

# Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?

Kensho Hara, Hirokatsu Kataoka, Yutaka Satoh

National Institute of Advanced Industrial Science and Technology (AIST)

Tsukuba, Ibaraki, Japan

{kensho.hara, hirokatsu.kataoka, yu.satou}@aist.go.jp

## Abstract

The purpose of this study is to determine whether current video datasets have sufficient data for training very deep convolutional neural networks (CNNs) with spatio-temporal three-dimensional (3D) kernels. Recently, the performance levels of 3D CNNs in the field of action recognition have improved significantly. However, to date, conventional research has only explored relatively shallow 3D architectures. We examine the architectures of various 3D CNNs from relatively shallow to very deep ones on current video datasets. Based on the results of those experiments, the following conclusions could be obtained: (i) ResNet-18 training resulted in significant overfitting for UCF-101, HMDB-51, and ActivityNet but not for Kinetics. (ii) The Kinetics dataset has sufficient data for training of deep 3D CNNs, and enables training of up to 152 ResNets layers, interestingly similar to 2D ResNets on ImageNet. ResNeXt-101 achieved 78.4% average accuracy on the Kinetics test set. (iii) Kinetics pre-trained simple 3D architectures outperforms complex 2D architectures, and the pretrained ResNeXt-101 achieved 94.5% and 70.2% on UCF-101 and HMDB-51, respectively.

The use of 2D CNNs trained on ImageNet has produced significant progress in various tasks in image. We believe that using deep 3D CNNs together with Kinetics will retrace the successful history of 2D CNNs and ImageNet, and stimulate advances in computer vision for videos. The codes and pretrained models used in this study are publicly available<sup>1</sup>.

## 1. Introduction

The use of large-scale datasets is extremely important when using deep convolutional neural networks (CNNs), which have massive parameter numbers, and the use of CNNs in the field of computer vision has expanded significantly in recent years. ImageNet [4], which includes more

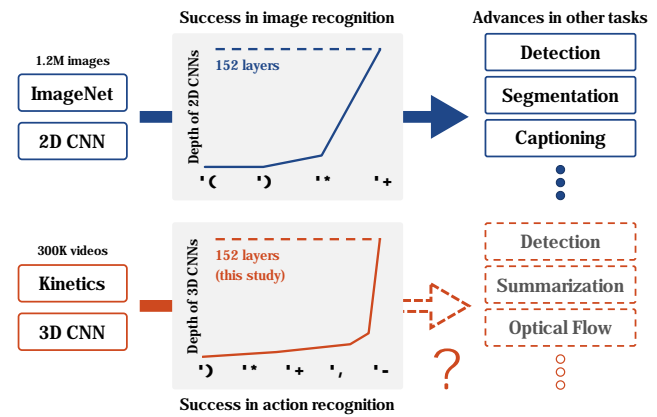


Figure 1: Recent advances in computer vision for images (top) and videos (bottom). The use of very deep 2D CNNs trained on ImageNet generates outstanding progress in image recognition as well as in various other tasks. Can the use of 3D CNNs trained on Kinetics generates similar progress in computer vision for videos?

than a million images, has contributed substantially to the creation of successful vision-based algorithms. In addition to such large-scale datasets, a large number of algorithms, such as residual learning [10], have been used to improve image classification performance by adding increased depth to CNNs, and the use of very deep CNNs trained on ImageNet have facilitated the acquisition of generic feature representation. Using such feature representation, in turn, has significantly improved the performance of several other tasks including object detection, semantic segmentation, and image captioning (see top row in Figure 1).

To date, the video datasets available for action recognition have been relatively small when compared with image recognition datasets. Representative video datasets, such as UCF-101 [21] and HMDB-51 [17], can be used to provide realistic videos with sizes around 10 K, but even though they are still used as standard benchmarks, such datasets are obviously too small to be used for optimizing CNN representations from scratch. In the last couple of years, ActivityNet [5], which

<sup>1</sup><https://github.com/kenshohara/3D-ResNets-PyTorch>

is a somewhat larger video dataset, has become available, and its use has made it possible to accomplish additional tasks such as untrimmed action classification and detection, but the number of action instances it contains is still limited. More recently, the Kinetics dataset [16] was created with the aim of being positioned as a de facto video dataset standard that is roughly equivalent to the position held by ImageNet in relation to image datasets. More than 300 K videos have been collected for the Kinetics dataset, which means that the scale of video datasets has begun to approach that of image datasets.

For action recognition, CNNs with spatio-temporal three-dimensional (3D) convolutional kernels (3D CNNs) are recently more effective than CNNs with two-dimensional (2D) kernels [2]. From several years ago [14], 3D CNNs are explored to provide an effective tool for accurate action recognition. However, even the usage of well-organized models [23, 25] has failed to overcome the advantages of 2D-based CNNs that combine both stacked flow and RGB images [20]. The primary reason for this failure has been the relatively small data-scale of video datasets that are available for optimizing the immense number of parameters in 3D CNNs, which are much larger than those of 2D CNNs. In addition, basically, 3D CNNs can only be trained on video datasets whereas 2D CNNs can be pretrained on ImageNet. Recently, however, Carreira and Zisserman achieved a significant breakthrough using the Kinetics dataset as well as the inflation of 2D kernels pretrained on ImageNet into 3D ones [2]. Thus, we now have the benefit of a sophisticated 3D convolution that can be engaged by the Kinetics dataset.

However, can 3D CNNs retrace the successful history of 2D CNNs and ImageNet? More specifically, can the use of 3D CNNs trained on Kinetics produces significant progress in action recognition and other various tasks? (See bottom row in Figure 1.) To achieve such progress, we consider that Kinetics for 3D CNNs should be as large-scale as ImageNet for 2D CNNs, though no previous work has examined enough about the scale of Kinetics. Conventional 3D CNN architectures trained on Kinetics are still relatively shallow (10 [16], 22 [2], and 34 [9, 24] layers). If using the Kinetics dataset enables very deep 3D CNNs at a level similar to ImageNet, which can train 152-layer 2D CNNs [10], that question could be answered in the affirmative.

In this study, we examine various 3D CNN architectures from relatively shallow to very deep ones using the Kinetics and other popular video datasets (UCF-101, HMDB-51, and ActivityNet) in order to provide us insights for answering the above question. The 3D CNN architectures tested in this study are based on residual networks (ResNets) [10] and their extended versions [11, 12, 30, 31] because they have simple and effective structures. Accordingly, using those datasets, we performed several experiments aimed at training and testing those architectures from scratch, as well

Figure 2: Averaged accuracies of 3D ResNets over top-1 and top-5 on the Kinetics validation set. Accuracy levels improve as network depths increase. The improvements continued until reaching the depth of 152. The accuracy of ResNet-200 is almost the same as that of ResNet-152. These results are similar to 2D ResNets on ImageNet [10].

as their fine-tuning. The results of those experiments (see Section 4 for details) show the Kinetics dataset can train 3D ResNet-152 from scratch to a level that is similar to the training accomplished by 2D ResNets on ImageNet, as shown in Figure 2. Based on those results, we will discuss the possibilities of future progress in action recognition and other video tasks.

To our best knowledge, this is the first work to focus on the training of very deep 3D CNNs from scratch for action recognition. Previous studies showed deeper 2D CNNs trained on ImageNet achieved better performance [10]. However, it is not trivial to show deeper 3D CNNs are better based on the previous studies because the data-scale of image datasets differs from that of video ones. The results of this study, which indicate deeper 3D CNNs are more effective, can be expected to facilitate further progress in computer vision for videos.

## 2. Related Work

### 2.1. Video Datasets

The HMDB-51 [17] and UCF-101 [21] datasets are currently the most successful in the field of action recognition. These datasets gained significant popularity in the early years of the field, and are still used as popular benchmarks. However, a recent consensus has emerged that indicates that they are simply not large enough for training deep CNNs from scratch [16].

A couple of years after the abovementioned datasets were introduced, larger video datasets were produced. These include ActivityNet [5], which contains 849 hours of video,

including 28,000 action instances. ActivityNet also provides some additional tasks, such as untrimmed classification and detection, but the number of action instances is still on the order of tens of thousands. This year (2017), in an effort to create a successful pretrained model, Kay et al. released the Kinetics dataset [16]. The Kinetics dataset includes more than 300,000 trimmed videos covering 400 categories. In order to determine whether it can train deeper 3D CNNs, we performed a number of experiments using these recent datasets, as well as the UCF-101 and HMDB-51 datasets.

Other huge datasets such as Sports-1M [15] and YouTube-8M [1] have been proposed. Although these databases are larger than Kinetics, their annotations are slightly noisy and only video-level labels have been assigned. (In other words, they include frames that do not relate to target actions.) Such noise and the presence of unrelated frames have the potential to prevent these models from providing good training. In addition, with file sizes in excess of 10 TB, their scales are simply too large to allow them to be utilized easily. Because of these issues, we will refrain from discussing these datasets in this study.

## 2.2. Action Recognition Approaches

One of the popular approaches to CNN-based action recognition is the use of two-stream CNNs with 2D convolutional kernels. In their study, Simonyan et al. proposed a method that uses RGB and stacked optical flow frames as appearance and motion information, respectively [20], and showed that combining the two-streams has the ability to improve action recognition accuracy. Since that study, numerous methods based on the two-stream CNNs have been proposed to improve action recognition performance [6, 7, 8, 27, 28, 29].

Unlike the abovementioned approaches, we focused on CNNs with 3D convolutional kernels, which have recently begun to outperform 2D CNNs through the use of large-scale video datasets. These 3D CNNs are intuitively effective because such 3D convolution can be used to directly extract spatio-temporal features from raw videos. For example, Ji et al. proposed applying 3D convolution to extract spatio-temporal features from videos, while Tran et al. trained 3D CNNs, which they referred to as C3D, using the Sports-1M dataset [15]. Since that study, C3D has been seen as a de facto standard for 3D CNNs. They also experimentally found 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 [25]. Those authors also found that using optical flows as inputs to 3D CNNs resulted in a higher level of performance than can be obtained from RGB inputs, but that the best performance could be achieved by combining RGB and optical flows. Meanwhile, Kay et al. showed that the results of 3D CNNs trained from scratch on

their Kinetics dataset were comparable with the results of 2D CNNs pretrained on ImageNet, even though the results of 3D CNNs trained on the UCF101 and HMDB51 datasets were inferior to the 2D CNNs results. In still another study, Carreira et al. proposed inception [22] based 3D CNNs, which they referred to as I3D, and achieved state-of-the-art performance [2]. More recently, some works introduced ResNet architectures into 3D CNNs [9, 24], though they examined only relatively shallow ones.

## 3. Experimental configuration

### 3.1. Summary

In this study, in order to determine whether current video datasets have sufficient data for training of deep 3D CNNs, we conducted the three experiments described below using UCF-101 [21], HMDB-51 [17], ActivityNet [5], and Kinetics [16]. We first examined the training of relatively shallow 3D CNNs from scratch on each video dataset. According to previous works [9, 16], 3D CNNs trained on UCF-101, HMDB-51, and ActivityNet do not achieve high accuracy whereas ones trained on Kinetics work well. We try to reproduce such results to ascertain whether the datasets have sufficient data for deep 3D CNNs. Specifically, we used ResNet-18, which is the shallowest ResNet architecture, based on the assumption that if the ResNet-18 overfits when being trained on a dataset, that dataset is too small to be used for training deep 3D CNNs from scratch. See Section 4.1 for details.

We then conducted a separate experiment to determine whether the Kinetics dataset could train deeper 3D CNNs. A main point of this trial was to determine how deeply the datasets could train 3D CNNs. Therefore, we trained 3D ResNets on Kinetics while varying the model depth from 18 to 200. If Kinetics can train very deep CNNs, such as ResNet-152, which achieved the best performance in ResNets on ImageNet [10], we can be confident that they have sufficient data to train 3D CNNs. Therefore, the results of this experiment are expected to be very important for the future progress in action recognition and other video tasks. See Section 4.2 for details.

In the final experiment, we examined the fine-tuning of Kinetics pretrained 3D CNNs on UCF-101 and HMDB-51. Since pretraining on large-scale datasets is an effective way to achieve good performance levels on small datasets, we expect that the deep 3D ResNets pretrained on Kinetics would perform well on relatively small UCF-101 and HMDB-51. This experiment examines whether the transfer visual representations by deep 3D CNNs from one domain to another domain works effectively. See Section 4.3 for details.

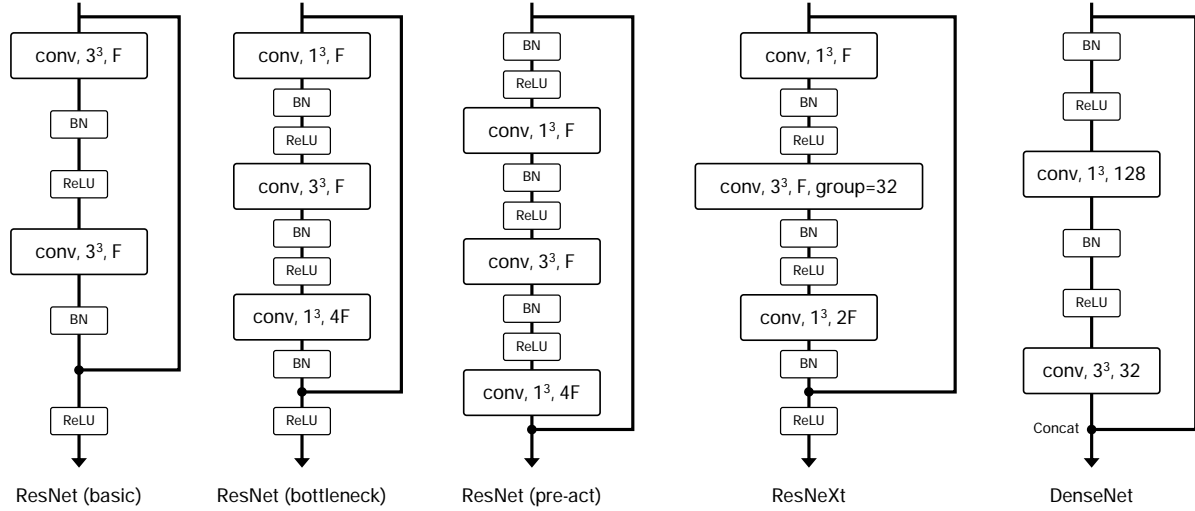


Figure 3: Block of each architecture. We represent conv,  $x^3$ ,  $F$  as the kernel size, and the number of feature maps of the convolutional filter are  $x \times x \times x$  and  $F$ , respectively, and group as the number of groups of group convolutions, which divide the feature maps into small groups. BN refers to batch normalization [13]. Shortcut connections of the architectures are summation except for those of DenseNet, which are concatenation.

Table 1: Network Architectures. Each convolutional layer is followed by batch normalization [13] and a ReLU [18]. Spatio-temporal down-sampling is performed by conv3\_1, conv4\_1, and conv5\_1 with a stride of two, except for DenseNet.  $F$  is the number of feature channels corresponding in Figure 3, and  $N$  is the number of blocks in each layer. DenseNet down-samples inputs using the transition layer, that consists of a  $3 \times 3 \times 3$  convolutional layer and a  $2 \times 2 \times 2$  average pooling layer with a stride of two, after conv2\_x, conv3\_x, and conv4\_x.  $F$  of DenseNet is the number of input feature channels of first block in each layer, and  $N$  is the same as that of the other networks. A  $3 \times 3 \times 3$  max-pooling layer (stride 2) is also located before conv2\_x of all networks for down-sampling. In addition, conv1 spatially down-samples inputs with a spatial stride of two.  $C$  of the fully-connected layer is the number of classes.

Model	Block	conv1	conv2_x		conv3_x		conv4_x		conv5_x	
			F	N	F	N	F	N	F	N
ResNet-{18, 34}	Basic		64	{2, 3}	128	{2, 4}	256	{2, 6}	512	{2, 3}
ResNet-{50, 101, 152, 200}	Bottleneck	conv, $7 \times 7 \times 7$ , temporal stride 1, spatial stride 2	64	3	128	{4, 4, 8, 24}	256	{6, 23, 36, 36}	512	3
Pre-act ResNet-200	Pre-act		64	3	128	24	36	512	3	
WRN-50	Bottleneck		128	3	256	4	6	1024	3	
ResNeXt-101	ResNeXt		128	3	256	24	36	1024	3	
DenseNet-{121, 201}	DenseNet		64	{6, 6}	128	{12, 12}	256	{24, 48}	{512, 896}	{16, 32}

global average pool,  
C-d fully-connected,  
softmax

### 3.2. Network architectures

Next, we explain the various ResNet-based architectures with 3D convolutions used in this study. 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. Unlike previous studies that examined only limited 3D ResNet architectures [9, 24], we examine not only deeper architectures but also some extended versions of ResNet. In particular, we explore the following architectures: ResNet (basic and bottleneck blocks) [10], pre-activation ResNet [11], wide ResNet (WRN) [31], ResNeXt [30], and DenseNet [12]. The architectures are summarized in Figure 3 and Table 1. In the following paragraphs, we will briefly introduce each



architecture.

A basic ResNets block consists of two convolutional layers, and each convolutional layer is followed by batch normalization and a ReLU. A shortcut pass connects the top of the block to the layer just before the last ReLU in the block. ResNet-18 and 34 adopt the basic blocks. We use identity connections and zero padding for the shortcuts of the basic blocks (type A in [10]) to avoid increasing the number of parameters of these relatively shallow networks.

A ResNets bottleneck block consists of three convolutional layers. The kernel sizes of the first and third convolutional layers are  $1 \times 1 \times 1$ , whereas those of the second are  $3 \times 3 \times 3$ . The shortcut pass of this block is the same as that of the basic block. ResNet-50, 101, 152, and 200 adopt the bottleneck. We use identity connections except for those that are used for increasing dimensions (type B in [10]).

The pre-activation ResNet is similar to bottleneck ResNet architectures, but there are differences in the convolution, batch normalization, and ReLU order. In ResNet, each convolutional layer is followed by batch normalization and a ReLU, whereas each batch normalization of the pre-activation ResNet is followed by the ReLU and a convolutional layer. A shortcut pass connects the top of the block to the layer just after the last convolutional layer in the block. In their study, He et al. showed that such pre-activation facilitates optimization in the training and reduces overfitting [11]. In this study, pre-activation ResNet-200 was evaluated.

The WRN architecture is the same as the ResNet (bottleneck), but there are differences in the number of feature maps for each convolutional layer. WRN increases the number of feature maps rather than the number of layers. Such wide architectures are efficient in parallel computing using GPUs [31]. In this study, we evaluate the WRN-50 using a widening factor of two.

ResNeXt introduces cardinality, which is a different dimension from deeper and wider. Unlike the original bottleneck block, the ResNeXt block introduces group convolutions, which divide the feature maps into small groups. Cardinality refers to the number of middle convolutional layer groups in the bottleneck block. In their study, Xie et al. showed that increasing the cardinality of 2D architectures is more effective than using wider or deeper ones [30]. In this study, we evaluate ResNeXt-101 using the cardinality of 32.

DenseNet makes connections from early layers to later layers by the use of a concatenation that is different from the ResNets summation. This concatenation connects each layer densely in a feed-forward fashion. DenseNets also adopt the pre-activation used in pre-activation ResNets. In their study, Huang et al. showed that it achieves better accuracy with fewer parameters than ResNets [12]. In this study, we evaluate DenseNet-121 and 201 using a growth rate of 32.

### 3.3 Implementation

**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. 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 randomly select a spatial position from the 4 corners or the center. In addition to the spatial position, we also select a spatial scale of the sample in order to perform multi-scale cropping. The procedure used is the same as [28]. The scale is selected from  $1, \frac{1}{2^{1/4}}, \frac{1}{2}, \frac{1}{2^{3/4}}, \frac{1}{2}$ . 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. 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  $112 \times 112$  pixels. The size of each sample is 3 channels  $\times$  16 frames  $\times$  112 pixels  $\times$  112 pixels, and each sample is horizontally flipped with 50% probability. We also perform mean subtraction, which means that we subtract the mean values of ActivityNet from the sample for each color channel. All generated samples retain the same class labels as their original videos.

In our training, we use cross-entropy losses and back-propagate their gradients. The training parameters include a weight decay of 0.001 and 0.9 for momentum. When training the networks from scratch, we start from learning rate 0.1, and divide it by 10 after the validation loss saturates. When performing fine tuning, we start from a learning rate of 0.001, and assign a weight decay of  $1e-5$ .

**Recognition.** We adopt the sliding window manner to generate input clips, (i.e., each video is split into non-overlapped 16-frame clips), and recognize actions in videos using the trained networks. Each clip is spatially cropped around a center position at scale 1. We then input each clip into the networks and estimate the clip class scores, which are averaged over all the clips of the video. The class that has the maximum score indicates the recognized class label.

### 3.4 Datasets

As stated above, this study focuses on four datasets: UCF-101 [21], HMDB-51 [17], ActivityNet [5], and Kinetics [16].

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.

HMDB-51 includes 6,766 videos from 51 human action classes. Similar to UCF-101, the videos were temporally

(a) UCF-101 (split 1).

(b) HMDB-51 (split 1).

(c) ActivityNet.

(d) Kinetics.

Figure 4: ResNet-18 training and validation losses. The validation losses on UCF-101, HMDB-51, and ActivityNet quickly converged to high values and were clearly higher than their corresponding training losses. The validation losses on Kinetics were slightly higher than the corresponding training losses, significantly different than those on the other datasets.

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.

ActivityNet (v1.3) provides samples from 200 human action classes with an average of 137 untrimmed videos per class and 1.41 activity instances per video. Unlike the other datasets, ActivityNet consists of untrimmed videos, which include non-action frames. The total video length is 849 hours, and the total number of action instances is 28,108. This dataset is randomly split into three different subsets: training, validation, and testing. More specifically, 50% is used for training, 25% is used for validation, and 25% is used for testing.

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.

The video properties of all these datasets are similar. Most videos were extracted from YouTube, except for HMDB-51, which includes videos extracted from movies. The videos include dynamic background and camera motions and the main differences among them are the numbers of action classes and instances.

We resized the videos to heights of 240 pixels without changing their aspect ratios and then stored them.

## 4. Results and discussion

### 4.1. Analyses of training on each dataset

We began by training ResNet-18 on each dataset. According to previous works [9, 16], 3D CNNs trained on UCF-101, HMDB-51, and ActivityNet do not achieve high accuracy whereas ones trained on Kinetics work well. We tried to reproduce such results in this experiment. In this process, we used split 1 of UCF-101 and HMDB-51, and the training and validation sets of ActivityNet and Kinetics.

Figure 4 shows the training and validation losses of ResNet-18 on each dataset. As can be seen in the figure, the validation losses on UCF-101, HMDB-51, and ActivityNet quickly converged to high values and were clearly higher than their corresponding training losses. These results indicate that overfitting resulted when the training on those three datasets. In addition to those losses, we confirmed per-clip accuracies, which are evaluated for each clip rather than for each video. The validation accuracies of UCF-101, HMDB-51, and ActivityNet are 40.1, 16.2, and 26.8%, respectively. It should be noted that direct comparisons between our results and those of previous studies would be unfair because the accuracies reported in most papers were per-video accuracies. However, since these accuracies are very low even compared with earlier methods [5, 26], our results indicate that it is difficult to train deep 3D CNNs from scratch on UCF-101, HMDB-51, and ActivityNet.

In contrast, the training and validation losses on Kinetics are significantly different than those on other datasets. Since the validation losses were slightly higher than the training losses, we could conclude that training ResNet-18 on Kinetics did not result in overfitting, and that it is possible for Kinetics to train deep 3D CNNs. In the next section, we will further investigate deeper 3D CNNs on Kinetics.

### 4.2. Analyses of deeper networks

Since the abovementioned experiment showed Kinetics could be used to train ResNet-18 without overfitting, we next examined deeper ResNets using the Kinetics training and validation sets.

Here, we will show ResNets accuracies changes based on model depths. Figure 2 shows the averaged accuracies over top-1 and top-5 ones. We can see that, essentially, as the depth increased, accuracies improved, and that the improvements continued until reaching the depth of 152. We can also see that deeper ResNet-152 achieved significant improvement of accuracies compared with ResNet-18, which was the previously examined architecture [9, 24]. In con-

trast, the accuracy of ResNet-200 was almost the same as that of ResNet-152. This result indicate that the training of ResNet-200 started to overfit. Interestingly, the results are similar to those for 2D ResNets on ImageNet [11]. More specifically, the accuracies of both 2D and 3D ResNets improved as the depth increased until reaching the depth of 152, and then the accuracies did not increase when increasing the depths of 200. These results indicate that the Kinetics dataset has sufficient data to train 3D CNNs in a manner similar to ImageNet.

Comparisons with other architectures are shown in Table 2. Here, it can be seen that the accuracies of pre-activation ResNet-200 are slightly low when compared with the standard ResNet-200 though He et al. reported that the pre-activation reduces overfitting and improves 2D ResNet-200 on ImageNet [11]. We can also see that the WRN-50 achieved higher accuracies when compared with the ResNet-152, which is similar to the results on ImageNet [31]. This result also supports that Kinetics is sufficient large for the training of 3D CNNs because the number of parameters of WRN-50 is larger than the ResNet-152. Furthermore, we can see that ResNeXt-101 achieved the best accuracies among the architectures tested. This result is also similar to that seen for ImageNet [30], and means that introducing the cardinality works well for the 3D ResNets on Kinetics. In contrast, the accuracies of the DenseNet-121 and 201 were slightly lower than the other architectures. The major advantage provide by dense connections is parameter efficiency, which contributes to reducing overfitting [12]. However, Kinetics did not need such techniques to train deep 3D CNNs.

Table 3 shows the results of the Kinetics test set used to compare ResNeXt-101, which achieved the highest accuracies, with the state-of-the-art methods. Here, it can be seen that the accuracies of ResNeXt-101 are clearly high compared with C3D with batch normalization [16], which is 10-layer network, as well as CNN+LSTM and two-stream CNN [16]. This result also indicates the effectiveness of deeper 3D networks trained on Kinetics. In contrast, RGB-I3D trained on Kinetics from scratch [3] were found to outperform ResNeXt-101 even though ResNeXt-101 is a deeper architecture than I3D. One of the reasons for this is the size differences of the network inputs. Specifically, the size of I3D is  $3 \times 64 \times 224 \times 224$ , whereas that of ResNeXt-101 is  $3 \times 16 \times 112 \times 112$ . Thus, I3D is 64 times larger than ResNeXt-101. To confirm the accuracies when using larger inputs, we also evaluated the ResNeXt-101 that used  $3 \times 64 \times 112 \times 112$  inputs, which are the largest available sizes in our environment (NVIDIA TITAN X  $\times 8$ ). We can see that the network, referred as ResNeXt-101 (64f) in Table 3, outperformed RGB-I3D even though the input size is still four times smaller than that of I3D. We can conclude that deeper 3D architectures trained on Kinetics are effective. In addition, it is felt that combining two-stream architec-

Table 2: Accuracies on the Kinetics validation set. Average is averaged accuracy over Top-1 and Top-5.

Method	Top-1	Top-5	Average
ResNet-18	54.2	78.1	66.1
ResNet-34	60.1	81.9	71.0
ResNet-50	61.3	83.1	72.2
ResNet-101	62.8	83.9	73.3
ResNet-152	63.0	<b>84.4</b>	<b>73.7</b>
ResNet-200	<b>63.1</b>	<b>84.4</b>	<b>73.7</b>
ResNet-200 (pre-act)	63.0	83.7	73.4
Wide ResNet-50	64.1	85.3	74.7
ResNeXt-101	<b>65.1</b>	<b>85.7</b>	<b>75.4</b>
DenseNet-121	59.7	81.9	70.8
DenseNet-201	61.3	83.3	72.3

Table 3: Accuracies on the Kinetics test set. Average is averaged accuracy over Top-1 and Top-5. Here, we refer the results of RGB- and Two-stream I3D trained from scratch [3] for fair comparison.

Method	Top-1	Top-5	Average
ResNeXt-101	–	–	74.5
ResNeXt-101 (64f)	–	–	<b>78.4</b>
CNN+LSTM [16]	57.0	79.0	68.0
Two-stream CNN [16]	61.0	81.3	71.2
C3D w/ BN [16]	56.1	79.5	67.8
RGB-I3D [3]	68.4	88.0	78.2
Two-stream I3D [3]	71.6	90.0	<b>80.8</b>

tures with ResNeXt-101 make further improvements based on higher accuracies of two-stream I3D.

### 4.3. Analyses of fine-tuning

Finally, in this section we confirm the performance of fine-tuning. In the experiments above, we showed that Kinetics can train deep 3D CNNs from scratch, but that it is difficult to train such networks on other datasets. In this section, we fine-tuned the Kinetics pretrained 3D CNNs on UCF-101 and HMDB-51. The results of this experiment are important for determining whether the 3D CNNs are effective for other datasets. It should be noted that, in this experiment, fine-tuning was only performed to train conv5\_x and the fully connected layer because it achieved the best performance during the preliminary experiments.

Table 4 shows the accuracies of Kinetics pretrained 3D CNNs, as well as ResNet-18 trained from scratch, in UCF-

Table 4: Top-1 accuracies on UCF-101 and HMDB-51. All accuracies are averaged over three splits.

Method	UCF-101	HMDB-51
ResNet-18 (scratch)	42.4	17.1
ResNet-18	84.4	56.4
ResNet-34	87.7	59.1
ResNet-50	89.3	61.0
ResNet-101	88.9	61.7
ResNet-152	89.6	62.4
ResNet-200	89.6	63.5
DenseNet-121	87.6	59.6
ResNeXt-101	<b>90.7</b>	<b>63.8</b>

101 and HMDB-51. Here, it can be seen that Kinetics pretrained ResNet-18 clearly outperformed one trained from scratch. This result indicate that pretraining on Kinetics is effective on UCF-101 and HMDB-51. We can also see that the accuracies basically improved as the depth increased, similar to the results obtained on Kinetics. However, unlike the results on Kinetics, ResNet-200 also improved the accuracies in HMDB-51. Because, as described above, the fine-tuning in this experiment was only performed to train conv5\_x and the fully connected layer, the numbers of trained parameters were the same from ResNet-50 to ResNet-200. Therefore, the pretrained early layers, which work as feature extractors, relate to the differences of performance. These results indicate that feature representations of ResNet-200 would be suitable for HMDB-51 even though the 200-layer network might start to overfit on Kinetics.

Table 4 also shows that ResNeXt-101, which achieved the best performance on Kinetics, achieved the highest levels of performance on both datasets when compared with the other networks. The performance difference, however, is smaller than that of Kinetics. It is considered likely that this result also relates to the sizes of datasets. We then compared the results with DenseNet-121 because it is a parameter-efficient network, and thus might achieve better performance on small datasets. However, the DenseNet-121 results were lower than those of ResNet-50, thereby indicating that its greater efficiency did not contribute on fine-tuning of 3D CNNs.

We shows the results of our comparison with state-of-the-art methods in Table 5. Here, we can see that ResNeXt-101 achieved higher accuracies compared with C3D [23], P3D [19], two-stream CNN [20], and TDD [27]. Furthermore, we can also see that ResNeXt-101 (64f), which utilize larger inputs described in previous section, slightly outperformed ST Multiplier Net [7] and TSN [29], which utilize more complex two-stream architectures. We can also see

Table 5: Top-1 accuracies on UCF-101 and HMDB-51 comared with the state-of-the-art methods. All accuracies are averaged over three splits. Dim indicate the dimension of convolution kernel.

Method	Dim	UCF-101	HMDB-51
ResNeXt-101	3D	90.7	63.8
ResNeXt-101 (64f)	3D	<b>94.5</b>	<b>70.2</b>
C3D [23]	3D	82.3	–
P3D [19]	3D	88.6	–
Two-stream I3D [3]	3D	<b>98.0</b>	<b>80.7</b>
Two-stream CNN [20]	2D	88.0	59.4
TDD [27]	2D	90.3	63.2
ST Multiplier Net [7]	2D	94.2	68.9
TSN [29]	2D	94.2	69.4

that two-stream I3D [3], which utilizes simple two-stream 3D architectures pretrained on Kinetics, achieved the best accuracies. Based on these results, we can conclude that simple 3D architectures pretrained on Kinetics outperform complex 2D architectures. We believe that development of 3D CNNs rapidly grows and contributes to significant advances in video recognition and its related tasks.

## 5. Conclusion

In this study, we examined the architectures of various CNNs with spatio-temporal 3D convolutional kernels on current video datasets. Based on the results of those experiments, the following conclusions could be obtained: (i) ResNet-18 training resulted in significant overfitting for UCF-101, HMDB-51, and ActivityNet but not for Kinetics. (ii) The Kinetics dataset has sufficient data for training of deep 3D CNNs, and enables training of up to 152 ResNets layers, interestingly similar to 2D ResNets on ImageNet. (iii) Kinetics pretrained simple 3D architectures outperforms complex 2D architectures on UCF-101 and HMDB-51, and the pretrained ResNeXt-101 achieved 94.5% and 70.2% on UCF-101 and HMDB-51, respectively.

We believe that the results of this study will facilitate further advances in video recognition and its related tasks. Following the significant advances in image recognition made by 2D CNNs and ImageNet, pretrained 2D CNNs on ImageNet experienced significant progress in various tasks such as object detection, semantic segmentation, and image captioning. It is felt that, similar to these, 3D CNNs and Kinetics have the potential to contribute to significant progress in fields related to various video tasks such as action detection, video summarization, and optical flow estimation. In our future work, we will investigate transfer learning not only for action recognition but also for other such tasks.



## References

- [1] S. Abu-El-Hajja, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. YouTube-8M: A large-scale video classification benchmark. *arXiv preprint, arXiv:1609.08675*, 2016. **3**
- [2] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017. **2, 3**
- [3] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. *arXiv preprint, arXiv:1705.07750*, 2017. **7, 8**
- [4] 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. **1**
- [5] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, 2015. **1, 2, 3, 5, 6**
- [6] 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)*, pages 3468–3476, 2016. **3**
- [7] 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. **3, 8**
- [8] 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)*, pages 1933–1941, 2016. **3**
- [9] 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. **2, 3, 4, 6**
- [10] 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)*, pages 770–778, 2016. **1, 2, 3, 4, 5**
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 630–645, 2016. **2, 4, 5, 7**
- [12] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017. **2, 4, 5, 7**
- [13] 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*, pages 448–456, 2015. **4**
- [14] 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*, 35(1):221–231, 2013. **2**
- [15] 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)*, pages 1725–1732, 2014. **3**
- [16] 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, arXiv:1705.06950*, 2017. **2, 3, 5, 6, 7**
- [17] 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)*, pages 2556–2563, 2011. **1, 2, 3, 5**
- [18] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning*, pages 807–814. Omnipress, 2010. **4**
- [19] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017. **8**
- [20] 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)*, pages 568–576, 2014. **2, 3, 8**
- [21] 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. **1, 2, 3, 5**
- [22] 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)*, pages 1–9, 2015. **3**
- [23] 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)*, pages 4489–4497, 2015. **2, 8**
- [24] D. Tran, J. Ray, Z. Shou, S. Chang, and M. Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint, arXiv:1708.05038*, 2017. **2, 3, 4, 6**
- [25] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *arXiv preprint, arXiv:1604.04494*, 2016. **2, 3**
- [26] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1):60–79, 2013. **6**
- [27] 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)*, pages 4305–4314, 2015. **3, 8**
- [28] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint, arXiv:1507.02159*, 2015. **3, 5**
- [29] 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), pages 20–36, 2016. 3, 8

- [30] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1492–1500, 2017. 2, 4, 5, 7
- [31] S. Zagoruyko and N. Komodakis. Wide residual networks. In Proceedings of the British Machine Vision Conference, 2016. 2, 4, 5, 7