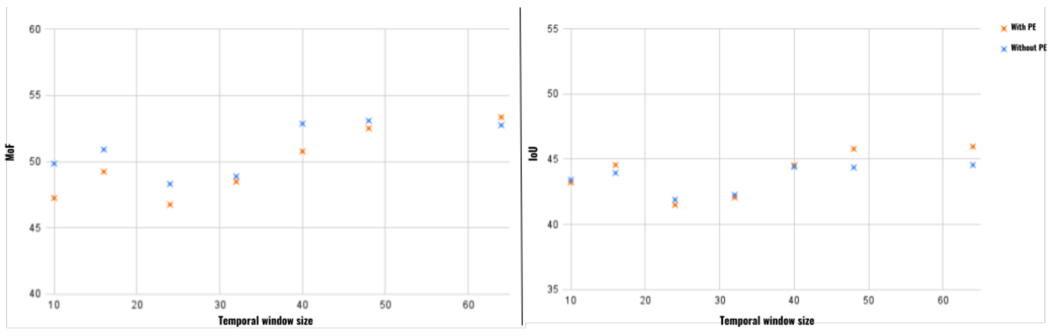

Figure 4.13: I3D results for the INRIA dataset, on the left the chart of MoF vs Temporal window size, and on the right of IoU vs Temporal window size.

5 Conclusions

In this dissertation, we present a method for the important task of temporal segmentation of actions with important applications in many applications such as video surveillance, human-robots interaction, medical diagnosis, video recommendation and video summarisation (4, 8). Our proposal is a cluster-based method, in which we address action segmentation as a grouping problem. A trivial way to generate predictions would be simply by generating frame-wise feature vectors and cluster them. However, actions are composed by visual and temporal information and such a method like this would be totally unaware of temporal information. For this reason, we explored ways to generate frames representations that are imbued with spatio-temporal information and are highly discriminative. Additionally, another important cue for segmenting actions is the positional information that a frame or a sequence of frames are relative to the timeline. In other words, the order of the embeddings we are generating is also valuable when predicting action boundaries.

Regarding the generation of video embeddings, we realize that short video snippets are of the exact same type of input that action recognition models are trained for. So we leverage mature action recognition architectures pre-trained in large scale datasets to generate our embeddings. We investigate two of them, the I3D (12) and the Slowfast (13) models. Moreover, we also explore how varying the length of video snippets that generate the feature vectors affects the quality of segmentation.

For the positional information signal we use a method originally proposed for NLP tasks: the positional encoding method, by Vaswani et. al (43). As far as we know, this is the first work to investigate this method in a temporal action segmentation task. We perform ablation experiments to understand how its usage affects the method.

Two different clustering algorithms, FINCH (54) and KMeans (53) were investigated, and our method (without any training or annotation) reaches competitive results in two of the main benchmark datasets (the Breakfast dataset (55), and the INRIA dataset (56)).

In summary, we highlight the following contributions of this dissertation:

1. A method for temporal action segmentation that needs no training or

annotated data;

2. An investigation of how different temporal windows affect the behavior of our proposal;

3. An investigation of the usage of the positional encoding method from the paper of Vaswani et. al (43) in a temporal action segmentation task;

4. An investigation of the efficacy of our method in two challenging benchmarks and reach competitive results with the state-of-the-art;

Our method, composed by the steps of video snippets sampling, feature extraction, positional encoding and clustering provide a backbone for temporal action segmentation of any video. Therefore, each step may be filled with a different algorithm than the ones investigated here, opening opportunities for future research. For example, a future work could investigate the usage of this method for IoT devices and see how efficient action recognition architectures would behave, explore new positional encoding strategies, try new modalities when generating features such as audio.

Furthermore, there are still open problems for temporal action segmentation. The over-segmentation problem caused by temporally distant frames of same actions amplified by the effect of positional encoder. This could be tackled by a post-processing step inside our method's pipeline, such as using a time-series clustering procedure and modelling each cluster as a time-series sample. Another future work in this specific problem is to investigate if different strategies for injecting the positional information are better suited here. For example, instead of summing the positional encoding matrix with the video embeddings, concatenate them might lead to better results. There is also the problem of under-segmentation caused by longer temporal windows when extracting features. A possible future work would be after generating the clusters, focus on the boundaries of each action and try to explore specifically the frames around it to mitigate the issue of a single feature vector with more than one action. Finally, there is still the huge problem of background frames that hinder so much the quality of current propositions.

Although our quantitative experiments demonstrated the remarkable results our method can achieve, there is still much to improve, specially when compare to the TW-FINCH (22). A good future work could be the combination of TW-FINCH with our method and investigate how deep feature extractors and positional encoding would behave with such a powerful clustering procedure specifically designed for temporal action segmentation.

Moreover, when thinking on future work, our method could be used to recommend videos or summarise videos. A proposition for video recommendation would be to generate the segmentation of videos, and cluster all these segmentations to generate clusters of segments, and sample recommendations from these groups that are based on the actions present in a video. In the challenging task of video summarisation our method could be used to generate segments, and a summary could be inferred by sampling frames from the segments generated, for example.

Finally, another usage for this method is to be a helper for humans to annotate video data. Since our method does not need any kind of training, any dataset can be segmented by it. A temporal segmentation annotation tool could have this method in the background generating suggestions of segmentation, and a human would just fine-tune over them. This could be helpful to create new datasets quicker, hence helping research in this area even further.

Bibliography

[1] G. N. d. Santos, P. V. de Freitas, A. J. G. Busson, Á. L. Guedes, R. Milidiú, and S. Colcher, "Deep learning methods for video understanding," in *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, 2019, pp. 21–23.

[2] P. R. C. Mendes, E. S. Vieira, P. V. A. de Freitas, A. J. G. Busson, Á. L. V. Guedes, C. d. S. S. Neto, and S. Colcher, "Shaping the video conferences of tomorrow with ai," in *Anais Estendidos do XXVI Simpósio Brasileiro de Sistemas Multimídia e Web*. SBC, 2020, pp. 165–168.

[3] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Comput. Vis. Image Underst.*, vol. 104, no. 2, p. 90–126, nov 2006. [Online]. Available: https://doi.org/10.1016/j.cviu.2006.08.002

[4] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, no. 6, pp. 976–990, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885609002704

[5] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1473–1488, 2008.

[6] A. A. Chaaraoui, P. Climent-Pérez, and F. Flórez-Revuelta, "A review on vision techniques applied to human behaviour analysis for ambient-assisted living," *Expert Syst. Appl.*, vol. 39, no. 12, p. 10873–10888, sep 2012. [Online]. Available: https://doi.org/10.1016/j.eswa.2012.03.005

[7] X. Wang, A. Farhadi, and A. Gupta, "Actions $\sim$ transformations," in *CVPR*, 2016.

[8] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition," *Image Vision Comput.*, vol. 60, no. C, p. 4–21, apr 2017. [Online]. Available: https://doi.org/10.1016/j.imavis.2017.01.010

[9] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The

Kinetics Human Action Video Dataset," *arXiv:1705.06950 [cs]*, May 2017, arXiv: 1705.06950. [Online]. Available: http://arxiv.org/abs/1705.06950

[10] H. Wang and C. Schmid, "Action Recognition with Improved Trajectories," 2013, pp. 3551–3558. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2013/html/Wang_Action_Recognition_with_2013_ICCV_paper.html

[11] K. Simonyan and A. Zisserman, "Two-Stream Convolutional Networks for Action Recognition in Videos," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 568–576. [Online]. Available: http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf

[12] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," 2017, pp. 6299–6308. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Carreira_Quo_Vadis_Action_CVPR_2017_paper.html

[13] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast Networks for Video Recognition," 2019, pp. 6202–6211. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/html/Feichtenhofer_SlowFast_Networks_for_Video_Recognition_ICCV_2019_paper.html

[14] K. Schindler and L. van Gool, "Action snippets: How many frames does human action recognition require?" in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[15] H. Xia and Y. Zhan, "A survey on temporal action localization," *IEEE Access*, vol. 8, pp. 70 477–70 487, 2020.

[16] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic, "Weakly Supervised Action Labeling in Videos Under Ordering Constraints," *arXiv:1407.1208 [cs]*, Jul. 2014, arXiv: 1407.1208. [Online]. Available: http://arxiv.org/abs/1407.1208

[17] A. Richard, H. Kuehne, and J. Gall, "Weakly Supervised Action Learning with RNN based Fine-to-coarse Modeling," *arXiv:1703.08132 [cs]*, Oct. 2017, arXiv: 1703.08132. [Online]. Available: http://arxiv.org/abs/1703.08132

[18] J. Li, P. Lei, and S. Todorovic, "Weakly Supervised Energy-Based Learning for Action Segmentation," *arXiv:1909.13155 [cs]*, Sep. 2019, arXiv: 1909.13155. [Online]. Available: http://arxiv.org/abs/1909.13155

[19] Y. Souri, M. Fayyaz, L. Minciullo, G. Francesca, and J. Gall, "Fast Weakly Supervised Action Segmentation Using Mutual Consistency," *arXiv:1904.03116 [cs]*, Mar. 2020, arXiv: 1904.03116. [Online]. Available: http://arxiv.org/abs/1904.03116

[20] J. Li and S. Todorovic, "Action Shuffle Alternating Learning for Unsupervised Action Segmentation," *arXiv:2104.02116 [cs]*, Apr. 2021, arXiv: 2104.02116. [Online]. Available: http://arxiv.org/abs/2104.02116

[21] R. G. VidalMata, W. J. Scheirer, A. Kukleva, D. Cox, and H. Kuehne, "Joint Visual-Temporal Embedding for Unsupervised Learning of Actions in Untrimmed Sequences," *arXiv:2001.11122 [cs]*, Sep. 2020, arXiv: 2001.11122. [Online]. Available: http://arxiv.org/abs/2001.11122

[22] M. S. Sarfraz, N. Murray, V. Sharma, A. Diba, L. Van Gool, and R. Stiefelhagen, "Temporally-Weighted Hierarchical Clustering for Unsupervised Action Segmentation," *arXiv:2103.11264 [cs]*, Mar. 2021, arXiv: 2103.11264. [Online]. Available: http://arxiv.org/abs/2103.11264

[23] A. Richard, H. Kuehne, A. Iqbal, and J. Gall, "NeuralNetwork-Viterbi: A Framework for Weakly Supervised Video Learning," *arXiv:1805.06875 [cs]*, May 2018, arXiv: 1805.06875. [Online]. Available: http://arxiv.org/abs/1805.06875

[24] F. Sener and A. Yao, "Unsupervised Learning and Segmentation of Complex Activities from Video," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, Jun. 2018, pp. 8368–8376. [Online]. Available: https://ieeexplore.ieee.org/document/8578971/

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: https://doi.org/10.1145/3065386

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *arXiv:1409.0575 [cs]*, Jan. 2015, arXiv: 1409.0575. [Online]. Available: http://arxiv.org/abs/1409.0575

[27] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[28] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, 2017.

[29] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[30] B. K. Horn and B. G. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, 1981. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370281900242

[31] J. Lin, C. Gan, and S. Han, “TSM: temporal shift module for efficient video understanding,” in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 7082–7092. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00718

[32] R. Goyal, S. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic, “The â€œsomething somethingâ€ video database for learning and evaluating visual common sense,” in *2017 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2017, pp. 5843–5851. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.622

[33] H. Kwon, M. Kim, S. Kwak, and M. Cho, “Motionsqueeze: Neural motion feature learning for video understanding,” in *ECCV*, 2020.

[34] C. Feichtenhofer, “X3d: Expanding architectures for efficient video recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[35] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, “A short note about kinetics-600,” *CoRR*, vol. abs/1808.01340, 2018. [Online]. Available: http://arxiv.org/abs/1808.01340

[36] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in homes: Crowdsourcing data collection for activity understanding," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 510–526.

[37] D. Kondratyuk, L. Yuan, Y. Li, L. Zhang, M. Tan, M. Brown, and B. Gong, "Movinets: Mobile video networks for efficient video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 16 020–16 030.

[38] O. Koller, S. Zargaran, and H. Ney, "Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent cnn-hmms," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3416–3424.

[39] A. Richard, H. Kuehne, and J. Gall, "Weakly supervised action learning with RNN based fine-to-coarse modeling," *CoRR*, vol. abs/1703.08132, 2017. [Online]. Available: http://arxiv.org/abs/1703.08132

[40] H. Kuehne, A. Richard, and J. Gall, "A Hybrid RNN-HMM Approach for Weakly Supervised Temporal Action Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 765–779, Apr. 2020, arXiv: 1906.01028. [Online]. Available: http://arxiv.org/abs/1906.01028

[41] A. Kukleva, H. Kuehne, F. Sener, and J. Gall, "Unsupervised Learning of Action Classes With Continuous Temporal Embedding," 2019, pp. 12 066–12 074. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Kukleva_Unsupervised_Learning_of_Action_Classes_With_Continuous_Temporal_Embedding_CVPR_2019_paper.html

[42] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 1243–1252.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv:1706.03762 [cs]*, Dec. 2017, arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762

[44] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *CoRR*, vol. abs/1803.02155, 2018. [Online]. Available: http://arxiv.org/abs/1803.02155

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, 1997.

[46] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[47] P. Mendes, A. Busson, S. Colcher, D. Schwabe, A. Guedes, and C. Laufer, "A cluster-matching-based method for video face recognition," in *Proceedings of the Brazilian Symposium on Multimedia and the Web*, 2020, pp. 97–104.

[48] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 3551–3558.

[49] A. Klaser, M. Marszalek, and C. Schmid, "A Spatio-Temporal Descriptor Based on 3D-Gradients," in *BMVC 2008 - 19th British Machine Vision Conference*, M. Everingham, C. Needham, and R. Fraile, Eds. Leeds, United Kingdom: British Machine Vision Association, Sep. 2008, pp. 275:1–10. [Online]. Available: https://hal.inria.fr/inria-00514853

[50] J. Sánchez Pérez, E. Meinhardt-Llopis, and G. Facciolo, "TV-L1 Optical Flow Estimation," *Image Processing On Line*, vol. 3, pp. 137–150, 2013, https://doi.org/10.5201/ipol.2013.26.

[51] D. Sun, X. Yang, M. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," *CoRR*, vol. abs/1709.02371, 2017. [Online]. Available: http://arxiv.org/abs/1709.02371

[52] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks: Towards good practices for deep action recognition," *CoRR*, vol. abs/1608.00859, 2016. [Online]. Available: http://arxiv.org/abs/1608.00859

[53] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[54] M. S. Sarfraz, V. Sharma, and R. Stiefelhagen, "Efficient Parameter-free Clustering Using First Neighbor Relations," Feb. 2019. [Online]. Available: https://arxiv.org/abs/1902.11266v1

[55] H. Kuehne, A. Arslan, and T. Serre, "The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 780–787, iSSN: 1063-6919.

[56] J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien, "Unsupervised Learning from Narrated Instruction Videos," in *CVPR2016 - 29th IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, United States, Jun. 2016. [Online]. Available: https://hal.inria.fr/hal-01171193

[57] H. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, 05 2012.

[58] C.-Y. Chang, D.-A. Huang, Y. Sui, L. Fei-Fei, and J. C. Niebles, "D3TW: Discriminative Differentiable Dynamic Time Warping for Weakly Supervised Action Alignment and Segmentation," *arXiv:1901.02598 [cs]*, Apr. 2019, arXiv: 1901.02598. [Online]. Available: http://arxiv.org/abs/1901.02598

[59] S. N. Aakur and S. Sarkar, "A Perceptual Prediction Framework for Self Supervised Event Segmentation," *arXiv:1811.04869 [cs]*, Apr. 2019, arXiv: 1811.04869. [Online]. Available: http://arxiv.org/abs/1811.04869

[60] L. Ding and C. Xu, "Weakly-Supervised Action Segmentation with Iterative Soft Boundary Assignment," *arXiv:1803.10699 [cs]*, Mar. 2018, arXiv: 1803.10699. [Online]. Available: http://arxiv.org/abs/1803.10699

[61] M. Fayyaz and J. Gall, "SCT: Set Constrained Temporal Transformer for Set Supervised Action Segmentation," *arXiv:2003.14266 [cs]*, Mar. 2020, arXiv: 2003.14266. [Online]. Available: http://arxiv.org/abs/2003.14266

[62] H. Kuehne, A. Richard, and J. Gall, "Weakly supervised learning of actions from transcripts," *arXiv:1610.02237 [cs]*, Jun. 2017, arXiv: 1610.02237. [Online]. Available: http://arxiv.org/abs/1610.02237

[63] J. Kruskal and M. Liberman, "The symmetric time-warping problem: From continuous to discrete," *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Jan. 1983.

[64] X. Wang, R. B. Girshick, A. Gupta, and K. He, "Non-local neural networks," *CoRR*, vol. abs/1711.07971, 2017. [Online]. Available: http://arxiv.org/abs/1711.07971