Timeception for Complex Action Recognition

Noureldien Hussein, Efstratios Gavves, Arnold W.M. Smeulders
QUVA Lab, University of Amsterdam
{nhussein,egavves,a.w.m.smeulders}@uva.nl

Abstract

This paper focuses on the temporal aspect for recognizing human activities in videos; an important visual cue that has long been either disregarded or ill-used. We revisit the conventional definition of an activity and restrict it to "Complex Action": a set of one-actions with a weak temporal pattern that serves a specific purpose. Related works use spatiotemporal 3D convolutions with fixed kernel size, too rigid to capture the varieties in temporal extents of complex actions, and too short for long-range temporal modeling. In contrast, we use multi-scale temporal convolutions, and we reduce the complexity of 3D convolutions. The outcome is Timeception convolution layers, which reasons about minute-long temporal patterns, a factor of 8 longer than best related works. As a result, Timeception achieves impressive accuracy in recognizing human activities of Charades. Further, we conduct analysis to demonstrate that Timeception learns long-range temporal dependencies and tolerate temporal extents of complex actions.

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 1. *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 [1, 2, 3]. Few attention has been paid to the recognition of long-range and complex actions, as in Charades and EventNet [4, 5], 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 [6, 7] falls short of learning temporal order. Neural temporal modeling [8, 9] and spatio-temporal convolutions of various types [10, 11, 12] successfully learns temporal order of 8 [13] or 128 timesteps [14]. 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 sub-actions. Related methods [11, 15] learn spatio-temporal convolutions with a fixed-size kernel, 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 of Charades [4] and achieving state-of-the-art-results.

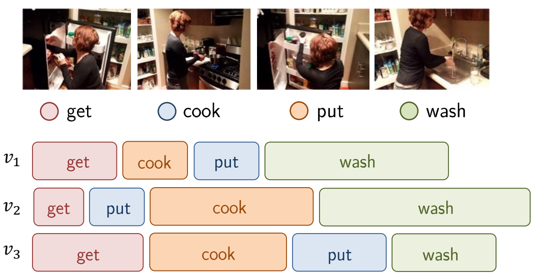

Figure 1: Properties of complex action "Cooking a Meal": **composition**: consists of several one-actions (Cook, ...), **order**: weak temporal order of one-actions (Get $\lll$ Wash), **extent**: one-actions vary in their temporal extents.

The novelties of of this paper 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 large variations in durations of action components. *iii.* We use temporal-only convolutions, which are better suited for complex actions than spatio-temporal counterparts.

2. Related Work

Temporal Modeling. The stark difference between video and image classification in the temporal dimension, which necessitates temporal modeling. A widely used approach is statistical pooling: max and average pooling [18, 19], attention pooling [6], rank pooling [20], dynamic images [21] and context gating [7], to name a few. Beyond statistical pooling, vector aggregation is also used. [22] uses Fisher Vector [23] to aggregate spatio-temporal features over time, while [9, 24, 25] extend VLAD [26] 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 [8]. While TA-DenseNet[27] extents DenseNet [28] to exploit the temporal dimension. To our knownedge, no substantial improvements have been reported recently.

Short-range Action Recognition. Few works [29] learn deep appearance features by frame-level classification of actions, using 2D CNNs. Others complement deep appearance features with shallow motion features, as IDT [30]. Also, auxiliary image representations is fused with RGB signals: [31] uses OpticalFlow channels, while [32] use Dynamic Images. 3D CNNs are natural evolution of their 2D counterparts. C3D [11, 10] proposes 3D CNNs to capture spatio-temporal patterns of 8 frames in a sequence. In the same vein, I3D [13] inflates the kernels of ImageNet-pretrained 2D CNN to jump-start the training of 3D CNNs. While effective in short-range video sequences (few seconds), 3D convolutions are too computationally expensive to address minute-long videos, which is the focus of this work.

Long-range Action Recognition. To learn long-range temporal patterns, [33] uses CRF on top of CNN feature maps to model human activities. To learn video-wide representations, TRN [34] learns relations between several video segments. TSN [35, 36] learns temporal structure in long videos. While LTC [37] considers different temporal resolutions as a substitute to bigger temporal windows. Inspired by self-attention [38], non-local networks [14] proposes deep convolution network with the capacity 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 simultaneously.

Convolution Decomposition. CNNs succeed in learning spatial [39, 29] and spatiotemporal [**?**, 11, 10, 40, 37, 41] 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 [42]. To control the computational complexity, several works propose the decomposition of 2D and 3D convolutions. Xception [17] argues that separable 2D convolutions are as effective as typical 2D convolutions. Similarly, S3D [15, 12] considers separable 2+1D convolutions to reduce the complexity of typical 3D convolutions. ResNet [42] reduces the channel dimension using 1×1 2D convolution before applying the costly 3×3 2D spatial convolution. ShuffleNet [16] models cross-channel correlation by channel shuffling instead of 1×1 2D convolution. ResNeXt [43] proposes grouped convolutions, while Inception [44, 45] advocates for decomposing large 2D spatial kernels with multi-scale spatial kernel of different sizes.

In this work we propose the decomposition of spatiotemporal 3D 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. Method

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 ($\mathcal{C}$). One spatiotemporal kernel $w \in \mathbb{R}^{T \times L \times L \times C}$ learns a latent concept by simultaneously convolving these three subspaces [11, 13], where T is the number of timesteps, C is the number of channels, and L is the size of spatial window. Although, there is no fundamental reason why these subspaces must be convolved simultaneously. Instead, as showcased in [15], 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 [15], it is not the only possibility. Therefore, Any other decomposition is permissible, namely:

$$\tilde{w} = w_\alpha \times w_\beta \times w_\gamma \times ..., \tag{1}$$

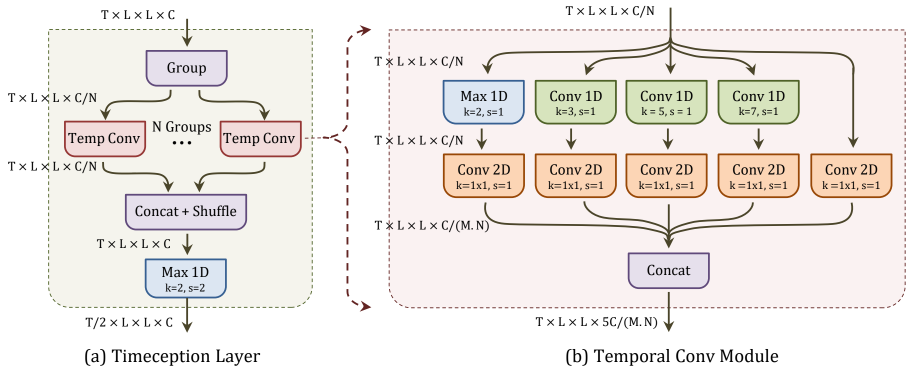

Figure 2: The core component of our method is Timeception layer (left). Simply, it takes as an input the features $\mathbf{X}$; corresponding to T timesteps from the previous layer in the network. Then, it splits them into N groups, and temporally convolves each group using temporal convolution module (right). It is 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 [16], to learn cross-channel correlations, efficiently than 1×1 spatial convolutions [17].

as long as some basic principles are maintained for the final cascade $\tilde{w}$. Generalizing on recent decomposed architectures [12, 17], 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 should retain 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 ($\mathcal{C}$) is expanded.

***iii*. Subspace Efficiency.** When designing the decomposition for a specific task at hand, we should make sure that the bulk of the available parameter budget is dedicated to subspaces that are directly relevant to the task. 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 2. First, we discuss the Timeception layer. Then we describe how to stack Timeception layers on top of existing 2D or 3D CNNs.

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 $1,000$ 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 [42], or short-range spatiotemporal; processing nearby bursts of frames, like I3D [13].

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 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 [13] with parameter complexity $\mathcal{O}(T \cdot L^2 \cdot C)$ or decomposed spatial and temporal kernels [15, 12] with parameter complexity $\mathcal{O}((L^2 + T) \cdot C)$. 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 [13, 15, 12], 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 $\mathcal{O}(T)$, it is computationally feasible to train a deep temporal model that encode relationships 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 a *channel grouping* operation before the temporal-only convolutions and a *channel shuffling* operation after the temporal-only convolutions. The purpose of the channel grouping is to encode cross-channel correlations. Clearly, as each group contains a random subset of channels, not all possible correlations are accounted for. This is mitigated by the 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 [16] is more cost-effective operation to learn cross-channel correlations than 1×1 2D convolutions [17].

Tolerating Variant Temporal Extents. The second objective for the Timeception layer is to tolerate the variances 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 kernels of fixed temporal scale with multi-scale temporal kernels. There are two possible ways to implement multi-scale kernels, see Figure 3. The first way, inspired by Inception [44] for images, is to adopt K kernels, each of a different size k. The second way, inspired by [46], is to employ dilated convolutions.

The temporal convolution module, see Figure 2b, 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 to 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 with a 1×1 spatial convolution . After the channel reduction, all five outputs are concatenated across channel dimension, resulting in an output $\mathbf{Y}_n \in \mathbb{R}^{T \times L \times L \times (5C/MN)}$.

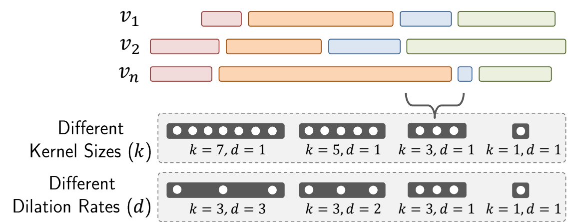

Figure 3: To tolerate temporal extents, we use multi-scale temporal kernels, with two options: *i.* different kernel sizes $k \in \{1, 3, 5, 7\}$ and fixed dilation rate $d = 1$, *ii.* different dilation rates $d \in \{1, 2, 3\}$ and fixed kernel size $k = 3$.

Summary of Timeception. A Timeception layer, see Figure 2a, expects an input feature $\mathbf{X} \in \mathbb{R}^{T \times L \times L \times C}$ from the previous layer in the network. The feature $\mathbf{X}$ across the channel dimension are then split into N channel groups $\mathbf{X}_n \in \mathbb{R}^{T \times L \times \times L \times [C/N]}$. Each channel group 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 $5/M$ after each Timeception layer, the temporal subspace shrinks by a factor 2.

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 baseline CNN models: a spatial 2D CNN and a short-range spatiotemporal 3D CNN.

2D CNN. The first baseline uses ResNet-152 [42] 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 $\mathbf{X} \in \mathbb{R}^{128 \times 7 \times 7 \times 2048}$. 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 $\mathbf{Y} \in \mathbb{R}^{8 \times 7 \times 7 \times 5000}$. To further reduce the spatial dimension, we follow the convention of CNNs by using spatial average pooling (per channel), which results in the feature $\mathbf{Y}' \in \mathbb{R}^{8 \times 5000}$. And to finally reduce the temporal dimension, we use depthwise-separable temporal convolution with kernel size $k \in \mathbb{R}^{8 \times 1 \times 1 \times 1}$ with no zero-padding. The result is a feature $\mathbf{Z} \in \mathbb{R}^{5000}$ which is fed to a two-layer MLP (with BatchNorm and ReLU) for classification.

3D CNN. The second baseline uses I3D [13] 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 as a backbone architecture 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 pre-train 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 0.00001 as learning rate, momentum and weight decay, respectively. The model is implemented using TensorFlow [47] and Keras [48]. Code will be made public upon acceptance.

4. Experiments

4.1. Datasets

The scope of this paper is complex actions with their three properties: composition, temporal extent and temporal order –see Figure 1. Thus, we choose to conduct our experiments on Charades [4] and MultiTHUMOS [49]. Other infamous datasets for action recognition do not meet the properties 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 [33, 9, 33, 14, 50].

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 [51] 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.

4.2. Tolerating Temporal Extents

In this experiment, we evaluate the capacity of the multi-scale temporal-only convolutions to tolerate the different temporal extents found in complex actions. The experiment is carried out on Charades dataset.

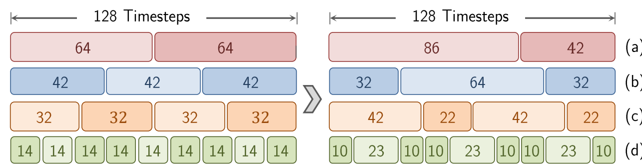

Figure 4: We split a video of 128 timesteps into segments of equal length (left, before alteration), and alter their temporal extents by expansion and shrinking (right, after alteration). We use 4 types of alterations:(a) very-coarse, (b) coarse, (c) fine, and (d) very-fine. Numbers in boxes are timesteps.

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 4.

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 1: 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/ResNet are backbone CNNs.

The results of this controlled experiment are shown in table 1. We observe that Timeception is more effective than fixed-size kernels in handling unexpected variations 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 typical configuration for temporal kernels used in all 3D CNNs [11, 13, 14, 15]. *ii*. Different kernel sizes ($k \in \{1, 3, 5, 7\}$) and fixed dilation rate ($d = 1$). *iii*. Fixed kernel size ($k = 3$) and fixed dilation rate ($d \in \{1, 2, 3\}$).

The result of this experiment are shown in Table 2. 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).

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

Table 2: 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.

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 models equipped with Timeception layers. These baselines use different number of input timesteps. We experiment on Charades with both ResNet-152 and I3D as backbone architectures for Timeception layers.

ResNet-152 is used, with an increasing number of timesteps as an input: $T \in \{32, 64, 128\}$, followed by Timeception layers. ResNet is a 2D model that 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, increasing number of timesteps as an input: $T \in \{256, 512, 1024\}$, followed by Timeception layers. I3D is a 3D model that 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.

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 3: 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 report results in Table 3 and we make two observations. First, stacking Timeception layers leads to an improved accuracy both when using ResNet and I3D as backbone architectures. As the only thing changing 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 feature maps returned from ResNet is twice as much as from I3D (2048 *v.s.* 1024). We conclude that Timeception layers allow for deep temporal and efficient models, able to learn long-range temporal abstractions needed to learn complex actions.

Learned Weights of Deep Timeception Model Figure 5 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, which uses 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, Timeception layer 1, we notice that long kernels ($k = 7$) captures fine-grained temporal dependencies, because of the rapid transition of kernel weights. But at for Timeception 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 Timeception layer 3, we observe that long-range and short-range temporal patterns are learned by short ker-

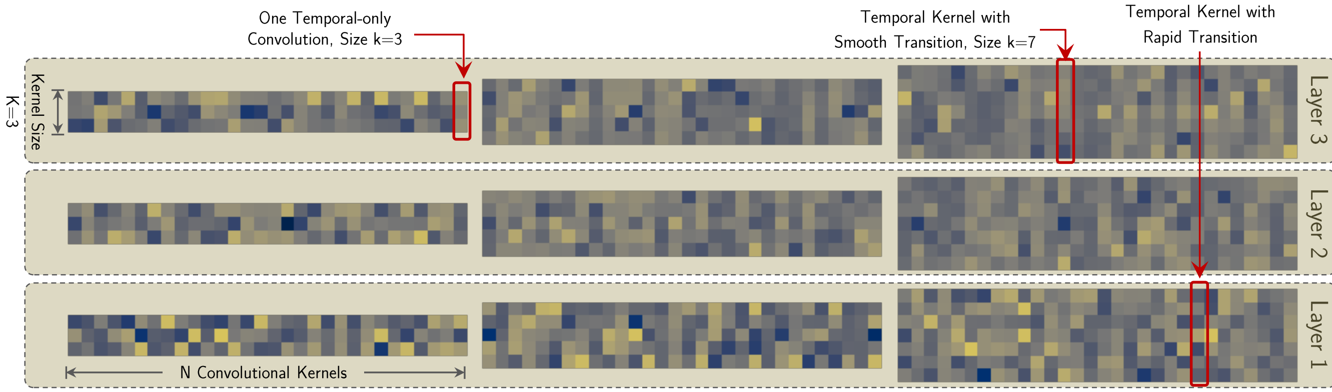

Figure 5: The learned weights by temporal convolutions of three Timeception layers. Each uses multi-scale convolutions with varying kernel sizes $k \in \{3, 5, 7\}$. In bottom layer (1), we notice that long kernels ($k = 7$) captures fine-grained temporal dependencies. But at the top layer (3), the long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

nels ($k = 3$) 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.

4.4. Effectiveness of Timeception

To quantify the effectiveness of Timeception, we compare it against related temporal convolution layers. To make a fair comparison, each of these layers processes both $\mathcal{T}, \mathcal{C}$ subspaces. These layers are: *i*. separable temporal convolution, as used in S3D [12], that models both $\mathcal{T}, \mathcal{C}$ simultaneously. *ii*. grouped separable temporal convolution to model $\mathcal{T}$. It is grouped to reduce computation, followed by $1{\times}1$ 2D convolution to model $\mathcal{C}$. *iii*. grouped separable temporal convolution to model $\mathcal{T}$, followed by channel shuffling to model $\mathcal{C}$.

Figure 6 shows the result of this comparison. Interestingly, Timeception is very efficient is maintaining a reasonable increase in number of parameters as the network goes deeper. While the expansion of other layers (*e.g.* separable, grouped separable) is out of hand. In theory, we can go as deep as 8 Timception layers. Which means our model has the capacity to temporally model 4096 timesteps, or 2.3 minutes. But there yet to exist a benchmark for complex actions where videos take many minutes to unfold. Currently, the biggest dataset is Charades, with 0.5 minutes per video, on average.

4.5. Experiments on 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 4. Timeception, monotonically improves the performance of the backbone CNN. That is, if Timeception is used on top of ResNet or I3D, the absolute gain is 8.8% and 4.3%, respectively.

Beyond the overall mAP, how beneficial is Timeception,

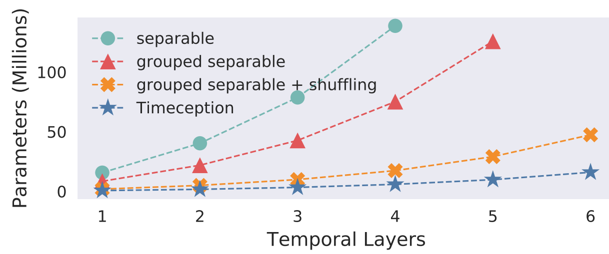

Figure 6: Timeception is an efficient temporal layer, as it maintains a manageable number of parameters as the network increases. It is compared against related temporal layers, each convolves temporal and channel subspaces ($\mathcal{T}, \mathcal{C}$).

Ours	Method	Modality	mAP (%)
	[33] Two-stream	RGB + Flow	18.6
	[33] Two-stream + LSTM	RGB + Flow	17.8
	[9] ActionVLAD	RGB + iDT	21.0
	[33] Temporal Fields	RGB + Flow	22.4
	[34] Temporal Relations	RGB	25.2
	[52] ResNet-152	RGB	22.8
✓	ResNet-152 + TC	RGB	31.6
	[13] I3D	RGB	32.9
✓	I3D + TC	RGB	37.2
	[14] 3D ResNet-101	RGB	35.5
	[14] 3D ResNet-101 + NL	RGB	37.5
	[50] 3D ResNet-50 + GCN	RGB + RP	37.5
	[50] 3D ResNet-101 + GCN	RGB + RP	39.7
✓	3D ResNet-101 + TC	RGB	41.1

Table 4: 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.

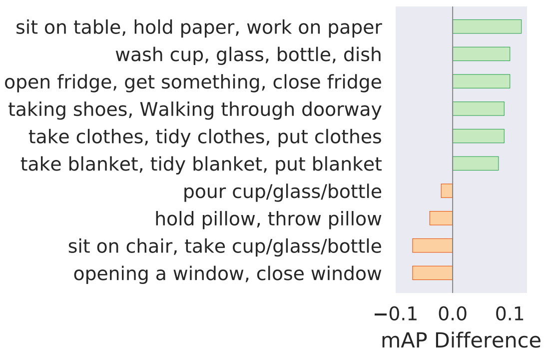

Figure 7: Multi-scale Timeception outperforms the fixed-kernel when complex actions are dynamic (in green). While in orange, complex actions with rigid temporal patters where fixed-size performs better than multi-scale.

and in what cases exactly does it help? To answer this question, we make two comparisons assess the relative performance of Timeception if we changed two factors: *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 7, 8, and we make two important observations.

First, when comparing the relative performance of multi-scale vs. fixed-size Timeception, as in Figure 7, we observe that multi-scale Timeception excels in complex actions which have dynamic video examples. As an example, "take clothes + tidy clothes + put clothes", one actor may take longer than others to tidy clothes. On the other hand, 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, we notice that the later 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".

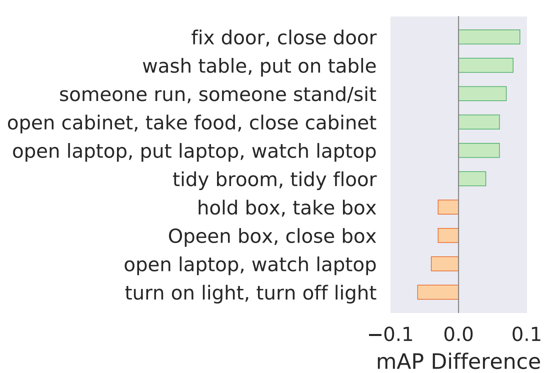

Figure 8: Long-range Timeception outperforms the short-range version when complex actions need the entire video to unfold (in green). However, in orange, we see one-actions that short-range Timeception can easily capture.

4.6. Experiments on MultiTHUMOS

We use MultiTHUMOS as a second dataset to experiment our model against. This helps in investigating the generality of our method 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. We follow the same procedures as in the literature to split the dataset and to train our model. Specifically, our model is first trained on the weakly-labeled training split, *i.e.* each video is annotated with only one label. Then, we fine-tune the model on the validation split, where each video is densely annotated with multi-labels of one-actions. Finally, we test on the test split.

To assess the performance of our model, we compare against two baselines: I3D and I3D equipped with Timeception with fixed-kernel size. As shown in the results in Table 5, the multi-scale version of Timeception achieves the best performance.

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	3,5,7	1	74.79

Table 5: Timeception helps baseline models to capture the long-range dependencies between one-actions in videos of MultiTHUMOS dataset. Timeception with multi-scale temporal kernels is better than fixed-size temporal kernels.

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 paper 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 in popular benchmarks, *i.e.* Charades and MultiTHUMOS, the proposed Timeception networks improve the state-of-the-art notably.

References

[1] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011. 1

[2] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. In *arXiv*, 2012. 1

[3] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. In *arXiv*, 2017. 1

[4] Gunnar A Sigurdsson, Gúl Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 1, 5

[5] Guangnan Ye, Yitong Li, Hongliang Xu, Dong Liu, and Shih-Fu Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *ACM MM*, 2015. 1

[6] Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. In *NIPS*, 2017. 1, 2

[7] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. In *arXiv*, 2017. 1, 2

[8] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 1, 2

[9] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017. 1, 2, 5, 7

[10] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. In *TPAMI*, 2013. 1, 2

[11] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2, 6

[12] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 1, 2, 3, 7

[13] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 3, 5, 6, 7

[14] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 1, 2, 5, 6, 7

[15] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. In *arXiv*, 2017. 1, 2, 3, 6

[16] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *arXiv*, 2017. 2, 3, 4

[17] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *arXiv*, 2016. 2, 3, 4

[18] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Unified embedding and metric learning for zero-exemplar event detection. In *CVPR*, 2017. 2

[19] Amirhossein Habibian, Thomas Mensink, and Cees GM Snoek. Video2vec embeddings recognize events when examples are scarce. In *IEEE TPAMI*, 2017. 2

[20] Basura Fernando, Efstratios Gavves, José Oramas, Amir Ghodrati, and Tinne Tuytelaars. Rank pooling for action recognition. In *TPAMI*, 2017. 2

[21] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *CVPR*, 2016. 2

[22] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013. 2

[23] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. In *IJCV*, 2013. 2

[24] Ionut Cosmin Duta, Bogdan Ionescu, Kiyoharu Aizawa, and Nicu Sebe. Spatio-temporal vector of locally max pooled features for action recognition in videos. In *PCVPR*, 2017. 2

[25] Ionut C Duta, Bogdan Ionescu, Kiyoharu Aizawa, and Nicu Sebe. Spatio-temporal vlad encoding for human action recognition in videos. In *ICMM*, 2017. 2

[26] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *CVPR*, 2013. 2

[27] Amir Ghodrati, Efstratios Gavves, and Cees GM Snoek. Video time: Properties, encoders and evaluation. In *BMVC*, 2018. 2

[28] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 2

[29] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[30] Christoph Feichtenhofer, Axel Pinz, and AP Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 2

[31] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2

[32] Hakan Bilen, Basura Fernando, Efstratios Gavves, and Andrea Vedaldi. Action recognition with dynamic image networks. In *TPAMI*, 2017. 2

[33] Gunnar A Sigurdsson, Santosh Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, 2017. 2, 5, 7

[34] Bolei Zhou, Alex Andonian, and Antonio Torralba. Temporal relational reasoning in videos. In *arXiv*, 2017. 2, 7

[35] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 2

[36] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. In *TPAMI*, 2018. 2

[37] Gul Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. In *TPAMI*, 2017. 2

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 2

[39] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. In *arXiv*, 2015. 2

[40] Christoph Feichtenhofer, Axel Pinz, Richard P Wildes, and Andrew Zisserman. What have we learned from deep representations for action recognition? In *CVPR*, 2018. 2

[41] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *arXiv*, 2017. 2

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 4

[43] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2

[44] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *CVPR*, 2015. 2, 4

[45] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 2

[46] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, 2016. 4

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

[48] François Chollet et al. Keras. keras.io, 2015. 5

[49] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. In *IJCV*, 2018. 5

[50] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, 2018. 5, 7

[51] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". In *CVIU*, 2017. 5

[52] Charades algorithms. github.com/gsig/charades-algorithms, 2017. 7