Temporal 3D ConvNets using Temporal Transition Layer

Ali Diba[1,4], Mohsen Fayyaz[2], Vivek Sharma[3], A.Hossein Karami[4], M.Mahdi Arzani[4], Rahman Yousefzadeh[4], Luc Van Gool[1,4]

[1]ESAT-PSI, KU Leuven, [2]University of Bonn, [3]CV:HCI, KIT, Karlsruhe, [4]Sensifai

{firstname.lastname}@esat.kuleuven.be, {lastname}@sensifai.com,
fayyaz@iai.uni-bonn.de, vivek.sharma@kit.edu

Abstract

The work in this paper is driven by the question how to exploit the temporal cues available in videos for their accurate classification, and for human action recognition in particular? Thus far, the vision community has focused on spatio-temporal approaches with fixed temporal convolution kernel depths. We introduce a new temporal layer that models variable temporal convolution kernel depths. We embed this new temporal layer in our proposed 3D CNN. We extend the DenseNet architecture - which normally is 2D - with 3D filters and pooling kernels. We name our proposed video convolutional network 'Temporal 3D ConvNet' (T3D) and its new temporal layer 'Temporal Transition Layer' (TTL). Our experiments show that T3D outperforms the current state-of-the-art methods on the HMDB51, UCF101 and Kinetics datasets.

1. Introduction

Compelling advantages of exploiting temporal rather than merely spatial cues for video classification have been shown lately [4, 20, 27]. Such insights are all the more important given the surge in multimedia videos on the Internet. Even if considerable progress in exploiting temporal cues was made [2, 6, 20, 21], the corresponding systems are still wanting. Recently, several variants of Convolutional Neural Networks (ConvNets) have been proposed that use 3D convolutions, but they fail to exploit long-range temporal information, thus limiting the performance of these architectures. Complicating aspects include: (i) these video architectures have many more parameters than 2D ConvNets; (ii) training the video architectures calls for extra large labeled datasets; and (iii) extraction and usage of optical-flow maps which is very demanding, and also difficult to obtain for large scale dataset, e.g. Sports-1M. All of these issues negatively influence their computational cost and performance.

Motivated by the above observations, we introduce a novel deep spatio-temporal feature extractor network illustrated in Figure 1. The aim of this extractor is to model variable temporal 3D convolution kernel depths over shorter and longer time ranges. We name this new layer in 3D ConvNets configuration 'Temporal Transition Layer' (TTL). TTL is designed to concatenate temporal feature-maps extracted at different temporal depth ranges, rather than only considering fixed 3D homogeneous kernel depths [2, 20, 21]. We embed this new temporal layer into the 3D CNNs. In this work, we extend the DenseNet architecture - which by default has 2D filters and pooling kernels - to incorporate 3D filters and pooling kernels, namely DenseNet3D. We used DenseNet because it is highly parameter efficient. Our TTL replaces the standard transition layer in the DenseNet architecture. We refer to our modified DenseNet architecture as 'Temporal 3D ConvNets' (T3D), inspired by C3D [20], Network in Network [15], and DenseNet [10] architectures. T3D densely and efficiently captures the appearance and temporal information from the short, mid, and long-range terms. We show that the TTL feature representation fits action recognition well, and that it is a much simpler and more efficient representation of the temporal video structure. The TTL features are densely propagated throughout the T3D architecture and are trained end-to-end. In addition to achieving high performance, we show that the T3D architecture is computationally efficient and robust in both the training and inference phase. T3D is evaluated on three challenging action recognition datasets, namely HMDB51, UCF101, and Kinetics. We experimentally show that T3D achieves the state-of-the-art performance on HMDB51 and UCF101 among the other 3D ConvNets and competitive results on Kinetics.

2. Related Work

Video Classification with and without ConvNets: Video classification and understanding has always been a hot topic. Several techniques have been proposed to come up with efficient spatio-temporal feature representations that capture the appearance and motion propagation across frames in videos, such as HOG3D [12], SIFT3D [16], HOF [14], ESURF [26], MBH [3], iDTs [22], and more. These were all hand-engineered. Among these, iDTs yielded the best performance, at the expense of being computationally expensive and lacking scalability to capture semantic concepts. It is noteworthy that recently several other techniques [7] have been proposed that also try to model the temporal structure

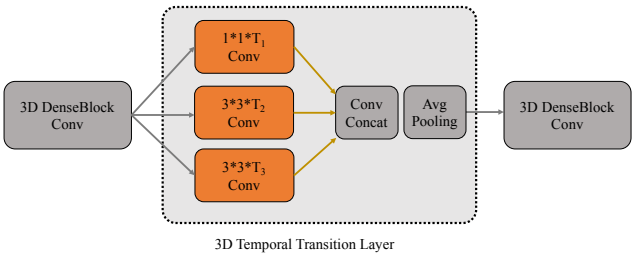

Figure 1: **Temporal Transition Layer (TTL)** is applied to our DenseNet3D. T3D uses video clips as input. The TTL operates on the different temporal depths, thus allowing the model to capture the appearance and temporal information from the short, mid, and long-range terms.

in an efficient way.

Temporal ConvNets: Recently, several temporal architectures have been proposed for video classification, where the input to the network consists of either RGB video clips or stacked optical-flow frames. The filters and pooling kernels for these architectures are 3D (x, y, time) with fixed temporal kernel depths throughout the architecture. The most intuitive are 3D convolutions ($s \times s \times d$) [27] where the kernel temporal depth d corresponds to the number of frames used as input, and s is the kernel spatial size. Simonyan et al. [17] proposed a two-stream network, cohorts of RGB and flow ConvNets. In their flow stream ConvNets, the 3D convolution has d set to 10. Tran et al. [20] explored 3D ConvNets with filter kernel of size $3 \times 3 \times 3$. Tran et al. in [21] extended the ResNet architecture with 3D convolutions. Feichtenhofer et al. [6] propose 3D pooling. Sun et al. [19] decomposed the 3D convolutions into 2D spatial and 1D temporal convolutions. Carreira et al. [2] proposed converting a pre-trained the 2D Inception-V1 [11] architecture to 3D by inflating all the filters and pooling kernels with an additional temporal dimension d. They achieve this by repeating the weights of 2D filters d times for weight initialization of 3D filters. All these architectures have fixed temporal 3D convolution kernel depths throughout the whole architecture. To the best of our knowledge, our architecture is the first end-to-end deep network that integrates variable temporal depth information over shorter and longer temporal ranges.

3. Temporal 3D ConvNets

Our goal is to capture short, mid, and long term dynamics for a video representation that embodies more semantic information. We propose a Temporal Transition Layer (TTL) inspired by GoogLeNet [11]. It consists of several 3D Convolution kernels, with diverse temporal depths (see Fig. 1). The TTL output feature maps are densely fed forward to all subsequent layers, and are learned end-to-end, as shown in Fig. 1. We employ this TTL layer in a DenseNet3D architecture. We name the resulting networks as Temporal 3D ConvNets (T3D). Fig. 1 sketches the steps of the proposed T3D.

In this work, we use the DenseNet architecture which has 2D filters and pooling kernels, but extend it with 3D filters and pooling kernels. We used the DenseNet architecture for several reasons, such as simpler and highly parameter efficient deep architecture, its dense knowledge propagation, and state-of-the-art performance on image classification tasks. In specific, (i) we modify 2D DenseNet by replacing the 2D kernels by 3D kernels in the standard DenseNet architecture and we present it as DenseNet3D; and (ii) introducing our new Temporal 3D ConvNets (T3D) by deploying 3D temporal transition layer (TTL) instead of transition layer in DenseNet. In both setups, the building blocks of the network and the architecture choices proposed in [10] are kept same.

Notation. The output feature-maps of the 3D Convolutions and pooling kernels at the l^{th} layer extracted for an input video, is a matrix $x \in \mathbb{R}^{h \times w \times c}$ where h, w and c are the height, width, and number of channels of the feature maps, resp. The 3D convolution and pooling kernels are of size ($s \times s \times d$), where d is the temporal depth and s is the spatial size of the kernels.

3D Dense Connectivity. Similar to 2D dense connectivity, in our network it is 3D dense connectivity that directly connects the 3D output of any layer to all subsequent layers in the 3D Dense block. The composite function H_l in the l^{th} layer receives the $\{x_i\}_{i=0}^{l-1}$ 3D feature maps of all preceding $(l-1)$ layers as input. The output feature-map of H_l in the l^{th} layer is given by:

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}]) \tag{1}$$

where $[x_0, x_1, \ldots, x_{l-1}]$ denotes that the features maps are concatenated. The spatial sizes of the x_i features maps are the same. The $H_l(\cdot)$ is a composite function of BN-ReLU-3DConv operations.

Temporal Transition Layer. Fig. 1 shows a sketch of Temporal Transition Layer (TTL). TTL is composed of several variable 3D Convolution temporal depth kernels and a 3D pooling layer, the depth of 3D Conv kernels ranges between $d, d \in \{T_1, \ldots, T_D\}$, where T_d have different temporal depths. The advantage of TTL is that it captures the short, mid, and long term dynamics, that embody important information not captured when working with some fixed temporal depth homogeneously throughout the network. The feature-map of l^{th} layer is fed as input to the TTL layer, $TTL : x \rightarrow x^{'}$, resulting in a dense-aggregated feature representation $x^{'}$, where $x \in \mathbb{R}^{h \times w \times c}$ and $x^{'} \in \mathbb{R}^{h \times w \times c^{'}}$. In specific, the feature-map from l^{th}, x_l is convolved with K variable 3D convolution kernel temporal depths, resulting to intermediate feature-maps $\{S_1, S_2, \ldots, S_K\}$, $S_1 \in \mathbb{R}^{h \times w \times c_1}$, $S_2 \in \mathbb{R}^{h \times w \times c_2}$, $S_K \in \mathbb{R}^{h \times w \times c_K}$, where c_1, c_2, and c_K have different channel-depths as x_l is convolved with different 3D convolution kernel temporal depths, while the spatial size (h, w) is same for all the $\{S_k\}_{k=1}^{K}$ feature-maps. These feature-maps $\{S_k\}_{k=1}^{K}$ are simply concatenated into a single tensor $[S_1, S_2, \ldots, S_K]$ and then fed into the 3D pooling layer, resulting to the output TTL feature-

map $x^{'}$. The output of TTL, $x^{'}$ is fed as input to $(l+1)^{th}$ layer in the T3D architecture. The TTL layer is learned in an end-to-end network learning, as shown in Fig. 1.

In our work, we also compare T3D with DenseNet3D i.e with the standard transition layer but in 3D. Compared to the DenseNet3D, T3D performs significantly better in performance, shown in Experimental Section 4. Although we agree that T3D model has 1.3 times more model parameters than DenseNet3D, but it is worth to have it because of its outstanding performance. It is also worth saying that, one can readily employ our TTL in other architectures too such as in Res3D [21] or I3D [2], instead of using fixed 3D Convolutions homogeneously through out the network.

4. Experiments

In this section, we demonstrate a study for the architecture of the proposed T3D model, and then the configurations for input data. Afterwards, we first introduce the datasets and implementation details of our proposed approach. Following, we test and compare our proposed methods with baselines and other state-of-the-art methods. For the ablation study of architecture search and configurations of input data, we report the accuracy of split-1 on UCF101.

4.1. Datasets

We evaluate our proposed method on three challenging video datasets with human actions, namely HMDB51 [13], UCF101 [18], and Kinetics [2]. For all of these datasets, we use the standard training/testing splits and protocols provided as the original evaluation scheme. For HMDB51 and UCF101, we report the average accuracy over the three splits and for Kinetics, we report the performance on the validation and test set.

4.2. Implementation Details

We use the PyTorch framework for 3D ConvNets implementation and all the networks are trained on 8 Tesla P100 NVIDIA GPUs. Here, we describe the implementation details of our Temporal 3D ConvNets.

Training: We train our T3D from scratch on Kinetics. Our T3D operates on a stack of 32 RGB frames. We resize the video to 256px when smaller, and then randomly apply 5 crops (and their horizontal flips) of size 224×224. For network weight initialization, we adopt the same technique proposed in [9]. For the network training, we use SGD, Nesterov momentum of 0.9, weight decay of 10^{-4} and batch size of 64. The initial learning rate is set to 0.1, and reduced by a factor of 10x manually when the validation loss is saturated. The maximum number of epochs for the whole Kinetics dataset is set to 200. Batch normalization also has been applied. We should mention that the proposed DenseNet3D shares the same experimental details as T3D.

Testing: For video prediction, we decompose each video into non-overlapping clips of 32 frames. The T3D is applied over the video clips by taking a 224×224 center-crop, and finally we average the predictions over all the clips to make a video-level prediction.

4.3. Architecture Search

To find the best architecture for our T3D, we conduct a large scale architecture search. We start the search by designing a new DenseNet3D based on the 2D DenseNet architecture, then we explore T3D architecture based on our DenseNet3D. Due to the high computational time of 3D ConvNets we limit the exploring space by exploiting a lot of insights about good architectures [2, 21].

DenseNet3D: As mentioned before, we have designed a new DenseNet3D architecture. To achieve the best configuration for the new architecture we have done a series of tests on the network-size, and temporal-depth of input data to the network. For the architecture study, the model weights were initialized using [9].

We employ two versions of 2D-DenseNet with network sizes of 121 and 169 for designing the DenseNet3D, namely T3D-121 and T3D-169. The models are trained on UCF101 first split from scratch and we have obtained 69.1% and 71.3% respectivley.

Temporal depth of series of input frames plays a key role in activity recognition tasks. Therefore, we have evaluated our T3D with configurations for different temporal depths. In the our evaluations the depth of 32 performs better than 16 with 69.1% versus 66.8%.

Frame Resolution: We use the DenseNet3D-121 for frame resolution study. We evaluate the model by varying the resolution of the input frames in the following set $\{(224 \times 224), (112 \times 112)\}$. In the DenseNet3D and later on the T3D setup, the higher frame size of 224px yields better performance. With DenseNet3D we have achieved 69.1% against 61.2%.

T3D: Seeking for the best configurations of our T3D, we have done experiments based on the results of experimental study which are exploited from DenseNet3D. The TTL layer is added to make more efficient spatial-temporal connection between 3D DenseBlocks of convolutions, and improvement is observed in the performance.

Method	Top1- Val	Avg-Test
DenseNet3D	59.5	-
T3D	**62.2**	71.5
Inception3D	58.9	69.7
ResNet3D-38 [8]	58.0	68.9
C3D* [8]	55.6	-
C3D* w/ BN [2]	-	67.8
RGB-I3D w/o ImageNet [2]	-	78.2

Table 1: Comparison results of our models with other state-of-the-art methods on Kinetics dataset. * denotes the pre-trained version of C3D on the Sports-1M.

Method	UCF101	HMDB51
DT+MVSM [1]	83.5	55.9
iDT+FV [23]	85.9	57.2
C3D [20]	82.3	56.8
Conv Fusion [6]	82.6	56.8
Conv Pooling [27]	82.6	47.1
Spatial Stream-Resnet [5]	82.3	43.4
Two Stream [17]	88.6	–
F_{ST}CV (SCI fusion) [19]	88.1	59.1
TDD+FV [24]	90.3	63.2
TSN-RGB [25]	85.7	-
Res3D [21]	85.8	54.9
ResNet3D	86.1	55.6
Inception3D	87.2	56.9
DenseNet3D	88.9	57.8
T3D (ours)	**91.7**	**61.1**
T3D+TSN (ours)	**93.2**	**63.5**

Table 2: Accuracy (%) performance comparison of T3D with state-of-the-art methods over all three splits of UCF101 and HMDB51.

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

Finally, after exploring and finding an efficient T3D architecture with the best configuration of input-data, we compare our DenseNet3D and T3D with the state-of-the-art methods by pre-training on Kinetics and finetuning on all three splits of UCF101 and HMDB51 datasets. For the UCF101 and HMDB51, we report the average accuracy over all three splits.

Table 1 shows the result on Kinetics dataset for T3D compared with state-of-the-art methods. C3D [20] employs batch normalization after each convolutional and fully connected layers (C3D w/ BN), and RGB-I3D which is without pretraining on the ImageNet (RGB-I3D w/o ImageNet) [2]. The T3D and DenseNet3D achieve higher accuracies than ResNet3D-34, Sports-1M pre-trained C3D and C3D w/ BN which is trained from scratch. However, RGB-I3D achieved better performance which might be the result of usage of longer video clips than ours (64 vs. 32). As mentioned earlier, due to high memory usage of 3D models we had to limit our model space search and it was not possible to checkout the longer input video clips. Moreover, [2] used larger number of mini-batches by engaging a large number of 64 GPUs that they have used, which plays a vital role in batch normalization and consequently training procedure.

Table 2 shows the results on UCF101 and HMDB51 datasets for comparison of T3D with other RGB based action recognition methods. Our T3D and DenseNet3D models outperform the Res3D [21], Inception3D and C3D [20] on both UCF101 and HMDB51 by 93.2% and 63.5% respectively. As shown in the Table 2, T3D performs better than Inception3D by almost 4% on UCF101. Furthermore, DenseNet3D and T3D achieve the best performance among the methods using only RGB input on UCF101 and HMDB51. Moreover it should be noted that, the reported result of RGB-I3D [2] pre-trained on ImageNet and Kinetics by Carreira et al. [2] is better than us on both UCF101 and HMDB51, this might be due to difference in usage of longer video clips and larger mini-batch sizes by using 64 GPUs. Furthermore, we note that the state-of-the-art ConvNets [2, 25] use expensive optical-flow maps in addition to RGB input-frames, as in I3D which obtains a performance of 98% on UCF101 and 80% on HMDB51. However the high cost of computation of such data limits their application at real world large scale applications. As additional experiments, we study the effect of feature fusion methods like TSN [25] on our T3D video features. TSN intends to encode the long term information coming from video clips. We employ the technique of TSN, but here we use our T3D features from 5 non-overlapping clips of each video for encoding via TSN aggregation method. The T3D+TSN results are reported in the same table. This simple feature aggregation method on T3D shows major improvement over using 2D CNN feature extraction from single RGB frames using the same aggregation method.

Note that, in our work we have not used dense optical-flow maps, and still achieving comparable performance to the state-of-the-art methods [25]. This shows the effectiveness of our T3D to exploit temporal information and capture long-range dynamics in video-clips. This calls for efficient methods like ours instead of computing the expensive optical-flow information (beforehand) which is very computationally demanding, and also difficult to obtain for large scale datasets.

5. Conclusion

In this work, we introduce a new 'Temporal Transition Layer' (TTL) that models variable temporal convolution kernel depths. We clearly show the benefit of exploiting temporal depths over shorter and longer time ranges over fixed 3D homogeneous kernel depth architectures. In our work, we also extend the DenseNet architecture with 3D convolutions, we name our architecture as 'Temporal 3D ConvNets' (T3D). Our TTL feature-maps are densely propagated throughout and learned in an end-to-end learning. The TTL feature-maps model the feature interaction in a more expressive and efficient way without an undesired loss of information throughout the network. Our T3D is evaluated on three challenging action recognition datasets, namely HMDB51, UCF101, and Kinetics. T3D architecture achieves state-of-the-art performance on HMDB51, UCF101 and comparable results on Kinetics , in comparison to other temporal deep neural network models. Even though, in this paper, we have employed TTL to T3D architecture, our TTL has the potential to generalize to any other 3D architecture too.

References

[1] Z. Cai, L. Wang, X. Peng, and Y. Qiao. Multi-view super vector for action recognition. In *CVPR*, 2014. 4

[2] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 3, 4

[3] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006. 1

[4] A. Diba, V. Sharma, and L. Van Gool. Deep temporal linear encoding networks. In *CVPR*, 2017. 1

[5] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in Neural Information Processing Systems*, pages 3468–3476, 2016. 4

[6] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 1, 2, 4

[7] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015. 1

[8] K. Hara, H. Kataoka, and Y. Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *ICCV*, 2017. 3

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 3

[10] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 1, 2

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[12] A. Klaser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008. 1

[13] H. Kuehne, H. Jhuang, R. Stiefelhagen, and T. Serre. Hmdb51: A large video database for human motion recognition. In *High Performance Computing in Science and Engineering*. 2013. 3

[14] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 1

[15] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2015. 1

[16] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM'MM*, 2007. 1

[17] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 4

[18] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012. 3

[19] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015. 2, 4

[20] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2, 4

[21] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv:1708.05038*, 2017. 1, 2, 3, 4

[22] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1

[23] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 4

[24] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 4

[25] 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. 4

[26] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008. 1

[27] 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. 1, 2, 4