Universitat Politècnica de Catalunya

Facultat de Matemàtiques i Estadística

Master in Advanced Mathematics and Mathematical Engineering

Master's thesis

Learning graph-based event representations for unconstrained video segmentation

Biel Tura Vecino

June, 2021

Abstract

Recent research has shown that, in particular domains, unsupervised learning algorithms are achieving on par, or even better performance than fully supervised algorithms, avoiding the need of human labelled data. The division of a video into events has been an active research topic through unsupervised algorithms, exploiting relations in the video itself for a temporal segmentation task. In particular, self-supervised learning has shown to be very useful learning video representations without any annotations assigned to it. This thesis proposes a self-supervised method for learning event representations of unconstrained complex activity videos. These are sequences of images with high temporal resolution and with very small visual variance between events, with a clear semantic differentiation for humans. The assumption underlying the proposed model is that a video can be represented by a graph that encodes both semantic and temporal similarity between events. Our method follows two steps: first, meaningful initial features are extracted by a spatio-temporal backbone neural network trained on a self-supervised contrastive task. Then, starting with this initial embedding, low-dimensional graph-based event representation features are iteratively learned jointly with its underlying graph structure. The main contribution in this work is to learn a function parameterized by a graph neural network that learns graph-based event feature representations by exploiting the semantic and temporal relatedness through a fully end-to-end self-supervised trainable approach. Experiments were performed in the challenging *Breakfast Action Dataset* and we show that the proposed approach leads to an effective low-dimensional feature representation of the input data, suitable for the downstream task of event segmentation. Moreover, we show that the presented method, followed by a downstream clustering task, achieves on par state-of-the-art metrics on video segmentation of complex activity videos.

Keywords

representation learning, graph embedding, video segmentation, event representations

Contents

1 Introduction **3**

2 Related Work **5**

2.1 Self-supervised representation learning 5

2.2 Video event segmentation 7

2.3 Clustering for unsupervised video segmentation 8

3 Proposed method **10**

3.1 Learning the Feature encoder 12

3.1.1 Spatiotemporal Contrastive Event Representation Learning 12

3.2 Graph-based event representation learning 14

3.2.1 Graph Convolutional Neural Networks 15

3.2.2 Graph and embedding initialization 15

3.2.3 Graph embedding update 16

3.2.4 Graph structure update 19

3.3 Downstream task 19

3.3.1 Spectral Clustering 20

4 Results **21**

4.1 Experimental setup 21

4.1.1 Dataset 21

4.1.2 Evaluation metrics 22

4.2 Experimental evaluation 23

4.2.1 Feature Encoder 23

4.2.2 Self-supervised graph-based representation learning 24

4.2.3 Experimental results 27

5 Conclusions and Future Work **29**

1. Introduction

Unconstrained videos capturing real-world scenarios are recently receiving lots of attention due to their increased availability and potential in various situations such as assistive applications or video-surveillance. They are usually long, untrimmed and contain a variety of events which can be effortlessly divided by a human observer to interpret it. This division of a sequence of images is a key part in the process of understanding and contextualizing a video. It is this process of organizing unstructured videos into semantically homogeneous units that is an active challenge in the computer vision field, not only for video interpretation but also to video browsing, indexing and summarization.

From cognitive psychology [1], an event is defined as a segment of time at a given location that is perceived to have a beginning and an end. Therefore, event segmentation is the process of identifying these boundaries. Evidently, in a video, these events are a set of adjacent frames in which the segmentation of every event in it will naturally lead to a time-related segmentation of the image sequence. Moreover, we can identify these events and label each temporal partition of the video sequence.

The process of manual annotating frames describing the events within the video is both time-consuming and costly. Such it is, there is a tremendous need for automatic video segmentation methods able to provide consistent temporal boundaries and accurate partition labelling. Temporal segmentation of long videos has largely been tackled through supervised learning, requiring large amounts of annotated training data. These approaches suffer from domain adaptation problems, being not the optimal techniques for a more general segmentation approach. On the other hand, self-supervised approaches have shown to be an effective way to learn embedded video representations suitable for the event segmentation task without any or very few human labeled data required.

Encouraged by previous developments [2] and motivated by Neuroscientific findings [3], self-supervised representations can be learnt for the task of event segmentation by learning its underlying graph structure, exploiting temporal and semantic priors on the learned representations themselves. This thesis will present a novel end-to-end trainable approach for learning, through a self-supervised manner, useful event representations with their underlying graph structure based on semantic (visual similarity) and temporal constrains.

Prior techniques based on a similar approach have shown remarkable results in image sequences where events are quite differentiated for its spatial characteristics (different scenes), or in sequences with a few frames per second (fps) or per minute (fpm) recorded. These have larger appearance changes between adjacent frames, and event a higher differentiation between events. However, our approach will target a set of complex activities, understanding them as a variety of untrimmed high temporal resolution videos ($\geq$ 15 fps) in which events have very small visual variance, that is events not differentiated by a frame or scenario change but by its semantic meaning in the video. An illustration of what we aim to achieve through the presented approach, as well as demonstrate the complexity of event differentiation is shown in Figure 1.

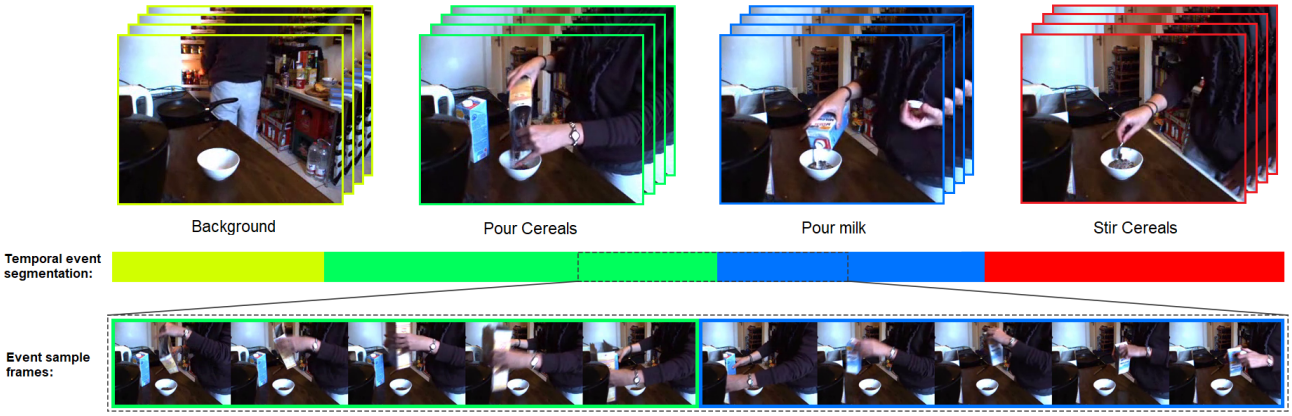


Figure 1: Example of a temporal event segmentation of one of the videos from the *Breakfast Action Dataset* (*P35_Cereals*). Frames corresponding to same event are grouped together in the temporal event segmentation. A zoom on the event sample frames captures the small visual variance that magnifies the difficulty of the event segmentation task of untrimmed complex activity videos.

The proposed method is performed through the following procedure: first learning to extract meaningful event representations from a video by modifying current state-of-the-art frameworks for video learning representations [4] in a self-supervised contrastive setup [5]. We will show that meaningful spatio-temporal event representations can be learned with a simple adjustment in the self-supervised training process. Then, by adopting different self-supervised training techniques from the image classification domain [6, 7], we propose a novel approach to improve the learned representations for the task of event segmentation. We propose to exploit temporal and semantic relatedness between the extracted features through a self-supervised graph-based representation learning approach. The presented novel technique will learn low-dimensional features and their temporal and semantic relation in an underlying graph structure. Finally, we will evaluate the learned representations for an unsupervised downstream task of action segmentation on the challenging *Breakfast Action Dataset* [8].

The main contributions of this thesis to the event representation domain are: (i) rather than learning frame-level features of video sequences, we show that learning spatio-temporal representations of a small set of frames is useful for the task of event representation learning. (ii) we propose a novel approach to improve learned spatio-temporal event representations by both exploiting semantic and temporal relations between them and learning their underlying graph structure. We demonstrate that the learnt graph is suitable for the task of temporal event segmentation of complex activity videos, on par with fully, weakly and self-supervised learning state-of-the-art approaches, as well as unsupervised clustering algorithms presented for this specific task.

2. Related Work

This thesis aims to learn meaningful representations of a sequence of images for the task of unconstrained video segmentation in a fully self-supervised manner. The proposed approach lies in the representation learning field, and the main contribution of our work is to learn meaningful features through an end-to-end trainable method, designed specifically for the event representation learning task. To evaluate the effectiveness of the learnt representations, we will use a clustering algorithm on them to perform the task of event segmentation in complex activity videos and thus compare our approach to supervised, weakly-supervised and self-supervised approaches evaluated in the same challenging task.

2.1 Self-supervised representation learning

Supervised learning algorithms reefer to those which requires input data to be labelled to the expected output. These algorithms are trained to learn the relation between these input-output pairs. While supervised algorithms have shown remarkable performance in the computer vision domain, these require a huge amount of manual data labeling which is slow, expensive and error prone. On the other hand, the term unsupervised learning has been widely used to describe techniques that do not use human-annotated data in the learning process. Self-supervised learning, a subset of unsupervised learning algorithms, aims to learn supervisory signals generated from exploiting relations in the input data itself, without the need of any human-annotated labels. Self-supervised learning is typically divided in two tasks presented in Figure 2: the pretext task and the downstream or target task. The first one is the process by which the training aims to extract relevant characteristics of the given statistics, thus learning a representation of the input data understood as feature vectors (representation learning). Then, these representations are typically used to solve a target problem through a downstream task. Self-supervised has shown to be particularly useful when labeled data is very scarce. Learning representations of unlabelled data through a self-supervised pretext task and then use them for a downstream task, sometimes in a weakly supervised fashion, has shown remarkable results against the same end-to-end supervised approach.

Self-supervised image representation learning In the image field, early works learned visual representations from unlabeled images via auto-encoding methods [9] that leverage contexts, thus suitable for image inpainting or color filling tasks. Latest research on the self-supervised image representation learning field have shown that a huge variety of the pretext tasks can be integrated into a contrastive learning framework [5, 10]. Contrastive self-supervised learning approaches exploits the relative consistency between the representations of an image and its augmented view. Idealizing the contrastive framework, different downstream tasks have been evaluated on the learned image representations such as image classification [6] by exploiting semantic relatedness between learned features, or image segmentation [11] through a clustering procedure of the learned self-supervised image representations.

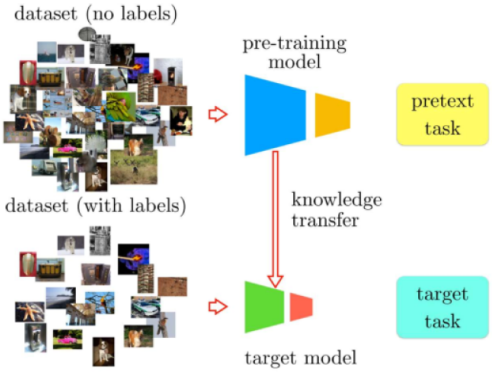


Figure 2: Self supervised pipeline: a pretext task is learned with non-labeled input data for a representation learning approach. This knowledge is then transferred to perform a downstream or target task in a weakly-supervised fashion, thus fine-tuning the model with the known ground truth labeled data.

Self-supervised video representation learning It is natural to extend representation learning approaches in images to the video domain by exploiting the temporal dimension. Several works have investigated representation learning on videos through different approaches. For instance, exploiting the pixel-wise geometry through the temporal dimension as an optical flow field and disparity maps [12] was proposed for learning video representations as an alternative to simple frame representation learning. This approach however, required a few labeled data to obtain decent results on the event segmentation task. Extending the idea of optical flow, learning video representation by motion prediction and appearance statics [13] was the first step of introducing spatio-temporal features to the video representation learning domain through 3-dimensional convolutional neural networks. The work concluded that the optical flow together with a set of frames can outperform previous works on video representations. However, the learned representations where not suitable for naive event segmentation and the resulting architecture was proposed as a starting point to be fine tuned for a downstream task. On the other hand, new methods tackling the task of learning spatio-temporal embeddings suitable for action recognition [14] are based on learning dense encoding of spatio-temporal blocks by recurrently predicting future representations. More recently, the current state of the art model for learning visual representations of videos [4] has managed to capture spatio-temporal characteristics in the learned representations in a much simpler contrastive video representation learning approach through very strong temporal consistent data augmentation techniques.

Self-supervised event representation learning Extending the idea of representation learning, the task of learning event representations in a self-supervised manner have also been studied. Early works showed a learning approach in image representations suitable for the event segmentation task in a self-supervised manner [15] through a pretext task trained to predict representations of temporal neighboring images. More recent works [2], aims to learn new event embeddings from frame-level representations jointly with their underlying graph structure that encodes both semantic and temporal similarity. This approach, named Dynamic Graph Embedding (DGE), learns iteratively from the input video through a two step approach: updating the underlying graph structure of the frame representations by enforcing semantic and temporal

relatedness and then updating the learned representations taking into account the graph structure. DGE has shown to be particularly excellent learning of low-dimensional embedded event representations suitable for the downstream task of event segmentation for low temporal resolution egocentric videos. Moreover, DGE learns through an iterative process from the extracted video frame-level features, thus it does not require any training data and does not suffer domain adaptation issues. Either methods learn event representations by improving image representations extracted at a frame-level.

We are, to the best of our knowledge, the first to extend latest works on contrastive video learning to the event representation domain, suitable for unconstrained video segmentation. Moreover, we propose to follow a similar approach to DGE by learning spatio-temporal event representations and their underlying graph structure to incorporate semantic and temporal meaning to the learned embeddings. Our method is the first to target the event representation learning task for complex activity videos and therefore we unravel further research to exploit the contrastive setup on video representation for event segmentation.

2.2 Video event segmentation

Video and image representations learned through a pretext task have shown good results in the event segmentation problem by using the learned knowledge and then training the downstream task in a weakly-supervised fashion. However, self-supervised approaches for the task of event segmentation has not been studied extensively, and even fewer research has been carried for the same task in complex video activities, which is the challenge our work focuses on. As of our knowledge, we are the first to propose an approach on learning spatio-temporal event representations on complex activity videos suitable for the downstream task of event segmentation in a fully end-to-end self-supervised manner. In this section we will show recent supervised, weakly supervised and self-supervised approaches for the event segmentation task.

Fully supervised approaches Event Segmentation has been traditionally tackled as a supervised learning problem were frame-labelled data is provided by human-annotations. There have been different approaches to supervised event segmentation. The firsts attempts on temporal segmentation of a video relied on Markov models [16] on top of frame-level classifiers. Later, new models were proposed to exploit the temporal information in a video sequence. For instance, techniques based on time-series methods built upon temporal convolutional neural networks [17] were used to capture long range temporal frame-level patterns. Another approach to capture temporal dependencies is the usage of recurrent neural networks [18]. More recently and due to the increase of new model architectures for video understanding, spatio-temporal convolutional layers [19] have shown decent results on capturing temporal dependencies while being easier to train than previous methods. Another recent approach is the usage of graph convolutional neural networks [20] to model temporal relation between representations. This graph-based approach learns a semantic and temporal graph structure used to map frame representations to the ground truth labels. Tackling the event segmentation problem with supervised learning leads to good results overall. However all this techniques require a large amount of labeled data and heavily rely on the quality of the annotations.

Weakly-supervised approaches To alleviate the need of large annotated datasets, weakly supervised techniques have also been investigated for the task of event segmentation. These usually rely in knowing the order of the events (with their respective labels) and a few annotated frames. Some weakly-supervised approaches have two-stage training process where pseudo-labels are generated using transcripts (a list of the order that the actions occur in the video) and then learn the segmentation models [21]. Other approaches lie in the incorporation of visual similarity to explicitly capture dependencies between consecutive frames given the order of the events [22]. More recently and using the mentioned transcripts, building a fully connected graph whose path represent a valid or invalid frame labelling [23] and then tracking the similarity between the learned path and the ground truth transcript have also lead to results on par with fully supervised techniques.

Self-supervised learning approaches Research on self-supervised learning for event segmentation has not been explored to the same extent as supervised or weakly-supervised techniques. Self-supervised approaches assume knowledge only of the number of events presents in the input video. The primary approach is to use a clustering algorithm to segment the self-supervised frame learned representations using discriminant features. Models proposed following the clustering procedure incorporate a temporal consistency term into the segmentation by using Low Short-Term Memory units (LSTMs) [24]. Other more complex approaches use Generalized Mallows Models [25] to discriminatively learn the appearance of sub-activities from the videos' visual features and then model the temporal structure of the sub-activities in it. More recent approaches aims to fine tune a frame-level representation model with a LSTM using future frame prediction, idealizing a self-supervised prediction error signal [26]. The model assigns the segmentation boundaries purely based on the prediction error of next frame, resulting in a temporal segmentation of the video. While this last work achieves comparable results to both supervised and weakly-supervised techniques, LSTMs are very difficult to train as they require a huge amount of memory bandwidth (even more for large videos) and they are often not suitable for transfer the learned knowledge into a different task.

2.3 Clustering for unsupervised video segmentation

Clustering methods lie under unsupervised algorithms as they do not require any extra information than the input data itself. Understanding representations as data points in an embedded space, clustering approaches generate a partition of the data based on a specific similarity metric. Every clustering algorithm is such that given N points in an embedded space, the objective is to group these observations into K unique groups known as clusters ($K < N$). Temporal video segmentation is inherently a clustering problem in which every cluster is the set of frames corresponding to an event. Given a good set of learned representations, a clustering algorithm can be used as a downstream task to achieve an end-to-end self-supervised video temporal segmentation approach. Unless the clustering approach takes into account temporal and semantic factors of the representations, clustering algorithms rely on the quality of the representations to be clustered to achieve good metrics in a downstream event segmentation task.

Standard clustering algorithms, such as K-means [27], are based on the euclidean distance and, as a consequence, assume that a cluster reassembles a spherical shape around the cluster center. Given the nature of event segmentation, using such techniques is not optimal because there exists a relevant temporal relation between the representations of a video that is not exploited in the K-means algorithm. Standard clustering algorithms also require parameters such as the number of clusters to split the data into, which may or may not be known in an unsupervised problem.

More recently, a new efficient parameter-free Clustering technique that uses first neighbor relations [28] (FINCH) has been presented lately. This clustering approach belongs to the family of hierarchical agglomerative methods. The popularity of hierarchical clustering stems from the fact that it produces a clustering tree that provides meaningful ways to interpret data at different levels of granularity.

Temporal clustering Despite the presented approaches of clustering algorithms, none of those were specifically designed for the problem of event segmentation, but rather rely on the structure of the learned representations being suitable for the downstream clustering task. There has been some works introducing novel clustering approaches for video segmentation. Early approaches [29] proposed a clustering algorithm for motion segmentation based on temporal subspace and spectral clustering. Subspace clustering [30] is an extension of traditional clustering that seeks to find clusters in different subspaces within a dataset. Spectral Clustering [31] make use of the spectrum of a similarity matrix of the given representations. The relevant metric behind the Spectral Clustering is the similarity between points given by the pre-computed affinity matrix (similarity relation between pairs of points). This method is ideal when the data points go along an affinity matrix that resembles the similarity between each representation.

On the other hand, semantic clustering has also been explored [32] by exploiting the temporal coherence and contextual semantic attributes of photo-streams. Very recently however, a new clustering approach, similar in spirit to FINCH, has appeared specifically for the task of video event segmentation. This technique, named Temporally-Weighted hierarchical clustering (TW-FINCH) [33], was introduced as a novel approach for clustering unsupervised frame-wise representations for the downstream task of event segmentation. Moreover, this clustering approach achieved state-of-the-art results on the event segmentation problem of complex video activities without any additional training on the learned features. This clustering technique proposes to exploit spatio-temporal characteristics of the learned representations by linking frames in the same cluster based on their feature space proximity (semantic similarity) and their respective positions in time (temporal similarity).

3. Proposed method

We propose to understand events in a video as community structures on a graph. These communities consists of data points grouped by its semantic and temporal relatedness. Motivated by neuroscientific findings [3], a neural representations of events arise from temporal community structures and suggests that event representations that share context between them are grouped together in the same community. Figure 3 illustrates this idea where data points belonging to the same event are grouped together. We aim to learn representations such that they are grouped with semantic and temporal awareness in which applying an unsupervised clustering algorithm leads to a temporal video segmentation of the events in it.

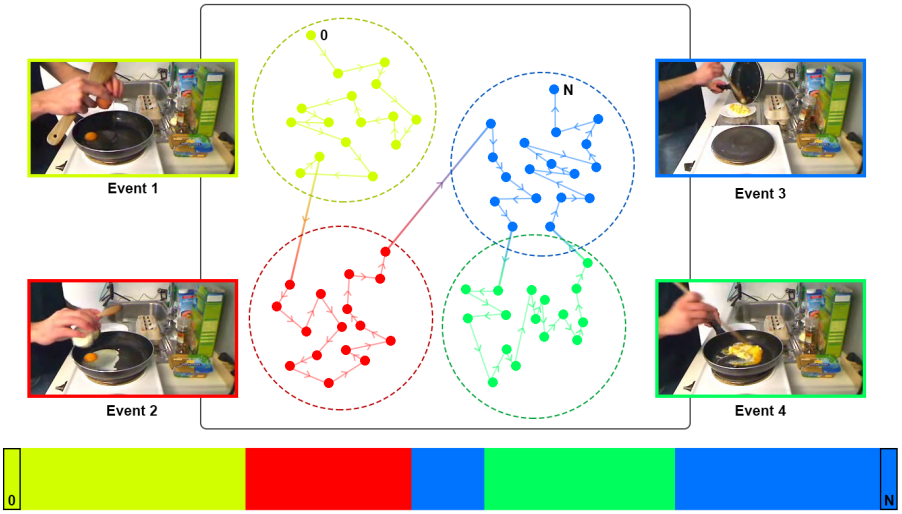


Figure 3: Learned event representations of clip sequences (color points) can be organized in a graph such that each community structure on it corresponds to a specific context. Distance between points correspond to semantic and temporal relatedness between them. Temporal transition between the points are shown by the arrow connections, such that the first embedding (0) is the start of the sequence and the last learned feature (N) is the end of it. Tracking where each representation lies in a temporal order ($0 \rightarrow N$) leads to the presented event segmentation plot shown.

The following sections present the basis of the proposed method for learning graph-based event representations through a completely end-to-end trainable self-supervised approach. Our pipeline is divided in four different parts shown in Figure 4: first, rather than extracting frame-level features of the input video, we propose to extract representations of a small set of frames together, exploiting the temporal characteristics of an image sequence. Next, we present an encoding method that learns meaningful features by training a backbone model (feature encoder) in a contrastive setup [5, 4]. Then, our main contribution in this thesis, we show how we can improve these initial embedding representations by exploiting semantic and temporal priors through a Graph Convolutional Neural Network (GCN) [34] as well as learning an affinity matrix representing the connections between these data points in the new representational space. Finally, we will show that the graph-based learned representations are suitable for the task of event segmentation through an unsupervised downstream task.

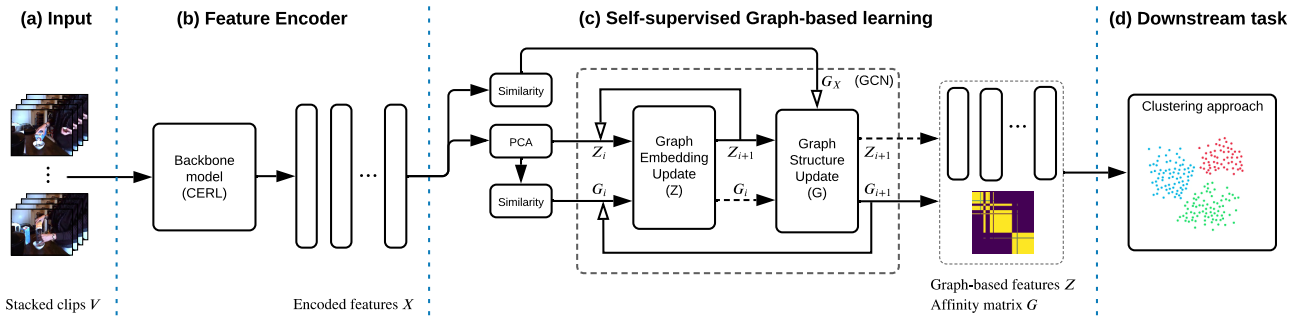


Figure 4: Pipeline of the proposed method for learning graph-based event representations. **Input**: Set of N clips stacked together as $V \in \mathbb{R}^{N\times(T\times W\times H)}$. **Feature encoder**: maps the input clips to an encoded feature vector matrix $X \in \mathbb{R}^{N\times n}$ through a model trained in a self-supervised contrastive setup (CERL). **Self-supervised Graph-based learning**: Extract from input features its graph structure $G_X \in \mathbb{R}^{N\times N}$, lower the dimensionality of X through applying PCA and then jointly learn a lower dimensional graph-based features $Z \in \mathbb{R}^{N\times d}$ with both semantic and temporal relatedness, and their underlying graph structure in form of an affinity matrix $G \in \mathbb{R}^{N\times N}$ via a graph convolutional neural network (GCN). **Downstream task**: Evaluate the learned features through an unsupervised clustering algorithm for a segmentation task.

Our approach is inspired by the Dynamic Graph Embedding (DGE) technique [2]. In it, temporal and semantic relations are imposed in the learned underlying graph. To foster temporal similarity, the affinity matrix is updated by local averaging the temporal neighbors edge weights. To reinforce the semantic similarity, a clustering algorithm is applied on the learned features to strengthen the edge weight between nodes belonging to the same cluster. Our approach, similar in spirit, aims to train a graph convolutional neural network that minimizes a custom loss which models temporal and semantic relatedness. Therefore, we propose an end-to-end learning approach, avoiding the need of non-trainable extra processing in the learned graph, resulting in a faster trainable method to exploit the same semantic and temporal priors.

Introducing the notation that will be used in the following sections, a single video is understood as 3-dimensional matrix of N frames with width W and height H (omitting the RGB channel dimension). A clip $v \in \mathbb{R}^{T\times W\times H}$ is a set of temporal adjacent frames in a video where $T < N$. Stacking all the clips together in a video with an overlap of $T-1$ and a padding technique, we can define the input we will use in our approach as the matrix $V = \{v_1, \cdots, v_N\} \in \mathbb{R}^{N\times(T\times W\times H)}$ being a set of clips with a length equal to the number of frames. This input shape is illustrated in Figure 4 (a).

We aim to find meaningful event representations (feature vectors) of each of these clips, that is given the encoded features $X \in \mathbb{R}^{N\times n}$ we want to learn new representations in a low-dimensional space, $Z \in \mathbb{R}^{N\times d}$, $(d \ll n)$ for the task of event segmentation. We will exploit two priors to do it: first, same events are semantically similar, thus two clips with a similar representation should belong to the same event. Events in a video are defined as a continuous set of clips, therefore temporal adjacent clips are likely to be of the same event as well. Consequently, we assume that these representations are organized by a graph in an underlying low-dimensional space. The number of nodes N in it is the amount of learned representations, and each node $i \in \{1, \cdots, N\}$ has its corresponding feature vector $z_i \in Z$. Therefore, we aim to find the affinity matrix G of this graph with feature representations as node embeddings such that the similarity between any pair of nodes of the graph reflects both semantic and temporal relatedness.

3.1 Learning the Feature encoder

In the first part of our representation learning approach, a feature encoder is learned in a self-supervised fashion by a function ϕ_θ, parameterized by a Neural Network with weights θ, that maps the input stacked clips V into its feature representation X as shown in Figure 4 (b).

$$X = \phi_\theta(V) \tag{1}$$

Representation learning has shown remarkable results for image problems. Models used are designed to exploit the spatial information of the input through 2-dimensional (2D) convolutional layers. The current state-of-the art models for these tasks are based on Residual Neural Networks (ResNets) [35]. In our problem, we are dealing with a sequence of images. The temporal dimension of videos gives rise to key differences between them and, therefore, it is natural to engineer self-supervised learning signals along the 3 dimensions of the video. Following the current state-of-the-art architecture for learning video representations [4], we will use as our backbone model a video encoder based on 3-dimensional Residual Neural Networks (3D-ResNets). This model expands the original 2D convolution layers in the residual neural network architecture to 3D in order to capture spatio-temporal characteristics of an image sequence.

The training of the backbone model will follow a self-supervised contrastive setup: the neural network will be trained to minimize the distance between pairs of input clips that are known to be representing the same event. This approach is usually tackled by taking the same clip and applying a strong data augmentation on it, resulting in two different clips that are known to represent the same input. However, we will present a new contrastive training framework for event representation learning (CERL), exploiting the characteristics that, two adjacent clips are likely to be of the same event, thus making the network learn temporal features of the input clip within the same event.

3.1.1 Spatiotemporal Contrastive Event Representation Learning

The first task to be solved on the event learning representation is inspired from the work on self-supervised learning video representations (CVRL) [4]. The idea behind it is to exploit the video's spatio-temporal characteristics in order to learn a good representations of each clip through a backbone neural network.

Our self-supervised approach models a Contrastive Event Representation Learning (CERL) task. This task will be trained on a Simple Framework for Contrastive Learning of Visual Representations (SimCLR) [5]. This framework learns representations by maximizing agreement between differently augmented views of the same data example $v_i \in V$ via a contrastive loss. In our case, we propose to exploit the fact that not only the same augmented input data correspond to a similar representation, but it's temporal adjacent input clip, v_{i+1}, is very likely to be of the same event. Therefore, we define a positive pair as two independently augmented temporal adjacent clips: $\tilde{v}_i, \tilde{v}_{i+1}$. An illustration of this approach is shown in Figure 5.

As suggested by the SimCLR framework, a projection head function ψ_θ, parameterized by a multi-layer

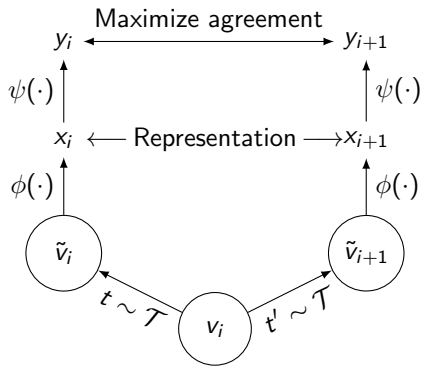


Figure 5: The modified SimCLR framework diagram for contrastive event representation learning: Instead of augmenting two times a randomly sampled input clip v_i and generate a positive pair, we propose to augment this clip once and its temporal adjacent v_{i+1} through independent transformations t and t' respectively.

perceptron network, is concatenated to the backbone model ϕ_θ solely in the training stage of the network. As stated in the original work [5], a nonlinear projection head improves the representation quality of the layer before it. This head maps the embedded space of the output representations $X \in \mathbb{R}^{N \times n}$ to a lower dimension projected representations $Y \in \mathbb{R}^{N \times m}$ where the contrastive loss is applied ($n > m$):

$$Y = \psi_\theta(X) = \psi_\theta(\phi_\theta(V)) \tag{2}$$

As mentioned earlier, the main difference between our approach and the recent work on video representations [4] is in the training process sample pick: a randomly chosen clip v_i is taken with its temporal adjacent counterpart v_{i+1}. Then, augmented examples $(\tilde{v}_i, \tilde{v}_{i+1})$ are obtained through independent augmentation operators t, t' sampled from the same family of augmentations $\mathcal{T}$. The contrastive prediction task is defined to minimize the distance in the embedded space between this pair of clips. Evaluating all the clips in V, this procedure derives to $2(N-1)$ data points in the projected space ($N-1$ (y_i, y_{i+1}) pairs of features). In the end, for every sample v_i and its network output y_i, there is one positive input sample v_{i+1} and output vector y_{i+1} and all the others are negative examples. The modified SimCLR loss function to be minimized for a every positive pair of feature vector is then defined as:

$$\mathcal{L}_{\text{CLR}} = -\sum_{i=1}^{N-1} \log \frac{\exp(\text{sim}(y_i, y_{i+1})/\tau)}{\sum_{j=1}^{2N} \mathbb{1}_{[j \neq i]} \exp(\text{sim}(y_i, y_j)/\tau)} \tag{3}$$

where $\mathbb{1}_{[j \neq i]} \in \{0, 1\}$ is and indicator function evaluation to 1 if $k \neq i$, τ denotes a temperature parameter and the operator $\text{sim}(\cdot, \cdot)$ is the cosine similarity between the two latent representations.

The presented loss function $\mathcal{L}_{\text{CLR}}$ will impose the network to learn optimal weights θ in both the head ψ_θ and the backbone ϕ_θ model. This trained weights will maximize the agreement between positive pairs of temporal adjacent projected feature vectors Y and thus learn a good representation X of the input V in the embedded space based on both spatial and temporal characteristics within the clip itself.

3.2 Graph-based event representation learning

Features extracted by the backbone model lack temporal and semantic awareness within the video itself. To solve this, we propose to learn new event representations exploiting the temporal and semantic priors encoded in a fully connected similarity graph, where every node represents each feature and the edge weights are the semantic and temporal similarity between each one of them. The presented approach does not rely on a training set, but instead learns solely from the input features. The following sections will describe in detail every step presented in the self-supervised graph-based event representation learning algorithm shown in Figure 4 (c) and detailed in the pseudo-code of Algorithm 1. The key novelty of the proposed algorithm is to learn jointly low-dimensional event representations Z and the graph G of Z that encodes the temporal and semantic relation between each representation. The presented approach follows an iterative procedure of two steps: the first one is to update the neural network parameters to output new representations with temporal and semantic awareness considering their affinity matrix G. Then, update this affinity matrix according to the new learned representations.

More formally, the graph-based event representation learning follows the learned features from the feature encoder, $X \in \mathbb{R}^{N \times n}$. These are now the input of the presented graph convolutional neural network Φ_η with trainable weights η. Therefore, we aim to find new node embeddings $Z \in \mathbb{R}^{N \times d}$ in a low-dimensional space ($n \ll d$) jointly with an affinity matrix $G \in \mathbb{R}^{N \times N}$ representing the weighted underlying graph of the learned event representations Z.

ALGORITHM 1: Graph-based event representation learning

Input : $X \in \mathbb{R}^{N \times n}$ — *initial feature matrix*

Output: $Z \in \mathbb{R}^{N \times d}$, $G \in \mathbb{R}^{N \times N}$ — *graph embedded feature and affinity matrix*

Graph and embedding initialization (Section 3.2.2)

$G_X = S_{l_x}(X) \quad \in \mathbb{R}^{N \times N}$ — *initialize graph in original space*

$\tilde{X} = \text{PCA}_d(X) \quad \in \mathbb{R}^{N \times d}$ — *reduce features dimensionality*

$\eta = \text{argmin}_\eta \, \mathcal{L}_G(S_{l_z}(\Phi_\eta(\tilde{X}, G_X)), G_X)$ — *initialize GCN weights*

$Z_0 \leftarrow \Phi_\eta(\tilde{X}, G_X) \quad \in \mathbb{R}^{N \times d}$ — *initialize embedding features given init graph*

$G_0 \leftarrow S_{l_z}(Z_0) \quad \in \mathbb{R}^{N \times N}$ — *initialize graph in embedding space*

for $i \leftarrow 0$ **to** T **do**

— **Graph embedding update** (Section 3.2.3):
Update networks Ψ, Φ parameters η to minimize loss that models feature's semantic and temporal priors:

$\mathcal{L} = (1-\gamma)\, \mathcal{L}_\text{S}(\Psi_\eta(\Phi_\eta(Z_i, G_i))) + \gamma\, \mathcal{L}_\text{T}(\Psi_\eta(\Phi_\eta(Z_i, G_i)))$

$Z_{i+1} \leftarrow \Phi_\eta(Z_i, G_i)$ — *update embedding features given graph G_i*

— **Graph structure update** (Section 3.2.4):
Update network Φ parameters η to minimize structural loss that maintains original graph G_X properties:

$\mathcal{L} = (1-\alpha)\, \mathcal{L}_\text{G}(S_{l_z}(\Phi_\eta(Z_{i+1}, G_i)), G_i) + \alpha\, \mathcal{L}_\text{G}(S_{l_z}(\Phi_\eta(Z_{i+1}, G_i)), G_X)$

$G_{i+1} \leftarrow S_{l_z}((\Phi_\eta(Z_{i+1}, G_i)))$ — *update graph structure given learned embeddings Z_{i+1}*

end

3.2.1 Graph Convolutional Neural Networks

In order to improved learned representations Z_{i+1} from the previous low-dimensional features Z_i based on the graph structure, we propose to use a Graph Convolutional Neural Network (GCN) following an autoencoder architecture. This means that the input and output features shape match in size ($Z_i \rightarrow Z_{i+1}$).

In general, graph convolutional neural networks learn their model parameters by inspecting the neighboring nodes given by the affinity matrix G of the input features Z. The adjacency matrix represents the edge weight between each node. By inserting G in the forward pass, graph convolutional neural networks learn the feature representation subject to the nodes connectivity given by the affinity matrix.

A graph convolutional neural network is defined as a function $\Phi_\eta(Z, G)$ with weights η based on a deep architecture of L stacked layers with hidden output states $H^{(l)}, l \in \{0, \cdots, L\}$ that follows the layer-wise propagation rule:

$$H^{(l+1)} = \sigma(D^{-\frac{1}{2}} G D^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{4}$$

where D is the degree matrix of the affinity matrix G computed following the Equation 18 and $W^{(l)} \in \eta$ is the layer-specific trainable weight matrix. The activation function $\sigma(\cdot) = max(0, \cdot)$ follows the Rectified Linear unit (ReLu) function. $H^{(l)}$ is the matrix of hidden activations in the lth layer, thus $H^{(0)} = Z_i$ is the input node features matrix and $H^{(L)} = Z_{i+1}$ is the learned output node features matrix.

$$Z_{i+1} = \Phi_\eta(Z_i, G) = \sigma(D^{-\frac{1}{2}} G D^{-\frac{1}{2}} \sigma(\cdots \sigma(D^{-\frac{1}{2}} G D^{-\frac{1}{2}} Z_i W^{(0)}) \cdots) W^{(L)}) \tag{5}$$

3.2.2 Graph and embedding initialization

The graph-based event representation learning algorithm initialization aims to obtain the first embedding features Z_0 and their affinity matrix G_0 both from the encoded features X, as well as initializing the graph neural network's parameters η. The first step is to obtain the affinity matrix G_X of the initial features X. The underlying affinity matrix G_X of a set of features X is given by the similarity function $S_{l_x}(\cdot)$ such that $G_{X(i,j)}$ encodes the similarity between any pair of nodes x_i, x_j. The similarity function, is defined element-wise as the pairwise similarity:

$$(S_{l_x}(X))_{(i,j)} = exp\left(-\frac{1 - \text{cdist}(x_i, x_j)}{l_x}\right) \tag{6}$$

where $\text{cdist}(\cdot, \cdot)$ is the cosine distance and l_x the filtering parameter of the exponential function, which directly depends on the dimensionality of the features X.

Then, we obtain the initial low-dimensional input embedding $\tilde{X}$ by applying PCA on the input X to keep the d major principal components. With that, we initialize the graph convolutional neural network Φ_η by optimizing the weights η such that the cross-entropy loss $\mathcal{L}_G$ between the affinity matrix given by

the output of the network and the original affinity matrix G_X is minimized. This approach results into low-dimensional representations Z_0 that resemble the starting initial features given by the feature encoder and its affinity matrix G_0 similar to the original high-dimensional features affinity matrix G_X.

3.2.3 Graph embedding update

The main goal of the graph convolutional neural network Φ_η is to learn new embedding representations Z that contain both semantic meaning and temporal awareness in order to learn suitable representations for a downstream event segmentation task.

To tackle this problem, we will define two loss functions. The first one is the Semantic Clustering Loss $\mathcal{L}_S$ and will capture the semantic dependencies between the representations. The second one is the Temporal Clustering Loss $\mathcal{L}_T$ which will enforce a temporal relationship between the learned representations.

Following the contrastive scheme of the backbone model, a new head Ψ_η is added to the graph convolutional neural network Φ_η. This head, parameterized by a multi-layer perceptron with weights η, maps the output features of the graph convolutional neural network Z to a set of vectors $Y \in \mathbb{R}^{N \times C}$ whose length is the number of desired clusters $\mathcal{C}$. On a self-supervised approach, the number of clusters C is generally unknown. However, following the strategy of evaluation by current self-supervised state-of-the-art approaches, we choose its length equal to the number of ground-truth clusters. In practice, it should be possible to obtain a rough estimate of the amount of clusters based on prior domain knowledge.

$$Y = \Psi_\eta(Z_{i+1}) = \Psi_\eta(\Phi_\eta(Z_i, G_i)) \tag{7}$$

The graph convolutional neural network and the head are trained together in the graph embedding update training process. The importance given to each of the two losses is determined by a parameter $\gamma \in (0, 1)$ and thus the resulting loss function to minimze is:

$$\mathcal{L} = (1 - \gamma)\, \mathcal{L}_S(Y) + \gamma\, \mathcal{L}_T(Y) \tag{8}$$

Semantic Clustering Loss The idea behind the Semantic Clustering Loss [6] is to enforce the learned representations to be more consistent between the same events. To do so, we mine the nearest neighbors of each input sample from the feature encoder based on feature similarity. We empirically found that, in most cases, these nearest neighbors belong to the same event and a fraction of those were not temporal adjacent. We therefore exploit this information as a prior into the presented learnable approach.

For every sample $x_i \in X$ from the encoded features, we mine its K nearest neighbors (in terms of euclidean distance) in the embedding space given by the backbone model trained in the constrastive setup. We define the set $\mathcal{N}_{x_i} \in \mathbb{N}$ as the neighboring samples indices of x_i in the embedded space.

We aim to learn a clustering function parameterized by a graph convolutional neural network Φ_η and a head Ψ_η with weights η that, given the representations Z and their adjacency matrix G, classifies a sample

z_i and its neighbors set in the encoded feature space together. The Semantic Clustering Loss $\mathcal{L}_S$ is then defined as the sum of two elements:

$$\mathcal{L}_S = \mathcal{L}_c + \lambda \mathcal{L}_e \tag{9}$$

The first term, $\mathcal{L}_c$, imposes the network to make consistent predictions y_i for a representation z_i and its neighboring feature $z_k, k \in \mathcal{N}_{x_i}$, prediction y_k, through a dot product operator. The dot product will me maximal (thus minimizing the negative log likelihood) when the predictions are assigned to the same cluster.

$$\mathcal{L}_c = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k \in \mathcal{N}_{x_i}} \log(y_i^\top y_k) \tag{10}$$

To avoid all the samples to be assigned into a single cluster, an entropy regularization term $\mathcal{L}_e$ is added and weighted by a scalar value λ. This expression is included to spread the predictions uniformly across the clusters. Given a single output $y_i \in \mathbb{R}^C$, the probability of it being assigned to a cluster $c \in \mathcal{C}$, $P(c|y_i)$, can be modeled by the Softmax operator on the output itself, thus the entropy loss term is defined as:

$$\mathcal{L}_e = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c \in \mathcal{C}} P(c|y_i) \log \left(\frac{1}{N} \sum_{j=1}^{N} P(c|y_j) \right) \tag{11}$$

The Semantic Clustering Loss function will make the model learn more consistent embeddings between its neighbors, resulting in a better grouping of the same event representations while also avoiding to cluster all the learned representations into the same cluster. To sum up, the Semantic Clustering Loss, $\mathcal{L}_S$, as the sum of the consistency (Eq. 10) and entropy (Eq. 11) terms is:

$$\mathcal{L}_S = -\frac{1}{N} \sum_{i=1}^{N} \left(\sum_{k \in \mathcal{N}_{x_i}} \log(y_i^\top y_k)) + \lambda \sum_{c \in \mathcal{C}} P(c|y_i) \log \left(\frac{1}{N} \sum_{j=1}^{N} P(c|y_j) \right) \right) \tag{12}$$

Temporal Clustering Loss The semantic prior between two events is not the only one we can exploit in the video domain. A video is a continuous set of images whose temporal relation determines which event or action is happening. Therefore, we can assume that there is a temporal relation between events: a set of frames temporally closer is more likely to be from the same event than another more distant sequence.

To capture this temporal relation in the event representations Z, a temporal window of length l_t for each anchor head output $y_i \in Y$ is determined centered at the time-frame i. Then, the set of outputs P_{y_i} is defined as those inside the temporal window and they are known as the positive set. The negative set N_{y_i} is then the set of samples outside the defined temporal window.

The aim of the Temporal Clustering loss is to map features in the positive set together and repeal those

in the negative one (by a larger distance than a margin m), exploiting the temporal relation in a video sequence. However, features in the positive window that are further from the anchor, are less likely to be part of the same event. To solve this, a weight function $w(\cdot)$ is defined to add a coefficient based on the temporal distance d_t between two representations, and it follows:

$$w(d_t) = -1 + 2\exp(-\frac{1}{\beta}d_t) \tag{13}$$

Where β is a constant that depends on the desired positive window length l_t. We want the weight function to be 0 at the boundary of the window, therefore the constant β can be obtained by imposing $w(l_t/2) = 0$ and solving for β:

$$0 = -1 + 2\exp(-\frac{1}{\beta}\frac{l_t}{2}) \implies \beta = -\frac{l_t}{2\ln(\frac{1}{2})}$$

Then, we can define a new weight function for a window of length l_t, $w_{l_t}(\cdot)$. An illustration of the function for a distance between a sample y_i with temporal time-frame i and any other sample is shown in Figure 6.

$$w_{l_t}(d_t) = -1 + 2\left(\frac{1}{2}\right)^{\frac{2d_t}{l_t}} \tag{14}$$

To determine whether two output vectors $y_i, y_j \in Y$ are similar or not, we follow the naive Triplet loss implementation [7] and define a distance metric, $d(y_i, y_j)$, as the euclidean distance temporally weighted by $w_{l_t}(\cdot)$ between them. Then, the Temporal Clustering loss function $\mathcal{L}_T$ with a margin m is defined as:

$$\mathcal{L}_T = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{y_j \in P_{y_i}}\sum_{y_k \in N_{y_i}} \max(0, w_{l_t}(|i-j|)d(y_i, y_j) + w_{l_t}(|i-k|)d(y_i, y_k) + m)\right) \tag{15}$$

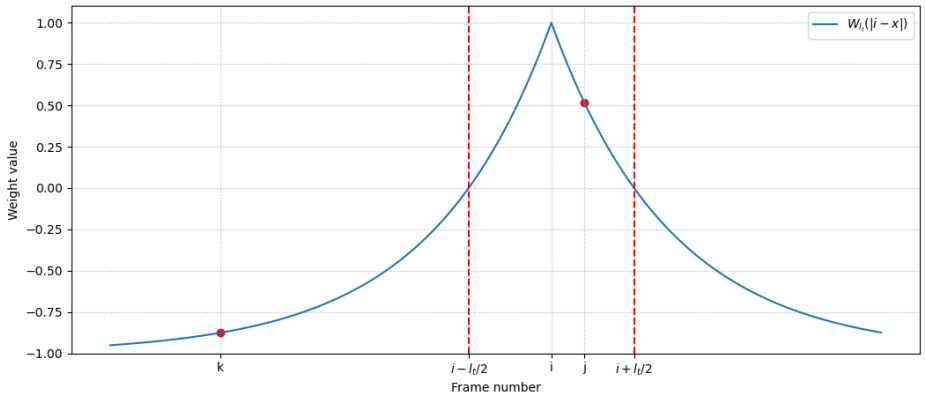


Figure 6: Plot of the weight function w_{l_t} for the absolute difference between the temporal index of i and any randomly picked feature index x with a window length of lt. Randomly chosen positive j and negative k indexes are plotted as an example.

3.2.4 Graph structure update

Learned event representations carry new semantic and temporal features from the graph embedding update. However, it is important that these representations are in a way similarly related to previous learned graphs and the original high dimensional graph. The original graph G_X was obtained by high level encoded representations that featured important spatio-temporal characteristics learned by the contrastive task. Although these representations were not suitable for event representation, they still carry meaningful characteristics that are useful to maintain in the new learned graph. The graph structure update aims to learn meaningful features from the graph convolutional neural network Φ_η suitable for the representation of the underlying graph G. The learned output features in this step are solely used for the graph structure update and are not kept as the new representations Z (see Figure 4 (c)).

The presented loss for the graph structure update has two terms weighted by a regularization parameter $\alpha \in (0, 1)$. The first term controls the fit of the generated affinity matrix from the output of the graph convolutional neural network given the learned features Z_{i+1} with the previous learnt graph G_i. The second loss term quantifies the fit of the same network's output features affinity matrix with the fixed initial graph G_X in the original high dimensional space. In general, the loss function is defined as:

$$\mathcal{L} = (1 - \alpha)\mathcal{L}_G(S_{l_z}(Z), G) + \alpha\mathcal{L}_G(S_{l_z}(Z), G_X) \tag{16}$$

The loss function used in the graph structure update (as well as in the initialization of the algorithm), $\mathcal{L}_G$, is a cross entropy loss function that measures how similar the affinity matrix given by the learned representations $S_{l_z}(Z)$ is against another affinity matrix G.

$$\mathcal{L}_G = -\sum_{i=1}^{N}\sum_{j=1}^{N} S_{l_z}(Z)_{(i,j)} log(G_{(i,j)}) \tag{17}$$

3.3 Downstream task

To evaluate the effectiveness of the learned graph-based event representations, we apply a clustering algorithm that classifies the learned features Z into C different clusters given by the ground truth labels of each video evaluated. Each cluster then represents an event in the video. This section describes the last part of the pipeline introduced, illustrated in Figure 4 (d).

Event segmentation is temporally smooth. This means that is not common to see a very small event change inside a larger event in the ground truth of each video. To enforce this smoothness and inspired by similar approaches [2], we found beneficial to foster similarity for temporally adjacent nodes by local averaging the edge weights on the learned affinity matrix G through a 2-dimensional smoothing kernel K_p of size $p \times p$. For the downstream task of event segmentation, we will reefer to the similarity graph as the smoothed version obtained after the convolution with the mentioned kernel: $G \leftarrow G * K_p$.

3.3.1 Spectral Clustering

The clustering approach chosen for the unsupervised event segmentation is the Spectral Clsutering technique. Given a set of data points $z_1, \dots, z_N$ and a notion of similarity between all pairs of data points z_i and z_j, the goal of a clustering procedure is to divide these data points into K groups known as clusters such that points grouped together are in a way similar between them. We then choose K equal to the number of ground truth events of the video ($K = C$).

An effective way of representing the data based on the similarity between them is in the form of the similarity fully connected graph G. Each vertex or node in the graph represents a data point z_i. Every vertex is connected by a weighted edge given by the similarity between the corresponding data points z_i and z_j. therefore, each edge carries a non-negative weight $G_{(i,j)} \geq 0$ in the weighted affinity matrix of the graph G. Then, the degree d_i of a node is denoted as:

$$d_i = \sum_{j=1}^{N} G_{(i,j)} \tag{18}$$

The degree matrix of the graph, $D \in \mathbb{R}^{N \times N}$, is defined as the diagonal matrix with the set of degrees for each vertex in the diagonal.

The problem of clustering can now be reformulated: the clustering goal is to find a partition of the graph such that the edges between different groups have very low weights and the edges within the same group have a strong weight connection. Spectral Clustering is an algorithm that obtains K different clusters from the similarity graph based on the edge weights connections [36]. The first step in the Spectral Clustering algorithm is to compute the Laplacian matrix of the affintiy matrix, L:

$$L = D - G \tag{19}$$

The degree matrix D is diagonal and positive, thus its square root $D^{-\frac{1}{2}}$ is the diagonal matrix whose entries are the reciprocals of the positive square roots of the diagonal entries of D. Then, the normalized symmetric Laplacian matrix L_{sym} is then computed as:

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \tag{20}$$

After that, we obtain the first K eigenvectors of L_{sym}: $u_1, \dots, u_K$ and then define $U \in \mathbb{R}^{N \times K}$ to be the matrix containing the vectors $u_1, \dots, u_K$ as columns. Normalizing the rows of U we obtain the set of vectors $y_i \in \mathbb{R}^K$ as the i-th column of the row-normalized matrix U.

The spectral clustering then follows the K-means algorithm [27] for the set of vectors $y_1, \dots y_N$, resulting into a K cluster divisions. Tracking the temporal order of each representation in the video, we can generate a segmentation sequence by assigning each representation to the cluster assigned by the algorithm.

4. Results

In this section, we will first introduce the dataset used in the evaluation of the presented approach, the metrics used to evaluate our method and the hyperparameters direct impact through the ablation studies of the proposed technique. Finally, we will illustrate the results for the downstream task of event segmentation and compare our approach to state-of-the-art approaches targeting the same task.

4.1 Experimental setup

In this part, we will present the dataset of complex activities: untrimmed videos recorded from the same point of view and no evident change between events, with high temporal resolution ($\geq$ 15 fps). We will also present the metrics used to evaluate our graph-based learned representations, which then will be used to benchmark our approach against other techniques that tackled the same challenging segmentation task.

4.1.1 Dataset

The dataset used for the training and evaluation of the proposed method, The *Breakfast Action Dataset* [8], consist on various videos suitable for the evaluation of the learned representations for the downstream task of video event segmentation. Each video in the dataset follows the definition of a complex activity: they are recorded in a high temporal resolution (15 fps) and the camera is static during all the recording (Figure 7).

This dataset is one of the largest fully annotated datasets available in both temporal and action segmentation terms. It comprises of 10 activities related to breakfast preparation, performed by 52 different individuals in 18 different kitchens. The place where the activity takes place as well as the activity performed is given by the name of the video file (*P03_cereals, P06_coffee, P11_tea...*). The amount of videos available for each activity is detailed in Table 1.

Activity	Nº of videos
Coffee	200
Orange Juice	187
Chocolate Milk	224
Tea	223
Cereals	214
Fried eggs	198
Pancake	173
Fruit Salad	185
Sandwich	197
Scrambled Eggs	188

Table 1: Activities in the *Breakfast Action dataset*

Figure 7: Sample images of the *Breakfast Action dataset*

Every video in the dataset describes an activity subdivided into different actions (e.g. the activity of making *Cereals* has different actions like *Take the milk*, *Take the bowl*, *Pour the cereals*). In this dataset, the temporal segmentation of the events will coincide with the timing of the actions happening in the activity as they are understood as the events of the video.

4.1.2 Evaluation metrics

The performance analysis of the presented approach for event segmentation will report two widely used metrics to benchmark complex activity videos. These are the same metrics used to evaluate other methods that tackle the same challenging task [33, 26, 25].

To begin with, the learned graph-based representations follow a clustering algorithm that inherently segments the video into different events. To match the predicted segmentation labels and the ground truth, we follow the Hungarian matching algorithm: construct a bipartite complete graph between the predicted clusters and the known labels. Then find a maximum-weight matching between the two partitions. This is the usual approach in unsupervised learning to obtain a one-to-one mapping between the predicted labels (cluster index), with the ground truth labels.

The first metric presented is highly related to the accuracy of the segmentation and event classification. The **Mean over Frames (MoF)** metric evaluates the ability of the presented approach to temporally segment and localize events within the video. The metric simply indicates the percentage of frames events correctly labeled after the Hungarian matching.

To evaluate both the similarity and diversity of the predicted segments, we also compute the **Intersection over Union (IoU)** metric, also known as the Jaccard Index or Jaccard similarity coefficient. Given two sets A and B, the Jaccard Index is defined as the size of the intersection divided by the size of the union of two sets. In our problem, the sets A and B are the samples between two consecutive detected boundaries in the predicted feature vector ($Pred$) and the Ground Truth (GT) respectively.

$$IoU = \sum_{A \in Pred} \sum_{B \in GT} \frac{|A \cap B|}{|A \cup B|} = \sum_{A \in Pred} \sum_{B \in GT} \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{21}$$

4.2 Experimental evaluation

The influence study of the presented hyperparameters, as well as describing the implementation details of each block of the proposed method will be discussed in this section. We will also show the resulting video segmentation results as well as compare the presented metrics with current state-of-the-art and baselines models for the segmentation task of the *Breakfast Action Dataset*.

4.2.1 Feature Encoder

The first stage of the proposed method is to train a contrastive model to extract meaningful spatial and temporal features from input clips. The backbone ϕ architecture used is based on a Resnet18-3D [4, 35] whose input is a set of video frames (clip) $V \in \mathbb{R}^{N\times(T\times W\times H)}$. The parameters' choice for the training follow the implementation details given by their original authors [4]: an input clip of temporal dimension (number of frames) $T = 16$ and frame dimensions $W = H = 180$. The output is a matrix $X \in \mathbb{R}^{N\times n}$ and $n = 512$ is the output dimension of each of the N feature vectors.

The training of the feature extractor for the Contrastive Event representation learning (CERL) follows a modified version of the SimCLR contrastive setup [5]. That is stacking a projection head ψ parameterized by a fully connected layer that maps the feature dimension n into an embedded dimension $m = 128$ (Equation 2). The contrastive loss function $\mathcal{L}_{\text{CLR}}$ defiend in Equation 3, has its temperature parameter τ set to 0.1. This loss is then minimized through the Stochastic Gradient Descend optimization method, with a starting learning rate of 0.05, a momentum of 0.9 and a L_2 weight decay of 10^{-3}. The training follows a cosine learning decay rate of 0.1. The model was trained for 150 epochs on a batch size of 10 pairs of positive clips (itself and its temporally adjacent), where the loss had already converged to a minimum.

Data augmentation Composition of data augmentation operations is crucial for learning good representations in a contrastive setup. For the event representation learning, we focus on the augmentations in both temporal and spatial dimensions.

For the temporal augmentation, we did not perform any additional operation than taking adjacent clips as positive pairs. Instead of taking the same clip and applying temporal transformations, we found reasonable to take it's adjacent temporal clip (likely of the same event) thus forcing the network to learn temporal relations between two adjacent clips.

Spatial augmentation is widely used in self-supervised learning. Although there are several ways to apply spatial augmentation in a sequence of images, we followed a temporally consistent spatial augmentation procedure, that is making each spatial augmentation consistent along the temporal dimension. Random augmentation in the temporal axis was shown to affect negatively in the video representation learning [4]. For spatial augmentations, we define a family of random transformations $\mathcal{T}$ that will be applied in conjunction to all frames in the clip. Algorithm 2 show a detailed description of the spatial augmentations applied in the training procedure.

ALGORITHM 2: Temporally consistent spatial augmentation method

Input : $v = \{f_1, f_2, ..., f_T\} \in \mathbb{R}^{T \times W' \times H'}$ *input video clip*

Output: $\tilde{v} = \{\tilde{f}_1, \tilde{f}_2, ..., \tilde{f}_T\} \in \mathbb{R}^{T \times W \times H}$, *spatially augmented video clip*

Spatial transformations $\mathcal{T}$**:** Random transformations are applied with a probability p to all frames

Resize: Randomly resize and center crop a spatial region of 180×180.

Horizontal flip: Draw a flag F_h from $0, 1$ with a $p = 0.5$.

Color Jitter: Draw a flag F_j from $\{0, 1\}$ with a $p = 0.8$.

Gray Scale: Draw a flag F_g from $\{0, 1\}$ with a $p = 0.2$.

Gaussian blur: Draw a flag F_b from $\{0, 1\}$ with a $p = 0.4$.

for $i \leftarrow 1$ **to** T **do**

- $\tilde{f}_i = \text{random_resized_crop}(f_i)$ *randomly crop from the center of the frame with a random scale in* $(0.3, 1.0)$
- $\tilde{f}_i = \text{horizontal_flip}(\tilde{f}_i)$ *randomly flip the frame horizontally if* $F_h = 1$
- $\tilde{f}_i = \text{color_jitter}(\tilde{f}_i)$ *brightness, contrast and saturation by a random factor of* $(0, 0.4)$ *if* $F_j = 1$
- $\tilde{f}_i = \text{grayscale}(\tilde{f}_i)$ *transform the frame to gray-scale if* $F_g = 1$
- $\tilde{f}_i = \text{graussian_blur}(\tilde{f}_i)$ *apply a Gaussian blur kernel size of* 3×3, $\sigma = 0.1$ *if* $F_b = 1$

end

4.2.2 Self-supervised graph-based representation learning

The graph based representation learning algorithm follows the training of a graph convolutional neural network Φ of $L = 2$ layers. The proposed GCN follows an autoencoder architecture in which the input shape is the same as the output shape. To map the high-dimensional $n = 512$ input learned features $X \in \mathbb{R}^{N \times n}$ to the low-dimensional event representations $Z \in \mathbb{R}^{N \times d}$, PCA_d is applied to the input features, where $d = 35$.

The similarity function $S_l(\cdot)$ (Equation 6) is used to get an affinity matrix from the representation features. The parameter l used in the similarity function has a direct relation with the feature space dimension. The choice of those were based on visual inspections of the resulting affinity matrix in both the high and low dimensional spaces. We found the best hyperparameter values $l_x = 2 \times 10^{-5}$ for n-dimensional features and $l_z = 0.002$ for d-dimensional features.

The training procedure follows two steps: the graph embedding update, by stacking a head Ψ parameterized by a fully connected layer that maps the low dimensional feature representations to the number of ground truth clusters (Equation 7), and the graph structure update, directly applied to the affinity matrix obtained on the output features of the graph convolutional neural network.

The graph embedding update loss function, presented in Equation 8, is composed of the Semantic clustering loss function $\mathcal{L}_S$ and the Temporal Clustering loss function $\mathcal{L}_T$ weighted by a parameter γ. Table 2 reports the evaluation metrics obtained respect to the choice of the parameter. We can see that, although performing a graph embedding update solely taking into account the semantic term ($\gamma = 0$) or the temporal term ($\gamma = 1$) leads to an improvement over naive CERL features, a larger improvement is achieved when taking both into account. Furthermore, a clear influence on the output graph is shown

Graph embedding weight update γ						
γ	0.0	0.2	0.4	0.6	0.8	1.0
MoF	56.0	56.5	**56.7**	56.4	56.4	55.7
IoU	44.85	45.3	45.4	**45.6**	45.4	44.3

Table 2: Ablation graph embedding update weight parameter γ. Underlined values reefer to extreme cases where only semantic ($\gamma = 0$) or temporal ($\gamma = 1$) terms are trained.

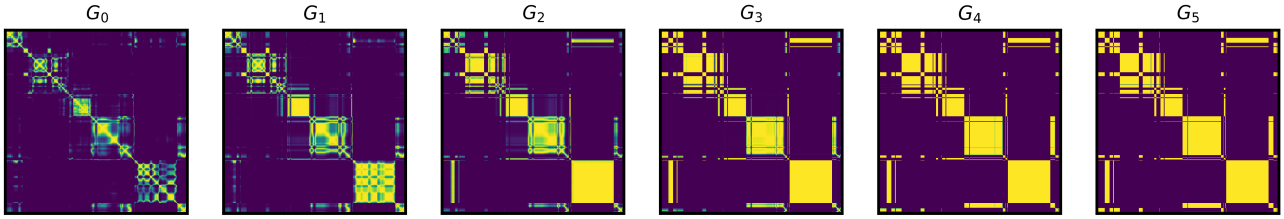
Graph structure weight update α						
α	0.0	0.2	0.4	0.6	0.8	1.0
MoF	54.7	56.0	**56.4**	55.6	55.7	56.2
IoU	44.18	45.0	**45.6**	44.7	45.0	44.8

Table 3: Ablation graph structure update weight parameter α. Underlined values reefer to extreme cases where only previous ($\alpha = 0$) or original ($\alpha = 1$) graphs are considered.

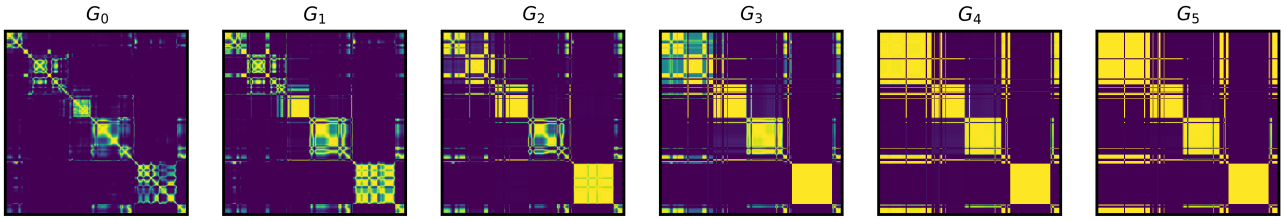
for the two extreme cases in Figure 8. Enforcing temporal relations, larger connected areas are learned in the diagonal of the affinity matrix, but semantic relations between events outside of the diagonal are lost. Finally, based on experimental results, a value of $\gamma = 0.6$ is set, which gives a bit more weight to the temporal term.

The Semantic clustering term (Equation 9) is composed of another two terms: the consistency $\mathcal{L}_c$ and the entropy $\mathcal{L}_e$ regularization term. For the first one, we followed the presented results in the original report [6] which states that the influence of the used number of neighbors K has no improvement after a value of 15, thus we set $K = 15$. The entropy term is weighted by a scalar value, which set to $\lambda = 5$ [6].

The Temporal Clustering term (Equation 15) follows the Triplet Margin loss implementation [7] with a margin $m = 1.0$. The weight function used to measure temporal similarity, w_{l_t} uses a window for the selection of the positive and negative sample pairs. We set the length of this window to the number of frames in the video divided the number of ground truth clusters in it, $l_t = \lfloor N/C \rfloor$.



(a) Semantic term learning ($\gamma = 0$)



(b) Temporal term learning ($\gamma = 1$)

Figure 8: Illustration of the graph-based representation affinity matrix at each iteration for one video of the *Breakfast Action Dataset* (*P46_milk*) for the two extreme cases of the graph embedding update: only semantic (a) or temporal (b) term.

The second loss function (Equation 16) is used in the graph structure update and was presented in the DGE approach [2]. This loss function is also a weighted sum of two terms by a parameter α. The first one captures the similarity between the learned representations with the previous affinity matrix. The

second term leverages the fit of the event features with the original high-dimensional representation graph. Motivated by the results in Table 3, the relative weight of each term is set to $\alpha = 0.4$.

Finally, the training process follows an iterative algorithm (Algorithm 1) that converges to the best affinity matrix and graph-based features in terms of evaluation metrics at the third epoch (Table 4). To optimize each of the loss functions described in the training process we used the Adam optimizer [37] with a learning rate of 0.01.

	Number of iterations T			
T	1	2	3	4
MoF	54.8	55.8	**56.4**	54.7
IoU	44.2	44.3	**45.6**	43.5

Table 4: Ablation study on training epochs for the graph-based event representation learning algorithm. Results obtained with the hyperparameters freezed to the best reported in the rest of the ablation studies.

After the parameters' choice discussed earlier, Figure 9 shows an example of the difference between the video segmentation of the encoded features learned through the contrastive task (CERL) by applying standard K-means clustering in them, and the resulting segmentation of the graph-based learned features through the spectral clustering technique on the smoothed affinity matrix (smoothing kernel K_p, $p = 50$). The learned affinity matrix is also shown at each iteration of the training process.

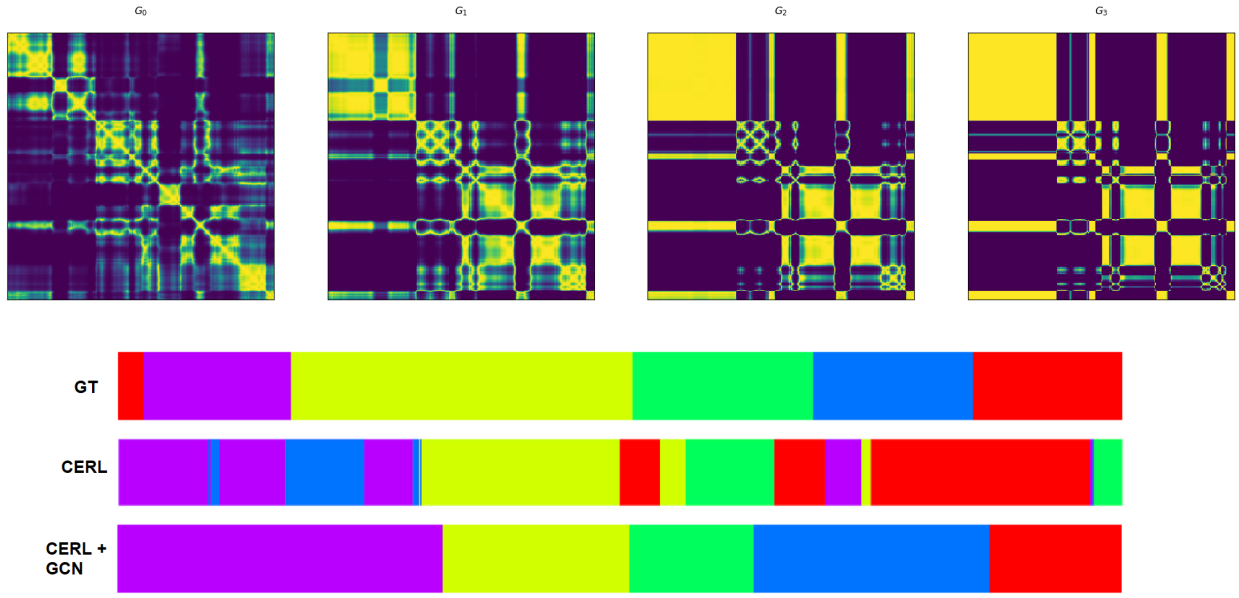


Figure 9: Resulting learned affinity matrix at each epoch (G_0, G_1, G_2, G_3) and segmentation output on a sample video from the Breakfast action dataset (*P03_cereals*). **GT** *reefers to the ground truth labels,* **CERL** *is the result of the segmentation clustering approach to the learned features from the backbone model and* **CERL+GCN** *the segmentation of the graph-based learned event representations.*

4.2.3 Experimental results

Table 5 compares our method to fully, weakly and self-supervised learning methods as well as unsupervised clustering approaches. Unsupervised baselines are based on clustering frame representations. Following the presented results in [33], three baseline methods are proposed: K-means [27], FINCH clustering [28] and Equal split. The latter simply splits the frame representations into K equal temporal continuous partitions, which surprisingly surpasses other supervised and weakly supervised baselines. We evaluate the results based on the presented metrics Mean over Frames (MoF) and Intersection of Union (IoU).

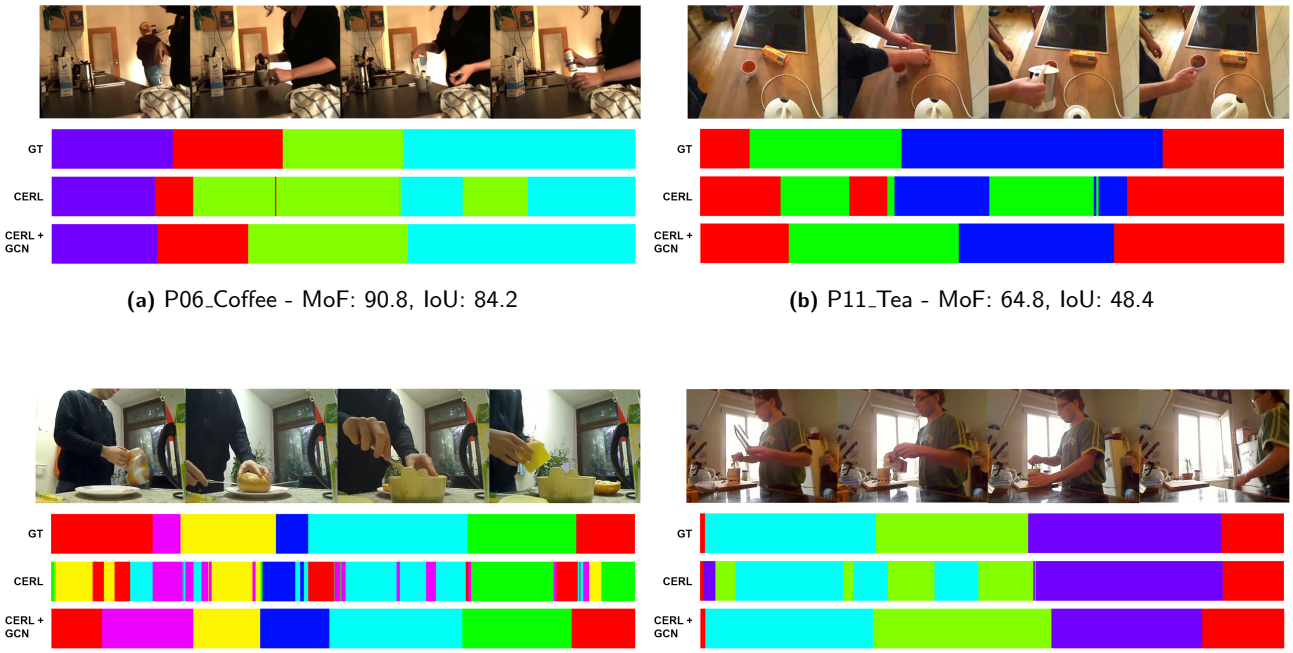
Supervision	Model	MoF	IoU
Fully supervised	SVM [38]	15.8	-
	HTK(64) [16]	56.3	-
	ED-TCN [17]	43.3	42.0
	GRU [18]	60.6	-
Weakly supervised	GMM + CNN [21]	12.9	28.2
	CDFL [23]	38.4	40.6
	TCFPN [39]	24.2	38.4
Unsupervised Clustering Baselines	Equal Split	34.8	21.9
	K-means [27]	42.7	23.5
	FINCH [28]	51.9	28.3
Unsupervised Clustering	TW-FINCH [33]	**62.7**	42.3
Self-supervised learning	KNN+GMM [24]	34.6	**47.1**
	LSTM+AL [26]	42.9	46.9
	Ours (CERL)	54.1	42.7
	Ours (CERL)+DGE [2]	55.6	44.1
	Ours (CERL+GCN)	56.4	45.6

Table 5: Segmentation results on the *Breakfast Action dataset*. MoF is the Mean over frames metric and IoU the Jaccard Index. CERL is the Contrastive Event representation learning model (Feature encoder) and GCN is the Graph Convolutional Neural network.

Our method significantly outperforms the presented unsupervised baselines. If we tackle current unsupervised and self-supervised state-of-the-art approaches, we see that we did not achieve the highest metric neither in terms of MoF or IoU. However, if we compare our approach to the best MoF performance method (TW_FINCH), we achieve a better IoU metric. Alternatively, the best method in terms of IoU (KNN+GMM) performs much worse than our method in metric terms of MoF. We can therefore conclude that the presented method is on point with current state-of-the-art approaches for the downstream task of event segmentation. We also tested DGE approach on the learned CERL features, achieving similar results than our method, but being much slower in the training stage due to its non-differentiable clustering step performed iteratively in the semantic update of the graph.

To finish the discussion, a few segmentation output samples of the *Breakfast Action Dataset* are shown in Figure 10. In it, we compare the results of the segmentation from the encoded CERL features with a

K-means clustering approach and the Spectral clustering on the learned affinity matrix from the graph-based event representations. We can see that our method not only results in a good estimate of the event segmentation, but the cluster assignment (color label) is also on point with the ground truth. Our method is reliable for short video sequences (10a) and long sequences (10c) where other methods tend to perform worse. On the other hand, videos whose metrics are not as good (10b) are temporal coherent with the segmentation events, thus the clustering order of the predicted actions is consistent with ground truth labels. We can clearly see that CERL spikes are completely removed in the presented approach. Moreover, if a short event is present in between two other equal larger events in the CERL segmentation, our method learns its temporal context and groups it into the same cluster, thus learning a similar event representations. On the other hand, semantic relatedness is preserved, clearly shown in 10d, where the few starting frames are clustered with the last frames by their semantic relatedness while being temporally very far, fully matching the ground truth annotation in extreme cases. We can then conclude that our graph-based representation learning approach successfully models semantic and temporal relations between clips, learning meaningful event representations jointly with their underlying graph structure that are suitable for the downstream task of event and action segmentation of complex activity videos.



(a) P06_Coffee - MoF: 90.8, IoU: 84.2

(b) P11_Tea - MoF: 64.8, IoU: 48.4

(c) P20_Sandwich - MoF: 81.0, IoU: 73.4

(d) P18_Milk - MoF: 92.2, IoU: 85.8

Figure 10: Illustration of the segmentation performance of our method and evaluation metrics for four different examples of different activities and scenarios from the *Breakfast Actions Dataset*. **GT** reefers to the ground truth labels. **CERL** is the segmentation obtained after applying a K-means clustering algorithm on the learned features through the contrastive event representation learning framework. **CERL+GCN** shows the resulting segmentation of the self-supervised graph-based event representation learning after a Spectral Clustering in the learned affinity matrix.

5. Conclusions and Future Work

The presented thesis shows a novel end-to-end trainable approach pipeline for learning event representations of complex activity videos jointly with their underlying graph structure. Furthermore, the presented approach operates in a fully self-supervised way and learns the event representations from the input video itself, avoiding the need of a large video dataset and expensive human annotations. Unlike other unsupervised approaches focused on the study of clustering techniques for event segmentation, the presented work shows that good event representations are enough to generate meaningful features which can be used in a downstream clustering task to segment the video sequence.

This work started by presenting a variation of the constrastive video representation learning for the task of event representation learning by introducing a modified training procedure (CERL). In it, we showed that learning similar representations of adjacent clips is of great value for learning event representations in the clip itself. While other approaches are based on learning frame-level features, we have shown that learning representations of 3D volumes (clips) are helpful in order to capture meaningful spatio-temporal characteristics for the event representation learning problem.

Then, we showed how to improve and contextualize this learned representations through a novel end-to-end trainable approach based on a graph convolutional neural network (GCN) and inspired on the Dynamic Graph Embedding (DGE) technique [2]. Our work shows that learning new features by exploiting semantic and temporal relatedness through a fully self-supervised manner ends up in the learning of significant event representations. Moreover, we demonstrated that learning the underlying graph structure of the representations, which encodes their semantic and temporal relation, is of great value for the downstream task of event segmentation in complex activity videos.

Our work achieves on par state-of-the-art metrics on the challenging task of segmenting each activity video of the *Breakfast Action Dataset*, as well as matching predicted labels of each partition to its ground truth action label. We are planning to scale these promising presented results to other complex activities datasets in order to perform a wider benchmarking of our method.

Future work will include the exploration of new clustering techniques on the learned gaph-based event representations, as well as extending the proposed approach to other challenging domains such as human motion segmentation. Moreover, we could benchmark the presented self-supervised graph-based learning approach on frame-level representations, which is the common baseline of the methods we compare to, and thus study how well the presented technique performs against the same initial features. On the other hand, future work could extend the contrastive proposed setup to include semantic and temporal relation within the CERL framework itself. However, training a feature extractor for each video on the dataset is computationally very expensive, therefore we find more interesting to further explore the CERL framework as presented, learning meaningful clip representations with spatial and temporal characteristics, and then further explore specific event segmentation clustering algorithm such as TW-FINCH [33] on these learned spatio-temporal features.

References

[1] Jeffrey Zacks, Barbara Tversky, and Gowri Iyer. Perceiving, remembering, and communicating structure in events. *Journal of experimental psychology. General*, 130:29–58, 2001.

[2] M. Dimiccoli and H. Wendt. Learning event representations for temporal segmentation of image sequences by dynamic graph embedding. *IEEE Transactions on Image Processing*, 30:1476–1486, 2021.

[3] Anna C. Schapiro, Timothy T. Rogers, Natalia I. Cordova, Nicholas B. Turk-Browne, and Matthew M. Botvinick. Neural representations of events arise from temporal community structure. *Nature Neuroscience*, 16:486–492, 2013.

[4] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. *Neural Information Processing Systems (NeurIPS)*, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning (ICML)*, 2020.

[6] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool. Scan: Learning to classify images without labels. *European Conference on Computer Vision (ECCV)*, 2020.

[7] Daniel Ponsa Vassileios Balntas, Edgar Riba and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 119.1–119.11, September 2016.

[8] H. Kuehne, A. B. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. *Computer Vision and Pattern Recognition Conference (CVPR)*, 2014.

[9] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] Måns Larsson, Erik Stenborg, Carl Toft, Lars Hammarstrand, Torsten Sattler, and Fredrik Kahl. Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization, 2019.

[12] C. Gan, B. Gong, K. Liu, H. Su, and L. J. Guibas. Geometry guided convolutional neural networks for self-supervised video representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5589–5597, 2018.

[13] Jiangliu Wang, Jianbo Jiao, Linchao Bao, Shengfeng He, Yunhui Liu, and Wei Liu. Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. *International Conference on Computer Vision (ICCV)*, 2019.

[15] Catarina Dias and Mariella Dimiccoli. Learning event representations by encoding the temporal context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[16] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. *Winter Applications for Computer Vision (WACV)*, 2016.

[17] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Colin Lea, Austin Reiter, Rene Vidal, and Gregory D. Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. *European Conference on Computer Vision (ECCV)*, 2016.

[20] Dong Wang, Di Hu, Xingjian Li, and Dejing Dou. Temporal relational modeling with self-supervision for action segmentation. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

[21] Hilde Kuehne, Alexander Richard, and Juergen Gall. Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding*, 163:78–89, 2017.

[22] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. *European Conference on Computer Vision (ECCV)*, 2016.

[23] Jun Li, Peng Lei, and Sinisa Todorovic. Weakly supervised energy-based learning for action segmentation. *International Conference on Computer Vision (ICCV)*, 2019.

[24] Bharat Lal Bhatnagar, Suriya Singh, Chetan Arora, and C.V. Jawahar. Unsupervised learning of deep feature representation for clustering egocentric actions. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1447–1453, 2017.

[25] Fadime Sener and Angela Yao. Unsupervised learning and segmentation of complex activities from video. *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[26] Sathyanarayanan N. Aakur and Sudeep Sarkar. A perceptual prediction framework for self supervised event segmentation. *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[27] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[28] Vivek Sharma M. Saquib Sarfraz and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2019.

[29] Sheng Li, Kang Li, and Yun Fu. Temporal subspace clustering for human motion segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4453–4461, 2015.

[30] Carlotta Domeniconi, Dimitris Papadopoulos, Dimitrios Gunopulos, and Sheng Ma. Subspace clustering of high dimensional data. *Proceedings of the Fourth SIAM International Conference on Data Mining*, 04 2004.

[31] S. Y. Charles J. Alpert. Spectral partitioning: The more eigenvectors, the better. In *32nd Design Automation Conference*, pages 195–200, 1995.

[32] Mariella Dimiccoli, Marc Bolaños, Estefania Talavera, Maedeh Aghaei, Stavri G Nikolov, and Petia Radeva. Sr-clustering: Semantic regularized clustering for egocentric photo streams segmentation. *Computer Vision and Image Understanding*, 155:55–69, 2017.

[33] M. Saquib Sarfraz, Naila Murray, Vivek Sharma, Ali Diba, Luc Van Gool, and Rainer Stiefelhagen. Temporally-weighted hierarchical clustering for unsupervised action segmentation. *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[34] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[38] 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*, pages 780–787, 2014.

[39] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. *Computer Vision and Pattern Recognition (CVPR)*, 2018.