Aspects of time for recognizing human activities

Hussein, N.M.E.

Publication date
2021
Document Version
Final published version

ASPECTS OF TIME FOR RECOGNIZING HUMAN ACTIVITIES

NOURELDIEN HUSSEIN

ASPECTS OF TIME FOR RECOGNIZING HUMAN ACTIVITIES

NOURELDIEN HUSSEIN

Aspects of Time for Recognizing Human Activities

Noureldien Hussein

This book was typeset by the author using LaTeX 2_{ε}.

Aspects of Time for Recognizing Human Activities

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen
op vrijdag 9 april 2021, te 13.00 uur

door Noureldien Mahmoud Elsayed Hussein
geboren te Dakahliya

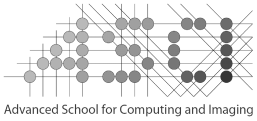
Promotiecommissie

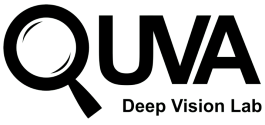
Promotor:	prof. dr. ir. A.W.M. Smeulders	Universiteit van Amsterdam
Copromotor:	dr. E. Gavves	Universiteit van Amsterdam
Overige leden:	prof. dr. M. Shah	University of Central Florida
	prof. dr. C.G.M. Snoek	Universiteit van Amsterdam
	prof. dr. M. Welling	Universiteit van Amsterdam
	dr. P.S.M. Mettes	Universiteit van Amsterdam
	dr. A. Habibian	Qualcomm Netherlands

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

UNIVERSITEIT VAN AMSTERDAM

This work was carried out in the ASCI graduate school, dissertation series number 417, at the Intelligent Sensory Information Systems lab, and at the QUVA lab of the University of Amsterdam.





CONTENTS

1

INTRODUCTION

What makes a single image fundamentally different form a collection of images?

(a) Single image *(b) Collection of successive images*

Figure 1: The left is a still picture of a sportsman – one can easily tell that it is a track and field sport. However, it is not that easy to guess what specific sport it is, actually. May be it is long jump, disc throw, high jump, or something else. Only by considering more images, on the right, you can conclude the answer[‡]*.*

It is often said that a picture is worth a thousand words. If that is so, then is a video worth a thousand pictures or a million words? Look at the single image on the left of figure 1, the first thought that comes to mind is a sportsman in one of the track and field sports. A second look might tell you that it is probably a long jump or a high jump. To make it easier, let's agree that it is a long jump. And yet, this opens more questions. For instance, the jump is successful or failed. Let us consider the rest of the images on the right of figure 1 – please turn the page upside down. Now that you have seen the rest of the images, why are we able to conclude the correct answer from the image collection on the right but not from the single image on the left? In other words, what is the essential difference between the left and the right? If the left is just a single image, while the right is a collection of successive images, what does the latter carry more than the former? Is it just more of that same; more observations of the same running person? Or is the difference something more fundamental?

From the perspective of computer vision, an image is a digital signal encoding the visual world around us, in a way that is comprehended by the computer. This digital signal has three dimensions: height, width, and color. A video is also considered as a digital signal. But compared to an image, it spans an extra dimension – that is the time. As such, a video encapsulates a multitude of information more than a single image. Having time in hand means having a much detailed and more refined view of the perceived visual world. Using the single image of figure 1a, we can predict endless possibilities of what will happen to the sportsman. He may succeed in the jump or may fall. However, using a video, *e.g.* the collection of images in figure 1b, we are able to tell

‡ The answer is high jump

a much more precise story of the sportsman, compared to the single image. Consequently, we would like to argue that *time* is the single most important, and most evident difference between a mere image and a collection of images, *i.e.* a video.

This thesis is concerned with studying time, within the context of human actions in videos. But why do we want to study time? One might argue that time brings a profoundly novel dimension of information. So, studying time gives us a far better understanding of human actions in videos. Complementary to studying time, a comprehensive understanding of its properties is needed. These properties include but are not limited to the (a) arrow of time, (b) cyclicity (or repetition), (c) rate of change, (d) structure, and last but not least (e) permutation. In the following, we emphasize on these properties, one by one.

The first fundamental property is the arrow of time. Let us revisit the single image in figure 1a. From the pose of the sportsman, you can probably tell that he is running. But you cannot for sure tell if he is running forward or backward – only time can tell. That is why it is important to study the arrow of time. There are countless opposite pairs of actions where only time can tell the difference between each pair. For example, swipe left *v.s.* swipe right, or open jar lid *v.s.* close jar lid.

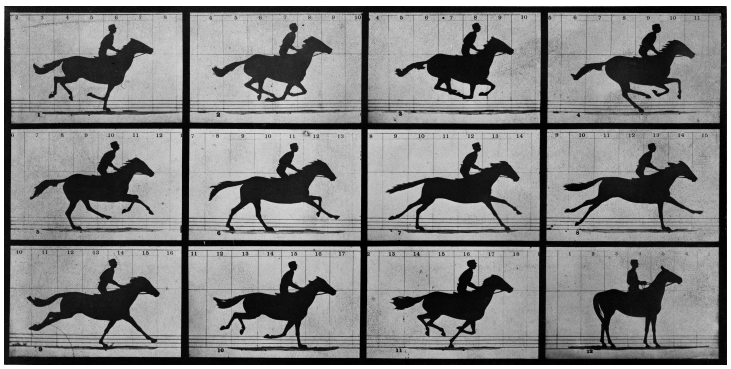


Figure 2: The top left image tells us that there is a jockey riding a horse. But the entire video, represented as an ordered collection of images, paints a much more clear picture, because it provides us with an extra, yet fundamental, signal – that is time. Only by considering time, important information about the video is revealed. For example, time tells us that: (a) the horse is moving forward, not backward (arrow of time), (b) the horse gait is gallop, not walk, trot, nor canter (structure of time), and (c) the horse is finishing one complete cycle of galloping (cyclicity or repetition of time).

Consider the top left image in figure 2, it shows a jockey riding a horse. But can you tell only from this image, what is the horse gait, and what is the periodic rate? The entire video, represented as an ordered collection of images, paints a much more clear picture, only because they provide us with an extra, yet fundamental, signal – that is time. Only when considering time, important information about the video is revealed. First, time tells us that the horse gait is gallop, not walk, trot, nor canter. Second, we can deduct from the video that the gallop is cyclic locomotion. That is to say, the gallop gait repeats itself exactly every nine successive images – notice the similarities in the horse gait between the second and the tenth image. Given that the order of images is left to right, then top to bottom, see how the second image and eleventh image are almost alike. This property of time is called cyclicity, or repetition. Third, the rate of change in the

horse gait from one image to another may tell us if the horse is about to change the gait or is coming to a complete stop. Notice the rate of change between the first two top left images, compared to that between the last two bottom-right ones. This property of time is called the rate of change. While the aforementioned properties, namely arrow of time, cyclicity, and rate of change are essential to understand time, they go beyond the scope of this thesis. While these properties are worth mentioning, this thesis discusses other aspects of time, which are discussed next.

How novel human activities can be recognized using previously learned ones?

Chapter 2 is concerned with how do we perceive novel visual evidences, and how do we recognize them. Imagine yourself riding a horse, such as that of figure 2. But this time, not as a jokey or equestrian, but rather as a zoo visitor. While you move from one place to another, you encounter new animals, see novel creatures, and hear new sounds. You will learn about these never-seen-before creature, and may memorize some of their names in your memory. For example, you may now have seen and defined what donkeys, lions, dogs, dears, ostriches, chickens may look like. Now, imagine you arrive at a new place and encountered a novel animal. In a split second, your brain, to avoid confusion and overcome uncertainties, will try to classify this animal as the closest look-alike from your memory. So, if this novel animal is, let's say, a kangaroo, here is the silent dialog that will happen in your brain. You will say, well, on one hand, it has the head of a donkey, or most properly a deer, but it is biped animal, so it is definitely, neither a donkey nor deer. On the other hand, this novel animal is biped, and very similar to an ostrich, but it does not have the head of the ostrich. This reasoning might go on for a while before you start calling the new animal a new name, such as ostrich-with-deer-face. Similarly, chapter 2 of this thesis looks into how do we recognize novel human events. It argues that a novel one can be cased as weighted combination of a previously known events. For instance, the human event of "baby shower" might be viewed as an event of "wedding shower" merged with another event of "baby birthday".

We are given a repository of test videos, each entails an event, like "birthday party" or "dog show". Also, we are given a set of textual queries describing these events. For each query, the task is to retrieve, *i.e.* rank, the most related videos, based on the semantic similarity between the query and the video contents. As none of the queries are accessible during training, this learning problem is known as zero-shot. We propose to learn a cross-modal feature space using external sources of knowledge, such that the space makes it possible to correctly retrieve the videos in test time. To learn this space, we rely on two external sources of data, namely the video repository of EventNet, and the event description articles of WikiHow. These two sources are heterogeneous, thus cannot be used directly for training. So, we notice that the categories of EventNet directly corresponds to the article of WikiHow. Using this analogy, these two sources were ready for training. Furthermore, we learn cross-modal neural embeddings between the visual and the textual modality, such that for a certain event, the article and its correlated videos fall closer to one another in the feature space. To learn the feature space, an off-the-shelf distance metric, such as Euclidean or cosine, is used. Differently, we find that learning the distance metric itself, along with the neural embeddings of the visual and textual modalities, in an end-to-end fashion, is the optimal solution. Experiments are conducted

on two benchmarks MED-13 and MED-14. In addition, analysis and comparisons are made, where our method comfortably outperforms previous methods.

In the method of chapter 2, to represent a certain video, frames are uniformly sampled and their features are extracted. The video-level features are simply the frame-level counterparts, pooled over the temporal dimension. So, the temporal structure of each video is clearly understated. This motivates us to pay attention to the temporal structure of videos, and how to recognize it. Upon preliminary research, this problem appeared to be much bigger than initially thought. For one, it is found that human motions in videos are expressed at different levels of granularity, *e.g.* micro actions, atomic actions, unit actions, actions, activities, and events. Accordingly, the thesis asks, for instance, what is the difference in granularity between atomic actions and activities? Are actions more coarse-grained or more fine-grained than activities? More importantly, the following question is asked:

Is time structured? How to recognize such a structure in videos of human activities?

Figure 3: Can you guess what is the human activity in this picture?† A complex human activity is analogous to a mosaic painting. The activity is made of small pieces, called atomic actions. Each, by itself, does not tell the complete story of the activity. But only when all are put together, these pieces paint a clear picture of the complex activity.

Look carefully at the picture in figure 3. It is full of small pieces, and full of colorful details, much like a mosaic painting. Can you see a general pattern out of this picture? Or can you tell that it shows a human doing an activity? Which activity is it? It might be that there is no structure governing such pieces, after all. Now take one step back and look at the big picture. Perhaps now you can tell which human activity is it†. If so, why you were able to figure out the activity only in the second look? Maybe because we can see the overall pattern or structure of the picture. A complex human activity in the video

† The answer is cooking dinner

is analogous to a mosaic painting. It is made of small pieces, called atomic actions. Each alone cannot tell the complete story. But only when all put together, these pieces paint a clear picture of the activity.

In Chapter 3, the difference between short-range atomic actions and long-range human activities is discussed. In addition, three important properties of the latter are outlined. These properties are temporal composition, temporal order, and temporal extent. Take for example the complex activity of "cooking meal". Composition means that it can be broken down into building blocks, called one-actions, *e.g.* "stir", "wash", "slice". None of which has an end goal by itself. But together, they make the complex activity more meaningful. Order means that these one-actions exhibit temporal order, albeit week. Usually, one washes the hands before start cooking. And the temporal extent means that the temporal duration of the same one-action may vary from one video exemplar to another. One person might take a little bit longer to wash hands than another person. Existing methods fall short of addressing these three properties, combined. Timeception, a novel neural network layer for temporal modeling, is proposed. Timeception uses multi-scale kernels to tolerate the temporal extents of one-actions. Multi-scale is achieved by either using different kernel sizes or different dilation rates. Additionally, Timeception decomposes the kernel of typical 3D convolution into a newly proposed temporal-only kernels. Since the temporal aspect of human activity is, arguably, the most important among all other aspects, these temporal-only kernels are dedicated to model only the temporal dimension. As such, the effect of temporal-only convolutions is a drastic reduction in the computational cost of 3D convolutions. This enables Timeception to live up to the computational demands of minute-long videos. Moreover, Timeception is a modern and modular layer for temporal modeling. It can be stacked on top of 2D or 3D CNNs alike. By conducting several experiments, the benefits of Timeception are demonstrated, and the technical novelties are verified. Besides, Timeception outperforms state-of-art methods on three benchmarks Charades, Breakfast, and MultiThumos.

It is concluded from chapter 3 that the temporal structure of complex activities is weak. To model such a structure, Timeception is proposed, where the main building blocks of Timeception are the temporal convolutions. The fact that these convolutions are temporal means that the structure of time is envisioned as an arrow. Along which, these convolutional operations try to recognize the temporal patterns. But the question is, why do we cast time as an arrow? Can it take other forms or structures? If yes, what do the structure of time look like? Graph, lattice, tree, or something else? Most importantly, the next question is:

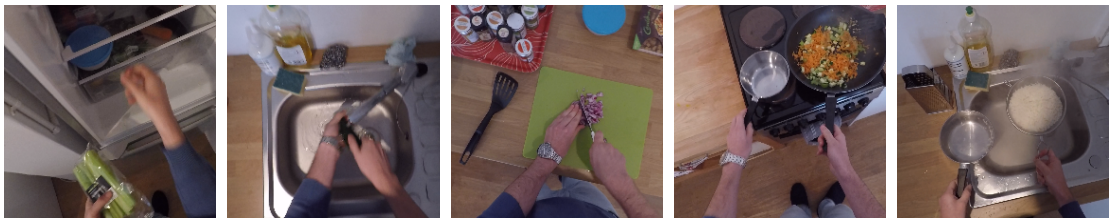
If time is structured, what is the ideal structure to represent it?

Chapter 4 tries to answer the question of how to ideally structure the time. But first, let us start with the following idea. The visual world around us is full of objects, each with a distinct shape or figure. The question is, if you look around the room you are in now, can you see the time as one of these integrated objects? Definitely not. So, if time is an agreed-upon concept that we use and experience in our daily life, why cannot we see it, and does it have a specific shape? Does in fit is a certain pattern or a does it have a unique mold? Is there a specific structure of time? And is it as simple as geometric shapes, *e.g.* triangle, square, or circle? Or does time need a more complex structure to be represented, such as trees, graphs, lattice, groups, or rings? In chapter 4, it is argued

that to better recognize the human activities in videos, the time is better to be cast as a graph, for the graph is the ideal structure to wrap the time of a seemingly endless human activity in video.

It is concluded from chapter 4 that the temporal structure of complex activities is weak. To model such a structure, VideoGraph is proposed as the ideal structure of time, where the main building blocks of this structure are the graph nodes. The nodes stand for the most discriminant visual evidence in a video of human action. In the graph structure, we also have the graph edges, representing the most important relationships between the visual evidence over time. While the graph structure succeeds in representing very long temporal patterns in videos, it falls short of tolerating the permutations in such patterns. So, to what extent is VideoGraph successful in tolerating the temporal perturbations of complex activities? In other words, this thesis asks next a more profound question:

Is time permutable in human activities? And how to achieve permutation invariance to recognize such activities?



(a) Get Food *(b) Wash Cucumber* *(d) Dice Meat* *(e) Cook Vegetables* *(f) Wash Rice*

Figure 4: A collection of images depicting the human activity of "cooking dinner".

Permutation is another important property of time. Consider the steps of preparing dinner, shown in the images of figure 4. It seems that some of the steps can be permuted without affecting the end-goal of preparing the dinner. For instance, one person might cook the rice before cooking the vegetables, another person might cook the meat first. However, we should also point out that some steps cannot be permuted. For example, one has to wash the cucumber, then cut it, and then cook it. Since real-life human actions do not follow a strict temporal order, we raise the question, what is the most suitable structure to represent time? Could it be a sequence, like arrays or lists, or a diagrammatical illustration, such as graphs, or trees, or lattices, or something completely different?

Chapter 5 focuses on a fundamental property of time, in the context of long-range human activities in videos. This property is the perturbation in the temporal structure of such activities. For example, a person might start the long-range activity of "make coffee" by "add milk" then "add coffee" and end up with "pour sugar". Another person might start first with "add milk", then "add coffee" and skip "pour sugar" altogether. These perturbations over time require some tolerance, or invariance, from the methods that learn to represent and classify the activities. To this end, a new temporal modeling layer is devised, namely Permutation Invariant Convolution, PIC. PIC is guided by three design principles *i.* invariance to permutation, *ii.* using shared kernels, and *iii.* respecting local connectivity. In addition, PIC addresses the shortcomings of three existing approaches for temporal modeling, namely vector-aggregation, self-attention, and convolution. Unlike

vector aggregation and self-attention, PIC respects local connectivity, thus is able to capture complex temporal patterns at multiple layers of abstractions. In contrast to self-attention, the kernels in PIC are shared and are not inferred from the input signal. Consequently, PIC is better than self-attention in detecting the most discriminant visual evidence from the noisy videos of long-range activities. After the experiments are carried out, it is concluded that PIC is better in modeling the complexities of long-range activities than the competing methods. Besides, PIC outperforms other methods in recognizing the activities of three benchmark Breakfast, MultiThumos, and Charades. Furthermore, by conducting quantitative and qualitative analysis, the design principles of PIC are thoroughly investigated, and their importance is confirmed.

Many neural models, such as those presented in chapters 2, 3, 4, and 5, are successful in detecting and recognizing human activities. Albeit this success comes at a great price; that is the high computational cost of such neural models. The main cause of this cost is the necessity to densely sample frames or segments from the video when recognizing its activity. That is because temporal redundancy, a fundamental problem of time, is not addressed by the aforementioned models. As a remedy, the thesis looks into better methods to handle the redundancy of visual signals across the time dimension. So, the thesis asks the following question:

How to overcome the redundancy of time, for the efficient recognition of human activities?

Chapter 6 sheds light on the task of understanding human activities in videos from a perspective different from the previous chapters. Rather than recognizing such activities, this chapter discusses the efficiency of already existing neural models in activity recognition. Generally, the model efficiency is materialized by four metrics, namely *i.* number of learning parameters in millions, *ii.* number of floating point operations (FLOPs), *iii.* feedforward time in milli-seconds, and *iv.* classification accuracy in percentage. This chapter, however, focuses on the trade-off between only two metrics, which are the FLOPs and the accuracy. In addition, TimeGate, a new method for the efficient recognition of long-range activities in videos, is proposed. Using TimeGate, realizing the efficiency is achieved by sampling the most representative segments from the activity's video. Then, only the sampled segments are considered for recognition, while all the other segments are discarded. Consequently, the recognition accuracy is retained at a fraction of the computational cost. TimeGate has two technical contributions. First, thanks to a carefully crafted gating mechanism, TimeGate is fully differentiable. Thus, it can be trained with existing video classifiers, such as 3D CNNs, in an end-to-end fashion. Second, the gating mechanism in TimeGate is context-conditional, which is considerably better for long-range activities. This is in contrast to the frame-conditional gating used by other methods, such as SCSampler. Context-conditioning means that when sampling a certain segment from a video, the visual evidence in both the segment and its context, *i.e.* the surrounding segments, are considered. While segment-conditioning means that the visual evidence of only the segment is used. In the end, in-depth analysis is conducted, and the benefits of TimeGate are verified. Besides, using qualitative and quantitative are experiments on three datasets, TimeGate outperforms related methods in reducing the computational cost of recognizing the long-range activities.

With chapter 7, this thesis comes to its proper ending. In this chapter, we summarize the lessons learned from the previous ones. Certainly, we emphasize on the importance of leveraging external sources of knowledge to improve the detection of events in zero-short learning (Chapter 2). In addition, we reiterate on the three properties of complex actions, and how they are addressed by Timeception (Chapter 3). Also, we recall how important it is to tolerate the temporal perturbations for better recognition of long-range activities (Chapter 5). Furthermore, we demonstrate how temporal selection can drastically reduce the high computational cost of neural models when recognizing minutes-long activities. Orthogonal to the conclusion, we briefly discuss five research topics. Not only do these topics remain open, but also they might forecast the progress in video understanding. The five topics are as follows.

- **New tasks:** is video recognition the correct task to understand human activities? Or other tasks are needed?
- **Structured representation:** can a single feature vector truly represent a human activity? Or is there are need for structure-oriented video representation?
- **Levels of human motion:** what are the differences between atomic action, unit action, one-action, action, activity, event ..., etc? What are the properties of each?
- **Multiple modalities:** is only the visual modality enough to understand videos? Or text and sound are also needed?
- **Semi-supervised learning:** is supervised learning sustainable? If unsupervised learning is even realistic? Or is self-supervised learning the way to consider?

1.1 RELATED WORK

There is a huge body of literature on video understanding, in general, and action recognition, in particular. Historically, motion in videos is represented as feature vectors using hand-crafted extractors. In static images, these extractors encode local points of interests, *i.e.* keypoints. An examples of such methods are SIFT [25], SRUF [9] and HOG [25]. Other extractors are more geared towards encoding motion in videos, for example DT [169], IDT [170] or optical flow [65]. In addition, vector aggregation methods are use to pool frame-based features and arrive at the video-level counterpart, for example BoVW [131], FV [145], VLAD [8]. To classify the actions represented by the aforementioned features, methods use feature classifiers, such as SVM [139, 156] and MLP [127]. While the majority of literature depend on the appearance images, *i.e.* RGB images, as the main source of data input, other sources are considered, such as optical flow [39], dynamic images [13], depth maps [167], and infrared images.

Recently, deep learning has successfully been employed to understand and recognize objects in images. The overwhelming majority of literature make use of Convolutional Neural Networks (CNNs) [40, 41, 105] as the de facto method of feature representation. CNNs are successful in detecting visual patterns in the spatial dimensions of images [62, 98]. Soon, they are extended to understand videos, by complementing the temporal dimension with the spatial counterparts. As a result, a new family of CNNs is devised, namely 3D CNNs [81, 158]. For action understanding and recognition, many successful methods stem from 3D CNNs, such as C3D [60], I3D [15], TSN [173], S3D [159], Non-local Networks [175], and many others. Complementary to CNNs, methods have been

proposed for temporal modeling of visual signal in videos. Examples are LSTM [107, 108], context gating [118], temporal segments [104], temporal convolutions [104], and feature banks [179].

The success of deep learning could not be possible without large-scale annotated datasets. Therefore, a large body of literature is dedicated to proposing and studying such datasets. In video understanding, actions and activities of these datasets span a wide spectrum of topics, such as sports, cooking, instructions, consumer, video-logging, activities of daily living, movies, TV series, etc. Examples of these datasets are UCF101 [148], Sports1M [90], Thumos [77], and ActivityNet [64]. For cooking, Breakfast [99], YouCookII [195], Epic-Kitchens [26], and MPII-Cooking [134] are good examples. For the interaction of human-human or human-object, there exist datasets such as Charades [144], Kinetics [91], YouTube8M [6], EventNet [187], FCVID [86], and YFCC100M [155]. Movies and TV series are a good source of harvesting video datasets. A few examples are Hollywood [113], HMDB [100], AVA [52], MPII-MD [133], and M-VAD [132].

1.2 LIST OF PUBLICATIONS

- **Chapter 2** is based on "Unified Embedding and Metric Learning for Zero-Exemplar Event Detection", published in *Computer Vision and Pattern Recognition (CVPR)*, 2017 [72], by Noureldien Hussein, Efstratios Gavves and Arnold W. M. Smeulders.

  Contribution of authors

  Noureldien Hussein: all aspects,
  Efstratios Gavves: guidance and technical advice,
  Arnold W. M. Smeulders: supervision and insight.

- **Chapter 3** is based on "TimeCeption for Complex Action Recognition", published in *Computer Vision and Pattern Recognition (CVPR)*, 2019 [72], by Noureldien Hussein, Efstratios Gavves and Arnold W. M. Smeulders.

  Contribution of authors

  Noureldien Hussein: all aspects,
  Efstratios Gavves: guidance and technical advice,
  Arnold W. M. Smeulders: supervision and insight.

- **Chapter 4** is based on "VideoGraph: Recognizing Minutes-Long Human Activities in Videos", published in *International Conference on Computer Vision Workshop (ICCV-W)*, 2019, in submission to *British Machine Vision Conference (BMVC)*, 2020 [74], by Noureldien Hussein, Efstratios Gavves and Arnold W. M. Smeulders.

  Contribution of authors

  Noureldien Hussein: all aspects,
  Efstratios Gavves: guidance and technical advice,
  Arnold W. M. Smeulders: supervision and insight.

- **Chapter 5** is based on "Permutation Invariant Convolution for Recognizing Long-range Activities", in submission to *European Conference on Computer Vision (ECCV)*, 2020 [75], by Noureldien Hussein, Efstratios Gavves and Arnold W. M. Smeulders.

  Contribution of authors

  Noureldien Hussein: all aspects,
  Efstratios Gavves: guidance and technical advice,
  Arnold W. M. Smeulders: supervision and insight.

- **Chapter 6** is based on "TimeGate: Conditional Gating of Segments in Long-range Activities", in submission to *European Conference on Computer Vision (ECCV)*, 2020 [76], by Noureldien Hussein, Mihir Jain and Babak Ehteshami Bejnordi.

  Contribution of authors

  Noureldien Hussein: all aspects,
  Mihir Jain: guidance and technical advice,
  Babak Ehteshami Bejnordi: guidance and technical advice.

2

UNIFIED EMBEDDING AND METRIC LEARNING FOR ZERO-EXEMPLAR EVENT DETECTION

2.1 INTRODUCTION

TRECVID Multimedia Event Detection (MED) [125, 126] is a retrieval task for event videos, with the reputation of being realistic. It comes in two flavors: few-exemplar and zero-exemplar, where the latter means that no video example is known to the model. Although expecting a few examples seems reasonable, in practice this implies that the user must already have an index of any possible query, making it very limited. In this work, we focus on event video search with zero exemplars.

Retrieving videos of never-seen events, such as "renovating home", without any video exemplar poses several challenges. One challenge is how to bridge the gap between the visual and the textual semantics [53, 55, 84]. One approach [16–18, 84, 111, 114] is to learn a dictionary of concept detectors on external data source. Then, scores for test videos are predicted using these detectors. Test videos are then ranked and retrieved accordingly. The inherent weakness of this approach is that the presentation of a test video is reduced to a limited vocabulary from the concept dictionary. Another challenge is how to overcome the domain difference between training and test events. While Semantic Query Generation (SQG) [17, 18, 84, 85] mitigates this challenge by extracting keywords from the event query, it does not address how relevant these keywords are to the event itself. For example, keyword "person" is not relevant to event "car repair" as it is to "flash mob gathering".

Our entry to zero-exemplar events is that they generally have strong semantic correlations [44, 116] with other possibly seen events. For instance, the novel event "renovating home" is related to "fit wall tiles", "remove drywall", or even to "paint door". Novel events can, therefore, be casted on a repository of prior events, for which knowledge sources in various forms are available beforehand, such as the videos, as in EventNet [187], or articles, as in WikiHow [1]. Not only do these sources provide video examples of a large –but still limited– set of events, but also they provide an association of text description of events with their corresponding videos. A text article can describe the event in words: what is it about, what are the details and what are the semantics. We note that such a visual-textual repository of events may serve as a knowledge source, by which we can interpret novel event queries.

For Zero-exemplar Event Detection (ZED), we propose a neural model with the following novelties:

Published in *Computer Vision and Pattern Recognition (CVPR)*, 2017 [72]

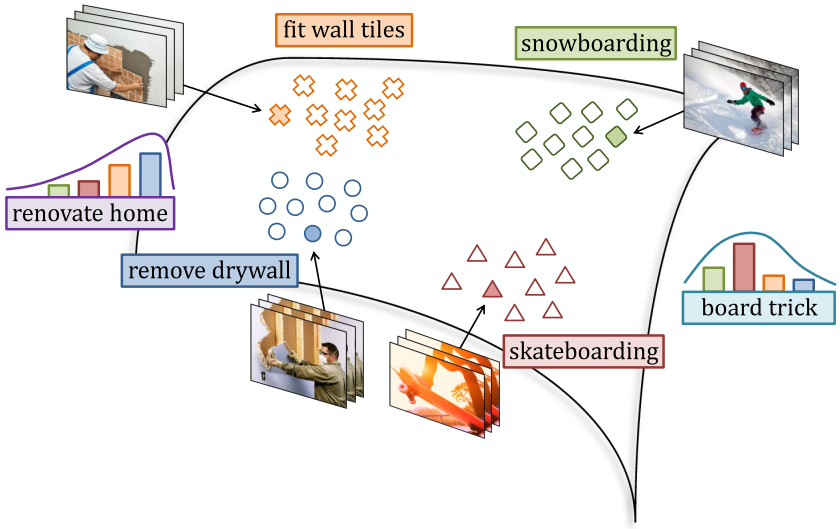


Figure 5: We pose the problem of zero-exemplar event detection as learning from a repository of pre-defined events. Given video exemplars of events "removing drywall" or "fit wall", one may detect a novel event "renovate home" as a probability distribution over the predefined events.

1. We formulate a unified embedding for multiple modalities (*e.g.* visual and textual) that enables a contrastive metric for maximum discrimination between events.
2. A textual embedding poses the representation of a novel event as a probability of predefined events, such that it spans a much larger space of admissible expressions.
3. We exploit a single data source, comprising pairs of event articles and related videos. A single source rather enables end-to-end learning from multi-modal individual pairs.

We empirically shows that our novelties result in performance improvement. We evaluate the model on TRECVID Multimedia Event Detection (MED) 2013 [125] and 2014 [126]. Our results show significant improvement over the state-of-the-art.

2.2 RELATED WORK

We identify three families of methods for ZED, as in figure 6 (a), (b) and (c).

Visual Embedding and Textual Retrieval. As in figure 6(a), given a video v_i represented as $x \in \mathcal{X}$ and a related text t represented as $y \in \mathcal{Y}$. Then, a visual model $f_{\mathcal{V}}$ is trained to project x as $y^v \in \mathcal{Y}$ such that the distance is minimized between (y^v, y). In test time, video ranking and retrieval is done using distance metric between the projected test video y^t and test query representation y.

VideoStory [54, 56] project the visual feature x of a web video v into term-vector representation y of the video's textual title t. However, during training, the model makes use of the text query of the test events to learn better term-vector representation. Consequently, this limits the generalization for novel event queries.

Textual Embedding and Visual Retrieval. As in figure 6(b), a given text query t is projected into $x^t \in \mathcal{X}$ using pre-trained or learned language model $f_{\mathcal{T}}$.

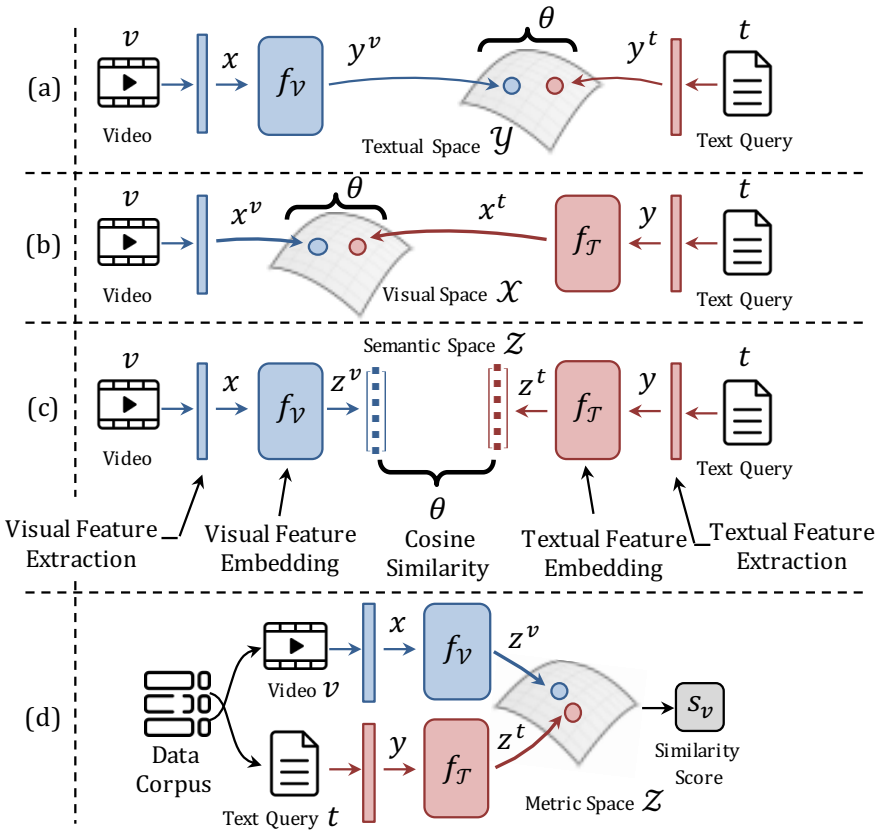


Figure 6: Three families of methods for zero-exemplar event detection: (a), (b) and (c). They build on top of feature representations learned a priori (i.e. initial representations), such as CNN features x for a video v or word2vec features y for event text query t. In a post-processing step, the distance θ is measured between the embedded features. In contrast, our model rather falls in a new family, depicted in (d), for it learns unified embedding with metric loss using single data source.

TagBook [115] makes use of freely-available weakly-tagged web videos. Then it propagates tags to test videos from its nearest neighbors. Methods [16–18, 84, 111] have similar approach. Given a text query t, Semantic Query Generation (SQG) extracts N most related concepts $\{c_i, i \in N\}$ to the test query. Then, pre-trained concept detectors predict probability scores $\{s_i, i \in N\}$ for a test video v. Aggregating these probabilities results in the final video score s_v, upon which videos are ranked and retrieved. [18] learns weighted averaging.

The shortcoming of this family is that expressing a video as probability scores of few concepts is under-representation. Any concept that exists in the video but is missing in the concept dictionary is thus unrepresented.

Visual-Textual Embedding and Semantic Retrieval. As in figure 6(c), visual $f_{\mathcal{V}}$ and textual $f_{\mathcal{T}}$ models are trained to project both of the visual x and textual y features into a semantic space $\mathcal{Z}$. During test, ranking score is the distance between the projections z^v, z^t in the semantic space $\mathcal{Z}$.

[180] projects video concepts into a high-dimensional lexicon space. Separately, it projects concept-based features to the space, which overcomes the lexicon mismatch between the query and the video concepts. [32] embeds a fusion of low and mid-level

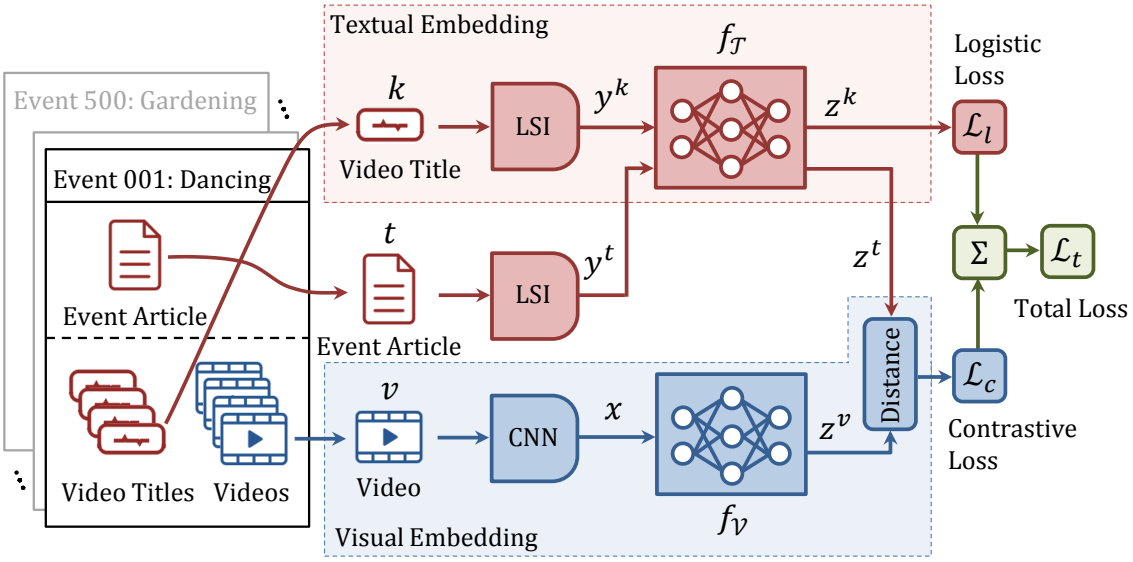


*Figure 7: Model overview. Using dataset $\mathcal{D}^z$ of M event categories and N videos. Each event has a text article and a few videos. Given a video x with text title k, belonging to an event with article t, we **extract** features x, y^k, y^t respectively. At the top, network $f_{\mathcal{T}}$ **learns** to classify the title feature y^k into one of M event categories. In the middle, we borrow the network $f_{\mathcal{T}}$ to **embed** the event article's feature y^t as $z^t \in \mathcal{Z}$. Then, at the bottom, the network $f_{\mathcal{V}}$ **learns** to embed the video feature x as $z^v \in \mathcal{Z}$ such that the distance between (z^v, z^t) is minimized, in the learned metric space $\mathcal{Z}$.*

visual features into distributional semantic manifold [119, 120]. In a separate step, it embeds text-based concepts into the manifold.

The third family, see figure 6(c), is superior to the others, see figure 6(a), (b). However, one drawback of [32, 180] is separately embedding both the visual and textual features z^v, z^t. This leads to another drawback, having to measure the distance between (z^v, z^t) in a post-processing step (*e.g.* cosine similarity).

Unified Embedding and Metric Learning Retrieval. Our method rather falls into a new family, see figure 6(d), and it overcomes the shortcomings of [32, 180] by the following. It is trained on a single data source, enabling a unified embedding for features of multiple modalities into a metric space. Consequently, the distance between the embedded features is measured by the model using the learned metric space.

Auxiliary Methods. Independent to the previous works, the following techniques have been used to improve the results: self-paced reranking [82], pseudo-relevance feedback [83], event query manual intervention [7], early fusion of features (action [36, 146, 157, 161, 169] or acoustic [89, 101, 122]) or late fusion of concept scores [56]. All these contributions may be applied to our method.

Visual Representation. ConvNets [63, 98, 105, 147] provide frame-level representation. To tame them into video-level counterpart, literature use: i- frame-level filtering [43] ii- vector encoding [8, 145] iii- learned pooling and recounting [111, 117] iv- average pooling [54, 56]. Also, low-level action [161, 169], mid-level action [146, 157] or acoustic [89, 101, 122] features can be used.

Textual Representation. To represent text, literature use: i- sequential models [150], ii- continuous word-space representations [103, 120], iii- topic models [14, 27], and iv- dictionary-space representation [56].

2.3 METHOD

2.3.1 *Overview*

Our goal is zero-exemplar retrieval of event videos with respect to their relevance to a novel textual description of an event. More specifically, for the zero-exemplar video dataset $\mathcal{D}^z = \{v_i^z\}, i = 1, \ldots, L$ and given any future, textual event description t^z, we want to learn a model $f(\cdot)$ that ranks the videos v_i^z according to the relevance to t^z, namely:

$$t^z : v_i^z \succ v_j^z \rightarrow f(v_i^z, t^z) > f(v_j^z, t^z). \tag{2.1}$$

2.3.2 *Model*

Since we focus on zero-exemplar setting, we cannot expect any training data directly relevant to the test queries. As such, we cannot directly optimize our model for the parameters $W_{\mathcal{T}}, W_{\mathcal{V}}$ in eq. (2.3). In the absence of any direct data, we resort to external knowledge databases. More specifically, we propose to cast future novel query descriptions as a convex combination of known query descriptions in external databases, where we can measure their relevance the database videos.

We start from a dataset $\mathcal{D}^z = \left\{v_i, k_i, l_j, t_j\right\},\ i = 1, \ldots, N,\ j = 1, \ldots, M$ organized by an event taxonomy, where we do not neither expect nor require the events to overlap with any future event queries. The dataset is composed of M events. Each event is associated with a textual, article description of the event, analyzing different aspects of it, such as: (i) the typical appearance of subjects and objects (ii) it's procedures (iii) the steps towards completing task associated with it. The dataset contains in total N videos, with v_i denoting the i-th video in the dataset with metadata k_i, *e.g.* the title of the video. A video is associated with an event label l_i and the article description t_i of the event it belongs to. Since multiple videos belong to the same event, they share the article description of such event.

The ultimate goal of our model is zero-exemplar search for event videos. Namely, provided unknown text queries by the user, we want to retrieve those videos that are relevant. We illustrate our proposed model during training in figure 7. The model is composed of two components, a textual embedding $f_{\mathcal{T}}(\cdot)$, a visual embedding $f_{\mathcal{V}}(\cdot)$. Our ultimate goal is the ranking of videos, $v_i \succ v_j \succ v_k$ with respect to their relevance to a query description, or in pairwise terms $v_i \succ v_j$, $v_j \succ v_k$ and $v_i \succ v_k$.

Let us assume a pair of videos v_i, v_j and query description t, where video v_i is more relevant to the query t than v_j. Our goal is a model that learns to put videos in the correct relative order, namely $(v_i, t) \succ (v_j, t)$. This is equivalent to a model that learns visual-textual embeddings such that $d_i^{tv} < d_j^{tv}$, where d_i^{tv} is the distance between visual-textual embeddings of (v_i, t), d_j^{tv} is the same for (v_j, t). Since we want to compare distances

between pairs $(v_i, t), (v_j, t)$, we pose the learning of our model as the minimization of a contrastive loss [23]:

$$\mathcal{L}_{con} = \frac{1}{2N} \sum_{i=1}^{N} h_i \cdot d_i^2 + (1 - h_i) \max(1 - d_i, 0)^2, \tag{2.2}$$

$$d_i = \| f_{\mathcal{T}}(t_i; W_{\mathcal{T}}) - f_{\mathcal{V}}(v_i; W_{\mathcal{V}}) \|_2, \tag{2.3}$$

where $f_{\mathcal{T}}(t_i; W_{\mathcal{T}})$ is the projection of the query description t_i into the **unified** metric space $\mathcal{Z}$ parameterized by $W_{\mathcal{T}}$, $f_{\mathcal{V}}(v_i; W_{\mathcal{V}})$ is the projection of a video v_i onto the same space $\mathcal{Z}$ parameterized by $W_{\mathcal{V}}$ and h_i a target variable that equals to 1 when the i-th video is relevant to the query description t_i and 0 otherwise. Naturally, to optimize eq. (2.2), we first need to define the projections $f_{\mathcal{T}}(\cdot; W_{\mathcal{T}})$ and $f_{\mathcal{V}}(\cdot; W_{\mathcal{V}})$ in eq. (2.3).

Textual Embedding. The textual embedding component of our model, $f_{\mathcal{T}}(\cdot; W_{\mathcal{T}})$, is illustrated in figure 7 (top). This component is dedicated to learn a projection of a textual input –including any future event queries t– on to the unified space $\mathcal{Z}$. Before detailing our model $f_{\mathcal{T}}$, however, we note that that the textual embedding can be employed not only with event article descriptions, but also with any other textual information that might be associated to the dataset videos, such as textual metadata. Although we expect the video title not to be as descriptive as the associated article, they may still be able to offer some discriminative information as previously shown [54, 56] which can be associated to the event category.

We model the textual embedding as a shallow (two layers) multi-layer perceptron (MLP). For the first layer we employ a `ReLU` nonlinearity. The second layer serves a dual purpose. First, it projects the article description of an event on the unified space $\mathcal{Z}$. This projection is *category-specific*, namely different videos that belong to the same event will share the projection. Second, it can project any *video-specific* textual metadata into the unified space. We, therefore, propose to embed the title metadata k_i, which is uniquely associated with a video, not an event category. To this end, we opt for `softmax` nonlinearity for the second layer, followed by an additional logistic loss term to penalized misprediction of titles m_i with respect to the video's event label y_i^j, namely

$$\mathcal{L}_{log} = \sum_{i=1}^{N} \sum_{j=1}^{M} -y_i^j \log f_{\mathcal{T}}^j(k_i; W_{\mathcal{T}}). \tag{2.4}$$

Overall, the textual embedding $f_{\mathcal{T}}^j$ is trained with a dual loss in mind. The first loss term, see eq. (2.2) (2.3) takes care that the final network learns event-relevant textual projections. The second loss term, see eq. (2.4), takes care that the final network does not overfit to the particular event article descriptions. The latter is crucial because the event article descriptions in $\mathcal{D}^z$ will not overlap with the future event queries, since we are in a zero-exemplar retrieval setting. As such, training the textual embedding to be optimal only for these event descriptions will likely result in severe overfitting. Our goal and hope is that the final textual embedding model $f_{\mathcal{T}}$ will capture both event-aware and video-discriminative textual features.

Visual Embedding. The visual embedding component of our model, $f_{\mathcal{V}}(\cdot; W_{\mathcal{V}})$, is illustrated in figure 7 (bottom). This component is dedicated to learn a projection from the visual input, namely the videos in our zero-exemplar dataset $\mathcal{D}^z$, into the unified metric space $\mathcal{Z}$. The goal is to project the videos belonging to semantically similar events; project them into a similar region in the space. We model the visual embedding $f_{\mathcal{V}}(v_i; W_{\mathcal{V}})$ using a shallow (two layers) multi-layer perceptron with `tanh` nonlinearities, applied to any visual feature for video v_i.

End-to-End Training. At each training forward-pass, the model is given a triplet of data inputs, an event description t_i, a related video v_i and video title k_i. From eq. (2.3) we observe that the visual embedding $f_{\mathcal{V}}(v_i; W_{\mathcal{V}})$ is encouraged to minimize its distance with the output of the textual embedding $f_{\mathcal{T}}(t_i; W_{\mathcal{T}})$. In the end, all the modules of the proposed model are differentiable. Therefore, we train our model in an end-to-end manner by minimizing the following objective

$$\begin{aligned} &\arg\min_{W_{\mathcal{V}}, W_{\mathcal{T}}} \ \mathcal{L}^{\mathcal{U}}, \\ &\mathcal{L}^{\mathcal{U}} = \mathcal{L}_{con} + \mathcal{L}_{log}. \end{aligned} \tag{2.5}$$

For the triplet input (v_i, t_i, k_i), we rely on external representations, since our ultimate goal is zero-exemplar search. Strictly speaking, a visual input v_i is represented as CNN [63] feature vector, while textual inputs t_i, k_i are represented as LSI [27] or Doc2Vec [103] feature vectors. However, given that these external representations rely on neural network architectures, if needed, they could also be further fine-tuned. We choose to freeze CNN and Doc2Vec modules to speed up training. Finally, in this work, we refer to our main model with unified embedding, as **model**$^{\mathcal{U}}$.

Inference. After training, we fix the parameters $(W_{\mathcal{V}}, W_{\mathcal{T}})$. At test time, we set our function $f(\cdot)$ from eq. (2.1) to be equivalent to the distance function from eq. (2.3). Hence, at test time, we compute the Euclidean distance in the learned metric space $\mathcal{Z}$ between the embeddings (z^v, z^t) of test video v and novel event description t, respectively.

2.4 EXPERIMENTS

2.4.1 *Datasets*

Before delving into the details of our experiments, first we describe the external knowledge sources we use.

Training dataset. We leverage videos and articles from publicly available datasets. EvenNet [187] is a dataset of ~90k event videos, harvested from YouTube and categorized into 500 events in hierarchical form according to the events' ontology. Each event category contains around 180 videos. Each video is coupled with a text title, few tags and related event's ontology.

We exploit the fact that all events in EventNet are harvested from WikiHow [1] – a website for *How-To* articles covering a wide spectrum of human activities. For instance:

"How to Feed a Dog" or "How to Arrange Flowers". Thus, we crawl WikiHow to get the articles related to all the events in EventNet.

Test dataset. As the task is zero-exemplar, the test sets are different from the training. While EventNet serves as the training, the following serve as the test: TRECVID MED-13 [125] and MED-14 [125]. In details, they are datasets of videos for events. They comprise 27k videos. There are two versions, MED-13 and MED-14 with 20 events for each. Since 10 events overlap, the result is 30 different events in total. Each event is coupled with short textual description (title and definition).

2.4.2 *Implementation Details*

Video Features. To represent a video v, we uniformly sample a frame every one second. Then, using ResNet [63], we extract `pool5` CNN features for the sampled frames. Then, we average pool the frame-level features to get the video-level feature x^v. We experiment different features from different CNN models: ResNet (`prob`, `fc1000`), VGG [147] (`fc6`, `fc7`), GoogLeNet [152] (`pool5`, `fc1024`), and Places365 [194] (`fc6`, `fc7`, `fc8`) except we find ResNet `pool5` to be the best. We only use ResNet `pool5` and we don't fuse multiple CNN features.

Text Features. We choose topic modeling [14,27], as it is well-suited for long (and sometimes noisy) text articles. We train LSI topic model [27] on Wikipedia corpus [2]. We experiment different latent topics ranging from 300 to 6000, expect we found 2500 to be the best. Also, we experiment other textual representations as LDA [14], SkipThoughts [94] and Doc2Vec [103]. To extract a feature from an event article k or video title t, first we preprocess the text using standard MLP steps: tokenization, lemmatization and stemming. Then, for k, t we extract 2500-D LSI features y^k, y^t, respectively. The same steps apply to MED text queries.

Model Details. Our visual and textual embeddings $f_{\mathcal{V}}(\cdot), f_{\mathcal{T}}(\cdot)$ are learned on top of the aforementioned visual and textual features (x^v, y^k, y^t). $f_{\mathcal{T}}(\cdot)$ is a 1-hidden layer MLP classifier with `ReLU` for hidden, `softmax` for output, `logistic` loss and `2500-2500-500` neurons for the input, hidden, and output layers, respectively. Similarly, $f_{\mathcal{V}}(\cdot)$ is a 1-hidden layer MLP regressor with `ReLU` for hidden, `contrastive` loss and `2048-2048-500` neurons for the input, hidden, and output layers, respectively. Our code is made public to support further research.

2.4.3 *Textual Embedding*

Here, we qualitatively demonstrate the benefit of the textual embedding $f_{\mathcal{T}}(\cdot)$. Figure 8 shows the similarity matrix between MED and EventNet events. Each dot represents how a MED event is similar to EventNet events. It shows that our embedding (right) is better than LSI (left) in mapping MED to EventNet events. For example, LSI wrongly maps "9: getting a vehicle unstuck" to "256: launch a boat" while our embedding correctly maps

`github.com/noureldien/unified_embedding`

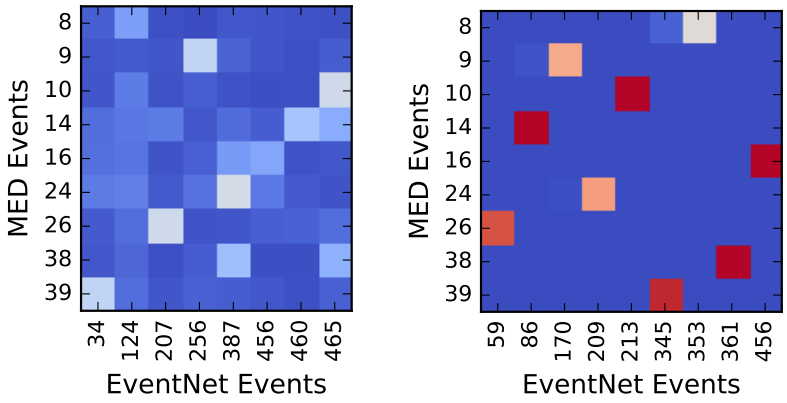


Figure 8: Our textual embedding (right) maps MED to EventNet events better than LSI features (left). Each dot in the matrix shows the similarity between MED and EventNet events.

it to "170: drive a car". Also, our embedding maps with higher confidence than LSI, as in "16: doing homework or study".

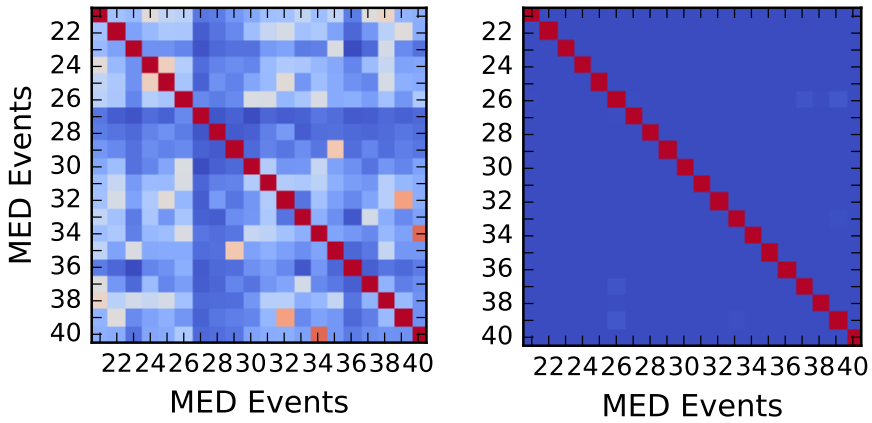


Figure 9: For 20 events of MED-14, our textual embedding (right) is more discriminant than the LSI feature representation (left). Each dot in the matrix shows how similar an event to all the others.

Figure 9 shows the similarity matrix for MED events, where each dot represents how related any MED event to all the others. Our textual embedding (right) is more discriminant than on the LSI feature representation (left). For example, LSI representation shows high semantic correlation between events "34: fixing musical instrument" and "40: tuning musical instrument", while our embedding discriminate them.

Next, we quantitatively demonstrate the benefit of the textual embedding $f_{\mathcal{T}}(\cdot)$. In contrast to the main model, see section 2.3, we investigate baseline **model**$^{\mathcal{V}}$, where we discard the textual embedding $f_{\mathcal{T}}(\cdot)$ and consider only the visual embedding $f_{\mathcal{V}}(\cdot)$. We project a video v on the LSI representation y of the related event t. Thus, this baseline falls in the first family of methods, see figure 6(a). It is optimized using mean-squared

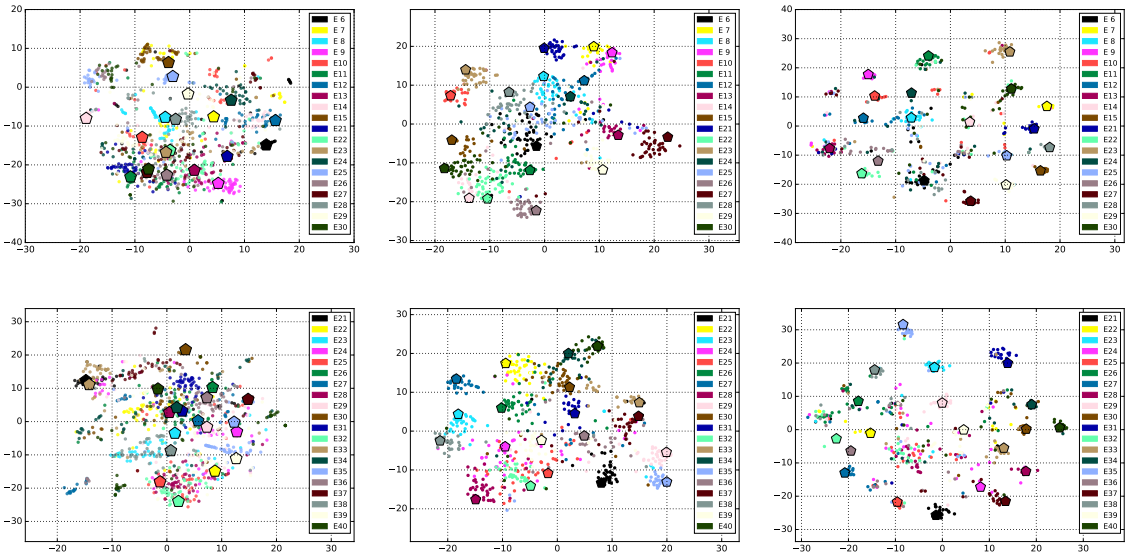


Figure 10: We visualize the results of video embedding using the unified embedding model$^{\mathcal{U}}$ and baselines model$^{\mathcal{V}}$, model$^{\mathcal{S}}$. Each sub-figure shows how discriminant the representation of the embedded videos. Each dot represents a projected video, while each pentagon-shape represents a projected event description. We use t-SNE to visualize the result.

error (MSE) loss $\mathcal{L}^{\mathcal{V}}_{mse}$, see eq. 2.6. The result of this baseline is reported in section 2.5, table 1.

$$\mathcal{L}^{\mathcal{V}}_{mse} = \frac{1}{N}\sum_{i=1}^{N} \|y_i - f_{\mathcal{V}}(v_i; W_{\mathcal{V}})\|_2^2. \tag{2.6}$$

Also, we train another baseline **model**$^{\mathcal{C}}$, which is similar to the aforementioned $^{\mathcal{V}}$ except instead of using MSE loss $\mathcal{L}^{\mathcal{V}}_{mse}$, see eq. (2.6), it uses contrastive loss $\mathcal{L}^{\mathcal{C}}_{con}$, as follows:

$$\begin{aligned} \mathcal{L}^{\mathcal{C}}_{con} &= \frac{1}{2N}\sum_{i=1}^{N} h_i \cdot d_i^2 + (1-h_i)\max(1-d_i, 0)^2, \\ d_i &= \|y_i - f_{\mathcal{V}}(v_i; W_{\mathcal{V}})\|_2. \end{aligned} \tag{2.7}$$

2.4.4 *Unified Embedding and Metric Learning*

In this experiment, we demonstrate the benefit of the unified embedding. In contrast to our model presented in section 2.3, we investigate baseline **model**$^{\mathcal{S}}$, where this baseline does not learn joint embedding. Instead, it separately learns visual $f_{\mathcal{V}}(\cdot)$ and textual $f_{\mathcal{T}}(\cdot)$ projections. We model these projections as a shallow (2-layer) MLP trained to classify the data input into 500 event categories, using logistic loss, same as eq. (2.4).

We conduct another experiment to demonstrate the benefit of learning metric space. In contrast to our model presented in section 2.3, we investigate baseline **model**$^{\mathcal{N}}$, where we discard the metric learning layer. Consequently, this baseline learns the visual embedding

is a shallow (2 layers) multi-layer perceptron with `tanh` non linearities. Also, we replace the contrastive loss $\mathcal{L}_c$, see eq. (2.2) with mean-squared error loss $\mathcal{L}_{mse}$, namely

$$\mathcal{L}^{\mathcal{N}}_{mse} = \frac{1}{N}\sum_{i=1}^{N} \|f_{\mathcal{T}}(t_i; W_{\mathcal{T}}) - f_{\mathcal{V}}(v_i; W_{\mathcal{V}})\|_2^2. \tag{2.8}$$

During retrieval, this baseline embeds a test video v_i and novel text query t_i as features z^v, z^t onto the common space $\mathcal{Z}$ using textual and visual embeddings $f_{\mathcal{T}}(\cdot), f_{\mathcal{V}}(\cdot)$, respectively. However, in a post-processing step, retrieval score s_i for the video v_i is the cosine distance between (z^v, z^t). Similarly, all test videos are scored, ranked and retrieved. The results of the aforementioned baselines model$^{\mathcal{S}}$ and model$^{\mathcal{N}}$ are reported in table 1.

Comparing Different Embeddings. In the previous experiments, we investigated several baselines of the unified embedding (model$^{\mathcal{U}}$), namely visual-only embedding (model$^{\mathcal{V}}$), separate visual-textual embedding (model$^{\mathcal{S}}$) and non-metric visual-textual embedding (model$^{\mathcal{N}}$). In a qualitative manner, we compare the results of such embeddings. As shown in figure 10, we use these baselines to embed event videos of MED-13 and MED-14 datasets into the corresponding spaces. At the same time, we project the textual description of the events on the same space. Then, we use t-SNE [112] to visualize the result on 2D manifold. As seen, the unified embedding, see sub-figures 10(c), 10(f) learns more discriminant representations than the other baselines, see sub-figures 10(a), 10(b), 10(d) and 10(e). The same observation holds for both MED-13 and MED-14 datasets.

2.4.5 *Mitigating Noise in EventNet*

Based of quantitative and qualitative analysis, we conclude that EventNet is noisy. Not only videos are unconstrained, but also some of the video samples are irrelevant to their event categories. EvenNet dataset [187] is accompanied by 500-category CNN classifier. It achieves top-1 and top-5 accuracies of 30.67% and 53.27%, respectively. Since events in EventNet are structured as an ontological hierarchy, there is a total of 19 high-level categories. The classifier achieves top-1 and top-5 accuracies of 38.91% and 57.67%, respectively, over these high-level categories.

Based on these observations, we prune EventNet to remove noisy videos. To this end, first we represent each video as average pooling of ResNet `pool5` features. Then, we follow the conventional 5-fold cross validation with 5 rounds. For each round, we split the dataset into 5 subsets, 4 subsets $\mathcal{V}_t$ for training and the last $\mathcal{V}_p$ for pruning. Then we train a 2-layer MLP for classification. After training, we forward-pass the videos of $\mathcal{V}_p$ and rule-out the mis-classified ones.

The intuition behind pruning is that we rather learn salient event concepts using less video samples than learn noisy concepts with more samples. Pruning reduced the total number of videos by 26%, from 90.2k to 66.7k. This pruned dataset is all what we use in our experiments.

2.4.6 *Latent Topics in LSI*

When training LSI topic model on Wikipedia corpus, a crucial parameter is the number of latent topics K the model constructs. We observe improvements in the performance directly proportional to increasing K. The main reason that the bigger the value of K, the more discriminant the LSI feature is. Figure 11 confirms our understanding.

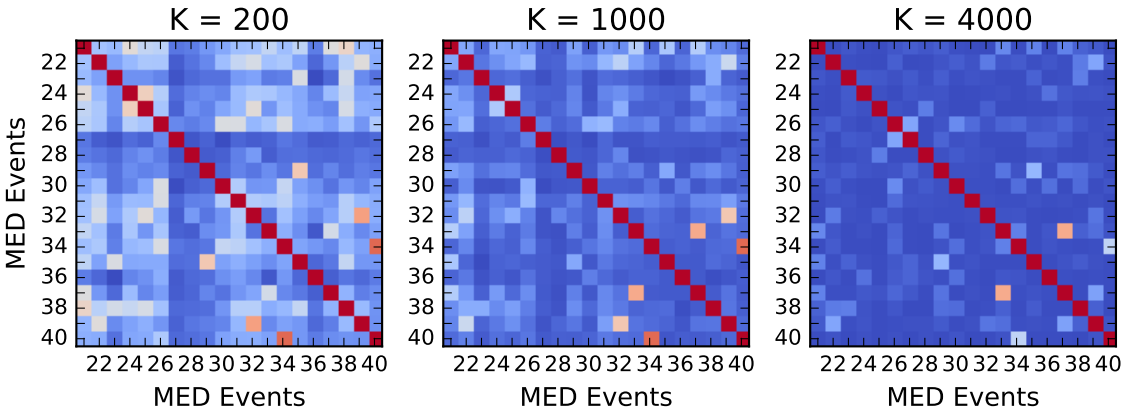


Figure 11: Similarity matrix between LSI features of MED-14 events. The more the latent topics (K) in LSI model, the higher the feature dimension, and the more discriminant the feature.

2.5 RESULTS

Evaluation metric. Since we are addressing, in essence, an information retrieval task, we rely on the average precision (AP) per event, and mean average precision (mAP) per dataset. We follow the standard evaluation method as in the relevant literature [88, 125, 126].

Comparing against model baselines. In table 1, we report the mAP score of our model baselines, previously discussed in the experiments, see section 2.4. The table clearly shows the marginal contribution of each of novelty for the proposed method.

Baseline	Loss		Metric	$f_{\mathcal{V}}(\cdot)$	$f_{\mathcal{T}}(\cdot)$	MED13	MED14
model$^{\mathcal{V}}$	$\mathcal{L}^{\mathcal{V}}_{mse}$	(2.6)	✗	✓	✗	11.90	10.76
model$^{\mathcal{C}}$	$\mathcal{L}^{\mathcal{C}}_{con}$	(2.7)	✓	✓	✗	13.29	12.31
model$^{\mathcal{S}}$	$\mathcal{L}_{log}$	(2.4)	✗	✓	✓	15.60	13.49
model$^{\mathcal{N}}$	$\mathcal{L}^{\mathcal{N}}_{mse}$	(2.8)	✗	✓	✓	15.92	14.36
model$^{\mathcal{U}}$	$\mathcal{L}^{\mathcal{U}}$	(2.5)	✓	✓	✓	17.86	16.67

Table 1: Comparison between the unified embedding and other baselines. The unified embedding model$^{\mathcal{U}}$ achieves the best results on MED-13 and MED-14 datasets. Metric means if the baseline uses metric-learning or not.

Comparing against related work. We report the performance of our method, the unified embedding model$^{\mathcal{U}}$ on TRECVID MED-13 and MED-14 datasets. When compared with the related works, our method improves over the state-of-the-art by a considerable margin, as shown in table 2 and figure 12.

Method		MED13	MED14
TagBook [115]	ToM '15	12.90	05.90
Discovary [16]	ICAI '15	09.60	–
Composition [17]	AAAI '16	12.64	13.37
Classifiers [18]	CVPR '16	13.46	<u>14.32</u>
VideoStory† [56]	PAMI '16	15.90	05.20
VideoStory* [56]	PAMI '16	**20.00**	08.00
This Work (model$^{\mathcal{U}}$)		<u>17.86</u>	**16.67**

Table 2: Performance comparison between our model and related works. We report the mean average precision (mAP%) for MED-13 and MED-14 datasets.

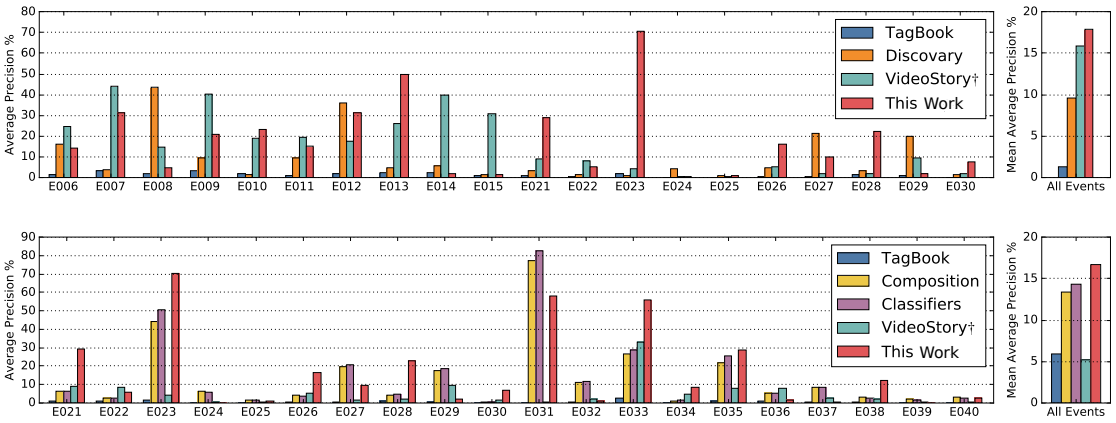


Figure 12: Event detection accuracies: per-event average precision (AP%) and per-dataset mean average precision (mAP%) for MED-13 and MED-14 datasets. We compare our results against TagBook [115], Discovary [16], Composition [17], Classifiers [18] and VideoStory [56].

It is important to point out that VideoStory† uses only object feature representation, so its comparable to our method. However, VideoStory* uses motion feature representation and expert text query (i.e. using term-importance matrix **H** in [56]). To rule out the marginal effect of using different datasets and features, we train VideoStory and report results in table 3. Clearly, CNN features and video exemplars in the training set can improve the model accuracy, but our method improves against VideoStory when trained on the same dataset and using the same features. Other works (Classifiers [18], Composition [17]) use both image and action concept classifiers. Nonetheless, our method improves over them using only object-centric CNN feature representations.

Method	Training Set	CNN	MED14
VideoStory	VideoStory46k [56]	GoogleNet	08.00
VideoStory	FCVID [87]	GoogleNet	11.84
VideoStory	EventNet [187]	GoogleNet	14.52
VideoStory	EventNet [187]	ResNet	15.80
This Work	EventNet [187]	ResNet	16.67

Table 3: Our method improves over VideoStory when trained on the same dataset and using the same feature representation.

2.6 CONCLUSION

In this work, we presented a novel approach for detecting events in unconstrained web videos, in a zero-exemplar fashion. Rather than learning separate embeddings form cross-modal datasets, we proposed a unified embedding where several cross-modalities are jointly projected. This enables end-to-end learning. On top of this, we exploited the fact that zero-exemplar is posed as retrieval task and proposed to learn metric space. This enables measuring the similarities between the embedded modalities using this very space.

We experimented the novelties and demonstrated how they contribute to improving the performance. We complemented this by improvements over the state-of-the-art by considerable margin on MED-13 and MED-14 datasets.

However, the question still remains, how can we discriminate between these two MED events “34: fixing musical instrument” and “40: tuning musical instrument”. We would like to argue that temporal modeling for human actions in videos is of absolute necessity to achieve such fine-grained event recognition. In future research, we would like to focus on human-object interaction in videos and how to model it temporally.

3

TIMECEPTION FOR COMPLEX ACTION RECOGNITION

3.1 INTRODUCTION

In ordinary life, activities of daily living pop up frequently. Our conversations include actions like "cooking a meal" or "cleaning the house" much more frequently than actions like "jumping" or "cutting a cucumber". The latter, which we call *one-actions*, exhibit one visual pattern, possibly repetitive. They are usually short in time, homogeneous in motion and coherent in form. In contrast, cooking a meal or cleaning the house are very different actions. We refer to them as *complex actions*, characterized by: *i.* They are typically composed of several one-actions, see figure 13. *ii.* These one-actions, contained in a complex action, exhibit large variations in their temporal duration and temporal order. *iii.* As a consequence of the composition, a complex action takes much longer to unfold. And, by the in-homogeneity in composition, the complex action needs to be sampled in full, not to miss crucial parts.

In the recent literature, the main focus is the recognition of short-range actions like in HMDB, UCF and Kinetics [91, 100, 148]. Few attention has been paid to the recognition of long-range and complex actions, as in Charades and EventNet [144, 187], which we study here. The first challenge is minute-long temporal modeling while maintaining attention to seconds-long details. Statistical temporal pooling, as applied in [49, 118] falls short of learning temporal order. Neural temporal modeling [29, 50] and spatio-temporal convolutions of various types [81, 158, 160] successfully learns temporal order of 8 [15] or 128 timesteps [176]. But the computational cost is far beyond scaling up to 1000 timesteps needed for complex actions. The second challenge is tolerating variations in temporal extent and temporal order of one-actions. Related methods [158, 182] learn spatio-temporal convolutions with fixed-size kernels, which would be too rigid for complex actions. To address these challenges, we present Timeception, a novel convolutional layer dedicated only for temporal modeling. It learns long-range temporal dependencies with attention to short-range details. Plus, it tolerates the differences in temporal extent of one-actions comprising the complex action. As a result, we demonstrate success in recognizing the long and complex actions, and achieving state-of-the-art-results in Charades [144], Breakfast Actions [99] and MultiTHUMOS [189].

The novelties of of this work are: *i.* We introduce a convolutional temporal layer effectively and efficiently learn minute-long action ranges of 1024 timesteps, a factor of 8 longer than best related work. *ii.* We introduce multi-scale temporal kernels to account for

Published in *Computer Vision and Pattern Recognition (CVPR)*, 2019 [73]

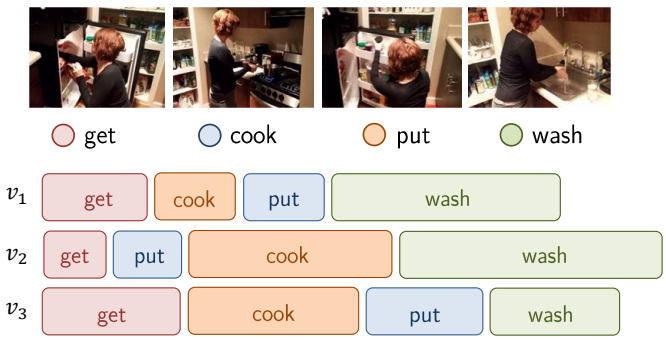


Figure 13: Given three video examples v_1, v_2, v_3 of a complex human action "Cooking a Meal", one can summarize the properties of this complex action as: ***i. composition****: consists of several one-actions (Cook, ...),* ***ii. order****: weak temporal order of one-actions (Get Wash),* ***iii. extent****: one-actions vary in their temporal extents.*

large variations in duration of action components. *iii*. We use temporal-only convolutions, which are better suited for complex actions than spatiotemporal counterparts.

3.2 RELATED WORK

Temporal Modeling. The stark difference between video and image classification is the temporal dimension, which necessitates temporal modeling. A widely used approach is statistical pooling: max and average pooling [57, 72], attention pooling [49], rank pooling [37], dynamic images [13] and context gating [118], to name a few. Beyond statistical pooling, vector aggregation is also used. [124] uses Fisher Vector [135] to aggregate spatio-temporal features over time, while [24, 31, 50] extend VLAD [8] to use local convolution features extracted from video frames. The downside of statistical pooling and vector aggregation is completely neglecting temporal patterns – an important visual cue.

Other strands of work use neural methods for temporal modeling. LSTMs are used to model the sequence in action videos [29]. While TA-DenseNet [46] extents DenseNet [71] to exploit the temporal dimension. To our knownedge, no substantial improvements have been reported recently.

Short-range Action Recognition. Few works [90] learn deep appearance features by frame-level classification of actions, using 2D CNNs. Others complement deep appearance features with shallow motion features, as IDT [35]. Also, auxiliary image representations are fused with RGB signals: [146] uses OpticalFlow channels, while [12] uses Dynamic Images. 3D CNNs are the natural evolution of their 2D counterparts. C3D [81, 158] proposes 3D CNNs to capture spatio-temporal patterns of 8 frames in a sequence. In the same vein, I3D [15] inflates the kernels of ImageNet-pretrained 2D CNN to jump-start the training of 3D CNNs. While effective in short-range video sequences of few seconds, 3D convolutions are too computationally expensive to address minute-long videos, which is our focus.

Long-range Action Recognition. To learn long-range temporal patterns, [142] uses CRF on top of CNN feature maps to model human activities. To learn video-wide

representations, TRN [193] learns relations between several video segments. TSN [172, 173] learns temporal structure in long videos. LTC [163] considers different temporal resolutions as a substitute to bigger temporal windows. Inspired by self-attention [164], non-local networks [176] proposes a 3D CNN with a long temporal footprint of 128 timesteps.

All aforementioned methods succeed in modeling temporal footprint of 128 timesteps (~4-5 sec) at max. In this work, we address complex actions with long-range temporal dependencies of up to 1024 timesteps, jointly.

Convolution Decomposition. CNNs succeed in learning spatial [90, 171] and spatiotemporal [34, 60, 81, 158, 163] action concepts, but existing convolutions grow heavy in computation, specially at the higher layers where the number of channels can grow as much as 2k [62]. To control the computational complexity, several works propose the decomposition of 2D and 3D convolutions. Xception [21] argues that separable 2D convolutions are as effective as typical 2D convolutions. Similarly, S3D [160, 182] considers separable 2+1D convolutions to reduce the complexity of typical 3D convolutions. ResNet [62] reduces the channel dimension using 1×1 2D convolution before applying the costly 3×3 2D spatial convolution. ShuffleNet [191] models cross-channel correlation by channel shuffling instead of 1×1 2D convolution. ResNeXt [181] proposes grouped convolutions, while Inception [152, 153] replaces the fixed-size 2D spatial kernels into multi-scale 2D spatial kernels of different sizes.

In this work, we propose the decomposition of spatiotemporal convolutions into depthwise-separable temporal convolutions, which we show to be better suited for long-range temporal modeling that 2+1D convolutions. Moreover, to account for the differences in temporal extents, we propose temporal convolutions with multi-scale kernels.

3.3 METHOD

3.3.1 *Motivation*

Modern 3D CNNs learn spatiotemporal kernels over three orthogonal subspaces of video information: the temporal ($\mathcal{T}$), the spatial ($\mathcal{S}$) and the semantic channel subspace (C). One spatiotemporal kernel $w \in \mathbb{R}^{T \times L \times L \times C}$ learns a latent concept by simultaneously convolving these three subspaces [15, 158], where T is the number of timesteps, C is the number of channels, and L is the size of spatial window. Though, there is no fundamental reason why these subspaces must be convolved simultaneously. Instead, as showcased in [182], one can model these subspaces separately, $w \propto w_s \times w_t$, by decomposing w into spatial $w_s \in \mathbb{R}^{1 \times L \times L \times C}$ and temporal $w_t \in \mathbb{R}^{T \times 1 \times 1 \times C}$ kernels. Strictly speaking, while replacing w with a cascade $\tilde{w} = w_s \times w_t$ is often referred to as *"decomposition"*, this operation is not tensor decomposition – there is no strict requirement that, at optimality, we have $w^* \equiv \tilde{w}^*$. Instead, as the cascade $\tilde{w}$ is, by definition, computationally more efficient than the full kernel w, the only practical requirement is that the resulting cascade $\tilde{w}$ yields equally good or better accuracies for the task at hand. In light of this realization, while the aforementioned decomposition along the spatial and temporal axes is intuitive and empirically successful [182], it is not the only possibility. Therefore, Any other

decomposition is permissible, namely: $\tilde{w} = w_\alpha \times w_\beta \times w_\gamma \times ...$, as long as some basic principles are maintained for the final cascade $\tilde{w}$. Generalizing on recent decomposed architectures [21, 160], we identify from the literature three intuitive design principles for the spatiotemporal CNNs:

***i*. Subspace Modularity.** In the context of deep network cascades, a decomposition should be modular, such that between subspaces, it retains the nature of the respective subspaces across subsequent layers. Namely, after a cascade of spatial and a temporal convolutions, it must be possible that yet another cascade (of spatial and temporal convolutions) is possible and meaningful.

***ii*. Subspace Balance.** A decomposition should make sure that a balance is retained between the subspaces and their parameterization in different layers. Namely, increasing the number of parameters for modeling a specific subspace should come at the expense of reducing the number of parameters of another subspace. A typical example is conventional 2D CNN, in which the spatial subspace ($\mathcal{S}$) is reduced while the semantic channel subspace (C) is expanded.

***iii*. Subspace Efficiency.** When designing the decomposition for a specific task, we should make sure that the bulk of the available parameter budget is dedicated to subspaces that are directly relevant to the task at hand. For instance, for long-range temporal modeling, a logical choice is a decomposition that increases the convolutional parameters for the temporal subspace ($\mathcal{T}$).

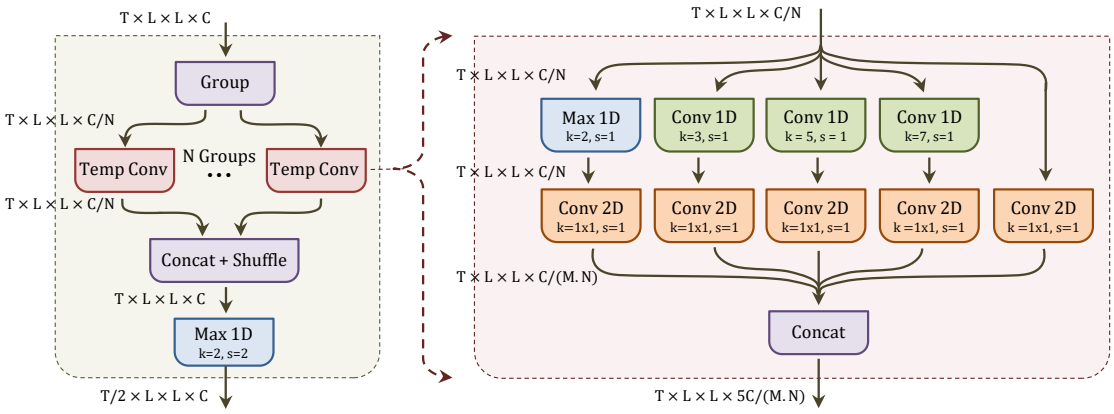
Motivated by the aforementioned design principles, we propose a new temporal convolution layer for encoding long-range patterns in complex actions, named Timeception, see figure 14. First, we discuss the Timeception layer. Then we describe how to stack Timeception layers on top of existing 2D or 3D CNNs.

3.3.2 *Timeception Layer*

For modeling complex actions in long videos, our temporal modeling layer faces two objectives. First, we would like to learn the possible *long-range temporal dependencies* between one-actions throughout the entire video, and for a frame sequence of up to 1000 timesteps. Second, we would like to *tolerate the variations in the temporal extents* of one-actions throughout the video.

Next, we present the Timeception layer, designed with these two objectives in mind. Timeception is a layer that sits on top of either previous Timeception layers, or a CNN. The CNN can be either purely spatial; processing frames independently, like ResNet [62], or short-range spatiotemporal; processing nearby bursts of frames, like I3D [15].

Long-range Temporal Dependencies. There exist two design consequences for modeling long-range temporal dependencies between one-actions throughout the video. The first consequence is that our temporal network must be composed of deeper stacks of temporal layers. Via successive layers, thereafter, complex and abstract spatiotemporal patterns can emerge, even when they reside at temporally very distant locations in the video. Given that we need deeper temporal stacks and we have a specific parameter



(a) Timeception Layer (b) Temporal Conv Module

Figure 14: The core component of our method is Timeception layer, left. Simply, it takes as an input the features $\mathbf{X}$*; corresponding to* T *timesteps from the previous layer in the network. Then, it splits them into* N *groups, and temporally convolves each group using temporal convolution module, right. It is a novel building block comprising multi-scale temporal-only convolutions to tolerate a variety of temporal extents in a complex action. Timeception makes use of grouped convolutions and channel shuffling to learn cross-channel correlations efficiently than* 1×1 *spatial convolutions.*

budget for the complete model, the second consequence is that the temporal layers must be as cost-effective as possible.

Revisiting the cost-effectiveness of spatiotemporal models, existing architectures rely either on joint spatiotemporal kernels [15] with parameter complexity $O(T \cdot L^2 \cdot C)$ or decomposed spatial and temporal kernels [160, 182] with parameter complexity $O((L^2 + T) \cdot C)$. To make the Timeception layer temporally cost-effective, according to the third design principle of *subspace importance*, we opt for trading spatial and semantic complexity for longer temporal windows. Specifically, we propose depthwise-separable temporal convolution with kernel $w_t^{TC} \in \mathbb{R}^{T \times 1 \times 1 \times 1}$. Hereafter, we refer to this convolution as *temporal-only*. What is more, unlike [15, 160, 182], we propose to focus only on temporal modeling and drop the spatial kernel $w_s \in \mathbb{R}^{1 \times L \times L \times C}$ altogether. Hence, the Timeception layer relies completely on the preceding CNN for the detection of any spatial pattern.

The simplified *temporal-only* kernel has some interesting properties. Each kernel acts on only one channel. As the kernels do not extend to the channel subspace, they are encouraged to learn generic and abstract, rather than semantically-specific, temporal combinations. For instance, the kernels learn to detect the temporal pattern of one *latent concept* represented by one channel. Last, as the parameter complexity of a single Timeception layer is approximately $O(T + logL)$, it is computationally feasible to train a deep model to encode temporal patterns of up to 1024 timesteps. This amounts to about 40 seconds of video sequences.

Unfortunately, by stacking *temporal-only* convolutions one after the other, we violate the first design principle of *subspace modularity*. The reason is that the semantic subspace in long-range spatiotemporal patterns is ignored. To this end, we propose to use *channel grouping* operation [181] before the temporal-only convolutions and *channel shuffling* operation [191] after the temporal-only convolutions. The purpose of channel grouping

is reducing the complexity of cross-channel correlations, by modeling it separately for each group. Clearly, as each group contains a random subset of channels, not all possible correlations are accounted for. This is mitigated by channel shuffling and channel concatenation, which makes sure that the channels are grouped altogether albeit in a different order. As such, the next Timeception layer will group a different subset of channels. Together, channel grouping and channel shuffling is more cost-effective operation to learn cross-channel correlations than 1× 2D convolutions [21].

Tolerating Variant Temporal Extents. The second objective for the Timeception layer is to tolerate the differences in temporal extents of complex actions. While in the previous description we assume a fixed length for the *temporal-only* kernels, one-actions in a complex video may vary in length. To this end, we propose to replace fixed-size temporal kernels with multi-scale temporal kernels. There are two possible ways to implement multi-scale kernels, see figure 15. The first way, inspired by Inception [152] for images, is to adopt K kernels, each of a different size k. The second way, inspired by [162], is to employ dilated convolutions.

The temporal convolution module, see figure 14(b), takes as an input the features of one group $\mathbf{X}_n \in \mathbb{R}^{T \times L \times L \times [C/N]}$. Then it applies five temporal operations in total. The first three operations are temporal convolutions with kernel sizes $k = \{3, 5, 7\}$, each maintaining the number of channels at C/N. The forth operation is a temporal max-pooling with stride $s = 1$ and kernel size $k = 2$. Its purpose is to max-out activations over local temporal window $(k = 2)$, instead of convolving them. The fifth operation is simply a dimension reduction for the input feature $\mathbf{X}_n$, using a 1×1 spatial convolution. To maintain a manageable number of dimensions for the output, the input to the first fours operations are shrinked by a factor of M using a 1×1 spatial convolution . After the channel reduction, all five outputs are concatenated across channel dimension, resulting in an output $\mathbf{Y}_n \in \mathbb{R}^{T \times L \times L \times (5C/MN)}$.

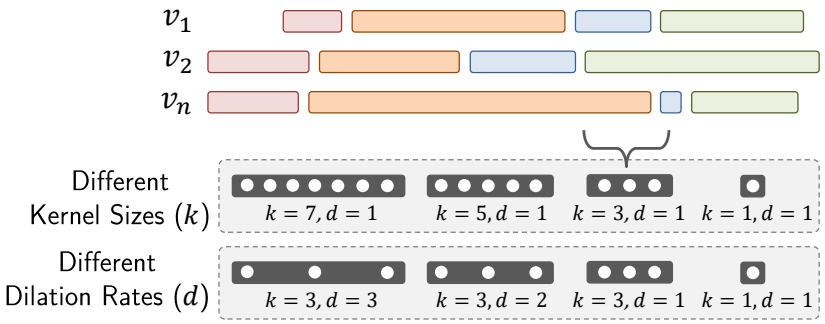


Figure 15: To tolerate temporal extents, we use multi-scale temporal kernels, with two options: i. different kernel sizes $k \in \{1, 3, 5, 7\}$ and fixed dilation rate $d = 1$, ii. different dilation rates $d \in \{1, 2, 3\}$ and fixed kernel size $k = 3$.

Summary of Timeception. A Timeception layer, see figure 14(a), expects an input feature $\mathbf{X} \in \mathbb{R}^{T \times L \times L \times C}$ from the previous layer in the network. The features $\mathbf{X}$ across the channel dimension are then split into N channel groups. Each group $\mathbf{X}_n \in \mathbb{R}^{T \times L \times \times L \times [C/N]}$ is convolved with the *temporal convolution module*, resulting in $\mathbf{Y}_n \in \mathbb{R}^{T \times L \times \times L \times [5C/MN]}$. This module expands the number of channels per group by a factor of $5/M$. After

that, the features of all groups $\mathbf{Y} = \{\mathbf{Y}_n \mid n \in [1, ..., N]\}$ are concatenated across the channel axis and then randomly shuffled. Last, to adhere to the second design principle of *subspace balance*, the Timeception layer concludes with a temporal max pooling of kernel size $k = 2$ and stride $s = 2$. The reason is that while the channel subspace expands by a factor of $5/M$ after each Timeception layer, the temporal subspace shrinks by a factor of 2.

3.3.3 *The Final Model*

The final model consists of four Timeception layers stacked on top of the last convolution layer of a CNN, used as backbone. We explore two backbone choices: a spatial 2D CNN and a short-range spatiotemporal 3D CNN.

2D CNN. The first baseline uses ResNet-152 [62] as backbone. It takes as an input 128 video frames, and processes them, up to the last spatial convolution layer `res5c`. Thus, the corresponding output for the input frames is the feature $\mathbf{X} \in \mathbb{R}^{128\times7\times7\times2048}$. Then, we proceed with four successive layers of Timeception, with BatchNorm and ReLU. Each has channel expansion factor of $5/M = 5/4 = 1.25, M = 4$ and temporal reduction factor of 2. Thus, the resulting feature is $\mathbf{Y} \in \mathbb{R}^{8\times7\times7\times5000}$. To further reduce the spatial dimension, we follow the convention of CNNs by using spatial average pooling, which results in the feature $\mathbf{Y}' \in \mathbb{R}^{8\times5000}$. And to finally reduce the temporal dimension, we use depthwise-separable temporal convolution with kernel size $k \in \mathbb{R}^{8\times1\times1\times1}$ with no zero-padding. The resulted feature $\mathbf{Z} \in \mathbb{R}^{5000}$ is classified with a two-layer MLP, with BatchNorm and ReLU.

3D CNN. The second baseline uses I3D [15] as backbone. It takes as an input 128 video segments (each has 8 successive frames), and independently processes these segments, up to the last spatiotemporal convolution layer `mixed-5c`. Thus, the corresponding output for the input segments is the feature $\mathbf{X} \in \mathbb{R}^{128\times7\times7\times1024}$. The rest of this baseline is no different than the previous one. The benefit of using I3D is that the Timeception layers learn long-range temporal combinations of short-range spatiotemporal patterns.

Implementation. When training the model on a specific dataset, first we pretrain the backbone CNN on this dataset. We use uniformly sampled frames for the 2D backbone and uniformly sampled video segments (each has 8 successive frames) for the 3D backbone. After pre-training, we plug-in Timeception and MLP layers on top of the last convolution layer of the backbone and fine-tune the model on the same dataset. At this stage, only Timeception layers are trained, while the backbone CNN is frozen. The model is trained with batch-size 32 for 100 epoch. It is optimized with SGD with 0.1, 0.9 and $1e$-5 as learning rate, momentum and weight decay, respectively. Our public implementation [4] uses TensorFlow [5] and Keras [22].

3.4 EXPERIMENTS

3.4.1 *Datasets*

The scope of this work is complex actions with their three properties: composition, temporal extent and temporal order –see figure 13. Thus, we choose to conduct our experiments on Charades [144], Breakfast Actions [99] and MultiTHUMOS [189]. Other infamous datasets for action recognition do not meet the properties of complex actions.

Charades is multi-label, action classification, video dataset with 157 classes. It contains 8k, 1.2k and 2k videos for training, validation and test splits, respectively (67 hrs for training split). On average, each complex action (*i.e.* each video) is 30 seconds and contains 6 one-actions. Thus, Charades meets the criteria of complex actions. We use mean Average Precision (mAP) for evaluation. As labels of test set are held out, we report results on the validation set, similar to all related works [50, 142, 142, 176, 177].

Breakfast Actions is a dataset for unscripted cooking-oriented human activities. It contains 1712 videos in total, 1357 for training and 335 for test. The average length of videos is 2.3 minutes. It is a video classification task of 12 categories of breakfast activities, where each video represents only one activity. Besides, each video has temporal annotation of one-actions composing its activity. In total, there are 48 classes of one-actions. In our experiments, we only use the activity annotation, and we do not use the temporal annotation of the one-actions.

MultiTHUMOS is a dataset for human activities in untrimmed videos, with the primary focus on temporal localization. It contains 65 action classes and 400 videos (30 hrs). Each video can be thought of a complex action, which comprises 11 one-actions on average. MultiTHUMOS extends the original THUMOS-14 [77] by providing multi-label annotation for the videos in validation and test splits. Having multiple and dense labels for the video frames enable temporal models to benefit from the temporal relations between one-actions across the video. Similar to Charades, mAP is used for evaluation.

3.4.2 *Tolerating Temporal Extents*

In this experiment, we evaluate the capacity of the multi-scale kernels to tolerate the differences in temporal extents of actions. The experiment is carried out on Charades.

Original v.s. Altered Temporal Extents First, we train two baselines, one with multi-scale temporal kernels (as in Timeception) and the other with fixed-size kernels. The training is done on the original temporal extent of training videos. Then, *at test time only*, we alter the temporal extents of test videos. Specifically, we split each test video into segments. Then, we temporally expand or shrink these segments. Expansion is done by repeating frames, while shrinking is done by dropping frames. We use 4 types of alterations with varying granularity to test the model in different scenarios: (a) very-coarse, (b) coarse, (c) fine, and (d) very-fine, see in figure 16.

The results of this controlled experiment are shown in table 4. We observe that Timeception is more effective than fixed-size kernels in handling unexpected variations

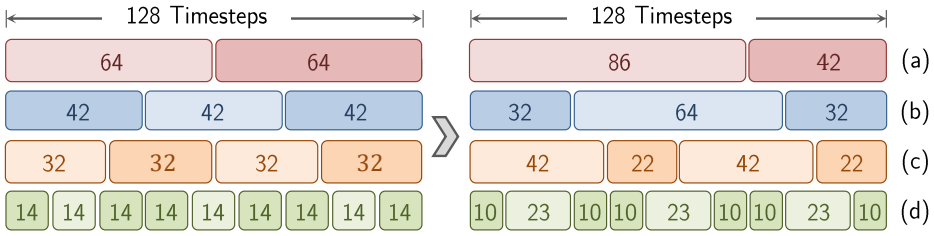


Figure 16: We split a video of 128 timesteps into segments of equal length (left, before alteration), and alter their temporal extents by expansion and shrinking (right, after alteration). We use 4 types of alterations:(a) very-coarse, (b) coarse, (c) fine, and (d) very-fine. Numbers in boxes are timesteps.

Altered Extent	Percentage Drop ↓ in mAP			
	I3D		ResNet	
	Fixed ↓	Multi ↓	Fixed ↓	Multi ↓
(a) very-coarse	2.09	1.75	1.52	1.08
(b) coarse	2.92	2.44	3.26	2.15
(c) fine	1.74	1.12	1.59	1.13
(d) very-fine	2.18	1.71	1.38	1.20

Table 4: Timeception, with multi-scale kernel, tolerates the altered temporal extents better than fixed-size kernels. We report the percentage drop in mAP (lower is better) when testing on original v.s. altered videos of Charades. I3D and ResNet are backbone CNNs.

in temporal extents. The same observations is confirmed using either I3D or ResNet as backbone architecture.

Fixed-size vs. Multi-scale Temporal Kernels This experiment points out the merit of using multi-scale temporal kernels. For this, we compare fixed-size temporal convolutions against multi-scale temporal-only convolutions, either with different kernel sizes k or dilation rates d. And we train 3 baseline models with different configurations of k, d: *i*. Fixed kernel size and fixed dilation rate $d = 1, k = 3$. This is the typical configuration used in 3D CNNs [15, 158, 176, 182]. *ii*. Different kernel sizes $k \in \{1, 3, 5, 7\}$ and fixed dilation rate $d = 1$. *iii*. Fixed kernel size $k = 3$ and different dilation rates $d \in \{1, 2, 3\}$.

The result of this experiment are shown in table 5. We observe that using multi-scale kernels is better suited for modeling complex actions than fixed-size kernels. The same observation holds for both I3D and ResNet as backbones. Also, we observe little to no change in performance when using different dilation rates d instead of different kernel sizes k.

3.4.3 *Long-range Temporal Dependencies*

In this experiment, we demonstrate the capacity of multiple Timeception layers to learn long-range temporal dependencies for complex actions. We train several baseline

Kernel Type	Kernel Size (k)	Dilation Rate (d)	mAP (%)	
			ResNet	I3D
Multi-scale	1,3,5,7	1	30.82	33.76
	3	1,2,3	30.37	33.89
Fixed-size	3	1	29.30	31.87

Table 5: Timeception, using multi-scale kernels (i.e. different kernel sizes (k) or dilation rates (d), outperforms fixed-size kernels on Charades. I3D/ResNet are backbone.

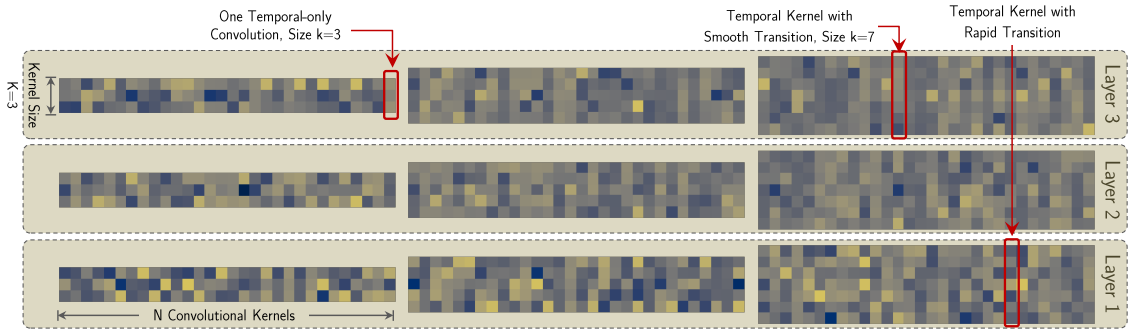


Figure 17: The learned weights by temporal convolutions of three Timeception layers. Each uses multi-scale convolutions with varying kernel sizes $k \in \{3, 5, 7\}$. In bottom layer (1), we notice that long kernels ($k = 7$) captures fine-grained temporal dependencies. But at the top layer (3), the long kernels tend to focus on coarse-grained temporal correlation. The same behavior prevails for the shot ($k = 3$) and medium ($k = 5$) kernels.

models equipped with Timeception layers. These baselines use different number of input timesteps. We experiment on Charades, with both ResNet and I3D as backbone.

ResNet is used, with a different number of timesteps as inputs: $T \in \{32, 64, 128\}$, followed by Timeception layers. ResNet processes one frame at a time. Hence, in one feedforward pass, the number of timesteps consumed by Timeception layers is equal to that consumed by ResNet.

I3D is considered, with a different number of timesteps as inputs: $T \in \{256, 512, 1024\}$, followed by Timeception layers. I3D processes 8 frames into one *super-frame* at a time. Thus, Timeception layers model $T' \in \{32, 64, 128\}$ super-frames, Practically however, as each super-frame is related to a segment of 8 frames, both I3D+Timeception process in total $T \in \{256, 512, 1024\}$ frames.

We report results in table 6 and we make two observations. First, stacking Timeception layers leads to an improved accuracy when using both ResNet and I3D as backbone. As the only change between these models is the number of Timeception layers, we deduce that the Timeception layers have succeeded in learning temporal abstractions. Second, despite stacking more and more Timeception layers, the number of parameters is controlled. Interestingly, using 4 Timeception layers on I3D processing 1024 timesteps requires half the parameters needed for a ResNet processing 128 timesteps. The reason is the number of channels from ResNet is twice as much as from I3D (2048 *v.s.* 1024).

Baseline		CNN Steps	TC Steps	Params	mAP (%)
ResNet	+ 3 TC	32	32	3.82	30.37
	+ 3 TC	64	64	3.82	31.25
	+ 4 TC	128	128	5.58	31.82
I3D	+ 3 TC	256	32	1.95	33.89
	+ 3 TC	512	64	1.95	35.46
	+ 4 TC	1024	128	2.83	37.19

Table 6: Timeception layers allow for deep and efficient temporal models, able to learn the temporal abstractions needed to learn complex actions. Columns are: Baseline: backbone CNN + how many Timception layers (TC) on top of it, CNN Steps: input timesteps to the CNN, TC Steps: input timesteps to the first Timeception layer, Params: number of parameters used by Timeception layers, in millions.

We conclude that Timeception layers allow for deep and efficient models, able to learn long-range temporal abstractions, which is crucial for complex actions.

Learned Weights of Timeception. Figure 17 visualizes the learned weights by our model. Specifically, three Timeception layers trained on top of I3D backbone. The figure depicts the weights of multi-scale temporal convolutions with different kernel sizes $k \in \{3, 5, 7\}$. For simplicity, only the first 30 kernels from each kernel-size, are shown. We make two remarks for these learned weights. First, at layer 1, we notice that long kernels ($k = 7$) captures fine-grained temporal dependencies, because of the rapid transition of kernel weights. But at layer 3, these long kernels tend to focus on coarse-grained temporal correlations, because of the smooth transition between kernel weights. The same behavior prevails for the short ($k = 3$) and medium ($k = 5$) kernels. Second, at layer 3, we observe that long-range and short-range temporal patterns are learned by short kernels ($k = 3$) and long kernels ($k = 7$), respectively. The conclusion is that for complex actions, both video-wide and local temporal reasoning, even at the top layer, is crucial for recognition.

3.4.4 *Effectiveness of Timeception*

To demonstrate the effectiveness of Timeception, we compare it against related temporal convolution layers: *i.* separable temporal convolution [160], that models both $\mathcal{T}, C$ simultaneously. *ii.* grouped separable temporal convolution to model $\mathcal{T}$, followed by 1×1 2D convolution to model C. *iii.* grouped separable temporal convolution to model $\mathcal{T}$, followed by channel shuffling to model C. Interestingly in figure 18 top, Timeception is very efficient in maintaining a reasonable increase in number of parameters as the network goes deeper. Also, figure 18 bottom shows how Timeception improves mAP on Charades, scales up temporal capacity of backbone CNNs while maintaining the overall model size.

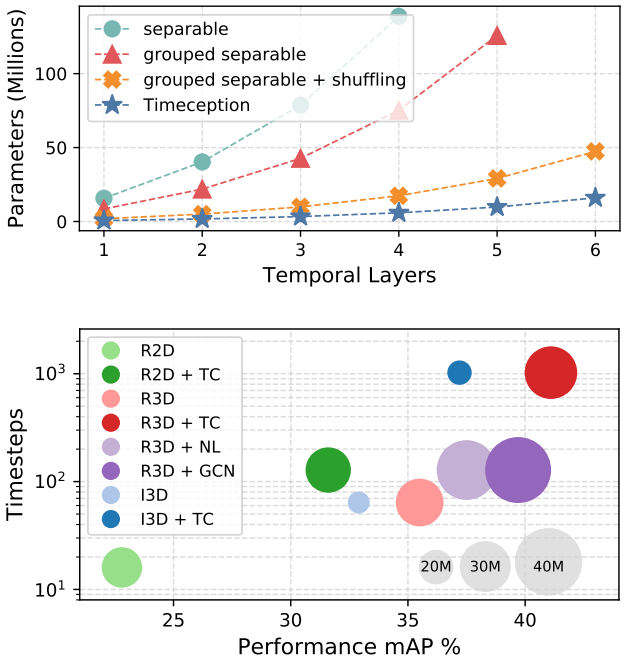


Figure 18: Top: the cost of adding new Timeception layers is marginal, compared to related temporal layers. Bottom: Timeception improves performance, scales up temporal capacity of backbone CNNs while maintaining the model size.

3.4.5 *Experiments on Benchmarks*

Charades is used to evaluate our model, and to compare against related works. In this experiment, our baseline networks use 4 Timeception layers. The number of convolutional groups is 8 for I3D and 16 for ResNet, be it 2D or 3D. The results in table 7 shows that Timeception monotonically improves the performance of the backbone CNN. The absolute gain on top of ResNet and I3D is 8.8% and 4.3%, respectively.

Beyond the overall mAP, how beneficial is Timeception? And in what cases exactly does it help? To answer this question, we make two comparisons to assess the relative performance of Timeception. We experiment two scenarios: *i*. short-range (32 timesteps) v.s. long-range (128 timesteps), *ii*. fixed-scale v.s. multi-scale kernels. The results are shown in figures 19, 20, and we make two observations.

First, when comparing the relative performance of multi-scale vs. fixed-size Timeception, see figure 19, we observe that multi-scale Timeception excels in complex actions with dynamic temporal patterns. As an example, "take clothes + tidy clothes + put clothes", one actor may take longer than others to tidy clothes. In contrast, fixed-size Timeception excels in the cases where the complex action is more rigorous in the temporal pattern, *e.g.* "open window + close window". Second, when comparing the relative performance of short-range (32 timesteps) v.s. long-range (1024 timesteps) Timeception, see figure 20, the latter excels in complex actions than requires the entire video to unfold, *e.g.* "fix door + close door". However, short-range Timeception would do better in one-actions, like "open box + close box" or "turn on light + turn of light".

Breakfast Actions is used as a second dataset to experiment our model. The average length of a video in this datataset is 2.3 sec. For this experiment, we use 3 layers of

Ours	Method		Modality	mAP (%)
	Two-stream	[142]	RGB + Flow	18.6
	Two-stream + LSTM	[142]	RGB + Flow	17.8
	ActionVLAD	[50]	RGB + iDT	21.0
	Temporal Fields	[142]	RGB + Flow	22.4
	Temporal Relations	[193]	RGB	25.2
	ResNet-152	[3]	RGB	22.8
✓	ResNet-152 + TC		RGB	31.6
	I3D	[15]	RGB	32.9
✓	I3D + TC		RGB	37.2
	3D ResNet-101	[176]	RGB	35.5
	3D ResNet-101 + NL	[176]	RGB	37.5
	3D ResNet-50 + GCN	[177]	RGB + RP	37.5
	3D ResNet-101 + GCN	[177]	RGB + RP	39.7
✓	3D ResNet-101 + TC		RGB	41.1

Table 7: Timeception (TC) outperforms related works using the same backbone CNN. It achieves the absolute gain of 8.8% *and* 4.3% *over ResNet and I3D, respectively. More over, using the full capacity of Timeception improves* 1.4% *over best related work.*

Timeception. And as for the backbone, we use I3D and 3D ResNet-50. None of the backbones is fine-tuned on this dataset, only Timeception layers are trained. To make one video consumable by our baseline, from each video we uniformly sample 64 video snippet, each of 8 sucessive frames. That makes the total timesteps modeled by the baseline is 512. Finally, we report result in table 8.

Method	Activities (Acc. %)	Actions (mAP %)
I3D	64.31	47.71
I3D + TC	69.30	56.36
3D ResNet-50	66.73	53.27
3D ResNet-50 + TC	71.25	59.64

Table 8: Timeception outperform baselines in recognizing the long-range activities of Breakfast dataset.

MultiTHUMOS is used as a third dataset to experiment our model. This helps in investigating the generality on different datasets. Related works use this dataset for temporal localization of one-actions in each video of complex action. Differently, we use this dataset to serve our objective: multi-label classification of complex actions, *i.e.* the entire video. As such, the evaluation method used is mAP [130]. To assess the performance of our model, we compare against I3D as a baseline. As shown in the results in table 9, Timeception, equipped with multi-scale kernels outperforms that with fixed-size kernel.

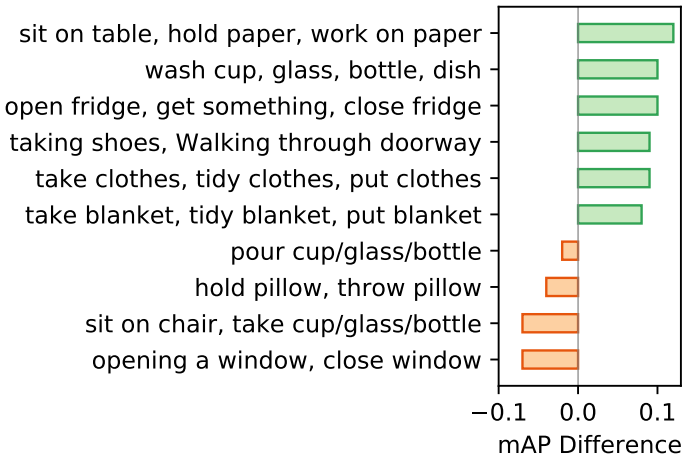


Figure 19: Multi-scale Timeception outperforms the fixed-kernel when complex actions are dynamic, in green. But when complex actions with rigid temporal patters, fixed-size performs better than multi-scale, in orange.

Method	Kernel Size (k)	Dilation Rate (d)	mAP (%)
I3D	–	–	72.43
I3D + Timeception	3	1	72.83
I3D + Timeception	3	1,2,3	74.52
I3D + Timeception	1,3,5,7	1	74.79

Table 9: Timeception, with multi-scale temporal kernels, helps baseline models to capture the long-range dependencies between one-actions in videos of MultiTHUMOS.

3.5 CONCLUSION

Complex actions such as "cooking a meal" or "cleaning the house" can only be recognized when processed fully. This is in contrast to one-actions, that can be recognized from a small burst of frames. This work presents Timeception, a novel temporal convolution layer for complex action recognition. Thanks to using efficient temporal-only convolutions, Timeception can scale up to minute-long temporal modeling. In addition, thanks to multi-scale temporal convolutions, Timeception can tolerate the changes in temporal extents of complex actions. Interestingly, when visualizing the temporal weights we observe that earlier timeception layers learn fast temporal changes, whereas later timeception layers focus on more global temporal transitions. Evaluating on popular benchmarks, the proposed Timeception improves the state-of-the-art notably.

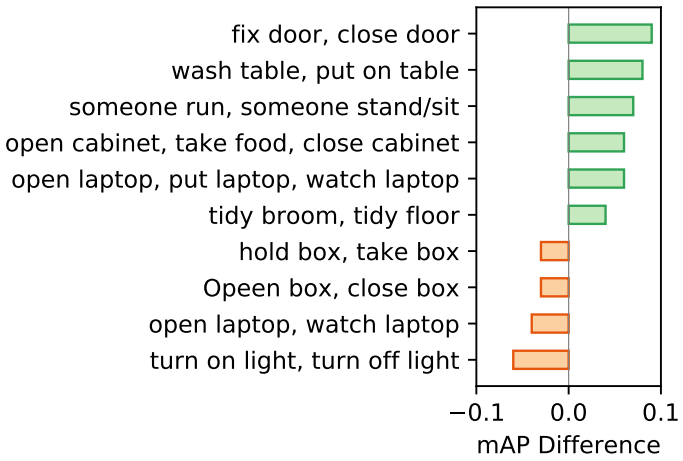


Figure 20: Long-range Timeception outperforms the short-range version when complex actions need the entire video to unfold, in green. However, we see one-actions that short-range Timeception can easily capture, in orange.

4

VIDEOGRAPH: RECOGNIZING MINUTES-LONG HUMAN ACTIVITIES IN VIDEOS

4.1 INTRODUCTION

Human activities in videos can take many minutes to unfold, each is packed with plentiful of fine-grained visual details. Take for example two activities: "making pancake" or "preparing scrambled eggs". The question is what makes a difference between these two activities? Is it the fine-grained details in each, or the overall painted picture by each? Or both?

The goal of this work is to recognize minutes-long human activities as defined by [99], also referred to as complex actions in [73]. A long-range activity consists of a set of unit-actions [99], also known as one-actions [73]. For example, the activity of "making pancakes" includes unit-actions: "cracking egg", "pour milk" and "fry pancake". Some of these unit-actions are crucial to distinguish the activity. For example, the unit-action "cracking egg" is all what is needed to discriminate the activity of "making pancakes" from "preparing coffee". Also, long-range activity is recognized only in its entirety, as its unit-actions are insufficient by themselves. For example, only a short video snippet of unit-action "cracking egg" cannot tell apart "making pancake" from "preparing scrambled eggs", as both activities share the same unit-action "cracking egg". Added to this, the temporal order of unit-actions for a specific activity may be permuted. There exist different orders of how we can carry out an activity, like "prepare coffee", see figure 21. Nonetheless, there exist some sort of temporal structure for such activity. One can start "preparing coffee" by "taking cup" and usually end up with "pour sugar" and "stir coffee". So, to recognize long-range human activities, goals to be met are: modeling the temporal structure of the activity in its entirety, and occasionally paying attention to its fine-grained details.

There exist two distinct approaches for long-range temporal modelling. The first approach is orderless modeling. Statistical pooling [72] and vector encoding [31, 50] are used to aggregate video information over time. The upside is the ability to address seemingly minutes- or even hours-long videos. The downside, however, is the inability to learn temporal patterns and the arrow-of-time [46]. Both are proven to be crucial for some tasks [70, 143]. The second approach is order-ware modelling. 3D CNN is proven to be successful in learning spatiotemporal concepts for short video snippets with strict temporal pattern [15]. Careful design choices enable them to model up to minute-long temporal dependencies [73]. But for minutes-long human activities, the strict temporal

Published in *International Conference on Computer Vision Workshop(ICCV Workshop) on Scene Graph Representation and Learning (SGRL)*, 2019 [74]

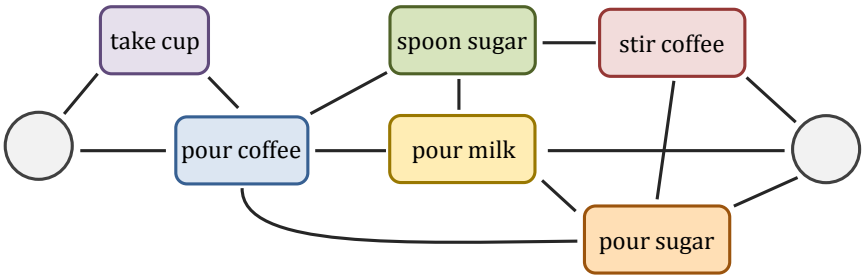


Figure 21: The activity of "preparing coffee" can be represented as undirected graph of unit-actions. We are inspired by graphs to represent this activity. The reason is that a graph can portray the many ways one can carry out such activity. More over, it preserves the temporal structure of the unit-actions. Reproduced from [99].

pattern no longer exists. So, the question arises: how to model the temporal structure of minutes or even hour-long human activities?

This work proposes VideoGraph, a graph-inspired representation to achieve the aforementioned goal. A soft version of undirected graph in learned completely from the dataset. The graph nodes represent the key latent concepts of which the human activity is composed. These latent concepts are analogous to one-actions. While the graph edges represent the temporal relationship between these latent concepts, *i.e.* the graph nodes. VideoGraph has the following novelties. *i.* In its graph-inspired representation, VideoGraph models human activity for up to thirty-minute videos, whereas the state-of-the-art is one minute [73]. *ii.* A proposed node embedding block to learn the graph nodes from data. This circumvents the node annotation burden for long-range videos, and makes VideoGraph extensible to video datasets without node-level annotation. *iii.* A novel graph embedding layer to learn the relationships between graph nodes. The outcome is representations of the temporal structure of long-range human activities. VideoGraph improvements on benchmarks for human activities: Breakfast [99], Epic-Kitchens [26] and Charades [144].

4.2 RELATED WORK

Orderless *v.s.* Order-aware Temporal Modeling. Be it short-, mid-, or long-range human activities, when it comes to temporal modeling, related methods are divided into two main families: orderless and order-aware. In orderless methods, the main focus is the statistical pooling of temporal signals in videos, without considering their temporal order or structure. Different pooling strategies are used, as max and average pooling [72], attention pooling [49], and context gating [118], to name a few. A similar approach is vector aggregation, for example: Fisher Vectors [124] and VLAD [31, 50]. Although statistical pooling can trivially scale up to extremely long sequences in theory, this comes at a cost of losing the temporal structure, reminiscent of Bag-of-Words losing spatial understanding.

In order-aware methods, the main attention is payed to learning structured or ordered temporal patterns in videos. For example, LSTMs [29, 107], CRF [142], 3D CNNs [15, 160, 176, 182, 184]. Others propose temporal modeling layers on top of backbone CNNs, as in Temporal-Segments [172], Temporal-Relations [193] and Rank-

Pool [37]. The outcome of order-aware methods is substantial improvements over their orderless counterparts in standard benchmarks [91, 100, 148]. Nevertheless, both temporal footprint and computational cost remain the main bottlenecks to learn long-range temporal dependencies. The best methods [73, 176] can model as much as 1k frames (~30 seconds), which is a no match to minutes-long videos. This work strives for the best of two worlds: learning the temporal structure of human activities in minutes-long videos.

Short-range Actions *v.s.* Long-range Activities. Huge body of work is dedicated to recognizing human actions that take few seconds to unfold. Examples of well-established benchmarks are: Kinetics [91], Sports-1M [90], YouTube-8M [6], Moments in Time [121], 20B-Something [51] and AVA [52]. For these short- or mid-range actions, [143] demonstrates that a few frames suffice for a successful recognition. Other strands of work shift their attention to human activities that take minutes or even an hour to unfold. Cooking-related activities are good examples, as in YouCook [196], Breakfast [99], Epic-Kitchens [26], MPII Cooking [134] or 50-Salads [149]. Other examples include instructional videos: Charades [144], and unscripted activities: EventNet [187], Multi-THUMOS [189].

In all cases, several works [42, 73, 99, 134] define the differences between short- and long-range human actions, albeit with a different naming or terms. We follow the same definition of [99]. More formally, we use *unit-actions* to refer to fine-grained, short-range human actions, and *activities* to refer to long-range complex human activities.

Graph-based Representation. Earlier, graph-based representation has been used in storytelling [92, 183], and video retrieval [128]. Different works use graph convolutions to learn concepts and/or relationships from data [28, 93, 123]. Recently, graph convolutions are applied to image understanding [19], video understanding [47, 68, 69, 177] and question answering [174]. Despite their success in learning structured representations from video datasets, the main limitation of graph convolution methods is requiring the graph nodes and/or edges to be known a priori. Consequently, when node or frame-level annotations are not available, using these methods is hard. In contrast, this work aims for a graph-inspired representation in which the graph nodes are fully inferred from data. The result is that our work is extensible to datasets without node-level annotations.

Self-Attention is used extensively in language understanding [109]. The recently proposed the transformer block shows substantial improvements in machine translation [164], image recognition [176] and video understanding [47, 178] or even graph representations [166]. The transformer block [178] attends to a local feature conditioned on both local and global context. That is why it outperforms the self-attention mechanism [30, 108, 186], which is conditioned on only the local feature.

A video of human activity consists of short snippets of unit-actions. This work is inspired by all these attention mechanisms to attend to a unit-action (*i.e.* local feature) based on the surrounding activity (*i.e.* global context).

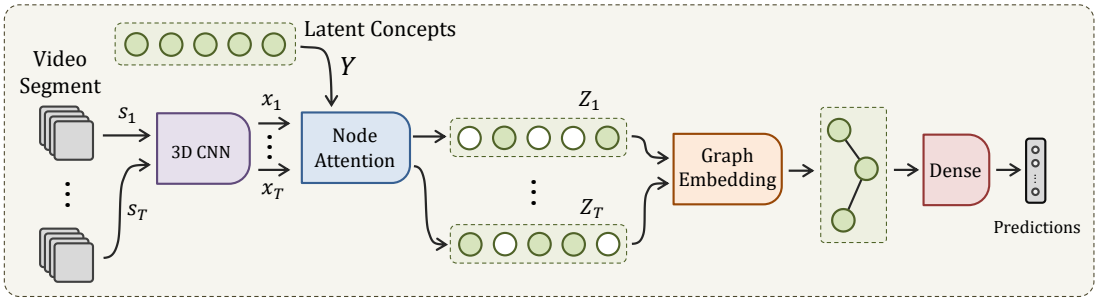


Figure 22: Overview of VideoGraph. It takes as input a video segment s_i of 8 frames from an activity video v. Then, it represents it using standard 3D CNN, .e.g I3D. The corresponding feature representation is x_i. Then, a node attention block attends to a set of N latent concepts based on their similarities with x_i, which results in the node-attenative representation Z_i. A novel graph embedding layer then processes Z_i to learn the relationships between its latent concepts, and arrives at the final video-level representation. Finally, an MLP is used for classification.

4.3 METHOD

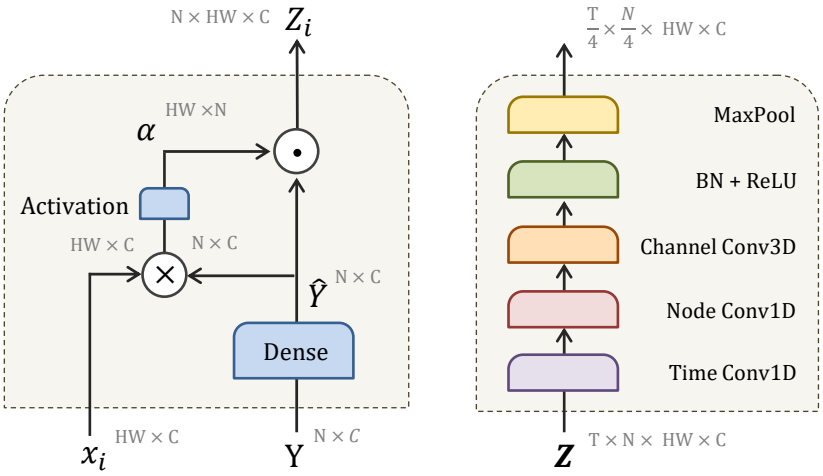
Motivation. We observe that a minutes-long and complex human activity usually is sub-divided into unit-actions. Similar understanding is concluded by [73,99], see Fig. 21. So, one can learn the temporal dependencies between these unit-actions using methods for sequence modeling in videos, as LSTM [107] or 3D CNN [182]. However, these methods face the following limitations. First, such activities may take several minutes or even hours to unfold. Second, as video instances of the same activity are usually wildly different, there is no single temporal sequence that these methods can learn. For example, one can "prepare coffee" in many different ways, as the various paths in Fig. 21 indicate. Nevertheless, there seems to be an over-arching weak temporal structure of unit-actions when making a coffee.

We are inspired by graphs to represent the temporal structure of the human activities in videos. The upside is the ability of a graph-based representation to span minutes- or even hour-long temporal sequence of unit-actions while preserving their temporal relationships. The proposed method, VideoGraph, is depicted in Fig. 22, and in the following, we discuss its details.

VideoGraph. We start from a video v comprising T randomly sampled video segments $v = \{s_i \,|\, i = 1, 2, ..., T\}$. Each segment s_i is a burst of 8 successive video frames, and represented as feature $x_i \in \mathbb{R}^{1\times H\times W\times C}$ using standard 3D CNN, for example I3D [15], where C is the number of channels, H, W are height and width of the channels. Our goal is to construct an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ to represent the structure of human activity in video v. The graph nodes $\mathcal{N}$ would then capture the key unit-actions in the activity. And the graph edges $\mathcal{E}$ would capture the temporal relationship between these nodes (*i.e.* unit-actions).

Learning The Graph Nodes. In a dataset of human activities, unit-actions can be thought of as the dominant *latent* short-range concepts. That is, unit-actions are the building blocks of the human activity. So, in a graph-inspired representation of the activity, these unit-actions can act as the graph nodes $\mathcal{N}$. Assuming that it is prohibitively

expensive to have unit-actions annotation for minutes-long videos, a challenge is how to represent them? In other words, how to represent the graph nodes? As a remedy, we opt for learning a set of N latent features Y, $Y = \{y_j \mid j = 1, 2, ..., N\}$, $Y \in \mathbb{R}^{N \times C}$. These features Y then become the vector representation of the graph nodes $\mathcal{N}$, *i.e.* $Y \equiv \mathcal{N}$.



(a) Node Attention Block (b) Graph Embedding Layer

Figure 23: (a) Node attention block measures similarities α between segment feature x_i and learned nodes $\hat{Y}$. Then, it attends to each node in $\hat{Y}$ using α. The result is the node-attentive feature Z_i expressing how similar each node to x_i. (b) Graph Embedding layer models a set of T successive node-attentive features $\mathbf{Z}$ using 3 types of convolutions. i. Timewise Conv1D learns the temporal transition between node-attentive features $\{Z_i, ..., Z_{i+t}\}$. ii. Nodewise Conv1D learns the relationships between nodes $\{z_{i,j}, ..., z_{i,j+n}\}$. iii. Channelwise Conv3D updates the representation for each node z_{ij}.

A problem, however, is how to correlate each video feature x_i with each node in Y. To solve this, we propose the node attention block, inspired by self-attention block [47, 164, 176], shown in Fig. 23a. The node attention block takes as an input a feature x_i and all the node features Y. Then, it transforms the initial representation of the nodes from Y into $\hat{Y}$, using one hidden layer MLP with weight and bias $w \in \mathbb{R}^{C \times C}, b \in \mathbb{R}^{1 \times C}$. This transformation makes the nodes learnable and better suited for the dataset inhand. Then, a dot product $\otimes$ is used to measure the similarity between x_i and $\hat{Y}$. An activation function σ is applied on the similarities to introduce non-linearity. The result is the activation values $\alpha \in \mathbb{R}^{H \times W \times N}$. The last step is multiplying all the nodes $\hat{Y}$ with the activation values α, such that we attend to each node $\hat{y}_j$ by how much it is related to the feature x_i. Thus, the node attention block outputs the attended nodes $Z_i = \{z_{ij} \mid j = 1, 2, ..., N\}, Z_i \in \mathbb{R}^{N \times H \times W \times C}$. We refer to Z_i as node-attentive feature, and we refer to z_{ij} as the j-th node feature in Z_i. More formally,

$$\hat{Y} = w * Y + b \tag{4.1}$$

$$\alpha = \sigma(x_i * \hat{Y}^T) \tag{4.2}$$

$$\begin{aligned} Z_i &= \alpha \odot \hat{Y} \\ &= \alpha_j \odot y_j, \quad j = 1, 2, ..., N \end{aligned} \tag{4.3}$$

Hence, the vector representation of all video segments is a 5D tensor $\mathbf{Z} = \{Z_1, Z_2, ..., Z_T\}$, $\mathbf{Z} \in \mathbb{R}^{T \times N \times H \times W \times C}$. The names of 5 dimensions in $\mathbf{Z}$ are: timesteps, nodes, width, height and channels. From now on, we use these 5 dimensions to express feature vectors and convolutional kernels.

In sum, the node attention block takes a feature x_i, corresponding to a short video segment s_i and measures how similar α it is to learned set of latent concepts $\hat{Y}$. The similarities α are then used to attend to the latent concepts. This is crucial for recognizing long-range videos, where the network is not feed-forwarded only with a short video segment x_i but with global representation Y. This gives the network the ability for focus on both local video signal x_i and global learned context $\hat{Y}$.

Our node attention block is different from the non-local counterpart [176] in twofold. First, the attention values are conditioned on local x_i and global $\hat{Y}$ signals. Second, non-local does tensor product between attention values α and local signal x_i, while we attend by scalar multiplication between $\alpha, \hat{Y}$ to retrain the node dimension. Lastly, our node attention block is much more simpler than the non-local, as we use only one fully-connected layer.

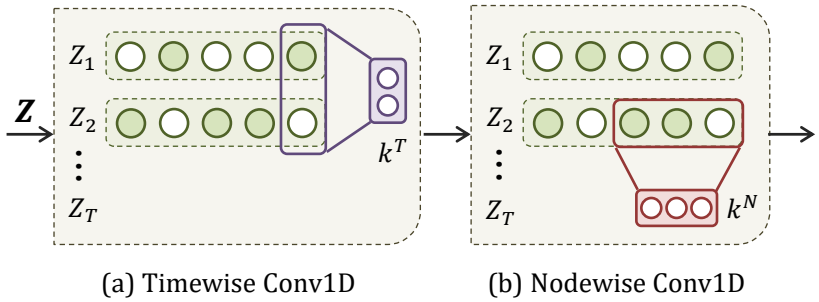


Figure 24: (a) Timewise Conv1D learns the temporal transition between successive nodes-embeddings $\{Z_i, ..., Z_{i+t}\}$ using kernel k^T of kernel size t. (b) Nodewise Conv1D learns the relationships between consecutive nodes $\{z_{i,j}, ..., z_{i,j+n}\}$ using kernel k^N of kernel size n.

Learning The Graph Edges. Up till now, we have learned the graph nodes $\hat{Y}$. We have also represented each video segment s_i in terms of the nodes, as node-attentive feature Z_i. Next, we would like to learn the graph edges $\mathcal{E}$, and arrive at the final graph structure. To this end, we propose a novel graph embedding layer, shown in Fig. 23b. Regarding the graph edges, we are interested in two types of relationships. First, we are interested in the relationship between graph nodes. Loosely speaking, if nodes stand for unit-actions as "pour milk", "crack egg", we would like to learn how correlated are these two unit-actions when used in different activities as "make pancake" or "prepare coffee". Second, we are interested in how the graph nodes transition over time. For instance, we want to encode the significance of unit action "pour milk" comes after or before "crack egg" when it comes to recognizing "make pancake". Let's take t successive video segments $\{s_i, ..., s_{i+t}\}$. When processed by CNN and node attention block, they are represented as $\{Z_i, ..., Z_{i+t}\}$. To learn the temporal transition between them, we apply a one-dimensional convolution, (Conv1D) on the temporal dimension only. These timewise Conv1D, proposed by [73], are efficient in learning temporal concepts. One kernel learned by timewise Conv1D is the 5D tensor $k^T \in \mathbb{R}^{t \times 1 \times 1 \times 1 \times 1}$, where t is the

kernel size. In total, we learn C kernels to keep the channel dimension of the features $\mathbf{Z}$ unchanged.

Besides learning the temporal transition between node-attentive features $\{Z_i, ..., Z_{i+t}\}$, we also want to learn the relationship between the nodes themselves $\{z_{ij} \mid j = 1, 2, ..., N\}$ inside each node-attentive feature Z_i. The problem is that the adjacency matrix, which defines the graph structure, is unkown. A naive approach is to assume all nodes are connected. This leads to an explosion of N^2 edges – prohibitive to learn. To overcome this, we restrict the number of adjacents (*i.e.* neigbours) each node z_{ij} can have. In other words, we assume that each node z_{ij} is adjacent to only n other nodes. This makes it possible to learn edge weights using one-dimensional convolution, applied on only the node dimension of Z_i. We call this convolution nodewise Conv1D. One kernel learned by nodewise Conv1D is the 5D tensor $k^N \in \mathbb{R}^{1\times n\times 1\times 1\times 1}$, where n is the kernel size. In sum, we learn C kernels to keep the channel dimension of the features $\mathbf{Z}$ unchanged.

Both timewise and nodewise Conv1D learn graph edges separately for each channel in the features $\mathbf{Z}$. That is why we follow up with a typical spatial convolution (Conv2D) to model the cross-channel correlations in each node feature z_{ij}. Spatial Conv2D learns C different kernels, each is the 5D tensor $k^C \in \mathbb{R}^{1\times 1\times 1\times 1\times C}$.

Having learned the graph edges using convolutional operations, we proceed with BatchNormalization and ReLU non-linearity. Finally, we downsample the entire graph representation $\mathbf{Z}$ over both time and node dimensions using MaxPooling operation. It uses kernel size 3 and stride 3 for both the time and node dimensions. Thus, after one layer of graph embedding, the result graph representation is reduced from $T \times N \times H \times W \times C$ to $(T/3) \times (N/3) \times H \times W \times C$.

4.4 EXPERIMENTS

Implementation. When training VideoGraph on a video dataset, we uniformly sample $T = 64$ video segments from each video v. One segment s_i is a burst of 8 successive frames. When the 64 segments are fed-forward to I3D up to the last convolutional layer `res5_c`, the corresponding convolutional features for the entire video is $\mathbf{X} = \{x_i \mid i = 1, 2, ..., 64\}, \mathbf{X} \in \mathbb{R}^{64\times 7\times 7\times 1024}$. We use $N = 128$ as the number of latent concepts. Both the video-level features $\mathbf{X}$ and latent concepts $Y \in \mathbb{R}^{128\times 1024}$ are fed-forward to the node attention block. The result is the graph representation $\mathbf{Z} \in \mathbb{R}^{128\times 64\times 7\times\times 1024}$. Then, $\mathbf{Z}$ is passed to graph embedding layers to learn node edges and reduce the feature representation. In graph embedding layer, we use kernel size $t = 7$ for the timewise Conv1D and kernel size $n = 7$ for the nodewise Conv1D. In total, we use 2 successive layers of graph embedding. Their output feature is then feed-forwarded to a classifier to arrive at the vide-level predictions. The classifier uses 2 fully-connected layers with BatchNormalization and ReLU non-linearity. We use softmax as the final activation for single-label classification or sigmoid for multi-label classification.

VideoGraph is trained with batch-size 32 for 500 epoch. It is optimized with SGD with 0.1, 0.9 and 0.00001 as learning rate, momentum and weight decay, respectively. It is implemented using TensorFlow [5] and Keras [22].

4.4.1 *Datasets*

As this work focus on human activities spanning many minutes, we choose to conduct our experiments on the following benchmarks: Breakfast [99], Epic-Kitchens [26] and Charades [144]. Other benchmarks for human activities contain short-range videos, *i.e.* a minute or less, thus do not fall within the scope of this work.

Breakfast is a dataset for task-oriented human activities, with the focus on cooking. It is a video classification task of 12 categories of breakfast activities. It contains 1712 videos in total, 1357 for training and 335 for test. The average length of videos is 2.3 minutes. The activities are performed by 52 actors, 44 for training and 8 for test. Having different actors for training and test splits is a realistic setup for testing generalization. Each video is represents only one category of focus activity. Besides, each video has temporal annotation of unit-actions comprising the activity. In total, there are 48 classes of unit-actions. In our experiments, we only use the activity annotation, and we don't use the temporal annotation of unit-actions.

Epic-Kitchens is a recently introduced large-scale dataset for cooking activities. In total, it contains 274 videots performed by 28 actors in different kitchen setups. Each video represents a cooking different cooking activity. The average length of videos is 30 minutes, which makes it ideal for experimenting very long-range temporal modeling. Originally, the task proposed by the dataset is classification on short video snippets, with average length of ~3.7 seconds. The provided labels are, therefore, the categories of objects, verbs and unit-actions in each video snippet. However, the dataset does no provide video-level category. That is why we consider all the object labels of a specific video as video-level label. Hence, posing the problem as multi-label classification of these videos. This setup is exactly the same used in Charades [144] for video classification. For performance evaluation, we use mean Average Precision (mAP), implemented in Sk-Learn [130].

Method	Modality	mAP (%)
Two-stream [142]	RGB + Flow	18.6
Two-stream + LSTM [142]	RGB + Flow	17.8
ActionVLAD [50]	RGB + iDT	21.0
Temporal Fields [142]	RGB + Flow	22.4
Temporal Relations [193]	RGB	25.2
ResNet-152 [3]	RGB	22.8
ResNet-152 + Timeception [73]	RGB	31.6
I3D [15]	RGB	32.9
I3D + ActionVLAD [50]	RGB	35.4
I3D + Timeception [73]	RGB	37.2
I3D + VideoGraph	RGB	**37.8**

Table 10: VideoGraph outperforms related works using the same backbone CNN. Results are for Charades dataset.

Method	Breakfast Acc. (%)	Breakfast mAP (%)	Epic-Kitchens mAP (%)
ResNet-152 [3]	41.13	32.65	–
ResNet-152 + ActionVLAD [50]	55.49	47.12	–
ResNet-152 + Timeception [73]	57.75	48.47	–
ResNet-152 + VideoGraph	**59.12**	**49.38**	–
I3D [15]	58.61	47.05	48.86
I3D + ActionVLAD [50]	65.48	60.20	51.45
I3D + Timeception [73]	67.07	61.82	**55.46**
I3D + VideoGraph	**69.45**	**63.14**	55.32

Table 11: VideoGraph outperforms related works using the same backbone CNN. We experiment 2 different backbones: I3D and ResNet-152. We experiment on two different tasks of Breakfast: single-label classification of activities and multi-label classification of unit-actions. And for Epic-Kitchens, we experiment on the multi-label classification.

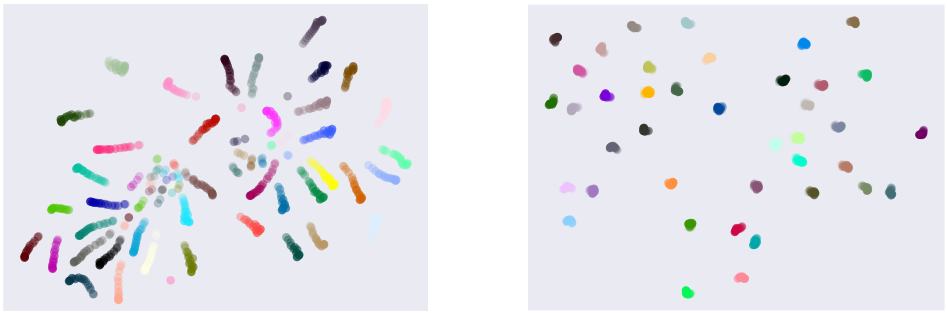
Charades is a dataset for multi-label classification of action videos. It consists of 8k, 1.2k and 2k video for training, validation and testing, respectively. is multi-label, action classification, video dataset with 157 classes. Each video spans 30 seconds and comprises of 6 unit-actions, on average. This is why we choose Charades, as it fits perfectly to the needs of this work. For evaluation, we use mAP, as detailed in [144].

4.4.2 *Experiments on Benchmarks*

In this section, we experiment and evaluate VideoGraph on benchmark datasets: Breakfast, Charades and Epic-Kitchens, and we compare against related works. We choose two strong methods to compare against. The first is Timeception [73]. The reason is that it can model 1k timesteps, which is up to a minute-long video. Another reason is that Timeception is an order-ware temporal method. The second related work is ActionVLAD [50]. The reason is that it is a strong example of orderless method. It also can aggregate temporal signal for very long videos.

VideoGraph resides on top of backbone CNN, be it spatial 2D CNN, or spatio-temporal 3D CNN. So, in our comparison, we use two backbone CNNs, namely ResNet-152 [62] and I3D [15]. By default, I3D is designed to model a short video segment of 8 frames. But thanks to the fully-convolutional architecture, I3D can indeed process minutes-long video. This is made possible by average pooling the features of many videos snippets, in logit layer, *i.e.* before softmax activation [15]. ResNet-152 is a frame-level classifier. To extend it to video classification, we follow the same approach used in I3D and average pool the logits, *i.e.* before softmax. In all the following comparisons, we use 512 frames, or 64 segments, per video as input to I3D. And we use 64 frames per video as and input to ResNet-152.

Breakfast. Each video in this dataset depicts a complex breakfast activity. Thus, the task inhand is single-label classification. The evaluation metric used is the classification



(a) *(b)*

Figure 25: Visualization of the learned graph nodes. In the first 20 epoch during training (left), VideoGraph updates the node features $\hat{Y}$ to increase the pairwise distance between them. That is, VideoGraph learns discriminant representations of the nodes. In the last 20 epoch during training (right), the learning cools down and barely their representation is updated. We visualize using t-SNE [112].

accuracy. We experiment our model on Breakfast, and we compare against baseline methods. The results are reported in table 11.

Epic-Kitchens. When comparing VideoGraph against related works, see table 11, Timeception and VideoGraph, we notice that we are on bar with Timeception. VideoGraph performs better when trained on single-label video dataset, where each video has one label. This gives VideoGraph an ample opportunity to tailor the graph-inspired representation for each class. However, as mentioned, we pose the task in Epic-Kitchen as multi-label classification. That is, no single category for a video. That's when VideoGraph does not perform as good.

Charades. In this experiment, we evaluate our model on Charades dataset. And we compare the performance against recent works. The results are reported in Table 10. VideoGraph improves the performance of the backbone CNN. For VideoGraph, Charades is particularly challenging dataset, for two reasons. First, the average video length is 30 seconds, and VideoGraph learns better representstion for long-range videos. Second, it is a multi-label classification, and that's when VideoGraph is not able to learn category-specific unique graph.

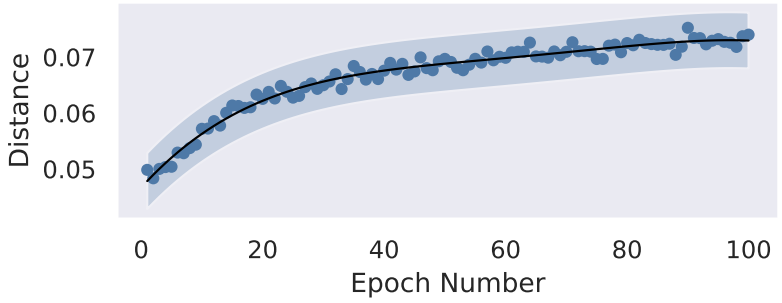
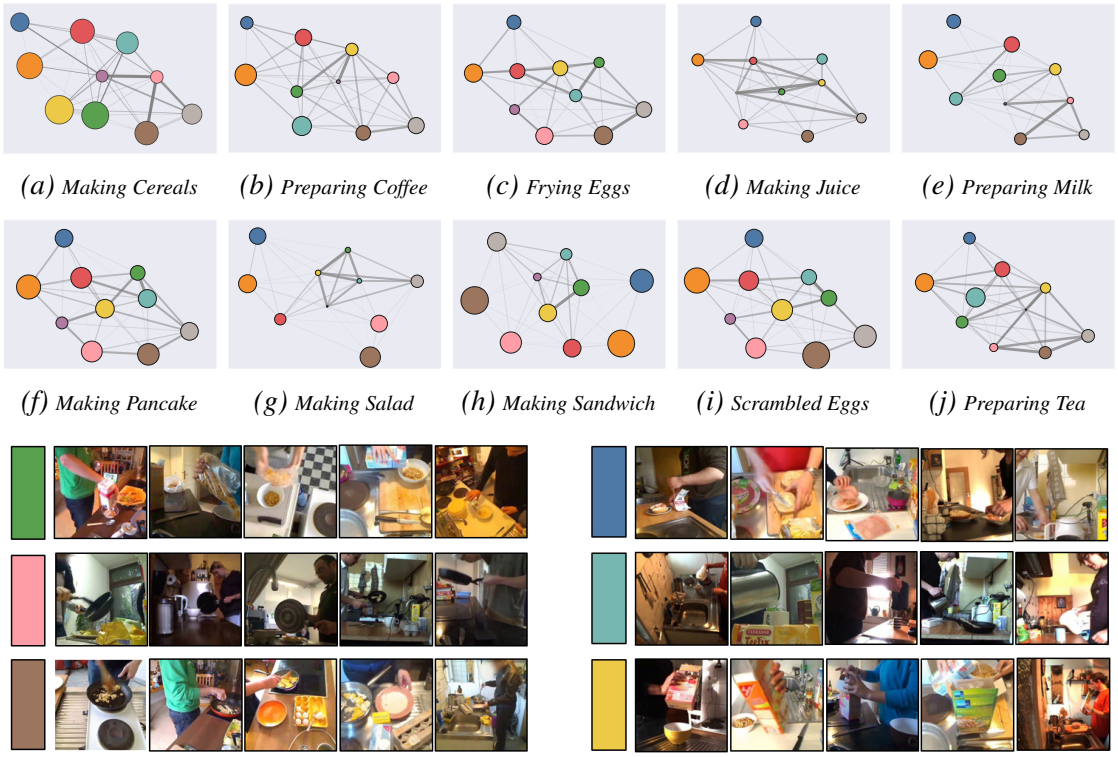


Figure 26: The pairwise Euclidean distances between normalized latent concepts $\hat{Y}$ increases rapidly in the beginning of the training, but it converges in the end.



(a) Making Cereals *(b) Preparing Coffee* *(c) Frying Eggs* *(d) Making Juice* *(e) Preparing Milk*

(f) Making Pancake *(g) Making Salad* *(h) Making Sandwich* *(i) Scrambled Eggs* *(j) Preparing Tea*

(k) Top related images to the nodes. These nodes are related to: cereal, pan, eggs, sandwich, kettle, and foodbox.

Figure 27: We visualize the relationship discovered by the first layer of graph embedding. Each sub-figure is related to one of the 10 activities in Breafast dataset. In each graph, the nodes represent the latent concepts learned by graph-attention block. Node size reflects how dominant the concept, while graph edges emphasize the relationship between these nodes.

4.4.3 *Learned Graph Nodes*

The proposed node attention block, see figure 24a, learns latent concept representation $\hat{Y}$ using fully-connected layer. This learning is conditioned on the initial value Y. We found that this initial value is crucial for VideoGraph to converge. We experiment with 3 different types of initialization: *i.* random values, *ii.* Sobol sequence and *iii.* k-means centroids. Random values seems to be a natural choice, as all the learned weights in the model are randomly initialized before training. Sobol sequence is a plausible choice, as the sequence guarantees low discrepancies between the initial values. The last choice has proven to be successful in ActionVLAD [50]. The centroids are obtained by clustering the feature maps of the last convolutional layer of the backbone CNN. However, we do not find one winning strategy across the benchmarks used. We find that Sobol sequence is the best choice for training on Epic-Kitchens and Charades. While the random initialization gives the best results on Breakfast. In table 10, we report the performance of VideoGraph using different initialization choices for the latent concepts Y. In all cases, we see in figure 25 that the node attention layer successfully learns discriminant representations of latent concepts, as the training proceedes. In other words, the networks learns to increase the Euclidean distance between each pair of latent concepts. This is further demonstrated in figure 26.

Initialization	Epic-Kitchen mAP	Breakfast Acc.
Random	54.12	**69.45**
Sobol	**55.46**	65.61
K-means Centroids	52.47	—

Table 12: The initialization of the latent concepts is crucial for learning better representation $\hat{Y}$. We experimented with 3 choices: random, sobol, and k-mean clustering. Yet, there seems not to be one winning choice across different datasets.

4.4.4 *Learned Graph Edges*

There are two types of graph edges, *i.e.* relationships, uncovered by VideoGraph. First, the timewise edges, *i.e.* how the nodes transition over time. Second, the nodewise edges, *i.e.* relationships between nodes themselves. To this end, we depend on the activation output of the second graph embedding layer. In other words, we extract the ReLU activation values. For M videos belonging to a specific human activity, the activation values are $z_1 \in \mathbb{R}^{M \times N \times T \times C}$, where C is the number of channels, T is the number of timesteps, and N is the number of nodes. First, we average the activations for all the videos, resulting in $z_2 \in \mathbb{R}^{N \times T \times C}$. Then, we average pool the activations over the temporal dimension, so we have $z_3 \in \mathbb{R}^{N \times C}$, summarizing the nodes representations for all the videos belonging to the specific activity. Finally, we measure the pairwise Euclidean distance between each pair in z_3. To plot the graph depicting the activity, we use these distances as the edge between the nodes. And to plot the nodes, we sum up the activations over the channel dimension in z_3. The result $z_4 \in \mathbb{R}^N$ is a scalar value reflecting the importance of the node to the activity. The graph is plotted using

Fruchterman-Reingold force-directed algorithm, implemented in [58]. Figure 27 shows 10 different graph, each belonging to one human activity.

Importance of Temporal Structure. In this experiment, we validate by how much VideoGraph depends on the temporal structure and weak temporal order to recognize the human activities. To this end, we choose Breakfast, as it is temporally well-structured dataset. VideoGraph is trained on ordered set of 64 timesteps. We alter the temporal order of these timesteps and test the performance of VideoGraph. We use different alterations: *i.* random order, and *ii.* reversed order. Then, we measure the performance of VideoGraph, as well as baselines, on Breakfast testset.

Temporal Structure	Reversed (↓%)	Random (↓%)
I3D	0.0	0.0
I3D + ActionVLAD	0.0	0.0
I3D + Timeception	44.1	56.2
I3D + VideoGraph	22.5	55.9

Table 13: The drop of performance of VideoGraph and other models when changing the temporal order of the input video. Both VideoGraph and Timeceptions suffer huge drop in performace, as both are order-aware methods. On the other hand, ActionVLAD retains the same performance, as it is orderless method.

We notice, from table 13, a huge drop in performance for both VideoGraph and Timeception. However, as expected, no drop in performance for ActionVLAD, as it is completely orderless model. The conclusion is VideoGraph encodes the temporal structure of the human activities in breakfast. Added to this, it suffered slightly less drop in performance than Timeception. More importantly, figure 28 shows the confusion matrix of classifiyng the videos of Breakfast using two cases: *i* natural order of temporal video segments, and *ii.* random order of the video segments. We notice video graph makes more mistakes when trained on random order. It mistakes "scrambled egg" for "fried egg" if temporal order is neglected.

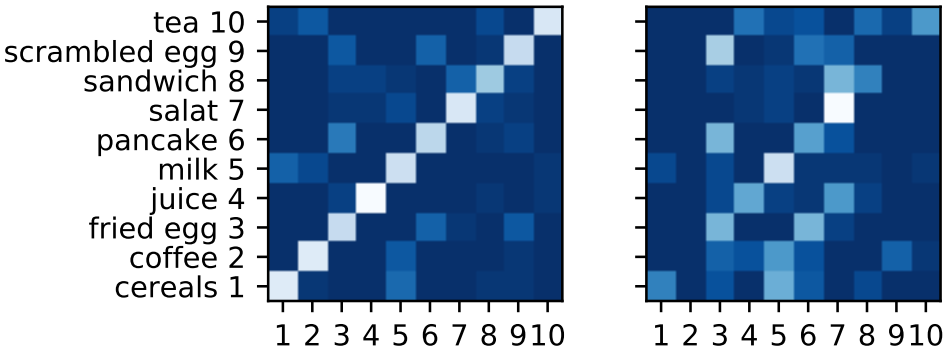


Figure 28: Confusion matrix for recognizing the 10 activity of Breakfast. VideoGraph is trained on random (right) v.s. correct temporal order (left). It mistakes "scrambled egg" for "fried egg" if temporal order is neglected.

4.5 CONCLUSION

To successfully recognize minutes-long human activities such as “preparing breakfast” or “cleaning the house”, we argued that a successful solution needs to capture both the whole picture and attention to details. To this end, we proposed VideoGraph, a graph-inspired representation to model the temporal structure of such long-range human activities. Firstly, thanks to the node attention layer, VideoGraph can learn the graph nodes. This alleviate the need of node-level annotation, which is prohibitive and expensive in nowadays video dataset. Secondly, we proposed graph embedding layer. It learns the relashionship between graph nodes and how these nodes transition over time. Also, it compresses the graph representation to be feed for a classifier. We demonstrated the effectiveness of VideoGraph on three benchmarks: Breafast, Epic-Kitchens and Charades. VideoGraph achieves good performance on the three of them. We also discussed some of the upsides and downside of VideoGraph.

5

PERMUTATION INVARIANT CONVOLUTION FOR RECOGNIZING LONG-RANGE ACTIVITIES

5.1 INTRODUCTION

Long-range human activities are famous for being lengthy [99]. Their composition is diverse [73], and their temporal structure is weak [74]. For example, the activity of "preparing coffee" shown in figure 29. This activity takes some ten minutes to unfold. It consists of many short segments of actions, called *unit actions* [99]. For "preparing coffee", unit actions include "taking cup" or "pouring milk", among many others. A unit action, by itself, spans only a few seconds, and exhibits a coherent temporal structure. However, the temporal order of all unit actions in the context of "preparing coffee" is not strict, and changes from one video exemplar to another. By how many ways one can "prepare coffee"? And by which specific order of unit actions? That is to say, there is a considerable variation of inputs when processing long-range videos. And learning these variations using temporal neural networks is progressively more challenging.

There are three main approaches for modeling the temporal structure of long-range activities. The first approach relies on temporal convolutions [15, 73]. They encode long-range activities as rigid patterns [15] by decomposing them on the receptive fields and by cascading over multiple time scales [73,104]. However, the temporal convolutions in the reference are not capable of handling changes in the temporal order. The second approach is vector aggregation [31, 50], which ignores temporal locality at all. With the temporal locality, however, also temporal refinement is lost including the ability to learn hierarchical temporal abstractions of varying granularity. Hence, vector aggregation is less suited for long-range activities. The third approach relies on self-attention and transformer networks [176, 179] using key-value pair of vectors to capture long-range dependencies. As the key-value pairs are inferred from the input signal [164] similar to meta-learning [38, 137, 168], self-attention approaches are easily disturbed by noisy inputs in long-range activities.

This work proposes a new approach for long temporal modeling, coined Permutation Invariant Convolution, or PIC for short. PIC is designed to have the same benefits and none of the drawbacks of the three aforementioned approaches. Unlike standard convolutions [15], PIC is designed to be invariant to temporal permutations within the local temporal receptive field. And, different from vector aggregation [31, 50] and self-attention [176] approaches, PIC respects temporal locality. Therefore it is able to learn temporal abstraction and deep temporal structure. And last, while adopting a key-value

In submission to *European Conference on Computer Vision (ECCV)*, 2020 [75]

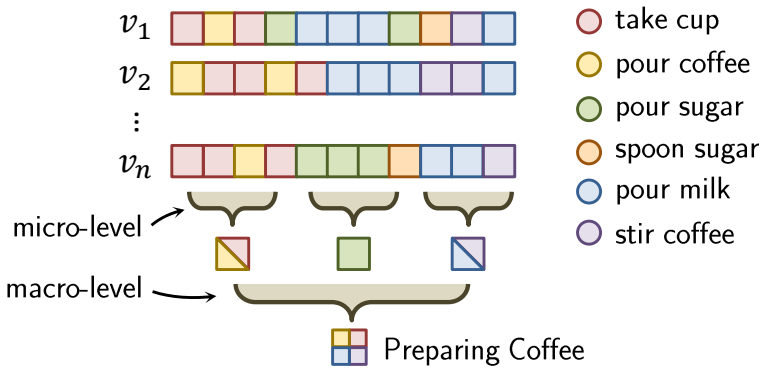


Figure 29: Given a few video examples v_1, v_2, v_3 of the human activity "Preparing Coffee", one video consists of a some unit actions, e.g. "take cup", "stir coffee" or "pour milk". PIC, Permutation Invariant Convolution, recognizes long-range activities using multiple levels of abstractions. On the micro-level of a short segment s_1, PIC models the correlation between unit-actions, regardless of their order, repetition or duration, s_1={○,○} = {○,○,○}. On the macro-level, PIC learns the interactions between segments.

pairs strategy [176], PIC does not infer them from the input data. As a result, PIC is more robust to spatio-temporal noise than existing approaches when recognizing long-range activities. In the end, PIC enables off-the-shelf spatial (2D) and short spatio-temporal (3D) convolutional networks (CNNs) to comfortably process long frame sequences, thus recognizing long-range human activities.

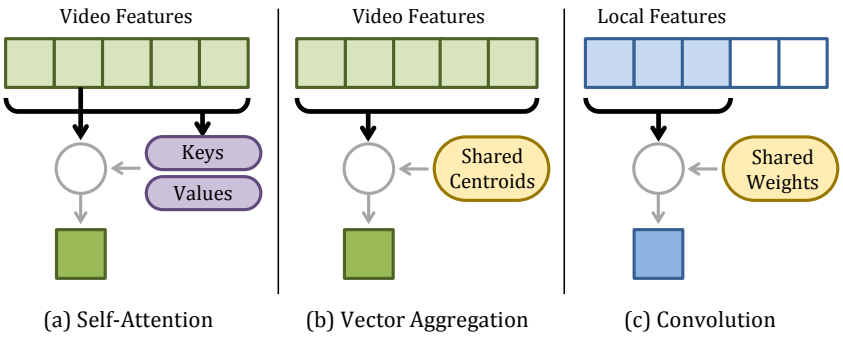
5.2 RELATED WORK

Short-range Activities. Recognizing short- and mid-range actions is a fundamental task in video understanding. These actions usually take up to 10 seconds to occur. For example, actions in sports, such as UCF [148], Sports-1M [90], or human interactions, such as Kinetics [91]. To address these benchmarks, literature propose methods for modeling the pattern, structure [104], order [46], and motion [78, 169] of the temporal signals.

Long-range Activities. There is a recent interest in understanding long-range activities, which brings news challenges. The reason is that these activities are complex [73], take longer to unfold [99] and are harder to model their temporal structure [74]. New benchmarks are proposed, such as Charades [144], Epic-Kitchens [26], Breakfast [99], MultiThumos [77, 188], YouCook [196] or Tasty [140].

This work focuses on modeling and recognizing long-range human activities. After a closer look into the related literature of only long-range modeling, one can conclude the prevalence of three approaches: convolution [73], self-attention [176, 179], and vector aggregation [31, 50], see figure 30.

Convolution. In this vein, convolutional kernels learn to detect patterns within a local window, *i.e.* receptive field. Then, using a cascade of layers, convolution can learn multiple levels of abstractions [97, 147]. So, one can simply attribute the success of



(a) Self-Attention (b) Vector Aggregation (c) Convolution

	(a) Self-attn.	(b) Vector Agg.	(c) Conv.	PIC
Permutation Invariance	✓	✓		✓
Shared Weights		✓	✓	✓
Temporal locality			✓	✓

Figure 30: Compared to other temporal modeling layers, PIC has three benefits. i. Temporal locality to learn long-range temporal abstractions using a cascade of layers. ii. Shared weights (i.e. key-value kernels) to detect the discriminant concepts. iii. Invariant to the temporal permutation within the receptive field, better for modeling weak structures.

convolutional models to two factors: respecting temporal locality and learning complex representations using cascaded layers. The outcome is many successful CNN architectures for image [62, 97] and action understanding [80, 146], temporal localization [31], and sequential modeling [45].

However, temporal convolutions are sensitive to the temporal order, even with multi-scale kernels [73, 151]. Differently, this work proposes PIC, which is invariant to temporal permutation and more permissive to the many temporal configurations exhibited by a long-range activity.

Self-attention. Attention is extensively used in many tasks as image captioning [185], temporal detection [141] and action recognition [30, 108]. Recently, self-attention shows success in machine translation [164] thanks to using a pair of vectors, namely key-value. Self-attention is adopted by various methods for graph representation [166], image recognition [176], and video understanding [48, 179].

Though, the limitation of self-attention [176] is twofold. First, it ignores temporal locality, which is fundamental to learning multiple levels of abstractions [129]. Second, the key-value pairs are inferred from the input, which limits their recognition ability [102]. In contrast, PIC uses weight sharing of the key-value pairs for a better filtering of the visual evidence in a noisy and long activity. Weight sharing is paramount to not only convolution but also to self-attention [102].

Vector Aggregation. This line of work pool feature representations of video frames over long-range sequence. Simple pooling methods is used as max [72], attention [49], and gating [118]. While others opt for more complex aggregation as Vlad [31, 50] and Fisher Vectors [124]. The upside of such methods is scaling up to long-range activities and being invariant to their scale, order and repetition.

Nevertheless, the downside of vector aggregation is that the temporal locality is ignored, and the temporal structure is overlooked. That's why Vlad methods opt for only one layer of temporal modeling. As an alternative, PIC regards temporal locality, thus able to learn multiple levels of abstractions using cascaded layers.

5.3 METHOD

First, we introduce PIC, Permutation Invariant Convolution, and discuss its novelties over existing layers for temporal modeling: convolution [73], self-attention [176] and vector aggregation [50]. Then, we describe how it can be fitted on top of modern CNNs. Finally, we detail the final model architecture and its implementation.

5.3.1 *Motivation*

The structure of the long-range activities can be thought of partially ordered sets, which constitute multiple levels of abstractions. On the macro-level, the entire video v of long-rage activity consists of a few segments v=$\{s_1, s_2, s_3, ...\}$, But on the micro-level, each segment s_i consists a few highly-correlated unit actions, albeit with no particular order or number of repetitions. Take for example the activity of "preparing coffee", see figure 29. One segment contains the unit actions s_1=$\{$"take cup", "pour coffe"$\}$, while another comprise s_2=$\{$"pour sugar", "spoon sugar", "pour milk"$\}$, and so on so forth. It is demonstrated by [73] that the multi-level structure of long-range activity can be learned using convolutional approach with a cascade of layers. The bottom layers learn the correlation between the unit actions within each segment, while the top layers learn the interactions between the segments. First, we discuss the standard temporal convolution, and its limitation in modeling the chaotic structure of long-range activities.

Standard Temporal Convolution. As we are interested in temporal modeling, we omit the spatial dimensions and focus only on the temporal dimension, for clarity. For which, the temporal convolution works as follows. It relies on a learned kernel $W = \{w_i \mid i \in [1, ..., T]\}, W \in \mathbb{R}^{T \times C}$, where T, C are the kernel size and dimension, respectively. At the i-th timestep, the input feature in a local window $X_w = \{x_i \mid i \in [1, ..., T]\}, X_w \in \mathbb{R}^{T \times C}$. This input feature X_w is convoluted $(\circledast)$ with the kernel W, the output feature map is $y \in \mathbb{R}^{1 \times 1}$. So, standard temporal convolution is formulated as

$$y = W \circledast X_w = \sum_{i=1}^{T} w_i \odot x_i^{\top}. \tag{5.1}$$

With this operation, the kernel W learns to detect the exact temporal order of the sequence X_w. However, the downside is that this operation is sensitive to the precise sequential order of X_w. In other words, standard temporal convolutions are not permissive to the many temporal configurations a sequence of unit actions can take place in a long-range activity. For example, there is no one particular order by which the sequence {"pour sugar", "take cup", "pour milk"} can occur in the activity of "preparing coffee". One possible solution is multi-scale convolutions [73]. They can model temporal sequences that differ in their temporal extent. However, they are still sensitive to the

temporal order. Another possible solution is using more convolutional kernels, such that each learns a different temporal order. This solution is computationally prohibitive, and cannot account for all possible permutations, especially for longer temporal patterns.

So, to successfully model long-range activities, a strong requirement is that the convolution operation has to be invariant to the temporal order of unit actions within the local window, *i.e.* within the receptive field. For there exist many ways one can perform the activity of "preparing coffee", with no strict order. To this end, we propose PIC, an invariant convolutional operation to replace the standard convolution for temporal modeling of long-range activities.

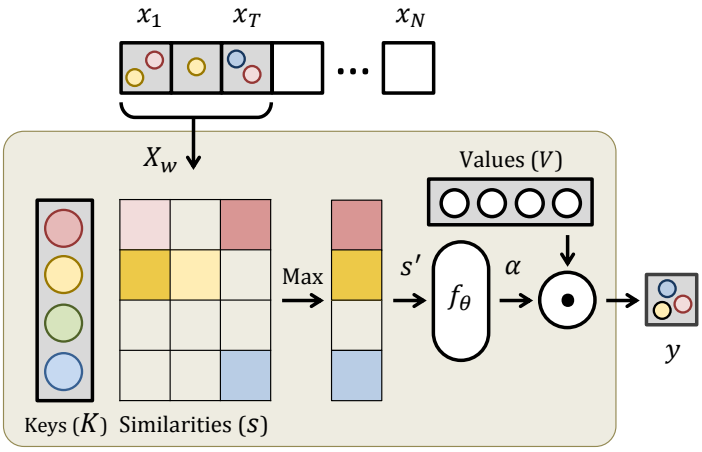


Figure 31: Given a segment of human activity of N frames, each is represented using off-the-shelf CNN as a feature x. Our method, Permutation Invariant Convolution (PIC), operates on the temporal axis of the features using sliding window approach, where the window size is T. In other words, at certain temporal window, it operates on the features $x_1, x_2, ...x_T$. Using a pair of Key-Value kernels (K, V), it models the correlation between the visual evidences {○, ○, ○} in a local window with $X_w = \{x_1, ..., x_T\}$ irrespective of their the temporal order.

5.3.2 *PIC: Permutation Invariant Convolution*

The goal is to make the standard convolution permissive to the weak temporal order of long-range activities. We propose PIC, Permutation Invariant Convolution, see figure 31. PIC takes as an input the features X_w in a local window. To model their correlations regardless of their order, PIC uses a pair of kernels, inspired by self-attention operation [164, 176]. The pair is demoted as the keys $K \in \mathbb{R}^{M \times C}$ and the values $V \in \mathbb{R}^{M \times C}$, where M is the number of kernels, and C is the kernel dimension. The keys K are known to act as a detector for M *latent* visual concepts. Using an outer product $\otimes$ between the keys K and all the features of the local window X_w, we get the similarity matrix $s \in \mathbb{R}^{M \times T}$. Intuitively, s encodes the possibility of any of the M latent concepts to ever exist in the local window. By max-pooling the similarities s over the temporal dimension of the local window, we get $s' \in \mathbb{R}^{M \times 1}$. We interpret s' as the maximum possibility of M concepts to take place in the local window.

After obtaining the maximum similarities s', we opt for values kernel V to represent only those detected. The main purpose of using a pair of kernels K, V instead of one is twofold. First, using a pair enables PIC to decouple detecting the concepts using the keys K, from representing them with the values V. Decoupling is proposed by [164] and successfully used in [176]. Second, by decoupling the kernels, we can have more keys $K \in \mathbb{R}^{M \times C}$ for detection and less values $V \in \mathbb{R}^{M' \times C}$ for representation, where $M' \ll M$.

The next step is using a dense layer $f_\theta(\cdot)$ to model the correlation between the maximum similarities s', and also to embed them from a higher dimension $\mathbb{R}^{M \times 1}$ to a lower dimension $\mathbb{R}^{M' \times 1}$. Then, an activation $\sigma = \texttt{ReLU}$ is used to rectify the similarities, resulting in the activated similarity $\alpha \in \mathbb{R}^{M \times 1}$. The final step is an inner product $\odot$ between the similarities α and the values V to arrive at the final representation $y \in \mathbb{R}^{1 \times C}$. PIC is formulated as

$$s = K \otimes X_w^\top \tag{5.2}$$
$$s' = \max_{\text{row}}(s) \tag{5.3}$$
$$\alpha = \sigma\left[f_\theta(s')\right] \tag{5.4}$$
$$y = \alpha^\top \odot V. \tag{5.5}$$

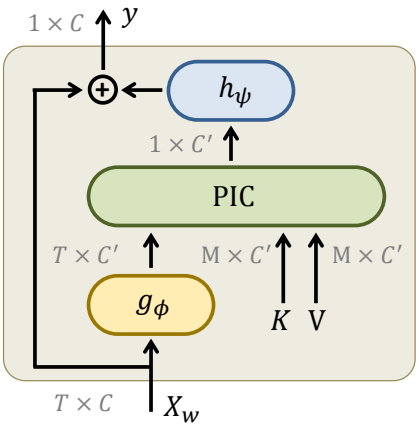


Figure 32: PIC layer models only the temporal dimension. It has shared kernels K, V to learn discriminant concepts. A residual bottleneck is used to reduce computation.

PIC Layer. After outlining the PIC operation, now we discuss how PIC can be used as a modular layer. PIC is a convolutional neural layer placed on top of backbone CNNs – be it 2D or 3D, see figure 32. It draws inspirations and design principles from a few related works [73, 176, 181]. In total, we list four design principles that govern PIC layer. *i.* PIC uses a residual bottleneck for reducing the computation [62, 181]. Before convolving the input features $X_w \in \mathbb{R}^{T \times C}$ with PIC, their dimension is reduced from C to $C' = C/4$ using a dense layer $g_\phi(\cdot)$. And to enable residual connection, the input dimension C is recovered by another dense layer $h_\psi(\cdot)$. *ii.* Instead of using one kernel as in standard convolution, PIC uses a pair of key-value kernels (K, V) [164, 176], to decouple concept detection from concept representation. *iii.* PIC focuses on modeling only the temporal dimension [73], leaving the spatial dimensions for the backbone CNN to handle. *iv.* Similar to the kernels of standard convolution, the kernels K, V learned by PIC are shared weights, *i.e.* model parameters, and are not inferred from the window

features X_w. While in [176], the keys and values K, V are inferred from the input X_w. The upside of having shared kernels K, V is the ability to detect the most representative visual concepts across the entire long-range activity, and not being conditioned on the visual signals in a narrow local window. This is an important design choice for modeling such activities, particularly when we do not know if X_w ever contains informative or noisy evidence. In addition, PIC respects temporal locality. In other words, it convolve the features of local windows X_w, in contrast to global windows used in self-attention [176]. temporal locality enables PIC to learn multiple levels of abstractions with cascaded layers.

5.3.3 *Final Model*

We start with an off-the-shelf backbone CNN, be it 2D CNN as ResNet [62] or 3D CNN as I3D [15]. Then, we stack a cascade of four PIC layer layers on top of the last convolution layer of the backbone CNN. Each layer consists of PIC convolution followed by `BatchNorm` for normalization, `LeakyReLU` for activation, and `MaxPool` with stride 2 for downsampling. Given a video v of long-range activity, we uniformly sample N segments $v = \{s_j \mid j \in [1, ...N]\}$. Each segment s_j consists of $L = 8$ successive video frames, and is processed by the backbone CNN, up to the last convolution layer. The output convolutional feature is $x_j \in \mathbb{R}^{1024\times7\times7}$. We call x_j the feature of the j-th timestep, because it corresponds to the j-th segment of the video. The video-level features are then $X = \{x_j \mid j \in \{1, ..., N\}$, where N is the temporal dimension, or the number of timesteps. To model the temporal structure of the entire video v, we feed-forward the features X to the cascade of PIC layers. Thanks to using a downsampling with stride 2, and four PIC layers in the cascade, the temporal footprint of the input features X is reduced to $N/4$. And so, the output feature is $Z \in \mathbb{R}^{1024\times7\times7\times N/4}$. For video classification, Z is pooled over the spatial and temporal dimensions, and feed-forwarded to a two-layer MLP with `BatchNorm` and `ReLU`. The MLP uses `softmax` and `sigmoid` as the last activation functions for the tasks of single-label and multi-label classification, respectively.

Implementation. For each dataset, we follow a two-stage procedure to train our final model. In the first stage, the backbone CNN is pre-trained on the dataset at hand. We follow the same training details provided by the authors of the backbone CNN, for example I3D [15]. In the second stage, the cascade of PIC layers is placed on top of the last convolutional layer of the backbone CNN. Only PIC layers, along with the classifier, are trained on the dataset at hand, while the backbone is kept frozen. The model is trained for 100 epochs and with batch size 32. For optimization, we opt for SGD with 0.1, 0.9 and 1e-5 as the learning rate, momentum and weight decay, respectively. Also, we experiment Adam with 0.01 and 1e-4 as the learning rate and epsilon ϵ, respectively. TensorFlow [5] and Keras [22] are used for implementation. Code is made public upon publication.

5.4 EXPERIMENTS

Next, we validate quantitatively and qualitatively PIC on three different benchmarks: Charades [144], Breakfast [99], and MultiTumos [188]. We compare against three

state-of-the-art approaches for modeling temporally long video sequences: the non-local networks [176] representing the self-attention approach, ActionVlad [50] representing the vector aggregation approach and Timeception [73] representing the temporal convolution approach. Moreover, we perform an ablation study of PIC properties.

5.4.1 *Datasets*

Charades [144] is video dataset for multi-label action classification, with total number of 157 unit-action classes. It contains 8k, 1.2k and 2k videos for training, validation and test splits, respectively (67 hrs for training split). Each video can be thought of a long-range human activity. On average, each video is 30 seconds and contains 6 different unit actions. Since the videos of Charades are multi-labeled, we opt for the mean Average Precision (mAP) as the evaluation metric.

Breakfast [99] contains 1712 video, split into 1357 and 335 videos for training and testing, respectively. The videos depict unscripted human activities of cooking in the kitchen. The videos are long-range, with 2.3 minutes as the average video length. The main task is to classify these videos into 10 classes of breakfast activities. It is a single-label classification task – each video has only one label. Thus, the classification accuracy (%) is the evaluation method.

MultiThumos [188] contains 400 videos, split into 200 and 213 for training and testing, respectively. The videos depict human activities in sports. Originally, the dataset is used for temporal localization and segmentation of these human actions. Over the entire dataset, there exist 65 action classes. Recently, this dataset is repurposed by [73] for multi-label classification of each long-range video. We adopt the same experimental setup as suggested by [73]. And we use the mAP for evaluation.

5.4.2 *Dissection of PIC*

As presented earlier, PIC is better suited for recognizing long-range activities, thanks to three favorable properties: *i.* invariance to permutation, *ii.* respecting local connectivity, *iii.* using shared weights of key-value pairs. In the following experiments, we dissect the PIC layer to highlight the individual importance of each of these three properties. We use Breakfast [188] as the main dataset to conduct the dissection experiments (*i.e.* ablation studies).

Permutation Invariance. PIC is, by design, invariant to the temporal permutations within windows of local connectivity. It achieves so by two operations: outer product $\otimes$ between the input X_w and the keys K in equation 5.2, and the $\max_{\text{row}}(\cdot)$ operation in equation 5.3. To examine the importance of invariance, we build a variant of PIC, named PIC-Ordered. In which, we convolve $\circledast$ the input X_w with K instead of using outer product $\otimes$. And we remove the $\max_{\text{row}}(\cdot)$ operation, thus making PIC-Ordered dependable on the temporal order within the local window X_w. PIC-Ordered is formalized as

$$\alpha = K \circledast X_w^\top, \qquad y = \alpha^\top \odot V. \tag{5.6}$$

Then, we train baselines accordingly and measure the performance. Timeception is included in this comparison, as it is a multi-scale convolutional layer, and able to handle slight temporal permutations.

Figure 33: Three different ways of sampling timesteps from a test video: uniform, coarse, and fine permutation.

Baseline	Uniform	Coarse Perm.	Drop ↓	Uniform	Fine Perm.	Drop ↓
Timeception	84.6	82.2	2.4	84.6	81.9	2.7
PIC-Ordered	80.2	77.6	2.6	80.2	76.3	3.9
PIC	87.5	87.0	**0.5**	87.5	86.7	**0.8**

Table 14: Being invariant to permutations, PIC is affected the least by altering the temporal order of test videos.

During testing, we use three different ways to sample N timesteps from a test video: *i* uniform, *ii.* coarse permutation, and *iii.* fine permutation, see figure 33. The reason is that we want to introduce randomness to the temporal order, and measure how the baseline methods perform in such cases. Different samplings are used, and the sampling mechanism is as follows. Given a test video v, if we uniformly sample 8 segments $v = \{1, ..., 8\}$, then coarse perm. is $v = \{1, 2, 5, 6, 3, 4, 7, 8\}$, while fine perm. is $v = \{1, 3, 5, 7, 2, 4, 6, 8\}$. Worth mentioning that the during training, the sampling is always uniform.

Results are reported in table 14. Our observation is that not only PIC outperforms other layers, but also has the lowest drop in performance in both cases of fine and coarse perturbations of the temporal information. In addition, we notice that Timeception is slightly more tolerant to perturbations, thanks to its multi-scale kernels. The conclusion is that PIC is more permissible than others to the many ways a long-range activity can happen.

Local *v.s.* Global Connectivity. PIC is a convolution layer with a temporal receptive field of size T. That's to say, given N features corresponding to N timesteps of a video, PIC operates on local windows, each of size T, where $T \ll N$. This gives PIC the ability to learn temporal abstractions of long-range activities at different layers of the network. Our assumption is, if a temporal layer is globally connected to the entire video, then there is no need to cascade multiple layers, as this layer would already summarize all the visual evidence in this video. Note that local connectivity is fundamental to convolutions as well as self-attention [129]. To test this assumption, we devise a variant of PIC, called PIC-Global, that is not restricted by a window size. Its receptive field is as big as the input video $T{=}N$. Then, we train baselines fitted with PIC-Global and PIC. In this comparison, we include ActionVlad and Nonlocal, as both are temporal layers with global receptive field.

Baseline	Accuracy (%) @ Layer			
	1	2	3	4
ActionVlad	83.07	—	—	—
Nonlocal	82.29	83.33	83.07	83.29
PIC-Global	86.76	85.68	85.68	85.42
Timeception	83.85,	84.90	85.30	86.93
PIC	86.20	87.72	88.02	**89.84**

Table 15: Having a local receptive field enables PIC to learn levels of abstractions at multiple layers. Thus, improving monotonically by stacking more layers. Others don't witness the same benefit, as they use global receptive field.

As shown in table 15, both Timeception and PIC improve monotonically by stacking more layers. In contrary, the other layers witness a performance plateau after the first or second layer is the stack. The conclusion is that, the complexity of long-range activities can be captured by a temporal layer of local receptive field. And over a cascade of layers, the entire complexity is learned.

Shared *v.s.* Inferred Kernels. Inspired by the self-attention [164, 176], PIC uses a pair of kernels K, V to learn latent concepts. But the difference is that, in PIC, these kernels are shared, and not inferred from the input video as in [176]. In short-range videos, it is acceptable to have K, V inferred from a sampled segment from the video, it usually contains most, if not all of the representative visual evidence. But in long-range video, the sampled segment might not contain all the evidences. To verify the importance of shared kernels, we construct a variant, named PIC-Inferred. In which, the pair K, V are inferred from the input features X_w using two dense layers $g_\gamma(\cdot), g_\lambda(\cdot)$, similar to Nonlocal [176]. It is formulated as

$$K = g_\gamma\left(X_w\right), \qquad V = g_\lambda\left(X_w\right). \tag{5.7}$$

Then, we train baselines and compare their results. We include Nonlocal in this comparison, as it also uses inferred kernels K, V. The outcome is reported in table 16. We observe that PIC outperforms the other baselines by a considerable margin. The conclusion is, when it comes to modeling the long-range activities, its important for the convolutional temporal layers to use shared kernels.

Baseline	Accuracy (%) @ Layer			
	1	2	3	4
Nonlocal	82.29	<u>83.33</u>	83.07	83.29
PIC-Inferred	82.55	83.85	<u>84.90</u>	84.64
PIC	86.20	87.72	88.02	**89.84**

Table 16: Thanks to sharing the kernels (K, V), PIC is better at learning concepts than layers with inferred kernels.

Studying PIC

5.4.3 *Analysis of PIC*

PIC in a modular temporal layer that resides on top of existing backbone CNNs – be it 2D as ResNet or 3D as I3D. To better utilize it for these CNNs, we analyze the upsides and downsides of PIC. And we study three factors: *i.* effectiveness *v.s.* efficiency., *ii.* optimal sizes of receptive field and downsampling, *iii.* extensibility to input video length, and *iv.* scalability with backbone CNNs.

Effectiveness *v.s.* Efficiency. In this analysis, we demonstrate that PIC is an efficient layer for temporal modeling. Also, we show that PIC scales sub-linearly using deeply cascaded layers. We compare against other layers for temporal modeling. Most notably, we include Timeception [73], as it is known for its efficiency. When quantifying the efficiency, we use four metrics: *i.* CPU feedforward time in milliseconds, *ii.* number of model parameters in millions, *iii.* number of floating point operations in mega FLOPs, and *iv.* classification accuracy of Breakfast activities.

As shown in figure 34, PIC is very efficient layer, and it scales sub-linearly when stacked. One observation is that Timeception and PIC are the most efficient layers, and both brings about monotonic improvements in the accuracy using cascaded layers. Nevertheless, PIC outperforms Timeception by a considerable margin. We conclude from this analysis that PIC achieves the best tradeoff between efficiency and effectiveness.

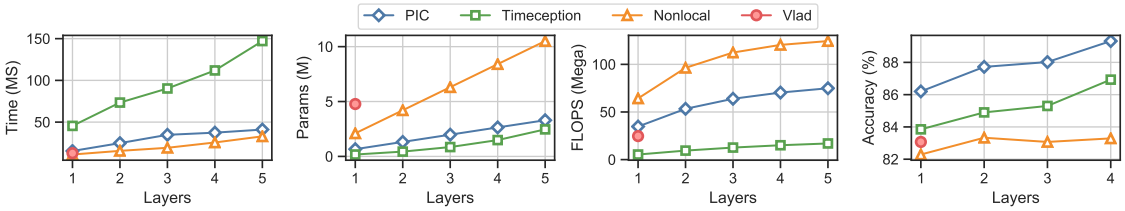


Figure 34: On x-axis, the number of stacked layers. While on y-axes, the the efficiency of temporal layers using four metrics: i. CPU feedforward time (milliseconds), ii. model parameters (millions), iii. number of operations (mega FLOPS), and iv. classification accuracy (%). PIC has the best tradeoff between efficiency and effectiveness.

Size of Receptive Field and Downsampling. PIC is, in principle, a convolutional operation applied to windows of local connectivity along the temporal dimension of long-range activities. As such, two of its most important hyperparameters are the window size and downsampling size. Here, we experiment different configurations to arrive at the best choice. For this, we use two layers of PIC, each of window size T and followed by a max-pooling operation for downsampling, with stride s.

Our observation is that, while increasing the window size helps PIC to have a bigger receptive field, this improvement degrades for $T > 9$. Based on this analysis of Breakfast dataset, the recommended window size is $T = 9$. As for the downsampling, we find that $s = 2$ is the optimal stride, while more aggressive strides $s = \{3, 4\}$ are detrimental.

Number of Latent Concepts. PIC makes used of shared pair of kernels (K, V), where M, M' are the number of K and V respectively. In our experiments, we found that

Stride Size (s)	Window Size (T)				
	3	5	7	9	11
2	86.98	86.98	87.24	**88.02**	87.50
3	86.98	86.72	86.98	86.20	86.20
4	85.68	85.68	85.94	85.42	85.68

Table 17: PIC accuracy when changing the convolution window size T and the down-sampling stride size s.

choosing these hyperparameters is of importance to the accuracy. A rule of thump is, for large datasets, as Charades, where there are many action categories, using large number of keys and values $M = M' = \{64, 128\}$ is important. While in medium-scale datasets, as Breakfast, we found that as little as $M = M' = \{16, 32\}$ would suffice.

5.4.4 *Quantitative Analysis*

Breakfast. We compare against three state-of-the-art temporal modeling layers. In this experiment, stacking four layers was optimal for Timeception [73]. Also we report results for Timeception after fine-tuning the backbone, as this yields higher accuracy than the numbers reported originally [73]. For Non-local networks, we found that stacking two layers yielded the best performance. For PIC we rely on a deep cascade of four layers. We follow the same experimental setup for all methods and report results in table 18. We observe that PIC comfortably outperforms all other methods, including the recent method Timeception.

Method	Backbone	Accuracy (%)
3D CNN*	I3D	64.31
3D CNN* + Timeception [73]	I3D	69.30
3D CNN* + **PIC**	I3D	**73.62**
3D CNN	I3D	80.64
3D CNN + Vlad [8]	I3D	82.67
3D CNN + Nonlocal [176]	I3D	83.79
3D CNN + Timeception [73]	I3D	86.93
3D CNN + **PIC**	I3D	**89.84**

Table 18: We report the accuracy of classifying the minutes-long activities of Breakfast. PIC outperforms the other baseline methods by a considerable margin. $*$ denotes that the backbone CNN is not-fined tuned, so we can be compared with [73].

Charades. Different from the Breakfast dataset, Charades is even more challenging, as it requires multi-label classification. On average, the complex video of Charades consists of six unit actions. As Charades videos are noticeably shorter than Breakfast videos, deep cascades of three PIC layers suffice. Furthermore, on this dataset, we compare with other temporal modeling methods, as Timeception [73], as well as with FeatureBanks [179]. We report results in table 19 using two different backbone CNNs,

namely R101 and R101-NL, as in the latest literature [176]. We observe that PIC attains the highest accuracy for all backbones.

Method	Backbone	mAP (%)
SlowFast [33]	—	42.1
SlowFast-NL [33]	—	42.5
3D CNN [176]	R101	35.5
3D CNN + Timeception [73]	R101	41.1
3D CNN + **PIC**	R101	**42.7**
3D CNN [179]	R101-NL	41.0
3D CNN + FeatureBanks [179]	R101-NL	42.5
3D CNN + **PIC**	R101-NL	**43.8**

Table 19: When classifying the complex multi-label actions of Charades, PIC layers bring improvements over previous works.

MultiThumos. Last, we evaluate PIC on MultiThumos. We follow the same experimental setup as suggested by [73] and use their backbone CNN without fine-tuning on MultiThumos. We report results in table 20. We observe PIC obtains state-of-the accuracy in recognizing the complex actions of MultiThumos.

Method	Backbone	mAP (%)
3D CNN	I3D	72.43
3D CNN + Timeception [73]	I3D	74.79
3D CNN + **PIC**	I3D	**78.31**

Table 20: PIC improves over related works in recognizing the multi-labeled, long-range videos of MultiThumos.

5.4.5 *Qualitative Analysis*

Visualizing Learned Concepts. PIC learns *latent* concepts using the key kernels K. We visually interpret these kernels on Breakfast dataset. The model used is a cascade of two PIC layers. We retrieve the top related video frames to each of the latent concept as per the similarity values s' in equation 5.3. We visualize results in figure 35. We observe that the bottom PIC layer learns fine-grained concepts that are shared across the activity categories of Breakfast. For example, in figure 35 the concept "pouring" is not specific to a particular category – be it "preparing coffee" or "making tea". Also, we found that the concept "cutting" can be shared among different activities, as "making salad" or "'making sandwich'. What is more, we observe that these discovered concepts can either be object-centric or action-centric. An object-centric concept means that it corresponds to single frames, like "food box". While action-centric concept means that it corresponds to short segments of unit actions, like "cutting".

Figure 35: In a cascade of PIC layers, we notice that in the bottom layer, the learned concepts are fine-grained and independent of the activity category. For example, the concept "pouring" is irrespective of activities "make coffee" or "make tea". Also, these concepts can be object-centric as "food box" or action-centric as "cutting".

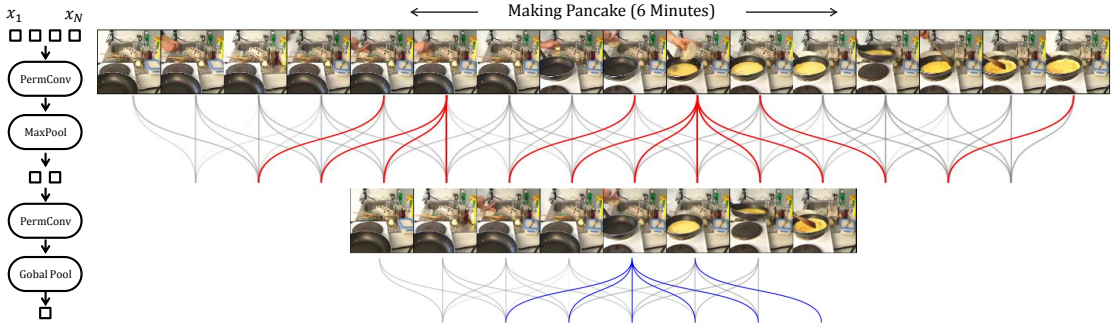


Figure 36: This figure shows 16 frames uniformly sampled from an activity of "Making Pancake". After one layer, M concept kernels are learned to detect relevant visual evidences. For simplicity, we show the activations of only one kernel per layer. In red, the activations of one kernel in bottom layer. While in blue, the activations of another kernel in the top layer.

Long-range Temporal Dependencies. Last, we visualize a conceptual overview of how a two-layer PIC cascade operates on the frames of a long-range activity, see figure 36. In each layer, we apply PIC convolution on sliding window along the temporal dimension. Each layer learns a set of N kernels, where each kernel k learns a specific latent visual concept. After the first layer, we downsample the temporal dimension and apply a second PIC layer that learns a new set of latent concepts. For clarity, we show the activation of only one kernel per layer. For example, the red color represents the activation of only one kernel in the bottom layer. Notice how the same concept can be detected over the entire video. While the blue color represents the activation of yet another kernel in the top layer.

5.5 CONCLUSION

This work introduces PIC, Permutation Invariant Convolution, a neural block dedicated to the temporal modeling of long-range activities in videos. It has three properties. First, being invariant to temporal permutations enable it to handle the chaotic temporal orders of long-range activities. Second, it respects temporal locality, so it can learn deep temporal abstractions using a cascade of layers. Third, it uses shared weights, namely

key-value pairs, to learn the most representative visual signals in long and noisy videos. We demonstrate the effectiveness of PIC layers, along with its three properties. Most notably, we show how PIC enables existing CNNs to model long-range activities and improve the performance. We benchmark on three datasets of long-range activities, where we improves on the previous methods.

6

TIMEGATE: CONDITIONAL GATING OF SEGMENTS IN LONG-RANGE ACTIVITIES

6.1 INTRODUCTION

A human can skim through a minute-long video in a few seconds, and still grasp its underlying story [154]. This extreme efficiency in temporal information processing raises a question. Can a neural model achieve such efficiency in recognizing minutes-long activities in videos?

Related works propose different CNN models with efficiency in mind [67, 192, 198]. However, such models [95] address only short-range actions, as in Kinetics [91], UCF-101 [148], or HMDB [100]. On average, these actions take ten seconds or less, where recognizing only a few frames would suffice [138]. However, this work focuses on long-range activities, as in Charades [144], Breakfast [99] or MultiThumos [189]. These activities can take up to a few minutes to unfold. Current methods fully process the entire video of long-range activity to successfully recognize it [15, 175]. As a result, the major computational bottleneck of such methods is the sheer number of video frames to be processed.

Another solution is frame sampling [190]. The recently proposed SCSampler [96] achieves efficiency by sampling the most salient segments from an untrimmed video of short-range action. The sampling is conditioned on only the segment level, which is plausible for short-range actions in trimmed videos, such as Kinetics [91] or untrimmed videos, such as Sports-1M [90]. The reason is that, on the segment level, one can easily tell if the segment is relevant to the action or it is just background, see figure 1. So, segment-level classification probabilities would suffice for sampling [96]. In contrast, long-range activities are known for being diverse and complex [73, 187]. Thus, the importance of one segment to a certain activity is not self-described but rather depends on the context, *i.e.* the long-range activity itself. That is to say, while a segment is relevant to one activity, it is not relevant to another. So, sampling conditioned only on the segment level is not the most optimal choice for long-range activities.

To address the limitations of the previous methods, we propose TimeGate, a two-stage neural network for the efficient recognition of long-range activities without compromising the performance. Different from previous sampling methods, such as SCSampler, TimeGate solves two problems. *i.* Conditional selection: when selecting segments from the long-range activity, TimeGate is conditioned on both the segment- and context-level features. Context-conditioning better suited for long-range activities than only

In submission to *European Conference on Computer Vision (ECCV)*, 2020 [76]

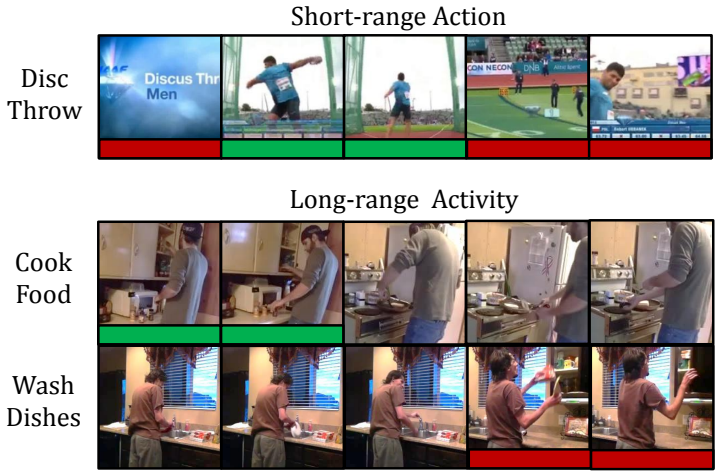


Figure 37: Top, short-range action "disc throw" in an untrimmed video. Based on each segment, you can tell whether is it relevant (green) to the action or not (red). But in long-range activities, middle and bottom, the importance of each segment is conditioned on the video context. The segment "get food from cupboard" is relevant to "cook food" but not to "washing dishes".

the segment-conditioning of SCSampler. *ii.* Differentiable gating: the selection mechanism of TimeGate is differentiable, so it is trained end-to-end with modern 2D and 3D CNNs [15, 62], resulting in a better performance.

Our novelties are: *i.* Gating module for the conditional sampling of segments in videos. Our gating is more suited to long-range activities than other methods, such as SCSampler. The reason is that the sampling is conditioned on the segment- and context-level features. *ii.* The proposed gating module is differentiable, which enables end-to-end training with existing CNNs. *iii.* The proposed model, TimeGate, reduces the computational cost of existing CNNs in recognizing long-range activities. In end-to-end training, the cost is reduced even further. Finally, we conduct experiments and report the results on three datasets for long-range activity recognition: Charades [144], Breakfast [99] and MultiThumos [189].

6.2 RELATED WORK

Long-range Activities. Short-range actions, such as Kinetics [91] and UCF-101 [148], have an average length of 10 seconds or less. Practically, they can be classified with CNNs using as little as ten frames per video [172], and in some cases, even one frame would suffice [138]. Therefore, building efficient CNNs is a plausible choice to reduce the computational cost of recognizing short-range actions. However, in long-range activities, such as Charades [144] and Breakfast [99], the activity can take up to five minutes to unfold. Thus, requiring as many as a thousand frames [73, 74] to be correctly classified. As such, analyzing all the frames using efficient CNNs is still computationally expensive.

Nevertheless, having a mechanism to select the most relevant frames can boost efficiency [11]. Therefore, this work focuses on reducing the number of video frames

needed for activity recognition. Though, our work is orthogonal to prior work of efficient CNNs for action recognition.

Efficient Architectures. CNNs are the go-to solution when it comes to video classification. Thus, one prospective of reducing the computation of video recognition is to build efficient CNNs. Methods for pruning less important weights [59, 61] or filters [106] were previously proposed. Careful design choices result in very efficient 2D CNNs such as MobileNet [66] and ShuffleNet [192]. These 2D CNNs are extended to their 3D counterparts, such as ShuffleNet3D and MobileNet3D [95], to learn spatio-temporal concepts for video classification. Neural architecture search [197] is used to find the lightweight NasNet-Mobile [198].

While efficient architectures are successful in the case of short-range actions, they are not the most viable solution for long-range activities. The reason is that these activities span up to a few minutes. Naively processing the video in its entirety undermines the computation saved by these efficient CNNs. In other words, in the case of long-range activities, the computational bottleneck is the sheer number of video segments needed to be processed.

Conditional Computing. Another solution to reduce the computation is to dynamically route the computational graph of a neural network. The assumption is that not all input signals require the same amount of computation – some are complicated while others are seemingly easy. Thanks to categorical reparameterization [79], it becomes possible to discretize a continuous distribution, and effectively learn binary gating. In [165], a dynamical graph is built by gating the layers of a typical CNN. While in [10, 20], the gating is achieved on the level of convolutional channels. In the same vein, GaterNet [20] proposes a separate gating network to learn binary gates for the backbone network.

Rather than gating the network layers, this work focuses on gating the video frames themselves, to realize the efficiency in recognizing long-range activities. In all cases, our work benefits from prior work of differentiable gating [79].

Sampling of Video Segments. Several works discuss frame sampling for short-range videos. In [11], a student-teacher model for trimmed video classification is presented. With reinforcement learning in [190], an agent predicts the next move. Most recently, SCSampler [96] proposes a method for sampling salient segments in the untrimmed videos of Sports-1M [90]. Conditioned on only the segment, it predicts a score for how salient this segment is to the action.

Conversely, in long-range activities, the importance of each segment is conditioned on not only the segment but also its context. Thus, SCSampler is less suited for such activities. This work presents TimeGate, a novel selection method tailored for these activities.

6.3 METHOD

6.3.1 *TimeGate*

Model Overview. TimeGate consists of two stages: timestep selector and video classifier, see figure 38. The first stage is the selector, which consists of a lightweight CNN,

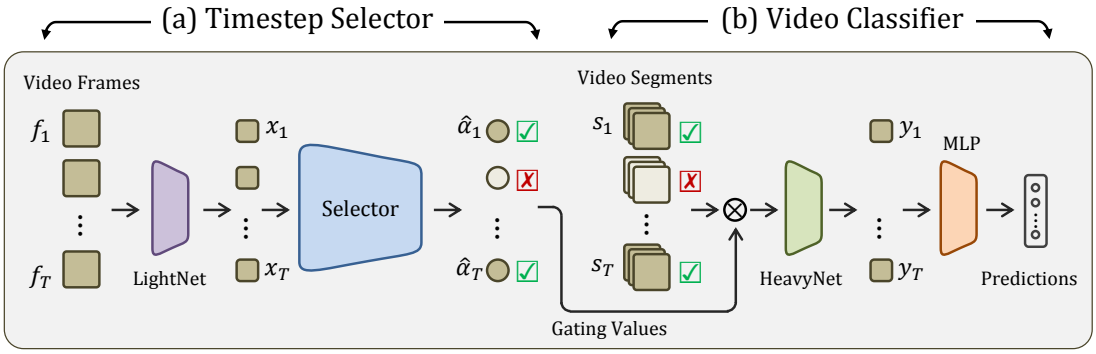


Figure 38: Overview of the proposed model, TimeGate, with two stages. The first stage is the timestep selector, left. Based on a lightweight CNN, LightNet, the model learns to select the most relevant timesteps for classifying the video. This selection is conditioned on both the features of timestep and its context. The second stage is the video classifier, right. In which, only the selected timesteps (☑) are considered, while the unselected timesteps (☒) are completely ignored. In this stage, a heavyweight CNN, HeavyNet is used for feature representation of only the selected timesteps, followed by MLP for classification.

LightNet, followed by a novel gating module, see figure 39. Its purpose is to select the most relevant timesteps from a minutes-long video. The second stage is the classifier. Its purpose is to learn deep and discriminatory video-level representations for maximum classification accuracy. Thus, it resides on top of a heavyweight CNN, *HeavyNet*, followed by a Multi-Layer Perceptron (MLP) for classification. Only the timesteps chosen by the first stage, the timestep selector, are considered by the second stage, the video classifier.

Timestep Selector. The selector takes as an input a uniformly sampled T frames from a long-range video $v = \{f_i \mid i \in [1, ..., T]\}$. All the frames are represented as convolutional features $X = \{x_i \mid i \in [1, ..., T]\}$, $X \in \mathbb{R}^{T \times C \times 1 \times 1}$, where C is the number of channels. The objective of the selector is to choose only a few of these features. In other words, we want to select only the timesteps that are most representative of the activity in the video, where each timestep is represented as a feature x_i. Our hypothesis is that, a lightweight feature representation using an efficient CNN, *LightNet*, would suffice for the selection. Thus, the features X are obtained from the last convolutional layer of the LightNet, and average-pooled globally over space, so the spatial dimensions of X are 1×1.

Concept Kernels. The next step is to take binary decision of considering or discarding the timesteps. But how to decide if a timestep feature x_i is relevant or not? Conceptually speaking, a long-range activity consists of few yet dominant and discriminative visual evidences, based on which, the video can be recognized [73]. Take for example "making pancake". One can easily discriminate it by observing the evidences "pancake", "eggs", "pan", and "stove". These evidences can be thought of as *latent* concepts. To represent them, we learn a set of concept kernels $K = \{k_1, k_2, ...k_N\}, K \in \mathbb{R}^{N \times C}$, where N is the number of kernels, and C is the kernel dimension. K are randomly initialized and are part of the network parameters. They are learned during the training of the selector. Our concept kernels K are reminiscent of the centroids in ActionVlad [50].

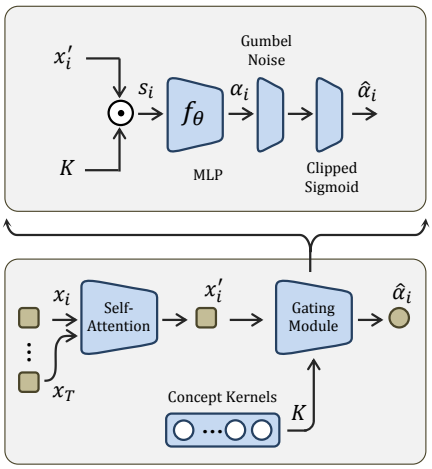


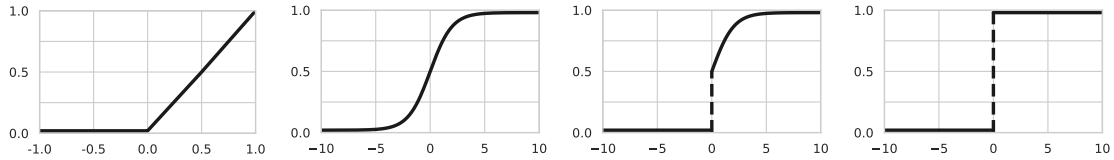
Figure 39: Bottom, the timestep selector learns concept kernels K to represent the most representative visual evidence. Top, the gating module learns to select only a timestep feature x_i according to its importance to the current video.

Gating Module. The purpose of the gating module is to select the video timsteps, see figure 39, top. The first step is to measure how relevant each timstep feature x_i is to all of the concept kernels K using an inner product $\odot$. The result is the similarity vector $s_i = K^\top \odot x_i$, $s_i \in \mathbb{R}^{N\times 1}$. Our understanding is that the vector s_i encodes how relevant a timestep is to each of N concept kernels. Then, based on this similarity vector s_i, we want to take a binary decision of considering or discarding the timestep feature x_i. That's why we model the similarity vector s_i using a two-layer MLP $f_\theta(\cdot)$, with a single neuron in the output layer, denoted as $\alpha_i = f_\theta(s_i)$, $a_i \in \mathbb{R}^1$.

Intuitively, α_i is the gating decision corresponding to the timestep feature x_i. Since α_i is a continuous variable, we cannot make a binary gating decision. Thus, we make use of [79] to discretize α_i to binary gating variable $\hat{\alpha}_i$. More formally, following the gating mechanism of [10], we add gumbel noise G to α_i and follow with `sigmoid` activation, thus $\hat{\alpha}_i = \texttt{sigmoid}(\alpha_i + G)$ .Then, we apply binary thresholding using the threshold value $\delta = 0.5$. So, we arrive at the binary gating value $\hat{\alpha}_i = \mathbb{I}_{(\delta>0.5)}(\delta)$, $\hat{\alpha}_i \in \{0, 1\}$, see figure 39, top. Finally, for gating the i-th timestep, we multiply its feature x_i with the binary value, resulting in the gated feature $\hat{x}_i = x_i \cdot \hat{\alpha}_i$, $\hat{x}_i \in \mathbb{R}^{C\times 1\times 1}$.

Gating Activation. A problem with using binary thresholding for gating, as in [10], is that during training, the classifier does not know out of the gated timestep features, which is more relevant than the other. Each x_i is multiplied by a binary value $\hat{\alpha}_i \in \{0, 1\}$. As a remedy, we propose `clipped-sigmoid` activation to replace the `sigmoid` activation used in [10]. We find that this simply modified activation `clipped-sigmoid` is better suited for timestep gating due to three desirable properties, see figure 3. *i.* Being a relaxed version of the `step function` makes it differentiable. *ii.* Retaining the `sigmoid` value above the threshold means that the classifier gets the chance to know, out of the selected timesteps, which is relatively more important than the other. *iii.* Conversely to `ReLU`, the activation `clipped-sigmoid` is upper-bounded by one, thus preventing a

single timestep feature x_i from dominating the others by being multiplied by unbounded gating value $\hat{\alpha}_i$.



(a) ReLU (b) Sigmoid (c) Clipped sigmoid (d) Step Function

Figure 40: In training, we use `gated-sigmoid` *to activate the gating value α_i and to select the timesteps.* `gated-sigmoid` *has some desirable properties. i. Unlike* `ReLU`*, having upper bound does not allow a timestep feature to dominate others. ii. Different from* `sigmoid`*, being clipped allows the network to discard insignificant timesteps, i.e. those with gating values $\alpha_i < 0.5$. In test, we replace the* `gated-sigmoid` *with* `step function` *for binary gating of timesteps.*

Context-Conditional Gating. Up till now, the selector learns to gate each timestep regardless of its context, *i.e.* the other timesteps in the video. To achieve context-conditional gating, where both the timestep and its context affect the gating decision, we opt for a temporal modeling layer, self-attention [175], before the gating module, See figure 39, bottom. This layer learns to correlate each timestep x_i with all the others in the video $\{x_1, ..., x_T\}$ before gating.

Sparse Selection. The last component of the selector is to enforce sparsity on timestep selection, *i.e.* choose as few timesteps as possible, yet retain the classification accuracy. Simply put, the selector can cheat by predicting gating values just higher than the threshold $\alpha > \delta$, $\delta = 0.5$, resulting in all gates opened and all timesteps selected. The selector has a natural tendency to such a behaviour, as the only loss used so far is that of the classification. And the more timesteps used by the classifier, the better the classification accuracy. To prevent such a behaviour, we apply L_0 regularization [10, 110] to all the gating values $\{\hat{\alpha}_i \mid i \in [1, ..., T]\}$ to enforce sparsity on the selected timesteps. We note that the sparsity regularization is necessary for a properly functioning gating mechanism.

Video Classifier. The assumption of TimeGate is that having efficiently selected the most crucial timesteps from the video using the LightNet and the selector, one can opt for a much more powerful HeavyNet to effectively classify the video. Thus, the second stage of TimeGate is the video classifier, see figure 38, left. This classifier takes as input only the subset T' of timesteps chosen by the selector, $T' \subset T$, $T' \ll T$. Each timestep is represented as the feature of last convolutional layer of the HeavyNet. The video-level features are denoted as $Y = \{y_i \mid i \in [1, ..., T']\}$, $Y \in \mathbb{R}^{T' \times C' \times H \times W}$, where C' is the number of channels, T' is the number of selected timesteps, and H, W are the spatial dimensions. After the last convolutional layer, the video level features Y are max-pooled over the spatial dimension and fed-forwarded to a two-layer MLP for classification. We follow [175] and max-pool the temporal dimension before the MLP `logits`.

6.3.2 *TimeGate Implementation*

Backbone Choices. LightNet and HeavyNet are the backbone CNNs used by TimeGate. Our choice for the LightNet is MobileNet-V3 [136]. As for the HeavyNet, we explore three choices. A powerful 3D CNN I3D [15], an efficient 3D CNN ShuffleNet3D-V2 [95], and a powerful 2D CNN ResNet2D-50 [62]. Before training TimeGate on a specific dataset, the backbone CNNs are pre-trained on the dataset at hand. We use the same training procedures specified by the authors of these CNNs.

Timestep Alignment. When the HeavyNet is a 3D CNN, the i-th timestep feature y_i is obtained from processing the i-th video segment s_i of M successive frames $s_i = \{f_j, ..., f_{j+M}\}$. For I3D, $M = 8$, and for ShuffleNet3D, $M = 16$. But since the LightNet of the selector is a 2D CNN, how can we align the timestep of the selector, with that of the classifier? Simply put, for the aforementioned HeavyNet feature y_i, the aligned LightNet feature x_i has to be obtained from the middle frame of the video segment s_i. More formally, the frame $f_{j+\lceil M/2 \rceil}$.

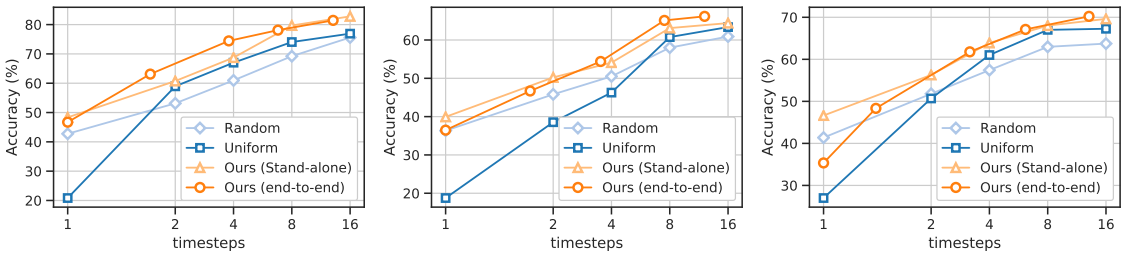
Model Training. TimeGate is trained with batch size 32 and for 100 epochs. We use Adam with learning rate $1e$-3 and epsilon $1e$-4. We use PyTorch and TensorFlow for our implementation. As for the number of concept kernels N, we found that $N = 128$ is a good choice for all the experiments, similar to [74]. As for the gating module, see figure 39, during the training phase, we use gumbel noise and `clipped-sigmoid` to get the activated gating value $\hat{\alpha}_i$. In the test phase, we don't use gumbel noise, and we replace `clipped-sigmoid` with `step-function`, to get the binary gating value $\hat{\alpha}_i = \mathbb{I}_{(\delta > 0.5)}(\delta)$. That means alpha is binarized $\hat{\alpha}_i \in \{0, 1\}$ with thresholding $\delta = 0.5$.

6.4 EXPERIMENTS

6.4.1 *Datasets*

Charades is a widely used benchmark for human action recognition. It is a diverse dataset with 157 action classes in total. The task is mult-label recognition, where each video is assigned to one or more action class. It is divided into 8k, 1.2k and 2k videos for training, validation and test splits, respectively, covering 67 hours. On average, each video spans 30 seconds, and is labeled with 6 and 9 actions for training and test splits, respectively. Thus, Charades meets the criteria of long-range activities. We use Mean Average Precision (mAP) for evaluation.

Breakfast is a benchmark for long-range activities, depicting cooking activities. Overall, it contains 1712 videos, divided into 1357 and 335 for training and testing, respectively. The task is video recognition into 10 classes of making different breakfasts. Added to the video-level annotation, we are given temporal annotations of 48 unit-actions. In our experiments, we only use the video-level annotation, and we do not use the temporal annotation of the unit-actions. The videos are long-range, with the average length of



(a) I3D (b) ShuffleNet3D (c) ResNet2D

Figure 41: Our stand-alone timestep selector helps improving the performance and reduces the computation of off-the-shelf CNN classifiers – be it 2D/3D heavyweight CNN or even lightweight 3D CNN. More over, if TimeGate is trained end-to-end, the selector learns a better gating to the benefit of the classifier. So, the performance is improved even further.

2.3 minutes per video. Which makes it ideal for testing the efficiency of recognizing long-range activities. The evaluation method is the classification accuracy.

MultiThumos is a benchmark for long-range videos, depicting sports activities. It consists of 413 videos, divided into 200 and 213 for training and testing, respectively. Each video has multi-labels, where the total number of action classes across the dataset is 65. The average length is 3.5 minutes per video. The original task of this dataset [189] is the temporal segmentation of these short-range actions. Recently, it is repurposed by [73] into multi-label classification of long-range videos. We adopt the same experimental setup of [73]. That is to say, each long-range video is classified into multi-labels, and the mAP is used for evaluation.

Ablation Studies. We use Breakfast as the primary dateset for the ablation experiments and studies. These experiments highlight our contributions, as follows. *i.* In § 6.4.3, we discuss to what extend the end-to-end training of TimeGate helps. *ii*. In § 6.4.4, we show how context-conditional gating is more important than frame-conditional. *iii.* In § 6.4.2, 6.4.5, we demonstrate the improvements of TimeGate over the current CNN classifiers, in terms of accuracy and efficiency.

6.4.2 *Stand-alone Timestep Selector*

One might raise an important question – will a timestep selector based on LightNet features X benefit a classifier based on HeavyNet features Y, given the differences between the feature spaces of LightNet and HeavyNet $C \neq C'$? To answer this question, we construct an experiment of two steps on Breakfast. The first step is training a stand-alone selector, where we choose MobileNet for both LightNet and HeavyNet. During training, we randomly sample $T = 32$ timesteps from each video. Since MobileNet is a 2D CNN, a timestep here is practically a video frame. With the help of the L_0 regularization, the selector achieves sparse selection of timesteps, by as little as $T' = 16$ without degrading the classification performance. The second step is testing how will the selector benefit off-the-shelf CNN classifiers: I3D, ShuffleNet3D and ResNet2D. Then, we measure their performance using different time scales. More formally, from each

Baseline	Accuracy (%) @ Timesteps						
	4	8	16	32	64	128	256
R2D	61.0	67.1	67.3	71.0	72.9	74.3	73.8
R2D+**TG**	63.9	68.2	70.2	73.3	74.3	**76.4**	74.3
S3D	46.3	60.8	63.4	67.2	67.3	65.8	66.3
S3D+**TG**	54.4	65.1	66.2	**69.8**	69.7	66.7	67.8
I3D	66.8	74.3	82.8	84.7	85.7	86.5	85.4
I3D+SCS [96]	61.4	74.7	81.8	84.4	84.4	85.4	84.6
I3D+**TG**	69.5	77.9	85.2	85.9	86.7	**88.1**	86.5

*Table 21: The stand-alone selector of our model TimeGate (**TG**) benefits off-the-shelf CNN classifiers. The benefit is consistent for various classifiers: I3D, ShuffleNet3D (S3D), and ResNet2D (R2D).*

test video, we sample T' timesteps , $T' \in \{1, 2, 4, 8, 16\}$, and we use different sampling methods: random, uniform, and timestep selector. During testing, the output of the timestep selector is a per-timestep binary value $\hat{\alpha}_i \in \{0, 1\}$ of whether to consider or discard the i-th timestep. So, if T timesteps are processed by the selector, it is able to choose a subset T' timesteps and discard the others, where $T' \subset T, T' \ll T$. And to evaluate the benefit of the selector, the off-the-self classifier then uses only T'.

As reported in table 21, and shown in figure 41, we observe that the stand-alone selector improves the accuracy of off-the-shelf classifiers. The reason is that the selector, based on LightNet, is able to select the most relevant timesteps from the video. Also, we notice that the improvements are consistent for three different classifiers: I3D, ResNet2D and ShuffleNet3D.

6.4.3 *End-to-End TimeGate*

Having experimented with the stand-alone selector, we pose another question. Is it possible to train TimeGate end-to-end, given that the selector and the classifier are based on two different CNNs, with two different feature spaces, $C \neq C'$?. Our experiments show that indeed, in end-to-end training, the gating module learns a better selection to the benefit of the classifier. The outcome is improvement in performance over the stand-alone selection, as reported in table 22. We conclude that in end-to-end, the gating module learns to determine the importance of the i-th heavyweight feature y_i based on the corresponding lightweight feature x_i.

In addition, figure 42 shows the average ratio of selected timesteps for each activity class of Breakfast dataset. The ratios of the stand-alone (red) is changed when it is trained end-to-end with different HeavyNet: ResNet2D, (blue), I3D (yellow), and ShuffleNet3D (blue). We observe that these ratios have similar trends when the HeavyNet is 3D CNN, regardless of which 3D CNN is used. Between yellow and blue, there is a similar trend in 8 out of 10 activities. However, these ratios tend to vary between 2D and 3D as HeavyNet – only 3 out of 10 actions tend to have similar trends, see green and yellow. From this

Baseline	Accuracy (%) @ Timesteps						
	4	8	16	32	64	128	256
SCSampler [96]	61.4	74.7	81.8	84.4	84.4	85.4	84.6
TimeGate **SA**	69.5	77.9	85.2	85.9	86.7	88.1	86.5
TimeGate **ETE**	74.4	78.1	82.9	86.7	87.4	**89.3**	86.1

*Table 22: Our stand-alone (**SA**) selector benefits off-the-shelf CNN classifiers. End-to-end (**ETE**) training is even better.*

Baseline	Accuracy (%) @ Timesteps						
	4	8	16	32	64	128	256
SCSampler [96]	61.4	74.7	81.8	84.4	84.4	85.4	84.6
TG Frame	69.2	73.8	80.7	81.5	83.9	83.1	83.6
TG Context	69.5	77.9	85.2	85.9	86.7	**88.1**	86.5

Table 23: TimeGate (TG) is better when the gating module is conditioned on both the frame-level and the context-level. More over, TimeGate outperforms SCSampler in long-range activities.

experiment, we conclude that the gating module, depending on LightNet features, learns to select better timesteps to the benefit of the HeavyNet classifier.

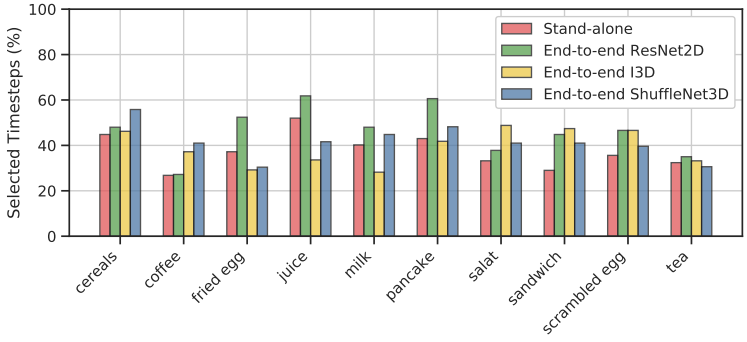


Figure 42: The ratios of selected timesteps for the activity classes of Breakfast. Note the change in these ratios from stand-alone selector (red) to end-to-end training with the HeavyNets: ResNet2D (green) I3D (yellow) and ShuffleNet3D (blue).

6.4.4 *Context-Conditional Gating*

When selecting the timesteps of long-range activities, TimeGate is conditioned on both the segment and its context. This context-conditioning is an important novelty of TimeGate. Also, this property is desired for long-range activities, because the importance of a certain segment is not always self-described, but rather depends on the context.

To validate this assumption, we design the following experiment. We devise a baseline model of our timestep selector, that does not have a temporal layer before the gating module. Thus, in this baseline, the gating is frame-conditioned. Also, we include SCSampler [96] in this comparison. We use I3D for the HeavyNet and we use MobileNet as the backbone CNN for both our timestep selector and SCSampler. As reported in table 23, we observe a drop in the performance when using the frame-conditioned TimeGate. The reason is that, for long-range activities, its important for the selector to pay attention to the context of the video segment before sampling it.

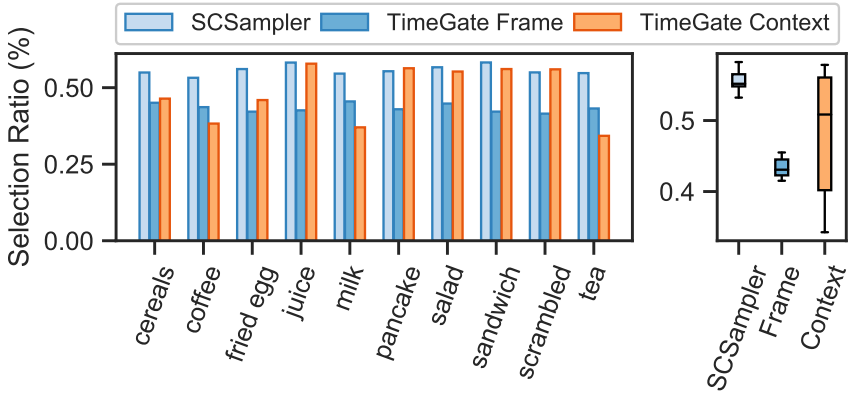


Figure 43: In both the frame-conditioned TimeGate and SCSampler, the ratio of the selected timesteps have small variance across the activity classes of Breakfast. In contrast, in context-conditioned TimeGate, the ratio is highly dependent on the activity, which means context-conditional gating is archived.

We report another analysis in figure 43. On the left, we show the ratio of selected timesteps for each activity class of Breakfast. The frame-conditioned gating (dark blue) tends to select similar ratios regardless of the category, so does the SCSampler (light blue). In contrast, we see more diverse ratios for the context-conditioned gating. Figure 43, right, shows the ratio variances. The much higher variance for context-conditional TimeGate means that it is more dependent on the activity class than the case of SCSampler or frame-conditional TimeGate.

6.4.5 *Computation-Performance Tradeoff*

When it comes to the recognition of long-range activities, the golden rule is the more timesteps the better the accuracy, and the heavier the computation. But given the huge redundancies of the visual evidences in these timesteps, there is a tradeoff between accuracy and computation. In this experiment, we explore what is the effect of such a tradeoff on TimeGate, and we compare against SCSampler. Figure 44 shows this tradeoff using I3D as the video classifier. We notice that both TimeGate and SCSampler can dramatically reduce the cost of I3D. However, TimeGate outperforms SCSampler.

In table 24, we report the exact computational budget of TimeGate *v.s.* SCSampler and I3D. We notice that with, carefully selected 16 timesteps out of 128, TimeGate is able to match the performance of off-the-shelf CNNs which use 64 uniformly sampled timesteps. Also, we notice the computational cost of selecting these timesteps is marginal to the cost

	Timesteps		FLOPS (G)		FLOPS ↓	Acc. (%) ↑
	LightNet	HeavyNet	LightNet+Gating	HeavyNet		
R2D	—	64	—	246.6	246.6	72.9
S3D+SCSampler [96]	128	16	7.5	61.7	69.2	68.6
R2D+TimeGate	128	16	7.8	61.7	69.5	70.2
S3D	—	64	—	61.8	61.8	67.3
S3D+SCSampler [96]	128	16	7.5	17.3	24.8	64.1
S3D+TimeGate	128	16	7.8	17.3	25.1	66.2
I3D	—	64	—	830.7	830.7	85.7
I3D+SCSampler [96]	128	16	7.5	207.8	215.3	81.8
I3D+TimeGate	128	16	7.8	207.8	215.6	85.2

Table 24: Breakdown of the computational cost of TimeGate v.s. SCSampler. Three choices of HeavyNet: ResNet2D (R2D), ShuffleNet3D (S3D) and I3D. The computational cost of LightNet and the gating module is marginal compared to that of the HeavyNet. TimeGate reduces the cost by almost half. Our selector improves over SCSampler.

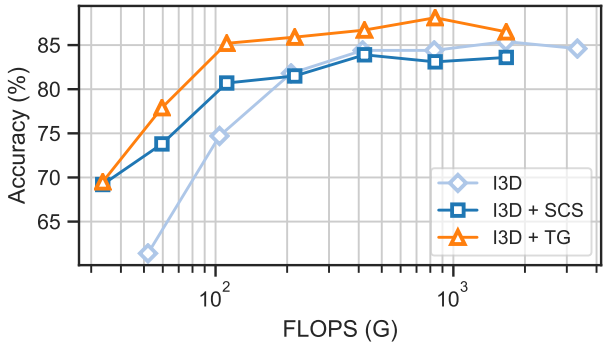


*Figure 44: TimeGate (**TG**) is better than SCSampler (**SCS**) in reducing computational cost of I3D.*

of the CNN classifier itself. For example, to select 8 out of 128 Timesteps, TimeGate spends 7.5 G-FLOPS, while to classify only one timestep using I3D, 3.9 G-FLOPS are needed.

Baseline	mAP (%) @ Timesteps						
	4	8	16	32	64	128	256
I3D	20.4	22.3	26.8	28.3	30.1	30.9	31.5
I3D + TimeGate	21.6	24.7	27.9	29.7	30.8	32.4	**33.1**

Table 25: TimeGate improves the performance of the backbone CNNs (i.e. I3D) on the challenging task of multi-label classification of Charades.

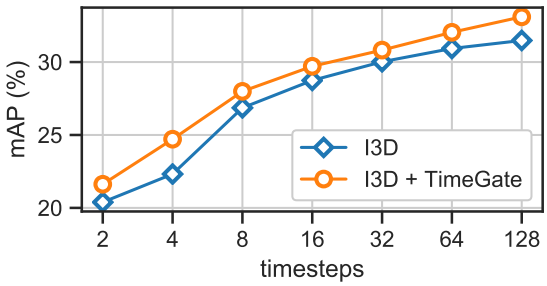


Figure 45: TimeGate improves the performance of the off-the-shelf I3D for recognizing the actions of Charades.

6.4.6 *Experiments on Charades*

In this experiment, we test how TimeGate would fair against off-the-shelf CNN for recognizing the multi-label action videos of Charades. This dataset is different from Breakfast in two ways. First, the videos are mid-range with average length of 0.5 minutes, compared to 2 minutes of Breakfast. Second, it is multi-label classification, but breakfast is single-label classification. So, it is more challenging to select unrelated timesteps from the videos of Charades than Breakfast. Most of the timesteps are already relevant to recognizing the mid-range videos of Charades. Still, TimeGate outperforms I3D at different time scales, see figure 45 and table 25. Worth mentioning that TimeGate consistently improves the efficiency of HeavyNet CNNs other than I3D. For example, if TimeGate uses 3D-ResNet-101 [175] as the HeavyNet, we achieve 36.2% using 256 timesteps compared to 35.5% achieved by [175] using dense sampling of 1024 timesteps. In other words, TimeGate retains the performance of 3D-ResNet-101 using only 25% of the computation. The reason is that, when TimeGate selects the most relevant segments from each video, it improves the signal-to-noise ratio. In analogy, [96] concluded that when the CNN video classifier considers the unrelated video segments, the accuracy degrades.

6.4.7 *Experiments on MultiThumos*

Our final experiment is to use TimeGate in classifying the long-range activities of MultiThumos. This dataset is particularly challenging, as each video is multi-labeled. Nevertheless, we observe that TimeGate is able to retain the performance of the HeavyNet (I3D) with much reduced computation, see table 26. In addition, it outperforms SC-Sampler in reducing the computational cost. Worth mentioning that TimeGate achieves 75.11% mAP using 256 timesteps compared to 74.79% mAP achieved by [73] using dense-sampling of 1024 timesteps. In other words, TimeGate retains the performance of [73] with almost 25% of the computational cost.

6.4.8 *Qualitative Results*

Examples of Gated Timesteps. In figure 46, we show a few visual examples of the timesteps selected, top, and discarded, bottom, by the gating module. We consider three

Baseline	mAP (%) @ Timesteps						
	4	8	16	32	64	128	256
I3D	41.85	45.02	52.75	58.41	64.74	67.19	69.32
I3D + SCSampler	43.51	47.68	54.14	60.87	67.23	69.83	72.46
I3D + TimeGate	45.38	50.02	57.63	63.34	69.07	73.20	**75.11**

Table 26: TimeGate improves the performance of I3D when classifying the long-range activities of MultiThmos. Also, it outperforms SCSampler.

Figure 46: Top, frames corresponding to the selected timesteps by TimGate. Bottom, are those discarded by TimeGate. The shown figures are for three activities: "making sandwich", "preparing coffee", and "making pancake". The general observation is that TimeGate tends to discard the segments with little discriminative visual evidences.

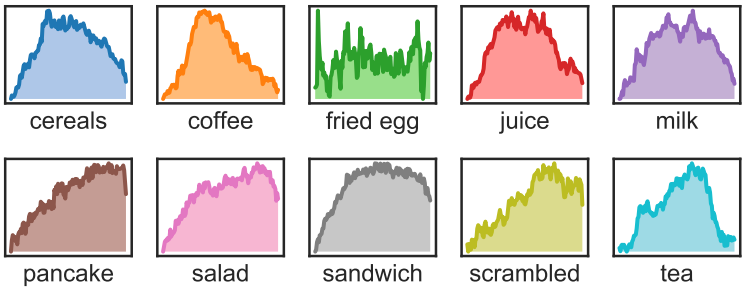


Figure 47: Distribution of the gating values across time for each activity of Breakfast. In simple activities, such as "making coffee", most of the selected segments happen in the middle of the video. This means the middle of the video is much more relevant than the other parts. While in complex activities, such as "making sandwich", the selected segments tend to distribute across the entire video. This means that almost the entire video contains relevant and important segments.

activities: "making sandwich", "preparing coffee", and "making pancake". The general observation is that TimeGate tends to discard the segments with little discriminative visual evidences.

Distribution of Gating Values. One might ask the question, how evenly distributed are the timesteps selected by TimeGate? To answer this question, we uniformly sample $T = 128$ timesteps from each test video. Then, we predict the gating value α_i for each timestep. After that, for all the videos of the same activity class, we average their gating values. Next, we normalize these values between zero and one, and visualize them in figure 47. Our observation is that, some activities are simple and usually happen in the

middle of the video, such as “preparing tea”, or “making coffee”. Others are complex and occupy the entire video, such as “fried egg” or “making sandwich”.

6.5 CONCLUSION

In this work, we proposed TimeGate, a neural model for efficient recognition of long-range activities in videos. Our approach for realizing the efficiency is sampling the most relevant segments from the activity video. We highlighted the differences between sampling for short-range actions *v.s.* long-range activities. We also stated the limitations of existing works, such as SCSampler. TimeGate overcomes these limitations using three contributions. First, a differentiable gating module for timestep selection. Second, the selection that is conditioned on both the timestep and its context. Third, TimeGate, an end-to-end neural model to retain the performance of existing CNN classifiers at a fraction of the computational budget. We experimented on three benchmarks and compared against related works. TimeGate consistently outperforms competing methods on all three benchmarks and reduces the computation of I3D by 50% while maintaining the classification accuracy. On MultiThumaos, TimeGate sets a new state-of-the-art mAP of 75.11% compared to 74.79% mAP of Timeception [73] while consuming only 25% of the computation cost. Our empirical evaluations and results demonstrate the efficiency of TimeGate in recognizing long-range activities.

7

CONCLUSIONS

7.1 SUMMARY

This thesis contributes to the literature of understanding and recognizing human activities in videos. More specifically, the thesis draw line between short-range atomic actions and long-range complex activities. For the classification of the latter, the mainstream approach in literature is to divide the activity into a handful of short segments, called atomic actions. Then, a neural model, such as 3D CNN, is trained to represent and classify each segment independently. Then, the activity-level classification probability scores are obtained by pooling over that of the segments. Differently, this work argues that long-range activities are better classified in full. That is to say, the neural model has to reason about the long-range activity, all at once, to better recognize it. Based on this argument, the thesis proposes different methods and neural network models for recognizing these complex activities.

In chapter 2, the task of zero-shot multimedia event detection (MED) is addressed. In this task, a repository of unlabeled videos is given. In addition, a set of text queries are given. Each query consists of a paragraph, or two, describing in details a certain human event. Examples of these events are “birthday party”, “wedding event” or “dog show”. The description includes possible subjects and objects expected to perform in the event. Also, it includes venues where the event can likely take place. The task is to retrieve, *i.e.* rank, the videos according to their semantic similarity to each text query. Furthermore, this task is zero-shot learning – neither the queries nor the videos are known in training time. This thesis proposes to learn a cross-modal manifold, *i.e.* feature space, using external sources of knowledge, on the hope that this space makes it possible to correctly retrieve the videos in test time. We notice that for EventNet, an existing dataset for event videos, the categories are originally crawled from WikiHow, a website with articles discussing many events in daily life. So, we draw the link between the videos of EventNet and the text articles of WikiHow. Then, we learn cross-modal neural embeddings between the visual and the textual modality, such that for a certain event, the article and its correlated videos fall closer on the manifold. Previous methods opt for an off-the-shelf distance metric, such as Euclidean or cosine, to learn such manifold. Differently, we find that learning the distance metric itself, along with the neural embeddings of the visual and textual modalities, in an end-to-end fashion, is the optimal solution. Experiments are conducted on two benchmarks MED-13 and MED-14. In addition, analysis and comparisons are made, where our method comfortably outperforms previous methods.

Chapter 3 addresses the difference between short-range atomic actions, and long-range human activities. In addition, three important properties of the latter are outlined. These properties are temporal composition, temporal order and temporal extent. Take for example the complex activity of "cooking meal". Composition means that it can be broken down into building blocks, called one-actions, *e.g.* "stir", "wash", "slice". None of which has an end goal by itself. But all together make the complex activity more meaningful. Order means that these one-actions exhibit temporal order, albeit week. Usually, one wash the hands before start cooking. And the temporal extent means that the temporal duration of the same one-action may vary from one video exemplar to another. One person might take a little bit longer to wash hands than another person. Existing methods fall short of addressing these three properties, combined. So, Timeception, a novel neural network layer for temporal modeling, is proposed. Timeception uses multi-scale kernels to tolerate the temporal extents of one-actions. Multi-scale is achieved by either using different kernel sizes, or different dilation rates. Additionally, Timeception decomposes the kernel of typical 3D convolution into a newly proposed temporal-only kernels. Since the temporal aspect of the human activity is, arguably, the most important among all other aspects, these temporal-only kernels are dedicated to model only the temporal dimension. As such, the effect of temporal-only convolutions is a drastic reduction in the computational cost of 3D convolutions. This enables Timeception to live up to the computational demands of seconds-long videos. Besides, Timeception is a modern and modular layer for temporal modeling. It can be stacked on top of 2D or 3D CNNs alike. By conducting several experiments, the benefits of Timeception are demonstrated, and the technical novelties are verified. Besides, Timeception outperforms state-of-art methods on three benchmarks Charades, Breakfast and MultiThumos.

While the proposed model Timeception of 3 can scale up to a minute-long human activities in videos, how about even longer activities. Chapter 4 is concerned with this problem – that is how to represent half-an-hour video and recognize it. Related works opt for statistical pooling, which neglects the temporal structure. Others opt for convolutional methods, as CNN and Non-Local. While successful in learning temporal concepts, they fall short of modeling minutes-long temporal dependencies. In this chapter, we propose VideoGraph, a method to achieve the best of two worlds: represent minutes-long human activities and learn their underlying temporal structure. To represent human activities, VideoGraph learns a soft version of an undirected graph. The graph nodes are deterministic and are learned entirely from video datasets, making VideoGraph applicable to video understanding tasks without node-level annotation. While the graph edges are probabilistic and are learned in a soft-assignment manner. The result is improvements over related works on benchmarks: Breakfast, Epic-Kitchens and Charades. Besides, we demonstrate that VideoGraph is able to learn the temporal structure of human activities in minutes-long videos.

In chapter 5, the focus is shifted to a fundamental property of long-range activities in videos. This property is the perturbations over time in short-range visual evidences, *i.e.* one-actions. In other words, a main conclusion of chapters 3 and 4 is that the long-range activity exhibits a weak temporal order of its building blocks, the one-actions. For example, a person might start the long-range activity of "make coffee" by "add milk" then "add coffee" and end up with "pour sugar". Anther person might start first with "add milk", then "add coffee" and skip "pour sugar" all together. These perturbations

over time requires some tolerance, or invariance, form the methods that learn to represent and classify long-range activities. As such, a new temporal modeling layer is devised, namely Permutation Invariant Convolution, PIC. Not only this layer, PIC, introduces the invariance to temporal perturbations, but also addresses the shortcomings of some of the existing temporal modeling methods. From the recent literature, three dominant approaches for recognizing the long-range activities are outlined. These approaches are vector aggregation, self-attention, and convolution. Their upsides and downsides are summarized. The proposed layer, PIC, maintains the upsides and addresses some of the downsides. Strictly speaking, it is found that both vector aggregation and self-attention disregard local connectivity – a fundamental design principle in recognizing temporal patterns. Variously, PIC respects local connectivity, thus is able to capture complex temporal patters at multiple layers of abstractions. In addition, self-attention depend on memory-like kernels to detect latent visual concepts. However, the kernels are conditioned on the input signal, *i.e.* the activity videos. Thus, these kernels are prune to failure in the case of noisy input signals. In contrast, the kernels in PIC layer are shared and not inferred. That is to say the kernels are freely learned from the training set, rather than being inferred from the input signal. As such, PIC is better suited for detecting the most discriminant visual evidences from the noisy videos of long-range activities. In summary, PIC incorporates three design principles *i.* invariance to permutation, *ii.* using shared concept kernels, and *iii.* respecting local connectivity. These principles are the reason why PIC is better in modeling the complexities of long-range activities than the competing methods. After the experiments are carried out, it is concluded that PIC outperforms other methods in recognizing the activities of three benchmark Breakfast, MultiThumos, and Charades. Further more, by conducting quantitative and qualitative analysis, the design principles of PIC are thoroughly investigated, and their importance is confirmed.

Chapter 6 sheds light on the task of understanding human activities in videos from a perspective different from the previous chapters. Rather than recognizing such activities, this chapter discusses the efficiency of already existing neural models in activity recognition. Generally, the model efficiency is materialized by four metrics, namely *i.* number of learning parameters in millions, *ii.* number of floating point operations (FLOPs), *iii.* feedforward time in milli-seconds, and *iv.* classification accuracy in percentage. This chapter, however, focuses on the trade-off between only two metrics, which are the FLOPs and the accuracy. In addition, TimeGate, a new method for the efficient recognition of long-range activities in videos, is proposed. Using TimeGate, realizing the efficiency is achieved by sampling the most representative segments from the activity's video. Then, only the sampled segments are considered for recognition, while all the other segments are discarded. Consequently, the recognition accuracy is retained at a fraction of the computational cost. TimeGate has two technical contributions. First, thanks to a carefully crafted gating mechanism, TimeGate is fully differntiable. Thus, it can be trained with existing video classifiers, such as 3D CNNs, in an end-to-end fashion. This is in contrast to SCSampler – a non-differentiable method for sampling salient segments. Second, the gating mechanism in TimeGate is context-conditional, which is considerably better for long-range activities. This is in contrast to the frame-conditional gating used by other methods, such as SCSampler. Context-conditioning means that when sampling a certain segment from a video, the visual evidences in both the segment

and its context, *i.e.* the video, are considered. While segment-conditioning means that the visual evidences of only the segment is used. In the end, in-depth analysis is conducted, and the benefits of TimeGate are verified. Besides, using qualitative and quantitative are experiments on three datasets, TimeGate outperforms related methods in reducing the computational cost of recognizing the long-range activities.

7.2 DISCUSSION

Over the past years, a large body of literature is dedicated to addressing the problem of video understanding. One perspective is to cast this problem as the downstream task of single-label or multi-label classification of videos, whether the videos are short-range atomic actions, or long-range complex activities. The outcome of this perspective is many methods and neural network models that achieve a near-perfect classification accuracy, and in some cases outperform human level. However, this thesis asks the following questions. Do the video feature representations learned from the classification tasks generalize well to other video-related tasks, such as object segmentation, object tracking, or video generation? More importantly, if the answer is no, what novel tasks might be better for understanding videos? And will these tasks inevitably require new datasets, benchmarks, and evaluation methods?

So far in video understanding, the majority of literature aim for representing a video using single feature vector. Each video, be it unit actions, or complex activities, entails a wealth of information, spanning both the spatial and temporal dimensions. This raises the question, can only one feature vector truly express all information given in a video? What about the temporal structure the video entails? Also, what about the spatial relationships between subject and object in the video? Maybe it is feasible to represent a short video clip using a single feature vector. But when a video spans a few minutes or even an hour, is only one vector still feasible? Wouldn't this vector be an understatement? All these questions argues for the need of structure-based representations to truly understand videos.

In this thesis, chapter 3 casts light on the difference between short segments of videos, called one-actions, and long-range videos, called complex activities. Nonetheless, the problem is much more bigger than that. In videos, human motions can be explained at different levels of granularity. In literature, there is lack of understanding these levels. The quote said by Ernest Hemingway "Never Mistake Motion for Action" is a good example for this lack of understanding. So, the thesis asks a few questions. What is the difference between unit-action, one-action, micro-action, action, activity and event? What are the properties of each? Consequently, to understand each of these levels, are different methods required?

The quote "seeing is believing", from the ancient Greek times, continues to hold true. For example, it holds true when a person watches a movie, or an episode of TV series, and tries to understand its content. But in computer vision, there are other sources, from which, the computer machine can still understand videos. These sources are called the video modalities. Audio, visual and textual modalities are good examples. So far, the main bulk of literature addresses only the visual modality. Great efforts is exerted, and the outcome of such efforts is many successful methods that successfully solve quite a

few tasks in video understanding, such as detection, recognition, segmentation, tracking, ...etc. Neverthless, the thesis asks this question. Can the computer machine understand videos from textual or audio modalities? Can literature devise methods to leverage such modalities for a better understand of videos? Is there an optimal way to fuse all these modalities to reach an even more better understanding?

Learning in computer vision, much like machine learning, is classically divided into supervised, unsupervised and semi-supervised. In video understanding, a huge body of literature is dedicated to proposing and discussing methods of supervised learning, or even weakly supervised learning. However, these methods come at the great cost of well-annotated, large-scale datasets. So far, a huge effort is made such that these datasets remain updated, expanded and available to the many tasks of video understanding. Though, with the rapid growth of data, problems, and tasks, the supervised learning might be running out of time, and coming to an abrupt ending. So, the thesis asks the following questions. Is supervised learning sustainable? If the answer is no, does the unsupervised learning provide a solution? Or the question would rather be, is the unsupervised learning realistic, to begin with? Or is there a compromise between the upsides and downsides of supervised *v.s.* unsupervised learning? Finally, is self-supervised learning the way to consider?

SAMENVATTING

ASPECTEN VAN TIJD IN DE HERKENNING VAN MENSELIJKE ACTIVITEITEN

Dit proefschrift draagt bij aan het begrijpen en herkennen van menselijke activiteit in video's.

In hoofdstuk 2 wordt zero-shot multimedia event detectie (MED) besproken. Elke zoekvraag bestaat uit een of twee alinea's die in detail een menselijke gebeurtenis beschrijven. Voorbeelden van zulke gebeurtenissen zijn een verjaardagsfeest, een huwelijksfeest of een honden show. Dit proefschrift stelt het leren van een afstandsmetriek voor met de inbedding van visuele en tekstuele modaliteiten. Onze methode is beter dan de eerdere methoden.

Hoofdstuk 3 behandelt het verschil tussen kortdurende, atomische acties en langdurende menselijke activiteiten in temporele compositie, temporele volgorde en temporele omvang. Neem bijvoorbeeld de complexe activiteit "een maaltijd koken" opgesplitst in roeren, wassen en snijden. Geen van deze heeft een doel op zich, maar alles bij elkaar maakt het de complexe activiteit betekenisvoller. Bestaande methoden schieten te kort bij het aanpakken van de drie eigenschappen. Dus wordt Timeception voorgesteld, een nieuw neuraal netwerk. Omdat het temporele aspect van menselijke activiteit het meest belangrijke is van alle aspecten, geven temporele convoluties een drastische verlaging van de rekenkosten en presteert de methode beter dan state-of-art methoden op drie benchmarks.

Hoofdstuk 4 bediscussiëren we hoe een video van een half uur kan worden herkend. Gerelateerd werk stelt statistisch samenvoegen voor. Dat laat de temporele structuur weg. Anderen stellen convolutionele methoden voor, zoals convolutionele en niet-locale neurale netwerken. Terwijl deze methoden succesvol zijn in het leren van temporele concepten, schieten ze te kort in het modelleren van minuten-lange, temporele afhankelijkheden. In dit hoofdstuk introduceren we VideoGraph, een methode die ontwikkeld is om het beste van beide werelden te bewerkstelligen. Om menselijke activiteiten weer te geven leert VideoGraph een zachte versie van een ongerichte graaf. De knopen van de graaf zijn deterministisch en worden volledig geleerd uit video datasets. Dit maakt VideoGraph toepasbaar op video taken zonder annotaties per knoop. Het resultaat is een verbetering vergeleken met gerelateerd werk op drie benchmarks.

In hoofdstuk5 wordt de focus verlegd naar een fundamentele eigenschap van langdurende activiteiten in video's. Een persoon kan bijvoorbeeld een langdurende activiteit "koffie zetten" beginnen door "melk toevoegen", vervolgens "koffie toevoegen" en te eindigen met "suiker gieten". Een andere persoon kan beginnen met "melk toevoegen", vervolgens "koffie toevoegen" om daarna "suiker gieten" helemaal over te slaan. Deze verstoringen over tijd vereisen enige tolerantie of invariantie van methoden die deze langdurende activiteiten representeren en classificeren. PIC is beter geschikt voor het detecteren van de meest onderscheidende visuele kenmerken voor de rommelige video's

van langdurende activiteiten: i. invariant onder permutatie, ii. gebruikmaken van conceptkernen en iii. het respecteren van lokale connectiviteit. PIC doet het beter dan andere methoden in het herkennen van activiteiten van drie benchmarks.

Hoofdstuk 6 belicht de taak om menselijke activiteiten in video's te begrijpen vanuit een ander perspectief dan de voorgaande hoofdstukken. TimeGate, een nieuwe methode voor het efficiënte herkennen van langdurende activiteiten in video's, realiseert de meest representatieve segmenten van de video. Als gevolg wordt de herkenningsnauwkeurigheid behouden met een fractie van de rekenkosten. In kwalitatieve en kwantitatieve experimenten op drie datasets uit te voeren, presteert TimeGate beter dan bestaande methoden in het reduceren van rekenkosten voor het herkennen van langdurende activiteiten.

ACKNOWLEDGMENTS

I would like to give thanks to all the persons that have become part of this thesis.

First and foremost, I would like to express my sincere gratitude to my supervisors Prof. Dr. Arnold Smeulders and Dr. Efstratios Gavves for the continuous support of my Ph.D. research. I thank them for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me greatly in all the time of research and writing this thesis. I cannot express enough how thankful I am for their continued guidance and immense support throughout the years of my Ph.D. studies. The outcome of this Ph.D. would not have been possible without their mentorship and supervision.

Second, I would like to thank the rest of the Ph.D. thesis committee: Prof. Dr. Mubarak Shah, Prof. Dr. Max Welling, Dr. Pascal Mettes, and Dr. Amirhossein Habibian for their encouragement, insightful comments, and valuable assessment.

Third, my sincere thanks go to Prof. Dr. Cees Snoek. I am grateful for his guidance and mentorship. Thanks to his advice, I was able to make better choices in life.

Fourth, I would like to thank all the professors and staff members I worked with and assisted in teaching: Prof. Dr. Arnoud Visser, Dr. Stevan Rudinac, Dr. Anthony van Inge, and Prof. Dr. Theo Gevers.

Fifth, my thanks go to all my collaborators. I very much enjoyed working with you and I learned a lot from you. Thank you Dr. Dennis Koelma, Dr. Babak Ehteshami bejnordi, Dr. Mihir Jain, Dr. Hamid Reza Vaezi Joze, and Dr. Davide Abati.

Sixth, I feel grateful to who ever gave me a helping hand at the time of need. Thank you Dr. Thomas Mensink and Dr. Sebastian Schelter. In addition, I would like to thank my fellow labmates for their help with the translation: Sarah Ibrahimi and Riaan Zoetmulder.

Seventh, my thanks go to the management team: Mrs. Virginie Mes, Mrs. Félice Arends, Mrs. Nicole Vastenhout, and Mr. Erik Hitipeuw. Only because of your help and support I was able to finish this Ph.D. in time.

Eighth, I would like to thank my students: Joop Pascha, Juan Buhagiar, Tony N. Guyen, and Joris Baan. Not only I enjoyed supervising your thesis projects, but also I learned from you. Also, thanks to all of my students in M.Sc. and B.Sc. cohorts of Computer Science and Artificial Intelligence.

Last but not the least, I thank all my colleagues for the wonderful years I spent working with you. Special thanks go to my fellow labmates at the Quva Lab: Mert Kilickaya,

Changyong Oh, Adeel Pervez, Dr. Peter O'Connor, Hendrik Heuer, Shaui Liao, Berkay Kicanaoglu, Kirill Gavrilyuk, Tom Runia, Matthias Reisser, Maurice Weiler, Dr. Ran Tao, Dr. Zhengyang Li, Dr. Amir Ghodrati, Dr. Deepak Gupta, and Dr. Andrew Brown. Besides, I would like to thank all my fellow researchers at the ISIS Lab, in no particular order: William Thong, Dr. Masoud Mazloom, Dr. Nanne van Noord, Fida Thoker, Gjorgji Strezoski, Riaan Zoetmulder, Sarah Ibrahimi, David Zhang, Sadaf Gulshad, Shuo Chen, Yunlu Chen, Devanshu Arya, Mehmet Altinkaya, Ivan Sosnovik, Artem Moskalev, Jiaojiao Zhao, Dr. Spencer Cappallo, Dr. Jörn Jacobsen, and Dr. Kandan Ramakrishnan.

BIBLIOGRAPHY

[1] Wikihow, 2016. http://wikihow.com.

[2] Wikipedia, 2016. http://wikipedia.com.

[3] Charades algorithms. github.com/gsig/charades-algorithms, 2017.

[4] Implementation of timeception. github.com/noureldien/timeception, 2017.

[5] M. Abadi et al. Tensorflow. tensorflow.org, 2015.

[6] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. In *arXiv*, 2016.

[7] A. Agharwal, R. Kovvuri, R. Nevatia, and C. G. Snoek. Tag-based video retrieval by embedding semantic content in a continuous word space. In *IEEE WACV*, 2016.

[8] R. Arandjelovic and A. Zisserman. All about vlad. In *IEEE CVPR*, 2013.

[9] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[10] B. E. Bejnordi, T. Blankevoort, and M. Welling. Batch-shaping for learning conditional channel gated networks. In *ICLR*, 2020.

[11] S. Bhardwaj, M. Srinivasan, and M. M. Khapra. Efficient video classification using fewer frames. In *CVPR*, 2019.

[12] H. Bilen, B. Fernando, E. Gavves, and A. Vedaldi. Action recognition with dynamic image networks. In *TPAMI*, 2017.

[13] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. In *JMLR*, 2003.

[15] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[16] X. Chang, Y. Yang, A. G. Hauptmann, E. P. Xing, and Y.-L. Yu. Semantic concept discovery for large-scale zero-shot event detection. In *IJCAI*, 2015.

[17] X. Chang, Y. Yang, G. Long, C. Zhang, and A. G. Hauptmann. Dynamic concept composition for zero-example event detection. In *arXiv*, 2016.

[18] X. Chang, Y.-L. Yu, Y. Yang, and E. P. Xing. They are not equally reliable: Semantic event search using differentiated concept classifiers. In *IEEE CVPR*, 2016.

[19] Y. Chen, M. Rohrbach, Z. Yan, S. Yan, J. Feng, and Y. Kalantidis. Graph-based global reasoning network. In *arXiv*, 2018.

[20] Z. Chen, Y. Li, S. Bengio, and S. Si. You look twice: Gaternet for dynamic filter selection in cnns. In *CVPR*, 2019.

[21] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[22] F. Chollet et al. Keras. keras.io, 2015.

[23] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE CVPR*, 2005.

[24] I. Cosmin Duta, B. Ionescu, K. Aizawa, and N. Sebe. Spatio-temporal vector of locally max pooled features for action recognition in videos. In *PCVPR*, 2017.

[25] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[26] D. Damen, H. Doughty, G. Maria Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018.

[27] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. In *JACS*, 1990.

[28] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

[29] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[30] Y. Du, C. Yuan, B. Li, L. Zhao, Y. Li, and W. Hu. Interaction-aware spatio-temporal pyramid attention networks for action classification. In *ECCV*, 2018.

[31] I. C. Duta, B. Ionescu, K. Aizawa, and N. Sebe. Spatio-temporal vlad encoding for human action recognition in videos. In *ICMM*, 2017.

[32] M. Elhoseiny, J. Liu, H. Cheng, H. Sawhney, and A. Elgammal. Zero-shot event detection by multimodal distributional semantic embedding of videos. In *AAAI*, 2016.

[33] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *ICCV*, 2019.

[34] C. Feichtenhofer, A. Pinz, R. P. Wildes, and A. Zisserman. What have we learned from deep representations for action recognition? In *CVPR*, 2018.

[35] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

[36] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. In *IEEE TPAMI*, 2016.

[37] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. In *TPAMI*, 2017.

[38] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[39] D. Fleet and Y. Weiss. Optical flow estimation. In *Handbook of mathematical models in computer vision*, 2006.

[40] K. Fukushima. Cognitron: A self-organizing multilayered neural network. In *Biological cybernetics*, 1975.

[41] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. In *Biological cybernetics*, 1980.

[42] A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011.

[43] C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. In *IEEE CVPR*, 2016.

[44] E. Gavves, T. E. J. Mensink, T. Tommasi, C. G. M. Snoek, and T. Tuytelaars. Active transfer learning with zero-shot priors: Reusing past datasets for future tasks. In *IEEE ICCV*, 2015.

[45] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *ICML*, 2017.

[46] A. Ghodrati, E. Gavves, and C. G. Snoek. Video time: Properties, encoders and evaluation. In *BMVC*, 2018.

[47] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. Video action transformer network. In *arXiv*, 2018.

[48] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. Video action transformer network. In *CVPR*, 2019.

[49] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. In *NIPS*, 2017.

[50] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017.

[51] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

[52] C. Gu, C. Sun, S. Vijayanarasimhan, C. Pantofaru, D. A. Ross, G. Toderici, Y. Li, S. Ricco, R. Sukthankar, C. Schmid, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *arXiv*, 2017.

[53] A. Habibian, T. Mensink, and C. G. Snoek. Composite concept discovery for zero-shot video event detection. In *ICMR*, 2014.

[54] A. Habibian, T. Mensink, and C. G. Snoek. Videostory: A new multimedia embedding for few-example recognition and translation of events. In *ACM MM*, 2014.

[55] A. Habibian, T. Mensink, and C. G. Snoek. Discovering semantic vocabularies for cross-media retrieval. In *ICMR*, 2015.

[56] A. Habibian, T. Mensink, and C. G. Snoek. Videostory embeddings recognize events when examples are scarce. In *IEEE TPAMI*, 2016.

[57] A. Habibian, T. Mensink, and C. G. Snoek. Video2vec embeddings recognize events when examples are scarce. In *TPAMI*, 2017.

[58] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In *SciPy*, 2008.

[59] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015.

[60] K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018.

[61] B. Hassibi, D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. In *ICNN*, 1993.

[62] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[64] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[65] B. K. Horn and B. G. Schunck. Determining optical flow. In *TAIU*, 1981.

[66] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al. Searching for mobilenetv3. In *CVPR*, 2019.

[67] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *arXiv*, 2017.

[68] D.-A. Huang, S. Buch, L. Dery, A. Garg, L. Fei-Fei, and J. Carlos Niebles. Finding it: Weakly-supervised reference-aware visual grounding in instructional videos. In *CVPR*, 2018.

[69] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *arXiv*, 2018.

[70] D.-A. Huang, V. Ramanathan, D. Mahajan, L. Torresani, M. Paluri, L. Fei-Fei, and J. C. Niebles. What makes a video a video: Analyzing temporal information in video understanding models and datasets. In *CVPR*, 2018.

[71] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

[72] N. Hussein, E. Gavves, and A. W. Smeulders. Unified embedding and metric learning for zero-exemplar event detection. In *CVPR*, 2017.

[73] N. Hussein, E. Gavves, and A. W. Smeulders. Timeception for complex action recognition. In *CVPR*, 2019.

[74] N. Hussein, E. Gavves, and A. W. Smeulders. Videograph: Recognizing minutes-long human activities in videos. In *ICCV Workhop*, 2019.

[75] N. Hussein, E. Gavves, and A. W. Smeulders. Permutation invariant convolution for recognizing long-range activities. In *arXiv*, 2020.

[76] N. Hussein, M. Jain, and B. E. Bejnordi. Timegate: Conditional gating of segments in long-range activities. In *arXiv*, 2020.

[77] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The thumos challenge on action recognition for videos "in the wild". In *CVIU*, 2017.

[78] M. Jain, H. Jegou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.

[79] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

[80] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In *TPAMI*, 2012.

[81] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In *TPAMI*, 2013.

[82] L. Jiang, D. Meng, T. Mitamura, and A. G. Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *ACM MM*, 2014.

[83] L. Jiang, T. Mitamura, S.-I. Yu, and A. G. Hauptmann. Zero-example event search using multimodal pseudo relevance feedback. In *ICMR*, 2014.

[84] L. Jiang, S.-I. Yu, D. Meng, T. Mitamura, and A. G. Hauptmann. Bridging the ultimate semantic gap: A semantic search engine for internet videos. In *ICMR*, 2015.

[85] L. Jiang, S.-I. Yu, D. Meng, Y. Yang, T. Mitamura, and A. G. Hauptmann. Fast and accurate content-based semantic search in 100m internet videos. In *ACM MM*, 2015.

[86] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang. Fcvid: Fudan-columbia video dataset. In *arXiv*, 2015.

[87] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. In *IEEE TPAMI*, 2017.

[88] Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A. C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *ICMR*, 2011.

[89] L. Jing, B. Liu, J. Choi, A. Janin, J. Bernd, M. W. Mahoney, and G. Friedland. A discriminative and compact audio representation for event detection. In *ACM MM*, 2016.

[90] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[91] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. In *arXiv*, 2017.

[92] G. Kim and E. P. Xing. Reconstructing storyline graphs for image recommendation from web community photos. In *CVPR*, 2014.

[93] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[94] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *NIPS*, 2015.

[95] O. Köpüklü, N. Kose, A. Gunduz, and G. Rigoll. Resource efficient 3d convolutional neural networks. In *arXiv*, 2019.

[96] B. Korbar, D. Tran, and L. Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *ICCV*, 2019.

[97] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural. In *NeurIPS*, 2014.

[98] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[99] H. Kuehne, A. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, 2014.

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

[101] A. Kumar and B. Raj. Audio event detection using weakly labeled data. In *arXiv*, 2016.

[102] G. Lample, A. Sablayrolles, M. Ranzato, L. Denoyer, and H. Jégou. Large memory layers with product keys. In *NeurIPS*, 2019.

[103] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, 2014.

[104] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 2017.

[105] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *IEEE*, 1998.

[106] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *arXiv*, 2016.

[107] X. Li, Y. Zhang, J. Zhang, S. Chen, I. Marsic, R. A. Farneth, and R. S. Burd. Concurrent activity recognition with multimodal cnn-lstm structure. In *arXiv*, 2017.

[108] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. In *CVIU*, 2018.

[109] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. In *arXiv*, 2017.

[110] C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. In *arXiv*, 2017.

[111] Y.-J. Lu. Zero-example multimedia event detection and recounting with unsupervised evidence localization. In *ACM MM*, 2016.

[112] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. In *JMLR*, 2008.

[113] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, 2009.

[114] M. Mazloom, E. Gavves, and C. G. M. Snoek. Conceptlets: Selective semantics for classifying video events. In *IEEE TMM*, 2014.

[115] M. Mazloom, X. Li, and C. Snoek. Tagbook: A semantic video representation without supervision for event detection. In *IEEE ToM*, 2015.

[116] T. Mensink, E. Gavves, and C. Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *IEEE CVPR*, 2014.

[117] P. Mettes, J. C. van Gemert, S. Cappallo, T. Mensink, and C. G. Snoek. Bag-of-fragments: Selecting and encoding video fragments for event detection and recounting. In *ICMR*, 2015.

[118] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. In *arXiv*, 2017.

[119] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. In *arXiv*, 2013.

[120] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[121] M. Monfort, B. Zhou, S. A. Bargal, A. Andonian, T. Yan, K. Ramakrishnan, L. Brown, Q. Fan, D. Gutfruend, C. Vondrick, et al. Moments in time dataset: one million videos for event understanding. In *arXiv*, 2018.

[122] L. Muda, M. Begam, and I. Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. In *arXiv*, 2010.

[123] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016.

[124] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013.

[125] P. Over, G. Awad, J. Fiscus, G. Sanders, and B. Shaw. Trecvid 2013–an introduction to the goals, tasks, data, evaluation mechanisms, and metrics. In *TRECVID Workshop*, 2013.

[126] P. Over, J. Fiscus, G. Sanders, D. Joy, M. Michel, G. Awad, A. Smeaton, W. Kraaij, and G. Quénot. Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID Workshop*, 2014.

[127] S. K. Pal and S. Mitra. Multilayer perceptron, fuzzy sets, classifiaction. 1992.

[128] J.-Y. Pan and C. Faloutsos. Videograph: a new tool for video mining and classification. In *ACM/IEEE-CS DL*, 2001.

[129] N. Parmar, P. Ramachandran, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

[130] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[131] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. In *CVIU*, 2016.

[132] S. Pini, M. Cornia, F. Bolelli, L. Baraldi, and R. Cucchiara. M-vad names: a dataset for video captioning with naming. In *MTA*, 2019.

[133] A. Rohrbach, M. Rohrbach, N. Tandon, and B. Schiele. A dataset for movie description. In *CVPR*, 2015.

[134] M. Rohrbach, A. Rohrbach, M. Regneri, S. Amin, M. Andriluka, M. Pinkal, and B. Schiele. Recognizing fine-grained and composite activities using hand-centric features and script data. In *IJCV*, 2016.

[135] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. In *IJCV*, 2013.

[136] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[137] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

[138] K. Schindler and L. Van Gool. Action snippets: How many frames does human action recognition require? In *CVPR*, 2008.

[139] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *IEEE ICPR*, 2004.

[140] F. Sener and A. Yao. Zero-shot anticipation for instructional activities. In *ICCV*, 2019.

[141] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. In *ICLR Worshop*, 2015.

[142] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, 2017.

[143] G. A. Sigurdsson, O. Russakovsky, and A. Gupta. What actions are needed for understanding human actions in videos? In *ICCV*, 2017.

[144] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016.

[145] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher vector faces in the wild. In *BMVC*, 2013.

[146] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[147] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv*, 2014.

[148] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR*, 2012.

[149] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *ACM PUC*, 2013.

[150] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[151] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

[152] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE CVPR*, 2015.

[153] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[154] E. Szelag, M. Kanabus, I. Kolodziejczyk, J. Kowalska, and J. Szuchnik. Individual differences in temporal information processing in humans. In *ANE*, 2004.

[155] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. In *Communications of the ACM*, 2016.

[156] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *JMLR*, 2001.

[157] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3d: generic features for video analysis. In *arXiv*, 2014.

[158] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[159] D. Tran, H. Wang, L. Torresani, and M. Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019.

[160] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

[161] J. Uijlings, I. Duta, E. Sangineto, and N. Sebe. Video classification with densely extracted hog/hof/mbh features: an evaluation of the accuracy/computational efficiency trade-off. In *IJMIR*, 2015.

[162] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, 2016.

[163] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. In *TPAMI*, 2017.

[164] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[165] A. Veit and S. Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018.

[166] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

[167] A. W. Vieira, E. R. Nascimento, G. L. Oliveira, Z. Liu, and M. F. Campos. Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences. In *ICPR*, 2012.

[168] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, 2016.

[169] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, 2011.

[170] H. Wang and C. Schmid. Action recognition with improved trajectories. In *CVPR*, 2013.

[171] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. In *arXiv*, 2015.

[172] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[173] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks for action recognition in videos. In *TPAMI*, 2018.

[174] S. Wang, W. Zhao, Z. Kou, and C. Xu. How to make a blt sandwich? learning to reason towards understanding web instructional videos. In *arXiv*, 2018.

[175] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018.

[176] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018.

[177] X. Wang and A. Gupta. Videos as space-time region graphs. In *ECCV*, 2018.

[178] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krähenbühl, and R. Girshick. Long-term feature banks for detailed video understanding. In *arXiv*, 2018.

[179] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krahenbuhl, and R. Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, 2019.

[180] S. Wu, S. Bondugula, F. Luisier, X. Zhuang, and P. Natarajan. Zero-shot event detection using multi-modal fusion of weakly supervised concepts. In *CVPR*, 2014.

[181] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[182] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning for video understanding. In *ECCV*, 2018.

[183] B. Xiong, G. Kim, and L. Sigal. Storyline representation of egocentric videos with an applications to story-based search. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[184] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017.

[185] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[186] Z. Yang, Y. Li, J. Yang, and J. Luo. Action recognition with spatio-temporal visual attention on skeleton image sequences. In *IEEE ToCS*, 2018.

[187] G. Ye, Y. Li, H. Xu, D. Liu, and S.-F. Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *ACM MM*, 2015.

[188] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. In *IJCV*, 2017.

[189] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. In *IJCV*, 2018.

[190] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.

[191] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[192] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[193] B. Zhou, A. Andonian, and A. Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018.

[194] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. In *arXiv*, 2016.

[195] L. Zhou and J. J. Corso. Youcookii dataset. In *arXiv*, 2017.

[196] L. Zhou, C. Xu, and J. J. Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI*, 2018.

[197] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *arXiv*, 2016.

[198] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.