Towards Good Practice for Action Recognition with Spatiotemporal 3D Convolutions

Kensho Hara, Hirokatsu Kataoka, Yutaka Satoh
National Institute of Advanced Industrial Science and Technology (AIST)
Tsukuba, Ibaraki, Japan
Email: {kensho.hara, hirokatsu.kataoka, yu.satou}@aist.go.jp

***Abstract*—The purpose of this study is to explore good practice for training convolutional neural networks (CNNs) with spatiotemporal three-dimensional (3D) kernels. Recently, 3D CNNs in the field of action recognition are rapidly developed, and the performance levels of them have improved significantly. However, to date, conventional research has mainly focused on their architecture, and has not sufficiently explored their training configurations. We conduct various experiments with different training configurations on Kinetics, UCF-101, and HMDB-51 datasets to share the knowledge of 3D CNNs for the research community. According to the results of those experiments, the following conclusions could be obtained. (i) Data augmentation by spatiotemporal random cropping improved the performance levels. (ii) Data augmentation by multi-scale spatial cropping increased the accuracies in most cases whereas multi-scale temporal cropping decreased them. (iii) A corner cropping strategy, which is previously shown as a good method for two-stream 2D CNNs, resulted lower accuracies for 3D CNNs compared with simple random cropping. (iv) Freezing early layers of 3D CNNs improved the performance levels when fine-tuning 3D CNNs on a relatively small dataset.**

I. Introduction

Vision algorithms for videos have achieved remarkable progress, as with those for images. In particular, action recognition accuracy has been significantly improved. For instance, accuracies in UCF-101 [1] and HMDB-51 [2], which are representative video datasets, improved from 88.0% and 59.4% [3] to 98.0% and 80.7% [4] in last three years, respectively. Developments in large-scale video datasets and convolutional neural networks (CNNs) mainly contribute to this progress.

Data-scale of video datasets has recently grown. Until several years ago, the data-scale of video datasets was relatively small compared with image datasets, such as ImageNet [5]. It is difficult to train CNNs from scratch using relatively small datasets, such as UCF-101 and HMDB-51. More recently, larger video datasets including Sports-1M [6], YouTube-8M [7], and Kinetics [8] were released and used for training CNNs. Some works showed that the use of the Kinetics dataset, which includes more than 300 K temporally trimmed videos, enables training of very deep CNNs even without pretraining on any datasets [9], and that Kinetics pretrained CNNs achieved state-of-the-art performance [4].

CNNs for action recognition have also been developed in several years. Similar to image recognition, CNNs with 2D convolutional kernels firstly proposed to recognize actions in videos [3], [10], [11]. Such 2D CNNs often utilize the two-stream architectures that combine both stacked flow and RGB images to capture not only appearance but also motion information. One of the advantages of the 2D CNNs is that they can be pretrained on large-scale image datasets, such as ImageNet, because the number of convolutional kernel dimensions are the same (i.e. 2D convolutions). More recently, CNNs with spatiotemporal 3D convolutional kernels, which can learn spatiotemporal feature representation from raw videos, achieved better performance than 2D CNNs. Though training of 3D CNNs is difficult because of the immense number of parameters in them, which are much larger than those of 2D CNNs, the use of recent large-scale video datasets enables the training and significantly improves their performance [4], [9].

Whereas 3D CNNs and large-scale datasets achieved significant performance improvements in action recognition, achieving good performance using 3D CNNs is not trivial. Conventional research has mainly focused on their architecture [4], [9], [12]. However, we have to tune hyper parameters and select training configurations to achieve the best performance even using well-organized architectures. Though some works showed such good practice for 2D CNNs [11], good practice for the training of 3D CNNs is not sufficiently explored.

In this study, we conduct various experiments using the Kinetics, UCF-101, and HMDB-51 datasets in order to explore good practice for training 3D CNNs. We focus on configurations of data augmentation and fine-tuning for 3D CNNs. Several data augmentation configurations for videos, such as spatiotemporal random cropping [4], [12] and spatial corner cropping [11], are used by different researchers. In the experiments, we compare these configurations and show the best configuration for 3D CNNs. We then compare fine-tuning configurations. Most works for 3D CNNs train all layers in fine-tuning [4] whereas some works reports that freezing early layers and training only later layers achieved the best performance [9]. We experimentally confirm the best configuration of fine-tuning. We believe that these experiments contribute to exploring good practice for the training of 3D CNNs, and that sharing these knowledge for the research

community facilitates development of action recognition.

II. Related Work

Developments of two-stream 2D CNNs proposed by Simonyan et al. [3] have been improved action recognition performance [10], [11], [13]–[16]. The two-stream CNNs use RGB and stacked optical flow frames as appearance and motion information, respectively, and showed that combining the two-streams has the ability to improve action recognition accuracy. Numerous methods based on the two-stream CNNs have been proposed to achieve further improvements by introducing an advantage of hand-crafted features [10], trying different combining method of the two streams [13]–[15], and modeling long-range temporal structure [16]. The abovementioned approaches are based on 2D CNNs, which are not our main focus. Similar to this study, Wang et al. tried to achieve good practice for the training of very deep two-stream 2D CNNs [11]. Though their practice contributes to improve action recognition performance, the practice was examined only for 2D CNNs.

3D CNNs, which are our main focus, have recently begun to outperform 2D CNNs through the use of large-scale video datasets. These 3D CNNs are intuitively effective because the 3D convolution can be used to directly extract spatio-temporal features from raw videos. Based on our knowledge, Ji et al. proposed the first method that utilizes 3D convolution to extract spatio-temporal features from videos. After the research, various researchers examined 3D CNNs to improve their performance [4], [9], [12], [17], [18]. In their study, Tran et al. trained a VGG-like 3D CNN, which they referred to as C3D, using the large-scale Sports-1M dataset [12]. Their experiments showed that a $3 \times 3 \times 3$ convolutional kernel achieved the best performance level. In another study, Varol et al. showed that expanding the temporal length of inputs for C3D improves recognition performance [17]. Meanwhile, Kay et al. showed that the use of their proposed Kinetics dataset for the training of 3D CNNs significantly improves recognition accuracy [8]. In still another study, Carreira et al. achieved state-of-the-art performance using inception [19] based 3D CNNs, which they referred to as I3D [4]. In our previous study, we examined 3D residual network (ResNet) architectures and showed that deeper architectures achieved better recognition accuracy [9].

Though, as described above, previous works mainly focused on architectures of 3D CNNs, few works explored good practice for training the 3D CNNs. Varol et al. showed results of 3D CNNs with different data augmentation configurations [17]. However, their experiments only examined data augmentation for only fine-tuning. In this study, we conduct experiments with different data augmentation configurations for both training from scratch and fine-tuning, as well as different configurations of fine-tuning.

III. Method

A. Network Architecture

In this study, we use 3D ResNet [9], which is a spatiotemporal extension of original 2D ResNet [20]. ResNet, which is one of the most successful architectures in image classification, provides shortcut connections that allow a signal to bypass one layer and move to the next layer in the sequence. Since these connections pass through the networks' gradient flows from the later layers to the early layers, they can facilitate the training of very deep networks. Because 3D ResNet achieved good performance in action recognition as shown in [9], we adopt the ResNet architecture.

We use the 18-layer ResNet with basic blocks of the ResNet. It is easy to utilize such relatively shallow network because of its relatively small number of parameters. Thus, exploring good practice based on the network is practically important. Table I shows the network architecture. Each block consists of two convolutional layers, and each convolutional layer is followed by batch normalization and a ReLU. The sizes of convolutional kernels in the blocks are $3 \times 3 \times 3$. A shortcut pass connects the top of the block to the layer just before the last ReLU in the block. We use identity connections and zero padding for the shortcuts of the ResNet block (type A in [20]) to avoid increasing the number of parameters. Strides of first convolutional layers of conv3, conv4, and conv5 are set to two to perform down-sampling of the inputs. A max pooling layer in conv2 also down-samples the inputs with a stride of two. Different from other convolutional layers, the size of conv1 is $7 \times 7 \times 7$. The temporal stride of conv1 is 1 whereas the spatial one is 2, similar to C3D [12].

B. Implementation

1) Training: We use stochastic gradient descent with momentum to train the networks and randomly generate training samples from videos in training data in order to perform data augmentation. In the experiments, we try different configurations of the data augmentation as described in Section IV. Here, we explain a common procedure among the configurations. First, we select a temporal position in a video by uniform sampling in order to generate a training sample. A 16-frame clip is then generated around the selected temporal position. If the video is shorter than 16 frames, then we loop it as many times as necessary. Next, we select a spatial position by the method that depends on the used data augmentation configuration. The sample aspect ratio is 1 and the sample is spatio-temporally cropped at the positions, scale, and aspect ratio. We spatially resize the sample at 112×112 pixels. The size of each sample is 3 channels $\times$ 16 frames $\times$ 112 pixels $\times$ 112 pixels. Note that we used the sample size because of our experimental environments, though using larger sizes of the samples contributes to improve recognition accuracies [17] and each sample is horizontally flipped with 50% probability. We also perform normalization, which subtracts the mean values of Kinetics from the sample for each color channel and divides the values by standard deviations of Kinetics. All generated samples retain the same class labels as their original videos.

TABLE I: Network Architectures. The dimensions of output sizes are $T \times Y \times X$, and the sizes are calculated based on a $16 \times 112 \times 112$-input. We represent $x \times x \times x, F$ as the kernel size, and the number of feature maps of the convolutional filter are $x \times x \times x$ and F, respectively. Each convolutional layer is followed by batch normalization [21] and a ReLU [22]. Spatio-temporal down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of two. A max-pooling layer (stride 2) is also located before conv2_x for down-sampling. In addition, conv1 spatially down-samples inputs with a spatial stride of two. C-d fc is a C-dimensional fully-connected layer, where C is the number of classes.

Layer Name	Output Size	Architecture
		18-layer
conv1	$16 \times 64 \times 64$	$7 \times 7 \times 7, 64$, stride 1 (T), 2 (XY)
conv2	$8 \times 32 \times 32$	$3 \times 3 \times 3$ max pool, stride 2 $\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 2$
conv3	$4 \times 16 \times 16$	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 2$
conv4	$2 \times 8 \times 8$	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 2$
conv5	$1 \times 4 \times 4$	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 2$
		average pool, C-d fc, softmax

In our training, we use cross-entropy losses and back-propagate their gradients. The training parameters include a batch size of 128, weight decay of 0.001, and 0.9 for momentum. When training the networks from scratch, we start from learning rate 0.03, and divide it by 10 after the validation loss does not improve during 10 epochs. Training is done for 250 epochs. When performing fine-tuning, we start from a learning rate of 0.0003 and decrease it by the same procedure. Training of fine-tuning is done for 100 epochs.

2) Recognition: We recognize actions in videos using the trained model. We adopt the sliding window manner to generate input clips, (i.e. videos are split into non-overlapped 16 frame clips.) Each clip is cropped around a center position with the maximum scale (i.e. the sample width and height are the same as the short side length of the frame). We estimate class probabilities of each clip using the trained model, and average them to recognize actions in videos.

IV. Experimental Configuration

We try different configurations of the data augmentation and fine-tuning, and compare accuracies of them in the experiments. In this section, we describe the details.

A. Data Augmentation

We try the following data augmentation configurations. The configurations are also shown in Figure 1.

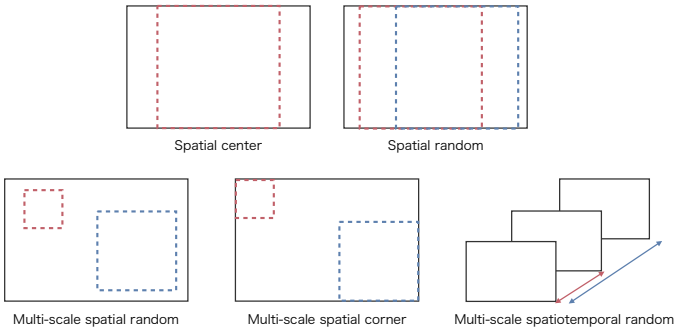


Fig. 1: Data augmentation configurations.

- *Spatial center*: We fix the spatial position of each sample in the center position. The spatial scale is also fixed in the maximum scale, which is the same size as the short side length of the frame. Thus, this configuration does not include spatial augmentation except for horizontal flipping, and perform only random temporal cropping.
- *Spatial random*: We randomly select the spatial position of each sample by uniform sampling. The position is sampled so that the cropping window is within a frame. The spatial scale is fixed in the maximum scale similar to *spatial center*.
- *Multi-scale spatial random*: We randomly select the spatial position and scale of each sample. The spatial position is selected by the same procedure as *spatial random*. The scale is randomly selected from $\left\{1, \frac{1}{2^{1/4}}, \frac{1}{\sqrt{2}}, \frac{1}{2^{3/4}}, \frac{1}{2}\right\}$. Scale 1 means that the sample width and height are the same as the short side length of the frame, and scale 0.5 means that the sample is half the size of the short side length.
- *Multi-scale spatial corner*: Different from uniform sampling of *multi-scale spatial crop*, we randomly select a spatial position from the 4 corners and a center. This procedure is proposed in [11] for two-stream 2D CNNs. The spatial scale is randomly selected similar to *multi-scale spatial random*.
- *Multi-scale spatiotemporal random*: Spatial augmentation of this configuration is the same as *multi-scale spatial crop*. In addition, we select temporal scale from $\left\{3, 2, 1, \frac{1}{2}, \frac{1}{3}\right\}$. To temporally crop based on the selected scale, we first select $s_t \times 16$-frame clip, where s_t is the temporal scale. We then perform temporal scaling for the clip by nearest neighbor scaling. For instance, scale 2 means that we first select a $16 \times 2 = 32$-frame clip, and then drop every three frames, and scale $\frac{1}{2}$ means that we first select a $16/2 = 8$-frame clip, and then make two copies of each frame. Therefore, the clip sizes of every scale are 16, but they contains different scale information.

B. Fine-tuning

In addition to data augmentation, we try different fine-tuning configurations and compare the results of them. Fine-tuning all layers would not be effective when using a small dataset. Therefore, we change the extents of fine-tuned layers. We freeze early layers and fine-tune layers above conv1, conv2, conv3, conv4, and conv5, as well as fine-tuning all layers.

V. Experiments

A. Dataset

In the experiments, we used the Kinetics [8], UCF-101 [1], and HMDB-51 [2] datasets. The Kinetics dataset has 400 human action classes, and consists of more than 400 videos for each class. The videos were temporally trimmed and last around 10 seconds. The total number of the videos is in excess of 300,000. The number of training, validation, and testing sets are about 240,000, 20,000, and 40,000, respectively.

UCF-101 includes 13,320 action instances from 101 human action classes. The videos were temporally trimmed to remove non-action frames. The average duration of each video is about 7 seconds. Three train/test splits (70% training and 30% testing) are provided in the dataset. Because of the relatively small-scale of UCF-101, we fine-tuned the Kinetics pretrained model on UCF-101 in the experiments.

HMDB-51 includes 6,766 videos from 51 human action classes. Similar to UCF-101, the videos were temporally trimmed. The average duration of each video is about 3 seconds. Three train/test splits (70% training and 30% testing) are provided in this dataset. We fine-tuned the Kinetics pretrained model on HMDB-51 similar to the experiments in UCF-101.

For all datasets, we resized the videos to 240 pixels height without changing their aspect ratios, and stored them.

B. Results

We began by training ResNet-18 on Kinetics with different data augmentation configurations. In this experiment, we trained ResNet-18 on the Kinetics training set and evaluated the model on the validation set.

Table II shows the accuracies of ResNet-18. We can see that *spatial random* achieved better accuracies than *spatial center*, and that *multi-scale spatial random* achieved the best accuracies. *Spatial center* only performs horizontal flipping and temporal random cropping whereas *spatial random* and *multi-scale spatial random* performs spatial random cropping with fixed and randomly selected scales, respectively. These results indicate that spatial random cropping improved recognition performance, and that multi-scale cropping further increased the accuracies. It is considered that adding spatial variations into training samples is important even for 3D CNNs.

The accuracies of *multi-scale spatial corner* were lower than *multi-scale spatial random*. This result indicate that the corner cropping strategy from 4 corners and a center is not effective for the training of 3D CNNs though its effectiveness for the training of two-stream 2D CNNs was shown in [11].

The accuracies of *multi-scale spatiotemporal random* also slightly lower than *multi-scale spatial random*. *multi-scale spatiotemporal random* adds multi-scale temporal cropping into *multi-scale spatial random*. This result indicate that the multi-scale temporal cropping decreased the accuracies even though the multi-scale spatial cropping improved recognition accuracies (comparison between *spatial random* and *multi-scale spatial random*). It is considered that nearest neighbor scaling of multi-scale temporal cropping makes the training samples artificial.

TABLE II: Comparisons of ResNet-18 with different data augmentation configurations in the Kinetics validation set. *Top-1*, *Top-5* means top-1 and top-5 accuracies, and *Average* are averaged accuracies over top-1 and top-5.

Configuration	Top-1	Top-5	Average
Spatial center	52.3	76.9	64.6
Spatial random	55.3	78.9	67.1
Multi-scale spatial random	**57.0**	**80.3**	**68.7**
Multi-scale spatial corner	53.2	77.2	65.2
Multi-scale spatiotemporal random	56.2	79.6	67.9

We next confirmed the results of fine-tuning with the different data augmentation configurations. We fine-tuned the Kinetics pretrained ResNet-18 on UCF-101 and HMDB-51. We used three splits of both datasets and show the results of them.

Table III shows the top-1 accuracies. We can see that *multi-scale spatial random* achieved the best accuracies in both datasets except for split 2 of UCF-101. We can also see that *spatial random* achieved the best averaged accuracies over three splits in UCF-101. These results indicate that the multi-scale spatial cropping often improve recognition accuracies but the single-scale spatial cropping is sufficient in some cases.

The accuracies of *Multi-scale spatial corner* and *multi-scale spatiotemporal random* were lower than those of *multi-scale spatial random* in most cases. These results support that the corner cropping and multi-scale temporal cropping are not effective even training in relatively small datasets.

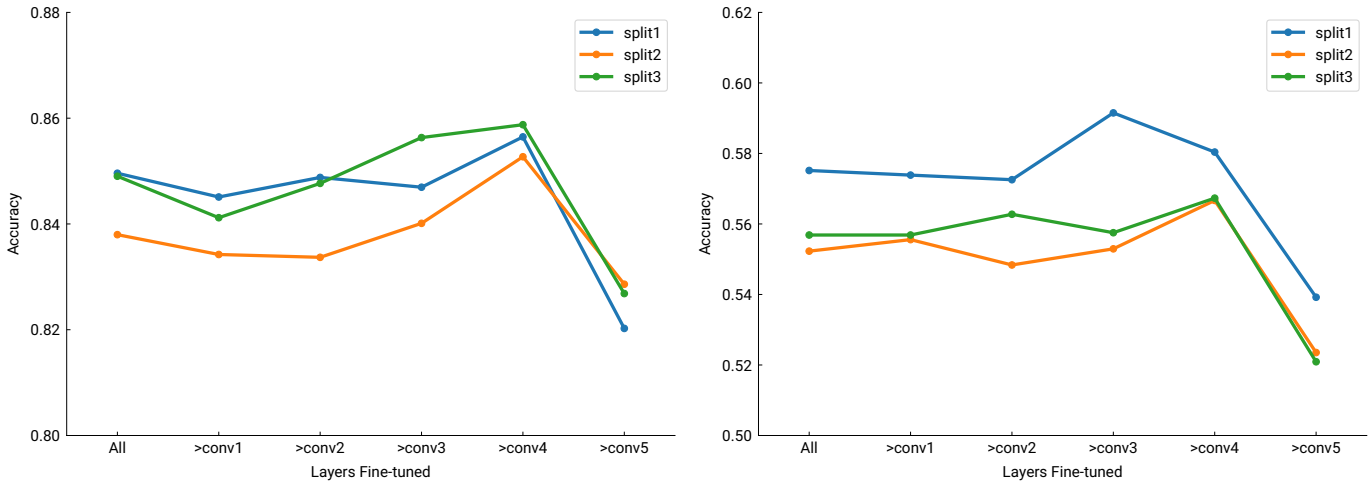
We also examine the fine-tuning configurations. In this experiment, we fine-tuned the Kinetics pretrained ResNet-18 on UCF-101 and HMDB-51 while changing the extents of the fine-tuned layers.

Figure 2 shows the accuracies of UCF-101 and HMDB-51. We can see that fine-tuning layers above conv4 achieved the best performance on both UCF-101 and HMDB-51 except for split 1 of HMDB-51. These results indicate that freezing early layers and fine-tuning later layers are effective for such relatively small datasets. In addition, it is considered that generalization ability of low-level features extracted by the early layers trained on Kinetics is high because the frozen features effectively worked on other datasets.

We finally show some recognition examples of Kinetics in Figure 3. The recognition results were output by ResNet-18 trained on the *multi-scale spatial random* configuration. Top three rows are correctly recognized examples. *Presenting weather forecast* and *bench pressing* are the classes that have the first and second best accuracies, respectively. These action classes clearly include a main person and object (a display and weight) Therefore, it is relatively easy to recognize these actions. *Salsa dancing* is the class that has the middle rank accuracy. Because Kinetics include various actions related to dances, such as *belly dancing*, *breakdancing*, and *robot dancing*, it is difficult to distinguish such actions compared with abovementioned classes. The bottom row of Figure 3

TABLE III: Comparisons with different data augmentation configurations in UCF-101 and HMDB-51. Fine-tuning is performed for all layers. *Average* means the averaged accuracies over all splits, and ± means that their standard deviations.

Configuration	UCF-101				HMDB-51			
	split 1	split 2	split 3	Average	split1	split 2	split 3	Average
Spatial center	82.4	82.9	81.8	82.4 ± 0.4	54.8	53.3	51.8	53.3 ± 1.2
Spatial random	84.5	**85.4**	84.5	**84.8** ± 0.4	56.6	54.2	55.2	55.3 ± 1.0
Multi-scale spatial random	**85.0**	83.8	**84.9**	84.6 ± 0.5	**57.5**	**55.2**	**55.7**	**56.1** ± 1.0
Multi-scale spatial corner	84.2	83.9	82.5	83.5 ± 0.7	56.3	52.1	54.6	54.3 ± 1.7
Multi-scale spatiotemporal random	84.4	83.2	83.1	83.6 ± 0.6	54.8	55.0	51.2	53.7 ± 1.8



(a) UCF-101.

(b) HMDB-51.

Fig. 2: Comparisons with different fine-tuning configurations. "*>convX*" means that all layers above convX are finetuned, and *All* means that the entire net is finetuned.

shows a misclassified example. *Slapping* is the class that has the worst accuracies. Because variation of this action class is very large, it is difficult to learn such ambiguous concept. In addition, a man woke other man up in this example video. Therefore, the misclassification to *baby waking up* is intuitively reasonable.

VI. Conclusion

In this study, we conducted various experiments with different training configurations on Kinetics, UCF-101, and HMDB-51 datasets to share the knowledge of 3D CNNs for the research community. According to the results of those experiments, the following conclusions could be obtained. (i) Data augmentation by spatiotemporal random cropping improved the performance levels. (ii) Data augmentation by multi-scale spatial cropping increased the accuracies in most cases whereas multi-scale temporal cropping decreased them. (iii) A corner cropping strategy, which is previously shown as a good method for two-stream 2D CNNs, resulted lower accuracies for 3D CNNs compared with simple random cropping. (iv) Freezing early layers of 3D CNNs improved the performance levels when fine-tuning 3D CNNs on a relatively small dataset.

In our future work, we will further investigate good practice for deeper models.

References

[1] K. Soomro, A. Roshan Zamir, and M. Shah, "UCF101: A dataset of 101 human action classes from videos in the wild," CRCV-TR-12-01, 2012.

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

[3] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 568–576.

[4] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the Kinetics dataset," *arXiv preprint*, vol. arXiv:1705.07750, 2017.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[6] 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 (CVPR)*, 2014, pp. 1725–1732.

[7] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "YouTube-8M: A large-scale video classification benchmark," *arXiv preprint*, vol. arXiv:1609.08675, 2016.

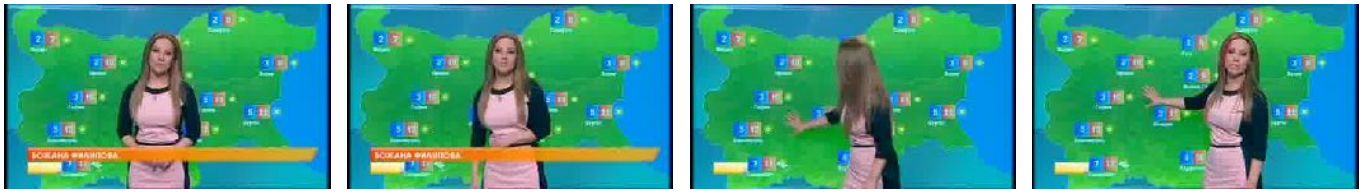


Fig. 3: Examples of recognition results on Kinetics using ResNet-18 trained by *multi-scale spatial random*.

[8] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The Kinetics human action video dataset," *arXiv preprint*, vol. arXiv:1705.06950, 2017.

[9] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and Imagenet?" *arXiv preprint*, vol. arXiv:1711.09577, 2017.

[10] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4305–4314.

[11] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, "Towards good practices for very deep two-stream convnets," *arXiv preprint*, vol. arXiv:1507.02159, 2015.

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

[13] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal residual networks for video action recognition," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 3468–3476.

[14] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1933–1941.

[16] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016, pp. 20–36.

[17] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. PP, no. 99, 2017.

[18] K. Hara, H. Kataoka, and Y. Satoh, "Learning spatio-temporal features with 3D residual networks for action recognition," in *Proceedings of the ICCV Workshop on Action, Gesture, and Emotion Recognition*, 2017.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[20] 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 (CVPR), 2016, pp. 770–778.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 448–4 56.

[22] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the International Conference on Machine Learning*. Omnipress, 2010, pp. 807–814.