Analysis on Temporal Dimension of Inputs for 3D Convolutional Neural Networks

Okan Köpüklü Gerhard Rigoll
Institute for Human-Machine Communication
Technical University of Munich
Munich, Germany
{okan.kopuklu, rigoll}@tum.de

***Abstract*—3D ConvNets provide a dedicated spatiotemporal representation in order to incorporate motion patterns within video frames. However, compared to 2D convolutions, the 3D convolution kernels increase the number of parameters in the architecture and the floating point operations during inference time, which are of critical importance for real-time applications requiring faster runtime. In this paper, we show a sparse sampling and stacking strategy to span large time intervals for 3D ConvNet architectures that can attain multiple times less inference time by relinquishing little amount of classification accuracy. The proposed approach is validated on action and gesture recognition tasks using two recent video datasets: Jester and Something-Something datasets.**

I. Introduction

Action and gesture recognition have received significant interest within computer vision field in the last few years. The current work in this domain is generally based on deep Convolutional Neural Networks (ConvNets), which outperform many state-of-the-art techniques based on hand-crafted features [1], [2].

ConvNets with three-dimensional convolution kernels (3D ConvNets) provide a decent way of temporal reasoning over video frames. For action and gesture recognition, 3D ConvNets achieve recently better performance compared to ConvNets with two-dimensional kernels (2D ConvNets) [3]. However, 3D ConvNets have many more parameters and longer inference time than 2D ConvNets, which limits their real-time capabilities.

The immense number of parameters in 3D ConvNet architectures would quickly lead to overfitting. With the availability of larger datasets (e.g. Sports-1M [1], Kinetics [3]) this overfitting problem is solved recently [4]. Nonetheless, the problem of large number of floating point operations during inference time of 3D ConvNets remains to be unsolved.

On the other hand, recent studies proved that expanding the temporal length of inputs for 3D ConvNets improves recognition performance [4], [5]. The intuition behind is that increased temporal length captures the content of actions that last long time durations better. However, increasing temporal length of the input also increases the number of floating point operations linearly. Correspondingly, the best option would be to have 3D ConvNets to cover larger time durations without increasing the number of floating point operations which can be achieved by sparse selection of input frames. As it can be seen in Fig. 9, reducing temporal dimension from 32 frames to 16 frames increases the speed more than two times, and the sparse selection of 16 frames (e.g., taking every second frame from 32 consecutive frames) achieves better performance than consecutive 16 frames. The reason behind this performance gain is that the sparse sampling spans longer time durations which is better to capture performed actions and gestures. Dropped frames can also be utilized by appending them to the selected ones creating a stacked structure which improves the performance further, as in [20]. The proposed approach is validated with two recent gesture and action datasets which are Jester [6] and Something-Something [7] datasets, respectively.

The main contributions of this paper can be summarised as follows: The speed of the 3D ConvNet architectures can be increased by reducing the temporal dimension of inputs (i.e., reducing the computation) with sparse selection and stacking of the input frames. This approach increases the speed of the architecture multiple times without significant performance degradation (e.g., compared to input dimension of 32 frames, sparsely selected 16 frames is 2 times faster with only performance degradation of 1.859% accuracy, Fig. 8). Moreover, we experimentally prove that 3D ConvNets still show relatively good performance even when the number of input frames reduces to 4. This shows that 3D ConvNets can capture the temporal relation between such a small number of frames under correct conditions. These conditions are discussed in detail in Section IV.

The rest of the paper is organised as follows: Section II reviews the action and gesture recognition that applies deep learning. Section III introduces the proposed approach and implementation details. Experimental results on the Jester and Something-Something datasets are provided in Section IV. Finally, Section V concludes the paper.

II. Related Work

ConvNets have been initially applied for static image analysis, which has provided better performance over hand-crafted features [8], [9]. Then, they have been extended for video action and gesture recognition [1], [2], [10].

After significant progress in 2D ConvNets in various tasks especially with ImageNet challenge [11], first deep learning based video analysis architectures were also based on 2D convolutional kernels. In [1], [2], [10], [12], video frames are treated as multi-channel inputs to 2D ConvNets for action classification. Temporal Segment Network (TSN) [13] divides video data into segments and combines information from color and optical flow modalities for action recognition. Recently, Temporal Relation Network (TRN) [14] builds on top of TSN to investigate temporal relational reasoning between video frames at multiple time scales. In [15], the authors propose to extract features from video frames by a 2D ConvNet and apply Recurrent Neural Networks (RNN) for global temporal modeling.

In [16], 3D ConvNets are explored to provide an effective tool for accurate action recognition. 3D ConvNets use 3D convolutional kernels and 3D pooling to capture discriminative features along both spatial and temporal dimensions [5], [17]. However, the number of parameters in 3D ConvNets are much larger compared to 2D ConvNets. This requires 3D ConvNets to be pretrained on large datasets first (e.g. Sports-1M [1], Kinetics [3]) for the application in small scaled datasets like UCF-101 [18] or HMDB-51 [19] in order to prevent overfitting. A recent work analysed 3D ConvNets on different scales of video datasets and proved that Kinetics dataset is large enough to train deep 3D ConvNets without overfitting [4].

Temporal dimension of inputs for 3D ConvNets are analysed in [4], [5], and it is proved that increased temporal dimension of inputs improves the accuracy of action recognition. However, increased temporal dimension also increases the number of floating point operations which results in slower architectures. Moreover, in order to train architectures with increased temporal dimension, either batch size or resolution of the video should be reduced to fit in the same hardware.

In another recent study [20], the authors propose to use data level fusion for color and optical flow modalities. This work was the main inspiration to append dropped frames to the selected ones and create a stacked structure. In Section 4, it is experimentally proved that stacking increases the recognition performance.

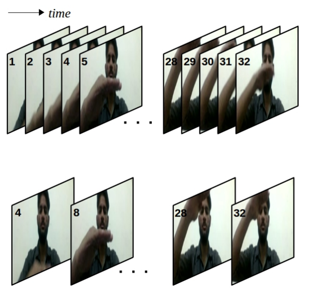

Fig. 1. Sparse selection of video frames. Sparse selection is achieved by equidistant downsampling. Every fourth frame is selected and temporal dimension of 32 frames is reduced to 8 frames.

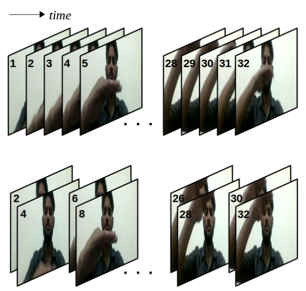

Fig. 2. Stacking dropped frames to the sparsely selected ones. Middle frames of the dropped ones are appended to the selected ones. Frames of 2,6,...,26,30 are appended to 4,8,...,28,32 respectively.

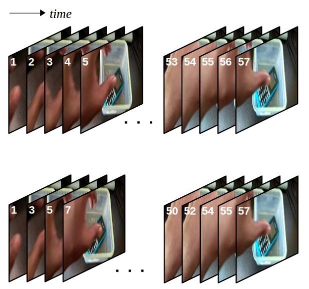

Fig. 3. Downsampling of large number of video frames to a smaller predefined number by dropping some of the frames.

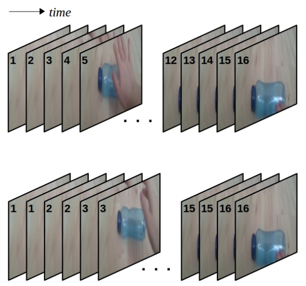

Fig. 4. Upsampling of small number of video frames to a larger predefined number by replicating some of the frames.

III. Methodology

In this section, we describe the sparse selection and stacking strategy and the network architecture used for gesture and action recognition. Particularly, we first describe how to apply sparse selection and stacking and explain how it speeds up the 3D ConvNet architecture. Then, we introduce the network architecture that we experimented on. Finally, we describe the training details.

A. *Sparse Selection and Stacking*

3D ConvNets require fixed temporal dimension inputs, which means the length of the input videos should be same. Therefore, selection of input frames to capture the essence of the performed actions plays a critical role in the recognition performance.

Selected input frames for long-term actions should represent the complete action in order to achieve better performance. In this manner, [4], [5] proposed to use longer temporal dimension. However, the performance gain achieved by increasing the number of input frames results in a slower runtime of the architectures. In order to remedy this, we proposed to use sparse selection and stacking.

Fig. 1 shows how to apply the sparse selection of 8 frames from 32 frames (i.e., selecting every 4^{th} frame). This way, longer time durations are covered without increasing the number of input frames. Instead of selecting consecutive 16 frames on Jester Dataset, the sparse selection of 16 frames (selecting every second frame of 32 frames) shows better performance, Fig. 8. Sparse selection is always applied by equidistant downsampling for all videos.

Dropped frames can also be used by appending them to the selected ones creating a stacked structure which improves the performance. Fig. 2 shows how to apply stacking where the middle frames of the dropped ones are appended to the selected ones. With this stacking strategy, overall used frames become uniformly distributed. If stacking is applied, the first convolutional layer of the 3D ConvNet should be modified in order to accommodate the stacked structure with the number of input channels as proposed in [20].

The above-mentioned strategy can be applied to datasets that have videos with relatively same temporal dimension. So that, choosing a predefined number of frames and applying sparse selection can capture the main motion cues without much of performance degradation. In other words, it is better

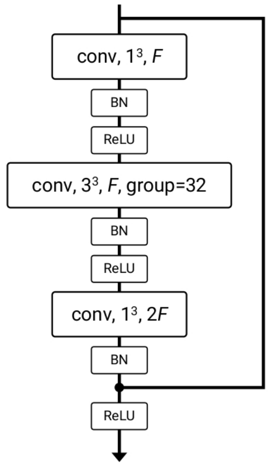

Fig. 5. ResNeXt block. F refers to the number of feature maps and group refers to the number of groups for group convolutions.

to use 4 frames distributed over the entire time period rather than 4 consecutive frames selected at any location within this time period. However, some datasets have videos with varying number of frames (e.g., Something-Something Dataset have videos with a temporal dimension between 20 and 75 frames). For such datasets, downsampling and upsampling by dropping and replicating some of the frames in order to get a predefined number of frames can be applied as in Fig. 3 and Fig. 4, respectively.

B. Network Architecture

The 3D ConvNet architecture in this study is based on residual networks (ResNets) [21] which showed good performance with large enough datasets [4]. As a building block, we used an extended version of ResNet which is called ResNeXt [22], as depicted in Fig. 5. For the shortcut connections in Fig. 5, the summation is used. BN and ReLU in Fig. 5 refers to batch normalization [23] and rectified linear unit [24].

Unlike the original ResNet block, the ResNeXt block introduces group convolutions, which divide the feature maps into small groups. Moreover, ResNeXt introduces cardinality, which refers to the number of middle convolutional layer groups in the bottleneck block. The authors in [22] showed that increasing the cardinality of 2D architectures is more effective than using wider or deeper architectures. Following the design choices as in [4], we used ResNeXt-101 in our experiments with the cardinality of 32. Table I shows the different network architectures for different input types used for the experiments on Jester Dataset. F and N in Table I refer to the number of feature channels corresponding in Figure 5 and the number of blocks in each layer, respectively.

The size of the first convolutional kernels is determined from experimentally best performing ones. For the first convolutional kernels of (7x7x7), (5x7x7), (3x7x7) and (1x7x7) zero padding of (3x3x3), (2x3x3), (1x3x3) and (0x3x3) are used, respectively. The stride of (1x2x2) remained same in the first convolutional kernel for all the experiments.

Input	Stacking	conv1	conv2-5	
32-frames	-	(7x7x7)	conv2_x (F:128, N:3) conv3_x (F:256, N:24) conv4_x (F:512, N:36) conv5_x (F:1024, N:3)	global average pool, C-d fully-connected, softmax
16-frames	2-stack	(3x7x7)		
16-frames	-	(5x7x7)		
8-frames	4-stack	(3x7x7)		
8-frames	2-stack	(3x7x7)		
8-frames	-	(3x7x7)		
4-frames	4-stack	(1x7x7)		
4-frames	2-stack	(3x7x7)		
4-frames	-	(3x7x7)		

TABLE I
THE NETWORK ARCHITECTURE APPLIED FOR THE EXPERIMENTS ON JESTER DATASET V1.

For the experiments on Something-Something Dataset, stacking is not applied. The reason is that during downsampling and upsampling, selected frames span a different time interval since Something Something Dataset have temporally variant video samples. For the first convolutional kernel in the network architectures, (7x7x7), (7x7x7), (3x7x7) and (3x7x7) are used for the temporal dimension of 32 frames, 16 frames, 8 frames and 4 frames, respectively. The remaining configuration is same as Table I for the experiments on Something Something Dataset.

C. Training Details

For all the experiments, we have used ResNeXt-101 architecture pretrained on Kinetics dataset which is made available by the authors of [4]. The final fully connected layer is modified according to the number of classes of the applied datasets.

For stacked structure, the first convolutional kernel of the network architecture is modified according to the number of input channels. For example, for 4-stack structure, the first convolutional kernel is modified to accommodate $4 \times 3 = 12$ input channels.

Learning. We use stochastic gradient descent (SGD) with standard categorical cross-entropy loss. Size of the mini-batch is chosen as high as possible according to the hardware limitations. As we have used NVIDIA Titan Xp GPU for training, we used 128, 96, 48 and 26 videos as mini-batches for 4 frames, 8 frames, 16 frames and 32 frames of temporal dimensions, respectively. For the momentum and weight decay, 0.9 and $1{\times}10^{-3}$ are used, respectively. Since we use pretrained models, we initialize the learning rate relatively small that is 1×10^{-2} for all the experiments. For both of the datasets, the learning rate is reduced twice with a factor of 10^{-1} after 5^{th} and 20^{th} epochs and optimization is completed after 5 more epochs.

Regularization. Several regularization techniques have been applied in order to reduce over-fitting. Weight decay ($\gamma = 1{\times}10^{-3}$) is applied on all parameters of the network. We apply transfer learning by using pretrained models on Kinetics dataset. Moreover, intensive data augmentation techniques are applied which is mentioned in the next part.

Data Augmentation. Several data augmentation techniques are applied to increase the variability of the training videos: (a) Random resizing ($\pm 10\%$), (b) random spatial rotation ($\pm 20°$), (c) spatial elastic deformation [25] with distortion field strength of α = 1.0 and standard deviation of the smoothing Gaussian kernel σ = 2.0 (with 50% probability), (d) random cropping and scale jittering as applied in [13], (e) flipping horizontally with probability 50% (for only Something-Something dataset), (f) temporal scaling ($\pm 10\%$) and jittering ($\pm$ 2 frames) (for only Something-Something). These data augmentation steps are applied on-the-fly, and finally the input is resized to 112 x 112 for network training.

Implementation. We have implemented our approach using PyTorch [26] with a single Nvidia Titan Xp GPU.

IV. Experiments

The performance of the proposed approach is validated on two publicly available datasets: Jester Dataset V1 and Something-Something Dataset V1. For the evaluation part, center cropping with temporally centered selection of the video frames are used for all the datasets.

A. Datasets

Jester Dataset V1 is currently the largest hand gesture dataset, which is recently made available [6]. It consists of 148,092 segmented gesture videos under 27 classes. The video clips are collected by a large number of crowd-workers performing predefined hand gestures in front of a camera. The dataset is divided into training, validation and test sets containing 118562, 14787 and 14743 videos, respectively.

Something-Something Dataset V1 is a large collection of densely-labeled video clips that show humans performing basic actions with everyday objects [7]. It allows machine learning models to develop fine-grained understanding of basic actions that occur in the physical world. There are in total 108,499 action videos under 174 classes, which is divided into training, validation and test sets containing 86017, 11522 and 10960 videos, respectively.

Jester dataset has gesture videos with similar temporal dimension concentrated between 30 - 40 frames. This makes it suitable to capture the performed hand gesture by choosing a temporal dimension of 32 frames. Afterwards, sparse selection and stacking can successfully be applied to these 32 frames as in Fig. 1 and Fig. 2. On the other hand, Something-Something dataset has videos with relatively varying temporal dimension. This steered us to apply a different strategy for frame selection: Downsampling and upsampling the input frames to the desired temporal dimension as in Table 3 and Table 4. The histogram of video lengths for Jester and Something-Something datasets are given in Fig. 6 and Fig. 7, respectively. We have conducted detailed experimental analysis only on the validation set of the both datasets, since the labels of the test sets are not made available by the dataset providers.

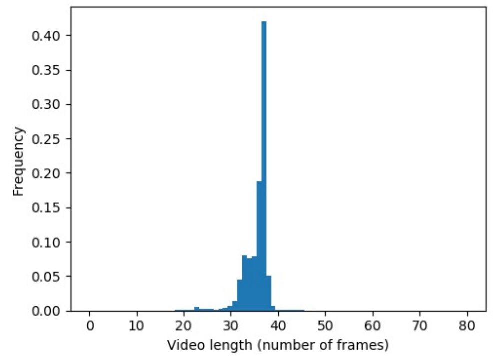

Fig. 6. Histogram of video lengths for Jester Dataset V1.

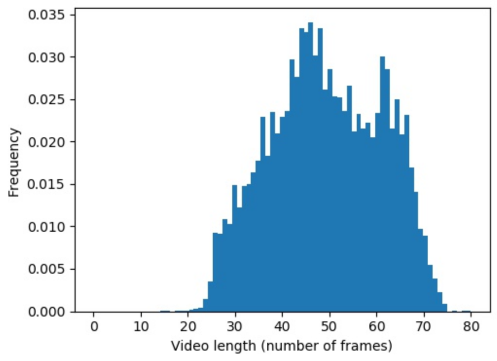

Fig. 7. Histogram of video lengths for Something-Something Dataset V1.

B. Results Using Jester Dataset

We initially investigated the performance of ResNeXt-101 architecture with consecutively selected 32 frames. It achieves a very good performance of 96.699% accuracy on the validation set of Jester dataset. We also investigated its performance on the test set by submitting our predictions to the official leaderboard. The ResNeXt-101 architecture competes with state-of-the-art results having a very similar performance with 96.24% accuracy as can be seen in Table III. So, our objective is to speed up this architecture without much of a performance degradation.

Secondly, we analyzed the sparse sampling strategy in order to cover large time durations without increasing the temporal dimension of inputs. So, we sparsely selected equidistant 16 frames, 8 frames and 4 frames out of 32 frames and reported performances of 94.840%, 91.817% and 87.215%, respectively, which is given in Table II. Although, there is some performance degradation, achieved performance with sparse selection is still better than the consecutive selection since we capture the main motion cues better with sparse selection, as seen in Fig. 8. On the other hand, we obtain considerable speed-up by applying sparse selection of 16 frames, 8 frames and 4 frames that are 2.09 times, 3.29 times and 4.79 times faster than 32 frames, respectively, as can be seen in Table II and Fig. 9. The speeds in Table II and Fig. 9

Input	Stacking	Speed (vps)	Acc.(%)
32-frames	-	73	**96.699**
16-frames	2-stack	104	95.250
16-frames	-	153	94.840
8-frames	4-stack	158	93.112
8-frames	2-stack	206	92.602
8-frames	-	245	91.817
4-frames	4-stack	271	91.000
4-frames	2-stack	303	89.572
4-frames	-	**350**	87.265

TABLE II
RESULTS ON THE VALIDATION SET OF JESTER DATASET V1.

Model	Acc.(%)
DRX3D	**96.60**
8-MFFs-3f1c	96.28
2 Stream CNN	96.28
ResNeXt-101 (32-frames)	**96.24**
MFNet	96.22
NUDT_PDL	95.34
DIN	95.31
Guangming Zhu	95.01

TABLE III
RESULTS ON THE TEST SET OF JESTER DATASET V1.

is calculated as videos per second (vps) using mini-batch of 16 videos.

Thirdly, we analyze the effect of stacking some of the dropped frames from sparse selection to the selected ones as in [20]. The selection of the stacked frames is always performed to keep the overall frame selection uniform. As an example, let's assume that we apply sparsely selected 4 frames and selected frames are 8, 16, 24 and 32. For 4 stack structure, appended frames to 16 are 10, 12 and 14. Performance gain achieved by stacking strategy can be seen in Table II.

Lastly, we compared the performance of sparse and consecutive frame selection which is depicted in Fig. 8. There is a significant performance gain achieved by sparse selection since it contains longer time durations which is better for capturing the full content of the actions.

C. Results Using Something-Something Dataset

Since Something-Something dataset contains videos having varying number of frames between 20 - 75 frames (see Fig. 7), we have applied downsampling and upsampling to obtain desired temporal dimension. In another perspective, downsampling is also a form of sparse sampling. However, the number of dropped frames in downsampling is not same and varies according to the video size. Therefore, we have decided not to apply stacking for this dataset.

We investigated the performance of ResNeXt-101 architecture for 32 frames, 16 frames, 8 frames and 4 frames and reported performance of 43.200%, 38.144%, 29.943% and 21.315%, respectively, as can be seen in Table IV. Although we achieve the same speed-up as in Jester dataset, performance degradation is significant for Something-Something dataset, especially for 8 frames and 4 frames temporal dimensions. There are several reasons for that. Firstly, Something-Something datasets have more action classes (174 compared to 27 for Jester Dataset) which are more complex than Jester dataset such that selected 4 frames and 8 frames cannot capture the full content of the action. Secondly, the number of training samples for Something-Something dataset is small (494 average number of training videos compared to 4391 for Jester Dataset) making it harder to train without overfitting. Thirdly, the temporal dimension of the video samples varies too much, making it inappropriate for networks requiring

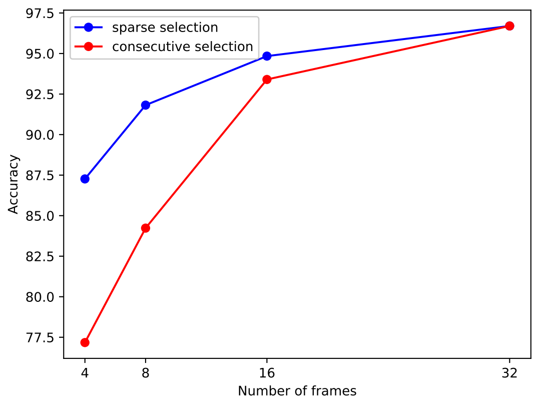

Fig. 8. Performance comparison of sparse and consecutive frame selections on the validation set of Jester Dataset V1. Sparse selection is applied on 32 frames.

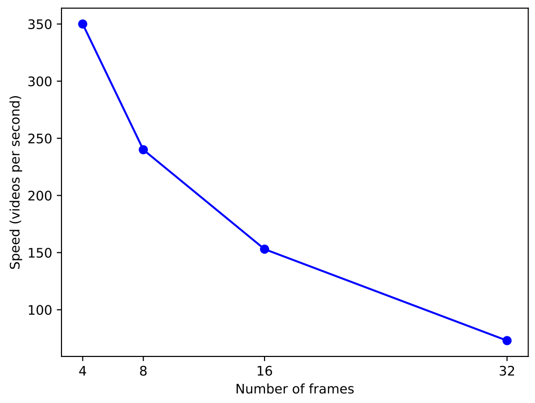

Fig. 9. Speed comparison of different temporal dimensions of inputs. As temporal dimension increases, the speed of the 3D ConvNet architecture drops.

Input	Stacking	conv1	Speed (vps)	Acc.(%)
32-frames	-	(7x7x7)	73	**43.200**
16-frames	-	(7x7x7)	153	38.144
8-frames	-	(3x7x7)	245	29.943
4-frames	-	(3x7x7)	**350**	21.315

TABLE IV
RESULTS ON THE VALIDATION SET OF SOMETHING-SOMETHING DATASET V1.

inputs with fixed temporal dimension.

The analysis on Something-Something dataset proves that dataset characteristics play a critical role on the performance achieved by sparse sampling. Also, dataset complexity should also be taken into consideration for the applied temporal dimension reduction by sparse sampling.

V. CONCLUSION

This paper presents an analysis on the temporal dimension of inputs for 3D ConvNets to obtain multiple times faster runtime by relinquishing little amount of classification accuracy. For this purpose, sparse sampling and stacking strategy are proposed.

We evaluated the proposed approach on two recent datasets and demonstrated that sparse sampling and stacking increases the speed of the architecture by reducing the temporal dimension but spanning the same time duration. With the same temporal dimension of inputs, sparse selection achieves better performance compared to consecutive selection. The main reason for this is that the important motion cues of the actions can be captured better with sparse selection by spanning longer time durations.

Experimental results proved that datasets should be large enough to apply sparse sampling without significant performance degradation. Moreover, dataset characteristics should also be taken into consideration for the degree of temporal dimension reduction by sparse sampling.

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

[1] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[2] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[3] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 4724–4733.

[4] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA*, 2018, pp. 18–22.

[5] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1510–1517, 2018.

[6] https://www.twentybn.com/datasets/jester/v1.

[7] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag *et al.*, "The something something video database for learning and evaluating visual common sense," in *The IEEE International Conference on Computer Vision (ICCV)*, vol. 1, no. 2, 2017, p. 3.

[8] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[10] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, 2009, pp. 248–255.

[12] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, "Towards good practices for very deep twostream convnets," vol. abs/1507.02159, 2015. [Online]. Available: http://arxiv.org/abs/1507.02159

[13] 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 *European Conference on Computer Vision*. Springer, 2016, pp. 20–36.

[14] B. Zhou, A. Andonian, and A. Torralba, "Temporal relational reasoning in videos," *arXiv preprint arXiv:1711.08496*, 2017.

[15] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[16] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[17] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4489–4497.

[18] 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.

[19] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2556–2563.

[20] O. Köpüklü, N. Köse, and G. Rigoll, "Motion fused frames: Data level fusion strategy for hand gesture recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5987–5995.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[25] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." in *ICDAR*, vol. 3, 2003, pp. 958–962.

[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.