Deep Temporal Linear Encoding Networks

Ali Diba[1,⋆], Vivek Sharma[2,⋆,∓], Luc Van Gool[1,3]
[1]ESAT-PSI, KU Leuven, [2]CV:HCI, KIT, [3]CVL, ETH Zürich
{ali.diba,luc.vangool}@esat.kuleuven.be, vivek.sharma@kit.edu

Abstract

The CNN-encoding of features from entire videos for the representation of human actions has rarely been addressed. Instead, CNN work has focused on approaches to fuse spatial and temporal networks, but these were typically limited to processing shorter sequences. We present a new video representation, called temporal linear encoding (TLE) and embedded inside of CNNs as a new layer, which captures the appearance and motion throughout entire videos. It encodes this aggregated information into a robust video feature representation, via end-to-end learning. Advantages of TLEs are: (a) they encode the entire video into a compact feature representation, learning the semantics and a discriminative feature space; (b) they are applicable to all kinds of networks like 2D and 3D CNNs for video classification; and (c) they model feature interactions in a more expressive way and without loss of information. We conduct experiments on two challenging human action datasets: HMDB51 and UCF101. The experiments show that TLE outperforms current state-of-the-art methods on both datasets.

1. Introduction

Human action recognition [6, 15, 25, 35] in videos has attracted quite some attention, due to the potential applications in video surveillance, behavior analysis, video retrieval, and more. Even if considerable progress was made, the performance of computer vision systems still falls behind that of people. On top of the challenges that make object class recognition hard, there are issues like camera motion and the continously changing viewpoints that come with it. Whereas Convolutional Networks (ConvNets) have caused several sub-fields of vision to leap forward, they still lack the capacity to exploit long-range temporal information, probably the main reason why end-to-end networks are still unable to outperform methods using hand-crafted features [35].

⋆Ali Diba and Vivek Sharma contributed equally to this work and listed in alphabetical order.
∓This work was carried out while he was at ESAT-PSI, KU Leuven.

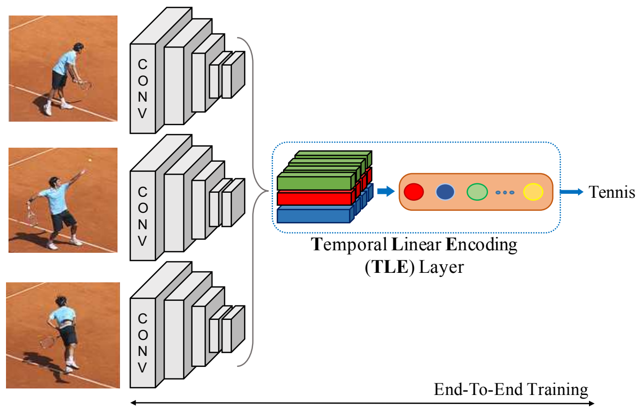

Figure 1: Temporal linear encoding for video classification. Given several segments of an entire video, be it either a number of frames or a number of clips, the model builds a compact video representation from the spatial and temporal cues they contain, through end-to-end learning. The ConvNets applied to different segments share the same weights.

Neural networks for action recognition can be categorized into two types, namely *one-stream* ConvNets [15, 33] (which use only one stream at a time: either spatial or temporal information), and *two-stream* ConvNets [25] (which integrate both spatial and temporal information at the same time).

As to the one-stream ConvNets, spatial networks perform action recognition from individual video frames. They lack any form of motion modeling. On the other hand, temporal networks typically get their motion information from dense optical flow. This reliance on dense temporal sampling leads to excessive computational costs for longer videos. One way to avoid processing the abundance of input frames is by extracting a fixed number of shorter clips, evenly distributed over the video [25, 33].

The two-stream ConvNets have shown to outperform one-stream ConvNets. They exploit fusion techniques like trajectory-constrained pooling [37], 3D pooling [8], and consensus pooling [38]. The fusion methods of spatial and motion information lie at the heart of the state-of-the-art two-stream ConvNets.

Motivated by the above observations, we propose the new spatio-temporal encoding illustrated in Figure 1. The design of the spatio-temporal deep feature encoding aims to aggregate multiple video segments (i.e. frames or clips) over longer time ranges. To that end, we use our 'temporal linear encoding' (TLE), which is inspired by previous works on video representations [35] and feature encoding methods [20, 31]. TLE is a form of temporal aggregation of features sparsely sampled over the whole video using feature map aggregation techniques, and then projected to a lower dimensional feature space using encoding methods powered by end-to-end learning of deep networks. Specifically, TLE captures the important concepts from the long-range temporal structure in different frames or clips, and aggregates it into a compact and robust feature representation by linear encoding. The compact temporal feature representation fits action recognition well, as it is a global feature representation over the whole video. The goal of the paper is not only to achieve high performance, but also to show that TLEs are computationally efficient, robust, and compact. TLE is evaluated on two challenging action recognition datasets, namely HMDB51 [18] and UCF101 [28]. We experimentally show that the two-stream ConvNets when combined with TLEs achieve state-of-the-art performance on HMDB51 (71.1%) and UCF101 (95.6%).

The rest of the paper is organized as follows. In Section 2, we discuss related work. Section 3 describes our proposed approach. Experimental results and their analysis are presented in Section 4 and Section 5. Finally, conclusions are drawn in Section 6.

2. Related Work

Action Recognition without ConvNets: Over the last two decades, several action recognition techniques in videos have been proposed by the vision community. Quite a few are concerned with effective representations using local spatio-temporal features, suc h as HOG3D [16], SIFT3D [24], HOF [19], ESURF [39], and MBH [4]. Recently, IDT [35] was proposed, which is currently the state-of-the-art among hand-crafted features. Despite this good performance, these features have several shortcomings: they are computationally expensive; they fail to capture semantic concepts; they lack discriminative capacity as well as scalability. To overcome such issues, several techniques have been proposed to model the temporal structure for action recognition, such as the actom sequence model [10] which considers sequence of histograms; temporal action decomposition [21] which exploits the temporal structure of human actions by temporally decomposing video frames; dynamic poselets [36] which uses a relational model for action detection; and the temporal evolution of appearance representations [9] which uses a ranking function capable of modeling the evolution of both appearance and motion over time.

ConvNets for Action Recognition: Recently several attempts have been made to go beyond individual image-level appearance information and exploit the temporal information using ConvNet architectures. End-to-end ConvNets have been introduced in [8, 25, 33, 38] for action recognition. Karpathy et al. [15] trained a deep network operating on individual frames using a very large sports activities dataset (Sports-1M). Yet, the deep model turned out to be less accurate than an IDTs-based representation because it could not capture the motion information. To overcome this problem, Simonyan et al. [25] proposed a two-stream network, cohorts of spatial and temporal ConvNets. The input to the spatial and temporal networks are RGB frames and stacks of multiple-frame dense optical flow fields, respectively. The network was still limited in its capacity to capture temporal information, because it operated on a fixed number of regularly spaced, single frames from the entire video. Tran et al. [33] explored 3D ConvNets on video streams for spatio-temporal feature learning for clips of 16 frames, and filter kernel of size $3 \times 3 \times 3$. In this way, they avoid to calculate the optical flow explicitly and still achieve good performance. Sun et al. [30] proposed a factorized spatio-temporal ConvNet and decomposed the 3D convolutions into 2D spatial and 1D temporal convolutions. Similar to [25] and [33] is Feichtenhofer et al.'s [8] work, where they employ 3D Conv fusion and 3D pooling to fuse spatial and temporal networks using RGB images and a stack of 10 optical flow frames as input. Wang et al. [38] use multiple clips sparsely sampled from the whole video as input for both streams, and then combine the scores for all clips in a late fusion approach.

Encoding Methods: As to prior encoding methods, there is a vast literature on BoW [3, 27], Fisher vector encoding [22] and sparse encoding [40]. Such methods have performed very well in various vision tasks. FV encoding [31] and VLAD [1, 12] have lately been integrated as a layer in ConvNet architectures, and CNN encoded features have produced superior results for several challenging tasks. Likewise, Bilinear models [20, 32] have been widely used and have achieved state-of-the-art results. Bilinear models are computationally expensive as they return matrix outer products, hence can lead to prohibitively high dimensions. To tackle this problem, compact bilinear pooling [11] was proposed which uses the Tensor Sketch Algorithm [23], to project features from a high dimensional space to a lower dimensional one, while retaining state-of-the-art performances. Compact bilinear pooling has shown to perform better than FV encoding and fully-connected networks [11]. Moreover, this type of feature representa-

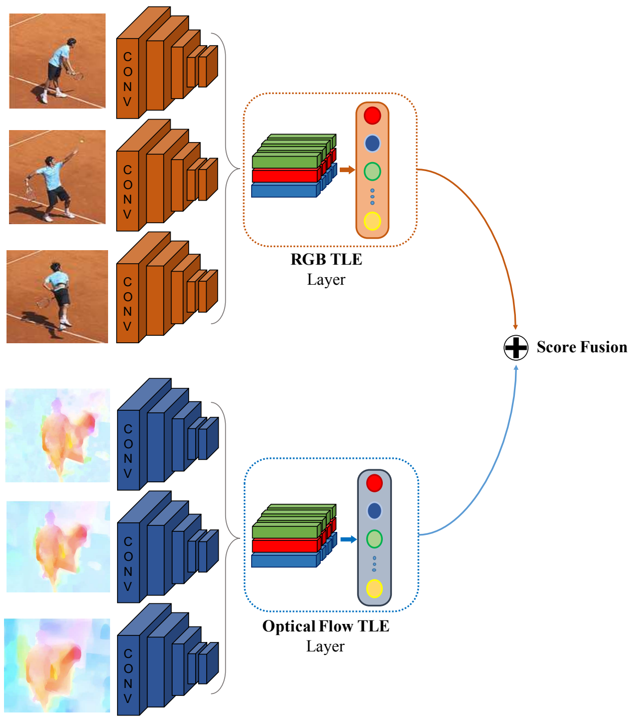

Figure 2: Our temporal linear encoding applied to the design of two-stream ConvNets [25]: spatial and temporal networks. The spatial network operates on RGB frames, and the temporal network operates on optical flow fields. The features maps from the spatial and temporal ConvNets for multiple such segments are aggregated and encoded. Finally, the scores for the two ConvNets are combined in a late fusion approach as averaging. The ConvNet weights for the spatial stream are shared and similarly for the temporal stream.

tion is compact, non-redundant, avoids over-fitting, and reduces the number of parameters of CNNs significantly, as it replaces fully-connected layers.

Our proposed temporal linear encoding captures more expressive interactions between the segments across entire videos, and encodes these interactions into a compact representation for video-level prediction. To the best of our knowledge, this is the first end-to-end deep network that encodes temporal features from entire videos.

3. Approach

In a video, the motion between consecutive frames tends to be small. Motivated by this, IDTs [35] showed that the densely sampling feature points in video frames and using optical flow to track them yields a good video representation. This suggests that we need a video representation that encodes all the frames together, in order to also capture long-range dynamics. To tackle this issue, recently some techniques have combined several consecutive [25] or sparsely sampled [38] frames into short clips. Unlike IDTs, these techniques use ConvNets with late fusion to combine spatial and temporal cues, but they still fail to efficiently encode all frames together.

Given earlier successes with deep learning, creating effective video representations should seem possible via the end-to-end learning of deep neural networks. The hope would be that such representations embody more of the semantic information extracted along the whole video. Our goal is to create a single feature space in which to represent each video using all its selected frames or clips, rather than scoring separate frames/clips with classifiers and label the video based on scores aggregation. We propose temporal linear encoding (TLE) to aggregate spatial and temporal information from an entire video, and to encode it into a robust and compact representation, using end-to-end learning, as shown in Fig. 2 and Fig. 3. Algorithm 1 sketches the steps of the proposed TLE. More details about the CNN encoding layer is given in Section 3.1.

3.1. Deep Temporal linear encoding

Consider the output feature maps of CNNs truncated at a convolutional layer for K segments extracted from a video V. The feature maps are matrices $\{S_1, S_2, ..., S_K\}$ of size $S \in \mathbb{R}^{h \times w \times c}$, where h, w and c denote the height, width, and number of channels of the CNN feature maps. A temporal aggregation function $T : S_1, S_2, \dots, S_K \rightarrow X$, aggregates K temporal feature maps to output an encoded feature map X. The aggregation function can be applied to the output of different convolutional layers. This temporal aggregation allows us to linearly encode and aggregate information from the entire video into a compact and robust feature representation. This retains the temporal relationship between all the segments without the loss of important information. We investigated different functions T for the temporal aggregation of the segments.

Algorithm 1 Deep Temporal Linear Encoding Layer

```
Input: CNN features for $K$ frames/clips $\{S_1, S_2, ..., S_K\}$ of video $V$, $S \in \mathbb{R}^{h \times w \times c}$, where $h$, $w$ and $c$ are height, width, and channels of feature maps respectively.
Output: Temporal linear encoded feature map $y \in \mathbb{R}^d$, where $d$ is the encoded feature dimension.
Temporal Linear Encoding:
1. $X = S_1 \diamond S_2 \diamond \dots \diamond S_K$, $X \in \mathbb{R}^{h \times w \times c}$, where $\diamond$ is an aggregation operator
2. $y = EncodingMethod(X)$, $y \in \mathbb{R}^d$, where $d$ denotes the encoded feature dimensions
```

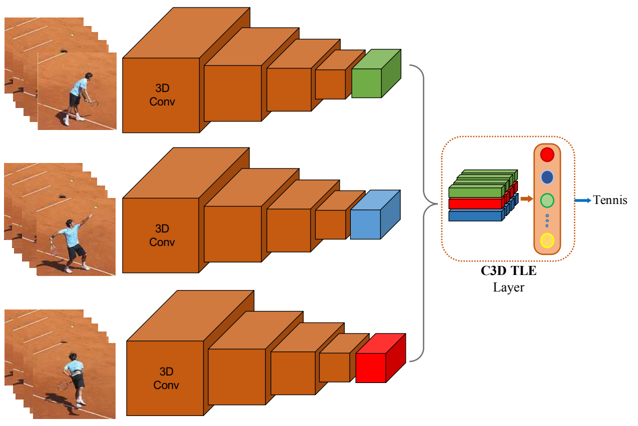

Figure 3: Our temporal linear encoding applied to 3D ConvNets [33]. These use video clips as input. The feature maps from the clips are aggregated and encoded. The output of the network is a video-level prediction. The ConvNets operating on the different clips all share the same weights.

- Element-wise Average of Segments:

$$X = (S_1 \oplus S_2 \oplus \ldots \oplus S_K)/K \tag{1}$$

- Element-wise Maximum of Segments:

$$X = max\{S_1, S_2, \ldots, S_K\} \tag{2}$$

- Element-wise Multiplication of Segments:

$$X = S_1 \circ S_2 \circ \ldots \circ S_K \tag{3}$$

Of all the temporal aggregation functions illustrated above, element-wise multiplication of feature maps yielded best results, and was therefore selected.

The temporal aggregated matrix X is fed as input to an encoding (or pooling) method $E : X \rightarrow y$, resulting in a linearly encoded feature vector $y, y \in \mathbb{R}^d$, where d denotes the encoded feature dimensions. The advantage of encoding is that every channel of the aggregated temporal segments interacts with every other channel, thus leading to a powerful feature representation of the entire video. In this work, we investigate two encoding methods E:

- **Bilinear Models**: A bilinear model [20, 32] computes the outer product of two feature maps, given by:

$$y = W[X \otimes X^{'}] \tag{4}$$

  Where $X \in \mathbb{R}^{(hw)\times c}$, $X^{'} \in \mathbb{R}^{(hw)\times c^{'}}$ are input feature maps, $y \in \mathbb{R}^{(cc^{'})}$ are bilinear features, $\otimes$ denotes the outer product, [ ] turns the matrix into a vector by concatenating the columns, and W represents model parameters to be learned (here linear). In our case, $X = X^{'}$. The resulting bilinear features capture the interaction of features with each other at all spatial locations, hence leading to a high-dimensional representation. For this reason, we use the Tensor Sketch algorithm [11, 23], which projects this high-dimensional space to a lower-dimensional space, without computing the outer product directly. That cuts down on the number of model parameters significantly. The model parameters W are learned with end-to-end back propagation.

- **Fully connected pooling**: As the network has fully-connected layers between the last convolutional layer and the classification layer, the model parameters of the fully-connected layer and classification layer are learned when training the network from scratch or when fine-tuning a pre-trained network.

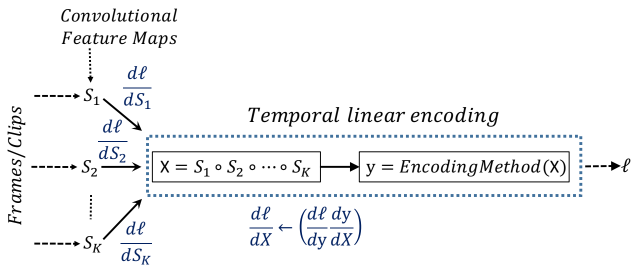
Convolutional Feature Maps
Frames/Clips
S_1
S_2
S_K
$\frac{d\ell}{dS_1}$
$\frac{d\ell}{dS_2}$
$\frac{d\ell}{dS_K}$
Temporal linear encoding
$X = S_1 \circ S_2 \circ \cdots \circ S_K$
$y = EncodingMethod(X)$
ℓ
$\frac{d\ell}{dX} \leftarrow \left(\frac{d\ell}{dy}\frac{dy}{dX}\right)$

Figure 4: Computing gradients for back-propagation in the temporal linear encoding.

Compared to the fully-connected pooling method, bilinear models projects the high dimensional feature space to a lower dimensional space, which is far fewer in parameters and still perform better than fully-connected layers in performance, apart from computational efficiency.

One can readily employ other encoding methods like deep fisher encoding [31] or VLAD [1, 12], instead of bilinear models or fully connected pooling. When bilinear models are used the features are passed through a signed squared root and L_2-normalization. In either case, we use softmax as a classifier.

End-to-end training: We use $K = 3$, following the advice from temporal modeling work [10]. Let the output feature maps of the CNNs be S_1, S_2, and S_3. The temporally aggregated features are given by $X = S_1 \circ S_2 \circ S_3$, and the temporal linearly encoded features are denoted by y. Let ℓ denote the loss function, and $d\ell/d(X)$ be the gradient of the loss function with respect to X. Algorithm 2 illustrates the forward and backward passes of our temporal linear encoding steps for the 3 segments setup.

The Back-propagation for the joint optimization of the

Algorithm 2 Forward & backward propagation steps for our deep temporal linear encoding with bilinear models for a scheme with 3 segments.

```
Input: Convolutional feature maps for a scheme of 3 segments, $\{S_1, S_2, S_3\}$, $S \in \mathbb{R}^{h \times w \times c}$
Output: $y \in \mathbb{R}^d$
Temporal Linear Encoding:
Forward Pass:
1. $X = S_1 \circ S_2 \circ S_3$, $X \in \mathbb{R}^{h \times w \times c}$
2. $y = [XX^T]$, $y \in \mathbb{R}^{c^2}$
Backward Pass:
1. $\frac{d\ell}{dS_1} = (S_2 \circ S_3) \frac{d\ell}{dX}$,
$\frac{d\ell}{dS_2} = (S_1 \circ S_3) \frac{d\ell}{dX}$,
$\frac{d\ell}{dS_3} = (S_1 \circ S_2) \frac{d\ell}{dX}$
```

K temporal segments can be derived as:

$$\frac{d\ell}{dS_k} = ((S_1 \circ ... \circ S_K) \backslash S_k) \frac{d\ell}{dX}, k \in K \tag{5}$$

In the end-to-end learning, the model parameters for the K temporal segments are optimized using stochastic gradient descent (SGD). Moreover, the temporal linear encoding model parameters are learned from the entire video. The scheme is illustrated in Fig. 4.

4. Evaluation

In this section, we first introduce the datasets and implementation details of our proposed approach. Then we demonstrate the applicability of our temporal linear encoding on 2D and 3D ConvNets using frames or clips to encode long-range dynamics across entire videos. Finally, we compare temporal linear encoding with the state-of-the-art methods.

4.1. Datasets

We conduct experiments on two challenging video datasets with human actions, namely HMDB51 [18] and UCF101 [28]. The HMDB51 dataset consists of 51 action categories with 6,766 video clips in all. The UCF101 dataset consists of 101 action classes with 13,320 video clips. Both of these datasets have at least 100 video clips for each action category. For both datasets, we use the three training/testing splits provided as the original evaluation scheme for these datasets, and report the average accuracy over these three splits.

4.2. Implementation details

We use the caffe toolbox [14] for ConvNet implementation and all the networks are trained on two Geforce Titan X GPUs. Here, we describe the implementation details of our two schemes, temporal linear encoding with two-stream ConvNets and temporal linear encoding with C3D ConvNets using bilinear models and fully-connected pooling. As mentioned earlier in the approach section, we use 3 segments for ConvNet training and testing.

Two-stream ConvNets: We employ three pre-trained models trained on the ImageNet dataset [5], namely AlexNet [17], VGG-16 [26], and BN-Inception [13], for the design of the two-stream ConvNets. The two-stream network consists of spatial and temporal networks, the spatial ConvNet operates on RGB frames, and the temporal ConvNet operates on a stack of 10 dense optical flow frames. The input RGB image or optical flow frames are of size 256×340, and are randomly cropped to a size 224×224, and then mean-subtracted for network training. To fine-tune the network, we replace the previous classification layer with a C-way softmax layer, where C is the number of action categories. We use mini-batch stochastic gradient descent (SGD) to learn the model parameters with a fixed weight decay of 5×10^{-4}, momentum of 0.9, and a batch size of 15 for network training. The prediction scores of the spatial and temporal ConvNets are combined in a late fusion approach as averaging before softmax normalization.

– **TLE with Bilinear Models:** In our experiments for bilinear models, we retain only the convolutional layers of each network, more specifically we remove all the fully connected layers, similar to [11, 20]. The convolutional feature maps extracted from the last convolutional layers (after the rectified output of the last convolutional layer, when there is one) are fed as input into the bilinear models. For example, the convolutional feature maps for the last layer of BN-Inception produces an output of size $14 \times 14 \times 1024$, leading to bilinear features 1024×1024, and 8,196 features for compact bilinear models. We follow two steps to fine-tune the whole model. First, we train the last layer using logistic regression. Secondly, we fine-tune the whole model. In both steps for training spatial ConvNets, we initialize the learning rate with 10^{-3} and decrease it by a factor of 10 every 4,000 iterations. The maximum number of iterations is set to 12,000. We use flip augmentation about the horizontal axis and RGB jittering for RGB frames. For the temporal ConvNet, we use a stack of 10 optical flow frames as input clip. We rescale the optical flow fields linearly to a range of $[0, 255]$ and compress as JPEG images. For the extraction of the optical flow frames, we use the TVL1 optical flow algorithm [42] from the OpenCV toolbox with CUDA implementation. In both steps for training the temporal ConvNets, we initialize the learning rate with 10^{-3} and manually decrease by a factor of 10 every 10,000 iterations. The maximum number of iterations is set to 30,000. We use batch normalization. Before the features are fed into the softmax layer, the features are passed through a signed squared root operation ($z \leftarrow sign(y)\sqrt{|y|}$) and L_2-normalization ($z^{'} \leftarrow z/||z||_2$).

– **TLE with Fully-Connected Pooling:** We follow the same two step fine-tuning scheme discussed earlier. For fine-tuning the fully-connected layers between the last convolutional layer and the C-way softmax layer for both spatial and temporal ConvNets, we initialize the learning rate with 10^{-3} and decrease it by a factor of 10 every 10,000 iterations in both model training steps. The maximum number of iterations is set to 30,000. We apply the same augmentation and discretization techniques for both RGB and optical flow frames, as discussed earlier.

C3D ConvNets: In our experiments, we use the C3D model [33] pre-trained on the Sport-1M dataset [15]. The convolution kernels are of size $3\times3\times3$ with stride 1 in both spatial and temporal dimensions, as suggested in [33]. The video is decomposed into non-overlapping, equal-duration clips of 16 frames. The C3D ConvNet operates on these video clips as input for network training. The video frames are of size 128×171. For network training, we randomly crop the video clips to a size $16 \times 112 \times 112$, and then mean-subtract. We use a single center crop per clip. For fine-tuning the network, we replace the previous classification layer with a C-way softmax layer, where C is the number of action categories. We use mini-batch stochastic gradient descent to learn the model parameters with a fixed weight decay of 5×10^{-4}, momentum of 0.9, and a batch size of 10 for network training.

Also in this work, following the same fine-tuning scheme as the original C3D ConvNets on UCF101, we fine-tune the C3D ConvNets on HMDB51 and report their average accuracy over the three splits.

– **TLE with Bilinear Models:** Similar to TLE with bilinear models from two-stream ConvNets, we retain the convolutional layers. For fine-tuning the model, we use the same two steps scheme explained earlier. In both steps for C3D ConvNets training, we initialize the learning rate with 3×10^{-3} and decrease by a factor of 10 every 10,000 iterations. The maximum number of iterations is set to 30,000. We use batch normalization. Before feeding the features to the softmax classifier, the features are passed through a signed squared root and L_2-normalization.

– **TLE with Fully-Connected Pooling:** To fine-tune the fully-connected layers of the C3D ConvNets, we follow the same two-step fine-tuning scheme discussed earlier. In both steps of model training, we initialize the learning rate with 10^{-3} and manually decrease by a factor of 10 every 10,000 iterations. The maximum number of iterations is set to 40,000.

Testing:

– **Two-stream ConvNets:** Given a video, we divide it into 3 parts of equal duration. The three parts are associated with the 3 segments. For TLE two-stream ConvNet testing, at a time, we extact 1 RGB frame or 10 optical flow frames from each part and feed these into the 3 segments network sequentially. In total, we sample 5 RGB frames or stacks of optical flow frames (i.e. 15 frames for the three-segments in total) from the whole video. For video prediction, we average predictions over all group of frame segments. The prediction scores of the spatial and temporal ConvNets are combined in a late fusion approach via averaging.

– **C3D ConvNets:** We decompose each video into non-overlapping clips of 16 frames, we then divide the number of clips into 3 equal parts. For TLE C3D ConvNets testing, 1 clip is extracted from each part and fed sequentally into the 3 segment network. In total, we extract 3 clips (i.e. 9 clips for three-segments in total) from the whole video. We average the predictions over all groups of clip segments to make a video-level prediction.

4.3. Evaluation of TLE

In this subsection, we explore (i) different aggregation functions T to linearly aggregate the segments into a compact intermediate representation for encoding; and (ii) different ConvNet architectures for both two-stream (spatial and temporal networks) and C3D networks. For this evaluation, we report the accuracy of split1 on UCF101 and HMDB51. The reported performance is for TLE with bilinear models using the tensor sketch algorithm.

Two-Stream ConvNets:

– **Aggregation Function:** In our evaluation, we explore three aggregations functions (i) element-wise average, (ii) element-wise maximum, and (iii) element-wise multiplication. In Table 1, we report the performance of the different aggregation strategies. We observe that the element-wise multiplication performs the best. Therefore, we choose element-wise multiplication as a default aggregation function. We believe combining the feature maps in this way allows us to aggregate the appearance and motion information accurately, hence leading to better results. Interestingly, we also found that aggregating rectified output of the last convolutional feature maps achieves around the same classification performance, in comparison to non-rectified ones.

Aggregation Function (T)	UCF101/HMDB51
Element-wise Maximum	91.3/67.4
Element-wise Average	92.6/68.1
Element-wise Multiplication	**94.8/70.4**

Table 1: Accuracy (%) performance comparison of the aggregation functions in TLE BN-Inception network for two-stream ConvNets using 3 segments on UCF101 and HMDB51 datasets (split1).

	UCF101/HMDB51	UCF101/HMDB51
Method	Spatial ConvNets	Temporal ConvNets
AlexNet	74.4/50.8	82.7/52.4
VGG-16	81.5/60.9	86.8/61.5
BN-Inception	**86.9/63.2**	**89.1/66.4**

Table 2: Different architecture accuracy (%) performance comparison of spatial and temporal ConvNets using 3 segments on the UCF101 and HMDB51 datasets (split1).

– **ConvNet Architectures:** Here, we compare the different ConvNet architectures for TLE. Specifically, we compare AlexNet [17], VGG-16 [26], and BN-Inception [13]. Among all architectures shown in Table 2, BN-Inception achieves the best performance, better than the AlexNet and VGG-16 architectures. BN-Inception is 5.4/2.3% (spatial ConvNets) and 2.3/4.9% (temporal ConvNets) better than VGG-16 on UCF101/HMDB51. Therefore, we choose BN-Inception as a default ConvNet architecture for TLE. We can observe that the deeper models, the higher is the performance gain on both datasets.

C3D ConvNets:

– **Aggregation Function:** We perform similar experiments to explore the aggregations functions in C3D ConvNets, as used in two-stream ConvNets. Table 3 summarizes the results of comparing different aggregation strategies. Similar to two-stream ConvNets, element-wise multiplication performs better in comparison to other candidate functions, and was therefore selected as a default aggregation function.

Aggregation Function (T)	Accuracy (%)
Element-wise Maximum	84.2
Element-wise Average	84.6
Element-wise Multiplication	**86.1**

Table 3: Performance comparison of different aggregation functions in TLE C3D ConvNet using 3 segments on the UCF101 dataset (split1).

– **ConvNet Architectures:** We use the C3D ConvNet [33] architecture as a default ConvNet architecture for TLE. The model obtains an accuracy of 86.1% for split1 on the UCF101 dataset.

4.4. Comparison with the state-of-the-art

Finally, after exploring the aggregation function and good ConvNet architectures, we compare our TLE with the current state-of-the-art methods over all three splits of the UCF101 and HMDB51 datasets. We report the average accuracy over the three splits of both datasets.

– **Two-stream ConvNets:** In Table 4, we compare the performance of TLE with the current methods using two-stream ConvNets and other traditional methods. TLE with bilinear models (TLE:Bilinear) performs the best among all methods. This model obtains an accuracy of 95.6% and 71.1% on UCF101 and HMDB51, respectively. The performance gap of TLE with bilinear models using the tensor sketch algorithm (TLE:Bilinear+TS) is however small 0.5/0.5%, and 3.4/2.3% for TLE with fully-connected pooling (TLE:FC-Pooling) when compared to TLE:Bilinear on UCF101/HMDB51. TLE:Bilinear is 7.6/11.7%, 1.6/2.6%, and 2.1/1.9% better than the Two-Stream [25], TSN [38], and 3DConv+3DPool [8] methods on UCF101/HMDB51. One can observe that the optical flow is better at capturing the motion information (shown in Table 2), and when combined with the appearance information in long-range temporal structure is effective to perform video-level learning. As another interesting comparison, our TLE with bilinear models yields very few parameters to train, in comparison to other methods which have several fully-connected layers to train with millions of parameters. Thus, our models are computationally efficient. Moreover, our models clearly show the power of encoded feature representations in video classification for entire videos, in end-to-end learning.

– **C3D ConvNets:** In Table 5, we summarize the performance of TLE in C3D ConvNets and compare it with the currently used 3D Conv [33], and other traditional methods. Similar to two-stream ConvNets, TLE:Bilinear outperforms other methods, and achieves an accuracy of 86.3% and 60.3% on UCF101 and HMDB51, respectively, which is 4/3.5%, and 0.4/3.1% better than the original C3D ConvNets [33] and iDT+FV [35] methods on

Method	UCF101	HMDB51
DT+MVSM [2]	83.5	55.9
iDT+FV [35]	85.9	57.2
Two Stream [25]	88.0	59.4
VideoDarwin [9]	–	63.7
C3D [33]	82.3	56.8
Two Stream+LSTM [41]	88.6	–
F$_{ST}$CV (SCI fusion) [30]	88.1	59.1
TDD+FV [37]	90.3	63.2
LTC [34]	91.7	64.8
KVMF [44]	93.1	63.3
TSN [38]	94.0	68.5
3DConv+3DPool [8]	93.5	69.2
TLE: FC-Pooling (ours)	92.2	68.8
TLE: Bilinear+TS (ours)	95.1	70.6
TLE: Bilinear (ours)	**95.6**	**71.1**

Table 4: **Two-stream ConvNets.** Accuracy (%) performance comparison of TLE BN-Inception network with state-of-the-art methods over all three splits of UCF101 and HMDB51.

Method	UCF101	HMDB51
SpatioTemporal ConvNet [15]	65.4	–
LRCN [7]	82.9	–
Composite LSTM Model [29]	84.3	44.0
iDT+FV [35]	85.9	57.2
Two Stream [25]	**88.0**	59.4
C3D [33]	82.3	56.8
TLE: FC-Pooling (ours)	83.1	58.6
TLE: Bilinear+TS (ours)	85.6	59.7
TLE: Bilinear (ours)	**86.3**	**60.3**

Table 5: **C3D ConvNets.** Accuracy (%) performance comparison of TLE with state-of-the-art methods over all three splits of UCF101 and HMDB51.

UCF101/HMDB51. The goal of this experiment is to show that TLEs can improve the performance of the original C3D ConvNets [33]. It is also interesting to see that, TLE:Bilinear improves the C3D ConvNet performance over the two-stream ConvNets [25] on the HMDB51 dataset. We think the reason for TLE:Bilinear performing better than other methods is that the model is essentially able to encode the dynamic appearance and motion using multiple aspects of long-range temporal cues in the video data, which were unavailable to the original C3D ConvNets [33].

5. Scene Context Embedding

This section describes an additional experiment to incorporate scene context in order to improve the success of action recognition.

ConvNets trained on individual frames in spatial networks tend to misclassify the contextual information from scenery and objects in videos, which could be an evident source of information for action recognition. As an application of our method, we study to incorporate the context information from the scenes to improve the action recognition performance. Our network architecture uses the TLE with 3 segments. In addition we add a fourth segment ConvNet pre-trained on the Places365 dataset [43]. The key reason behind using the latter is to supervise the learning of extra representations of scene-related information, to further boost the action recognition. In this way, we transfer the learned representations between the two tasks for better action recognition. We are aware that in this case we use additional data, but it is a nice way to demonstrate the capability of TLE to combine different data streams.

In this experiment, we exploit the context information from the scenes to improve the action recognition. We apply the same training scheme for two-stream ConvNets explained in Section 4.2. We use the VGG-16 network architecture in this experiment. In Table 6, we observe that the action recognition accuracy of this proposed method outperforms the TLE method with three segments (as shown in Table 2) on spatial ConvNets. The result indicates that the two streams of information are encoded as complementary information.

	UCF101/HMDB51
Method	Spatial ConvNets
TLE:Bilinear+TS,Action (ours)	81.5 / 60.9
TLE:Bilinear+TS,Action+Context (ours)	**83.8 / 63.6**

Table 6: Accuracy (%) performance comparison of the VGG-16 spatial ConvNets using 3 segments, when combined with context information pre-trained on Places365 [43]. The accuracy is reported for split1 on both datasets.

6. Conclusion

In this paper, we proposed Temporal Linear Encoding (TLE) embedded inside ConvNet architectures, aiming to aggregate information from an entire video, be it in form of frames or clips. The result is a global feature representation obtained in an end-to-end learning scheme. The model performs action prediction over an entire video. We demonstrated the TLE on two challenging action video datasets: HMDB51 and UCF101. In addition to yielding a better performance than the state-of-the-art methods, TLEs are computationally efficient, robust, compact, reduce the number of model parameters significantly below that of fully-connected ConvNets, and retain the feature interaction in a more expressive way without an undesired loss of information. Even though, in this paper, we have focused on two-stream and C3D ConvNets architectures, our method has the potential to generalize to other architectures, and can readily be employed with other encoding methods also. Thus, it can lead to more accurate classification. Another potential of this work is that TLEs are flexible enough to be readily employed to other forms of sequential data streams for feature embedding.

In future work, concerning the spatial and temporal segment aggregation, we plan to further investigate architectural alternatives. For instance, one could combine the spatial and temporal networks for each segment individually and then aggregate this spatio-temporal network in hierarchical fashion for a global spatio-temporal feature representation.

Acknowledgements: This work was supported by DBOF PhD scholarship, and KU Leuven CAMETRON project. The authors would like to thank Nvidia for GPU donation.

References

[1] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016.

[2] Z. Cai, L. Wang, X. Peng, and Y. Qiao. Multi-view super vector for action recognition. In *CVPR*, 2014.

[3] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshops*, 2004.

[4] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[6] A. Diba, A. M. Pazandeh, and L. Van Gool. Efficient two-stream motion and appearance 3d cnns for video classification. In *ECCV Workshops*, 2016.

[7] 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.

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

[9] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015.

[10] A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal localization of actions with actoms. *PAMI*, 2013.

[11] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *CVPR*, 2016.

[12] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014.

[15] 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.

[16] A. Klaser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.

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

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

[19] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[20] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015.

[21] J. C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*.

[22] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[23] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *ACM SIGKDD*, 2013.

[24] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007.

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

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[27] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[28] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[29] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[30] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015.

[31] P. Tang, X. Wang, B. Shi, X. Bai, W. Liu, and Z. Tu. Deep fishernet for object classification. *arXiv preprint arXiv:1608.00182*, 2016.

[32] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 2000.

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

[34] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *arXiv preprint arXiv:1604.04494*, 2016.

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

[36] L. Wang, Y. Qiao, and X. Tang. Video action detection with relational dynamic-poselets. In *ECCV*, 2014.

[37] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.

[38] 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.

[39] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008.

[40] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[41] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[42] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*, 2007.

[43] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016.

[44] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *CVPR*, 2016.