Spatiotemporal Initialization for 3D CNNs with Generated Motion Patterns

Hirokatsu Kataoka, Kensho Hara, Ryusuke Hayashi
National Institute of Advanced Industrial Science and Technology (AIST)
{hirokatsu.kataoka, kensho.hara, r-hayashi}@aist.go.jp

Eisuke Yamagata, Nakamasa Inoue
Tokyo Institute of Technology
eisukeyamagata@gmail.com, inoue@c.titech.ac.jp

Abstract

The paper proposes a framework of Formula-Driven Supervised Learning (FDSL) for spatiotemporal initialization. Our FDSL approach enables to automatically and simultaneously generate motion patterns and their video labels with a simple formula which is based on Perlin noise. We designed a dataset of generated motion patterns adequate for the 3D CNNs to learn a better basis set of natural videos. The constructed Video Perlin Noise (VPN) dataset can be applied to initialize a model before pre-training with large-scale video datasets such as Kinetics-400/700, to enhance target task performance. Our spatiotemporal initialization with VPN dataset (VPN initialization) outperforms the previous initialization method with the inflated 3D ConvNet (I3D) using 2D ImageNet dataset. Our proposed method increased the top-1 video-level accuracy of Kinetics-400 pre-trained model on {Kinetics-400, UCF-101, HMDB-51, ActivityNet} datasets. Especially, the proposed method increased the performance rate of Kinetics-400 pre-trained model by 10.3 pt on ActivityNet. We also report that the relative performance improvements from the baseline are greater in 3D CNNs rather than other models. Our VPN initialization mainly helps to enhance the performance in spatiotemporal 3D kernels. The datasets, codes and pre-trained models used in this study will be publicly available[1].

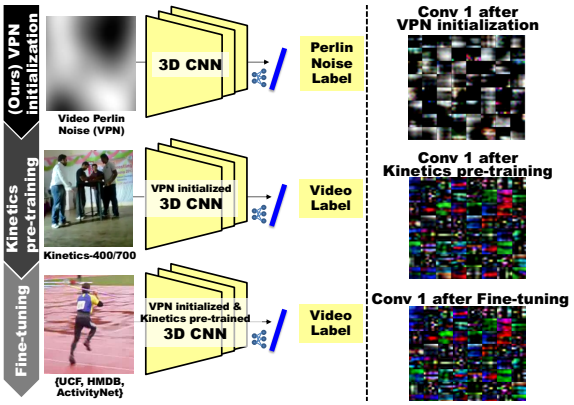
1. Introduction

In recent years, although the use of deep neural networks has become the mainstream of video recognition research, there are two major different options available for network architecture, namely, 2D and 3D CNNs, which are the methods for processing either only a space (x, y) volume or both a space and temporal (t) volume with a convolutional kernel. Initially, Two-stream ConvNets [30] were the reliable method and 2D CNNs with optical flows surpassed the methods with spatiotemporal 3D CNNs (e.g., [15, 32]).

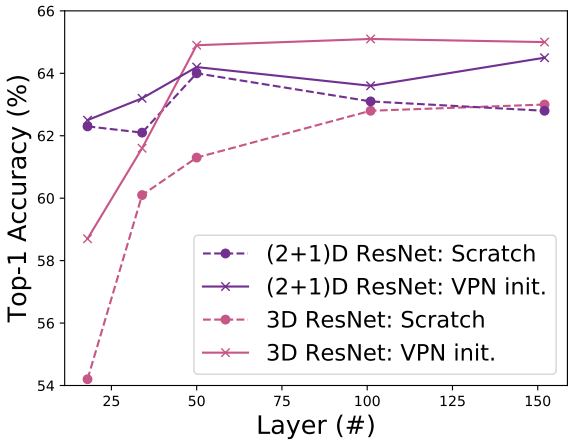
One of the main reasons why the early 3D CNNs were not able to perform better than 2D CNNs is that 3D CNNs have more parameters to be optimized and proper pre-training using a large-scale video dataset is critical to achieve meaningful learning. Indeed, even 2D CNNs, such as two-stream ConvNets were pre-trained with ImageNet dataset [7] followed by the transfer learning for video recognition. In the context of network initialization in spatiotemporal 3D CNNs, Inflated 3D ConvNet (I3D) has been proposed to improve the video recognition performance [5]. I3D employs network initialization with the parameters of 2D CNNs pre-trained using ImageNet; the parameters of 2D kernels are iteratively copied in temporal order for initializing 3D kernels. Thus a spatiotemporal initialization dedicated to 3D CNNs has not been considered thoroughly yet.

We also consider that designing a pre-training dataset that make the 3D CNNs acquire a feature representation relevant for motion perception in a biological brain would lead to better performance in video recognition tasks. It is known that visual images on the retina are encoded as electrophysiological activities of neurons and projected first to area V1, located in the posterior part of the brain [14]. Then visual information is hierarchically processed along the dorsal visual pathway including areas MT and MST for visual motion perception and action recognition [10, 34]. The neurons related to motion processing have receptive-field properties, such as orientation selectivity for stripes/bars or direction selectivity for moving visual stimuli, which can be characterized as localized oriented bandpass filters, comparable with the basis functions of Gabor wavelet transforms [1, 2]. Since these receptive-field properties are accounted for as being the results of sparse coding in response

[1] https://hirokatsukataoka16.github.io/Spatiotemporal-Initialization-for-3DCNNs/



(a) The flow of the proposed network training procedure. Video Perlin Noise(VPN) dataset consists of dynamic random textures determined only by frequencies in space and time dimensions. The dataset predispose a CNN to classify motion categories defined by the frequency characteristics of the noises (VPN initialization). The feature representation through the VPN initialization and Kinetics pre-training shows a more well-organized structure and clearer modulation in spatiotemporal dimensions. In fact, the video representation with VPN→Kinetics-400 is changed from those with the simple Kinetics-400 pre-training and acquires direction selectivity (see also Figure 5 and supplementary video).



(b) 3D and (2+1)D CNNs performances on Kinetics-400 recognition as function of #layer on Kinetics-400. The VPN initialization relatively enhances the performance of 3D ResNets more compared with those of (2+1)D ResNets. The accuracy gap is up to +4.5 pt.

Figure 1. The effects of VPN initialization.

to natural images [25], it is reasonable to think that the motion patterns made of naturalistic textures and temporal dynamics enable efficient initialization/pre-training for 3D CNNs.

One more important direction in computer vision must be Formula-Driven Supervised Learning (FDSL). The FDSL framework is highly expected to replace self-supervised learning since this does not require any natural images taken by a sensor. Although the related work [16, 3] have treated how to train visual representations, we believe that the FDSL framework must be applied in video recognition. Especially, by considering the work [16], the dataset pre-trained by Perlin noise can be extended into a video

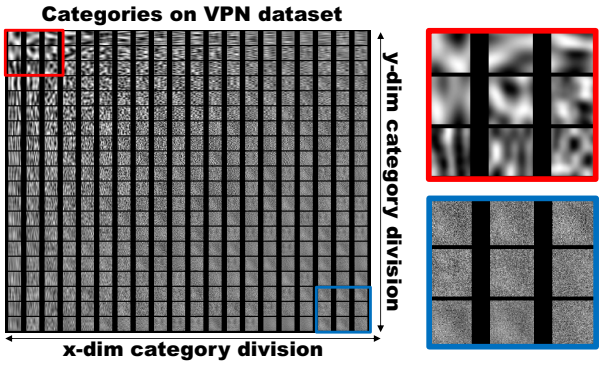


Figure 2. All categories in VPN dataset. The x and y coordinates correspond to the number of spatial divisions in (W, H).

dataset in perspective of its original usage [27, 28]. Moreover, as mentioned above, the rendering engine with Perlin noise allows us to generate similar motion patterns in terms of the mechanism of human brain, namely orientation selectivity and direction selectivity for moving visual stimuli.

In this paper, we use Perlin noise [27, 28] to generate frequency-defined dynamically changing textures to conduct training for initializing 3D CNNs. The advantages of using Perlin noise are that 1) the calculation cost for generating Perlin noise is low and the method is widely used in the computer graphics community; and 2) we can manipulate the spatiotemporal frequency of the Perlin noise to focus 3D CNNs on learning useful frequency characteristics such as orientation selectivity and direction selectivity. Here, we define the category of each motion pattern in the Video Perlin Noise (VPN) dataset based on its frequency in the (x, y, t) dimensions and conduct pretext training of the 3D CNNs to classify the defined categories to learn elemental features useful for motion processing in general[2].

The paper contributes to initializing spatiotemporal 3D CNNs by using the proposed synthetic movies (see also Figure 1(a)). Our network initialization method with the VPN dataset (see Figure 2) successfully improves the video-level classification accuracy by comparing to I3D initialization. Our method is also effective for various representative models with R(2+1)D [33] and 3D-ResNet [11] (see Figure 1(b)). According to our experimental results, the proposed method increases the top-1 video-level accuracy in the Kinetics-400 pre-training and in fine-tuning on {Kinetics-400 [18], UCF-101 [31], HMDB-51 [20], ActivityNet [13]} datasets (see Table 5). Especially in fine-tuning on ActivityNet, the accuracy is improved +10.3 pt by our method. The observed top-1 accuracies are all better than the performance of combined 3D-ResNet-50 and I3D [5][3].

[2]In the paper, we refer to the process as 'VPN initialization'.

[3]For a fair comparison, we have implemented ImageNet inflation on the 3D ResNet architecture since the original I3D was trained on InceptionV1 with many GPUs.

2. Related work

Spatiotemporal models. In the early approaches in video recognition, sparse [21, 23] and dense [35, 36] keypoint detection was used for spatiotemporal feature representations. After the rise of CNNs, video recognition studies had followed roughly two approaches: 2D (e.g., [30, 37]) and 3D CNNs (e.g., [5, 11, 32]). Then, recent video recognition technologies have shifted toward 3D CNNs. Specifically, 3D CNNs pre-trained with the Kinetics datasets [6, 18] allow successful transfer learning for many video recognition tasks. Furthermore, a method that initialize network parameters before pre-traing was proposed to improve the recognition performance. However, the previous method, I3D [5], merely used the parameters of 2D CNN trained with 2D images for the network initialization. Therefore, in order to exploit the advantages of using 3D kernels, we consider that a initialization method dedicated to learn a spatiotemporal representation should be applied before pre-training 3D CNNs on a large-scale video dataset.

Training with generated patterns. Recently, an interesting approach, FDSL was proposed [16] as an alternative framework to self-supervised learning. The paper discussed how to pre-train a 2D CNN without any natural images in order to prevent dataset-related problems such as offensive labels, privacy preservation, and annotation labor. The authors concluded that fractal geometry images [4, 22] are useful for pre-training a 2D CNN model compared to Perlin noises [27, 28] and Bezier curves [8].

Here, we make use of the idea that Perlin noise [27, 28] gives a better initial representation through pretext task training with spatiotemporal 3D CNNs. The proposed method enables us to easily generate various controllable motion patterns and their labels/categories for training. In what follows, we discuss how to construct the VPN dataset in terms of effective pretext task training for 3D CNNs to acquire adequate spatiotemporal frequency. Moreover, we compare our proposed method with video-extended FractalDB in the experimental section (see Table 10).

3. Video Perlin Noise (VPN) dataset

3.1. Overview

The proposed dataset is inspired by the knowledge of FDSL. Especially, we considered that the Perlin noise must be applied to construct video dataset as the original usage of animation generation. The proper initialization using a synthetic video dataset designed to effectively learn a visual representation must be useful for spatiotemporal 3D CNNs to improve the performance of video recognition.

Here, we use Perlin noise [27, 28] in order to generate motion patterns with minimal orientations and directions but various spatiotemporal frequencies based on a simple algorithm. Perlin noise is capable of generating natural moving textures based on the frequency only in the x, y, and t dimensions that are complex enough to give a proper initial representation for a network to further learn the target video recognition tasks.

In the network initialization, we introduce pretext task training to classify the categories of VPN dataset shown in Figure 2. Hereafter, we conduct Kinetics pre-training after the VPN initialization (VPN→Kinetics-400/700 (K4/K7)) for evaluation. Regarding the procedure see also Section 4 and Table 1 for more details.

3.2. Dataset construction with Perlin noise

Perlin noise. Perlin noise [27, 28] is random textured motion pattern that can be automatically generated from a set of parameters in the spatial (x, y) and temporal (t) dimensions through the following three steps. 1) We define a 3D grid of $W \times H \times D$ in terms of (x, y, t) dimensions. The 3D grid is initialized by random gradient vectors $v_{p,q,r}$ at each grid point, i.e., for $p = 0, 1, 2, \cdots, W$, $q = 0, 1, 2, \cdots, H$, and $r = 0, 1, 2, \cdots, D$. 2) Motion patterns are generated as gradients in the 3D grid. For each pixel, this step computes the dot product of the gradient vectors at the eight corners of the cube that the pixel belongs to and the distance vectors between the pixel and the corresponding corners. 3) Finally, the luminance value of each pixel is determined through linear interpolation between the eight values computed in the previous step.

How to divide a spatiotemporal category. The parameters in the Perlin noise constitute $\vec{f} = [W, H, D]$ and correspond to the frequencies of the motion patterns in the x, y, and t dimensions of the video clips. We assign one of 20 parameter values to each coordinate of (W, H) which correspond to the spatial (x, y) divisions in Figure 2: {1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 48, 60, 80, 120, 240}. The parameter (D) which corresponds to temporal dimension t is set to either {4, 8, 16}. The temporal parameter indicates the loop speed; $D = 16$ indicates 16 frames per motion loop. The parameter triplet of the video size is (240, 240, 240) in terms of (x, y, t). This corresponds to 240 temporal frames of 240×240-pixel images. We explore the settings of motion categories defined by these spatial and temporal parameters in Section 5.1 (ii).

In order to prevent overfitting to the Perlin noise pretext classification task and to increase the number of instances per category for training, we add together several randomly generated frequency noises of different frequencies to form a single video instance v,

$$v = \sum_{k=1}^{n} a_k Generator(\vec{f_k}), \tag{1}$$

where $Generator(\vec{f})$ is the function that generates the Per-

Table 1. Training procedures in the proposed and baseline methods. The dataset (task) associations are as follows: ImageNet (image classification), VPN (video classification in automatically generated videos), Kinetics-400 (video classification), {UCF-101, HMDB-51, ActivityNet} (video classification).

	K4	I3D→K4	VPN→K4
Initialization	–	ImageNet	VPN
Pre-training	Kinetics-400		
Fine-tuning	{UCF-101, HMDB-51, ActivityNet}		

lin noise based on frequencies $\vec{f}$ and a_k is the amplitude of noise $n_k = Generator(\vec{f_k})$. Here, $\vec{f_1}$ indicates the parameters for the primal Perlin noise and the other terms ($\vec{f_2}$, $\vec{f_3}$ ...) for the randomly generated frequency noises. These frequency noises are randomly generated by assigning parameter values of either {1, 2, 4, 8, 16} in the (x, y, t) dimensions and are overlapped with the primal Perlin noise. Adding the random frequency noises also serves as video instance augmentation. We set the amplitude parameter (a_k) as either {0.01, 0.02, 0.05, 0.1, 0.2}. Moreover, the number of overlapping random frequencies is set as 2 (i.e., we use $\vec{f_2}$ and $\vec{f_3}$ for the random frequency noises). We evaluate the effectiveness of adding the randomly generated frequency noises in Section 5.1 (iii).

The category of video v is defined by the parameters $[W, H, D]$ of f_1 in Eq. (2). To let the primal Perlin noise determine the category of each video instance and to scale the pixel values as ordinary grayscale images, a_k are constrained as follows:

$$\sum_{k=1}^{n} a_k = 1, (0 < a_k < a_1, (k = 2, 3, \cdots, n)) \quad (2)$$

4. Training settings

Training procedures. The datasets used in the experimental section can be roughly divided into pretext task training for network initialization, pre-training, and fine-tuning. As described in previous reports [11, 17], we employ Kinetics-400 (K4) as a main pre-training dataset. We only use Kinetics-700 (K7) as pre-training dataset in the final performance comparison part (see Section 5.3). We use UCF-101, HMDB-51, and ActivityNet as fine-tuning datasets for video recognition according to the standard evaluation method. We summarize the procedure of the proposed method (VPN→K4) and baseline methods (K4, I3D→K4) in Table 1. However, in consideration of computational resources, we first explored optimal dataset construction without Kinetics-400 pre-training, i.e. the procedure consists of **VPN**→{UCF-101, HMDB-51, ActivityNet} in Section 5.1.

Model architecture. In exploratory experiments for dataset configuration, we mainly use 3D-ResNet-50 since its architecture is the basic model in video recognition [11]. For a fair comparison between our VPN initialization and the conventional I3D, we implement the ImageNet inflation in the 3D-ResNet-50 architecture. We additionally employ and compare other 3D ResNet and (2+1)D ResNet architectures in Section 5.2 (iv). Finally, we further investigate larger video inputs such as 3D-ResNet-50 with 64 temporal frames and 224×224 pixels in Section 5.3.

Table 2. Effectiveness of VPN initialization.

	UCF-101	HMDB-51	ActivityNet
Scratch	33.9	16.9	20.3
VPN	**46.3**	**19.8**	**30.7**

Implementation details. An input image sequence consists of 112 pixels × 112 pixels × 3 channels × 16 frames obtained by cropping an original video sample. The 16-frame video clip is cropped from a random time position in the video. We apply ×10 data augmentation, i.e. images generated by adopting four-corner/center cropping and their horizontal flipping. In the training phase, we use stochastic gradient descent (SGD) and cross-entropy loss as an optimizer and loss function, respectively. The weight decay and momentum are set to 0.001 and 0.9, respectively. The learning rate starts at 0.3 and is then updated if the validation loss is saturated for 10 epochs in a row. The batch size for SGD is fixed as 128.

5. Evaluation

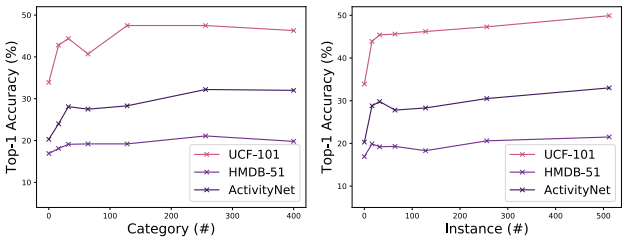
We evaluate the optimum settings of the categories in the VPN dataset. In Section 5.1, we first explore several configurations of VPN by changing the categories included for pretext task in a procedure of VPN→{UCF-101, HMDB-51, ActivityNet}. In Section 5.2, we report experiments conducted for initializing spatiotemporal 3D CNNs for further improvement of video recognition performance in a procedure of (VPN→K4→{UCF-101, HMDB-51, ActivityNet}).

5.1. Exploration of VPN configuration

We carried out a parameter exploration on the VPN. We conduct (i) comparisons against scratch training (Table 2), (ii) spatial and temporal #category (Figure 3 and Table 3), (iii) with and without randomly generated frequency noise (Table 4), and (iv) freezing layers.

(i) Comparison against scratch training. We confirm the effectiveness of the VPN initialization (400 categories, 200 instances per category), compared with the training from scratch in Table 2. The results clearly show how the performances increase with VPN initialization.

(ii) Spatial and temporal #category. We illustrate the relationship between several variations of the dataset with different spatial categories/instances and their performance rates in Figure 3. We investigated spatial divisions (W, H) up to 400 categories with {0, 16, 32, 64, 128, 256, 400}



(a) #Category (b) #Instance

Figure 3. Training accuracy as a function of the number of categories or instances on UCF-101, HMDB-51, and ActivityNet.

Table 3. The effect of temporal category settings.

	UCF-101	HMDB-51	ActivityNet
Slow	44.1	20.7	32.2
Middle	**49.9**	**21.5**	**33.0**
Fast	43.5	19.4	27.6
Full	46.5	20.2	29.3

Table 4. With or without randomly generated frequency noise.

	UCF-101	HMDB-51	ActivityNet
w/o frequency	**49.9**	21.5	33.0
w/ frequency	**49.9**	**23.0**	**33.9**

with 200 instances per category. Where 0 category indicates a training from scratch. Because of the construction method, we set #category up to 400. #category is determined from the 20 values {1, 2, ..., 120, 240} of spatial division in each of the x and y dimensions described in Section 3.2. We also examine the effect of varying #instances per category among {0, 16, 32, 64, 128, 256, 512} with the fixed 256 categories. Although there is a tendency of performance improvements as #category increase, 256 categories condition performs best. On the other hand, bigger #instances performs better than smaller #instances.

Next, we examine the effect of employing different temporal categories to be learned or not on the performance rate. Here, we divide the dataset into the temporal categories of {fast, middle, slow} which correspond to the values of {4, 8, 16} in the temporal dimension shown in Section 3.2. Table 3 shows the comparisons between three values + the combined temporal category variations (Full). We confirm that the involvement of Full temporal frequency variations gives no significant improvement even though #category is three times larger than in the baseline method (Middle).

Based on our experimental results, we used mainly a dataset of 256 categories, 512 instances per category, and a temporal parameter value of $D = 8$ in the initialization phase for further experiment in order to save computational time. Thus, we only change the spatial parameters (W, H) and keep temporal parameter D fixed hereafter.

(iii) With and without randomly generated frequency noise. Training on the pretext task classifying the category of Perlin noise may cause overfitting that is not valuable for more general video recognition task. Therefore, we overlap randomly generated frequency noises onto the primal Perlin noises in the basic VPN dataset as described in Section 3.2. Table 4 shows top-1 video-level accuracies with and without randomly generated frequency noise. The results clearly show the effectiveness of adding randomly generated frequency noises. In the case of Perlin noise without randomly generated frequency noise, we don't have a sufficient number of instance per category. Adding the randomly generated frequency noises allows us to give diverse instances to highly improve the fine-tuning rate. The performance rates in top-1 video-level accuracy are increased by {+1.5, +0.9} on the datasets {HMDB-51, ActivityNet}.

(iv) Freezing layers. To check the feature representation through the VPN initialization, we freeze parameters at each convolutional layer. The video-level accuracy of the full fine-tuning and freezing layer 1–4 are respectively {49.9, 46.1, 47.2, 47.3, 47.0} on UCF-101, {23.0, 22.0, 21.8, 21.9, 21.9} on HMDB-51, and {33.9, 33.6, 33.7, 33.1, 32.1} on ActivityNet. The scores show that the trained parameters serve as a good network initialization and the earlier layers tend to obtain good representations for general video recognition tasks, although full fine-tuning tends to give the best score.

Summary of exploration. We showed 1) Initialization of spatiotemporal 3D CNNs with the VPN dataset is much better than training from scratch (see Table 2). 2) Initialization using wider varieties of spatial frequency categories than those of temporal frequency are more critical, which is consistent with the properties of human motion perception. We also found that 3) we can improve the performance rates by adding randomly generated frequency patterns to Perlin noise. 4) Earlier convolutional layers tend to acquire good feature representations by VPN initialization.

Comparisons to self-supervised video representation. The setting of this exploration study is similar to self-supervised learning in videos. Essentially, we initialize 3D CNNs without any natural videos. Though we don't include self-supervised learning into our method, the VPN initialization and additional training recorded the accuracy of 50.4 on UCF-101 (VPN→UCF-101). In Comparison between the 3D-ResNet-based methods from [19], our method is equivalent to 3D Auto-Encoder + future prediction (50.1) [40] and 3D inpainting (50.9) [26].

5.2. Network initialization + K4 pre-training

In the context of network initialization, I3D achieved great success with ImageNet inflation for improving pre-training on the Kinetics datasets. In order to confirm that a spatiotemporal initialization of 3D CNNs surpasses a 2D initialization with ImageNet pre-trained model, we compare the results of network initialization with **VPN**→Kinetics-400 (K4)→{UCF-101, HMDB-51, ActivityNet} and those

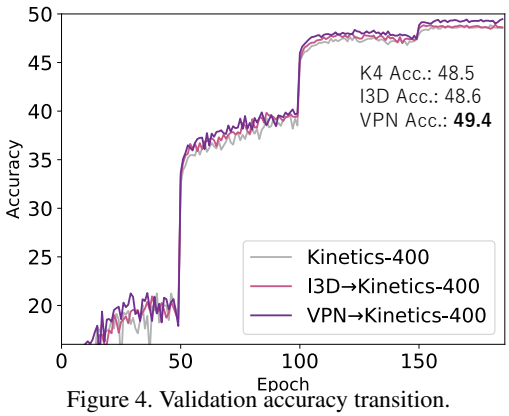


Figure 4. Validation accuracy transition.

of **I3D**→K4→{UCF-101, HMDB-51, ActivityNet}.

In this section, we show the experimental results with (i) accuracy transition (Figure 4), (ii) performance comparison between different network initializations (Table 5), (iii) high- and low-frequency VPN datasets (Table 6), and (iv) other architectures (Figure 1(b)).

(i) Accuracy transition. Figure 4 illustrates the accuracy transitions of the Kinetics-400 dataset training from scratch and training with ImageNet inflation and with our VPN initialization. Our VPN→K4 improves the final accuracy.

(ii) Performance comparison between different network initializations. We investigate the spatiotemporal initialization effects of 3D CNNs on video recognition performance. Table 5 lists the averaged top-1 accuracies in three trials[4] on Kinetics-400 (K4), UCF-101, HMDB-51, and ActivityNet (ANet). We conducted fine-tuning based on the Kinetics-400 pre-trained model on UCF-101, HMDB-51, and ActivityNet. The results in the table indicate that I3D and VPN initialized 3D-ResNet-50 improves the video-level accuracies compared with the scratch condition. Our method increases the scores by {+0.7, +0.6, +0.3, +1.2} relative to I3D→K4 and {+0.9, +4.8, +6.2, +10.3} relative to K4 on {Kinetics-400, UCF-101, HMDB-51, ActivityNet}.

(iii) High- and low-frequency (freq) datasets. According to the report [38], an accurate and robust CNN in image recognition tends to rely on low-spatial frequency features [38]. To test the frequency-dependency of our initialization method, we construct two subsets, high-freq and low-freq, from the VPN dataset. For the low- or high-freq dataset, we select 12 lowest or highest parameter values from the 20 spatial divisions, respectively. We adjust the #category for the initialization to 144, at both high- and low-freq settings. Table 6 compares the results of the high- and low-freq VPN datasets used in network initialization. The scores with the low-freq VPN dataset (64.9@K4) are higher than those with the high-freq VPN dataset (64.0@K4) and the full VPN dataset (64.4@K4 using 256 categories rather than 144 categories).

[4]The performance rates are the most important point in the paper, therefore, we calculated and showed averaged scores and variants in Table 5

Table 5. Comparison between Kinetics-400 (K4), I3D→K4, and VPN→K4 pre-trained models. ANet denotes ActivityNet dataset.

	K4	UCF-101	HMDB-51	ANet
K4	63.4±.11	85.1±.20	55.6±.26	62.3±.23
I3D	63.6±.15	89.3±.10	61.5±.65	71.4±.10
VPN	**64.3**±.05	**89.9**±.10	**61.8**±.20	**72.6**±.17

Table 6. High- and low-freq VPN datasets. We carry out classification of Kinetics-400 (K4) as well as fine-tuning classification on {UCF-101, HMDB-51, ActivityNet (ANet)} with the Kinetics-400 pre-trained model.

	K4	UCF-101	HMDB-51	ANet
High-freq	64.0	89.2	63.3	72.7
Low-freq	**64.9**	**90.2**	64.1	**73.3**
Full VPN	64.4	89.9	**64.2**	73.2

Table 7. Training from scratch (Scratch) and VPN initialization (VPN init.) with SlowFast [9].

	Scratch	VPN init.	Δ
SlowFast	**67.7**	67.2	-0.5
Slow-pathway	**65.1**	64.4	-0.7
Fast-pathway	48.9	**49.2**	+0.3

(iv) Performances in other architectures. The effects of initialization using the proposed dataset with other network architectures are shown in Figure 1(b). We use 3D-ResNet (R3D)-{18, 34, 50, 101, 152} and (2+1)D-ResNet-{18, 34, 50, 101, 152}. The results summarized in the table indicate that our initialization method with the VPN dataset is more effective than scratch training for all tested architectures. Though the 3D CNNs have more parameters than the (2+1)D CNNs, the VPN initialization and Kinetics-400 pre-training successfully improve the video recognition accuracies. The 3D-ResNet models are worse than the (2+1)D-CNN models in the case of training from scratch. Our spatiotemporal initialization allows us to improve the learning in 3D CNNs, especially with deeper layers. As the figure shows, fine-tuning accuracies of VPN initialized 3D-ResNet is better than those of (2+1)D conv.

Table 7 shows the comparisons between scratch training and VPN initialization with state-of-art models, i.e. SlowFast [9]. Two models in the table decreased the performance rates from scratch training when introducing VPN initialization. VPN initialized Fast-pathway in SlowFast, on the other hand, improved the score from scratch training by +0.3 pt. The architecture in Fast-pathway uses the 3D kernel in the first convolutional layer with $(7, 7, 5)$ which is similar to the kernel in 3D ResNet and biological brain, while SlowFast (Slow-pathway) employs the (x, y, t) kernels with $(3, 3, 1)$ in the first layer, thus practically not using 3D kernels. Therefore, it is considerable that the VPN initialization helps to train a relatively larger architecture with many 3D kernels like 3D ResNet rather than SlowFast.

Table 8. Comparison in architectures, pre-training datasets, and fine-tuning datasets. R3D indicates 3D ResNet. In the left column, we indicate the number of temporal frames (f) or spatial pixels (p) for an input if we utilize a larger video size, e.g., (64f, 224p), (128f, 256p).

Method	Pre-training	UCF-101	HMDB-51	ActivityNet	K4	K7
InceptionV1 (64f, 224p)	Kinetics-400	93.3	72.1	75.3	69.6	–
InceptionV1 (64f, 224p)	VPN→Kinetics-400	93.9	74.0	76.3	**70.4**	–
InceptionV1 (64f, 224p)	VPN→Kinetics-700	95.7	76.5	80.9	–	62.0
R3D-50	Kinetics-400	89.3	61.0	63.3	61.3	–
R3D-50	I3D→Kinetics-400	89.3	61.6	71.4	63.3	–
R3D-50	VPN→Kinetics-400	90.2	64.1	73.3	64.9	–
R3D-50 (64f, 224p)	VPN→Kinetics-400	93.8	71.0	75.1	69.5	–
R3D-50	Kinetics-700	92.0	66.0	75.9	–	54.7
R3D-50	I3D→Kinetics-700	92.1	66.7	76.8	–	55.9
R3D-50	VPN→Kinetics-700	92.8	68.7	77.8	–	56.5
R3D-50 (64f, 224p)	VPN→Kinetics-700	96.0	76.0	81.4	–	62.6
R3D-50 (128f, 256p)	VPN→Kinetics-700	**96.5**	**80.3**	**82.8**	–	**63.9**

Table 9. Comparison between VPN and Video Fractal (VF) as a pretext task training without K4 pre-training. The FractalDB proposed in [16] was upgraded to capture at every 1,000 points for video rendering.

	UCF-101	HMDB-51	ANet
VF	45.0	19.6	27.9
VPN	**49.9**	**23.0**	**33.9**

Table 10. Comparison between VPN and Video Fractal (VF). The FractalDB proposed in [16] was upgraded to capture at every 1,000 points for video rendering.

	K4	UCF-101	HMDB-51	ANet
I3D→K4	63.3	89.3	61.6	71.4
VF→K4	63.4	88.8	63.6	72.4
VPN→K4	**64.9**	**90.2**	**64.1**	**73.3**

5.3. Results using larger video input and dataset

We summarize the results in our experiments and list additional scores in Table 8. We show InceptionV1 and 3D-ResNet-50 (R3D-50) on K4/K7 pre-training. Although we cannot reproduce the results of InceptionV1 on I3D→K4 such as 95.6 (RGB input) on UCF-101 [5] due to the different computational resources, our VPN→K4 performed better than K4 baseline. In 3D-ResNet-50, we further implement a larger video input with 64 temporal frames and 224×224 pixels (64f, 224p), and K7 pre-training. We can confirm the improvements with VPN initialization for the larger video inputs with (64f, 224p) and (128f, 256p).

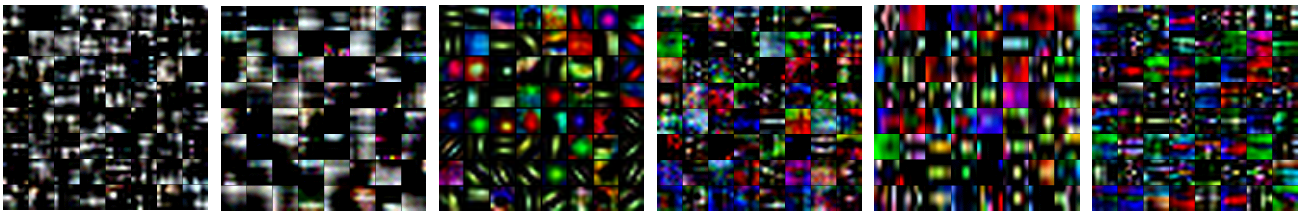
5.4. Additional experiments

Comparison of another FDSL method. We compare our VPN dataset to another FDSL method (FractalDB-1k dataset) in context of pretext task training with and without K4 pre-training. In order to expand 2D FractalDB to the dataset for 3D CNNs initialization, we upgraded the dataset as a video rendering engine (Video Fractal (VF) dataset). We record a checkpoint frame every 1,000 points rendered, of a total of 100k points. Therefore, each video contains 100 frames in temporal order. Following the original FractalDB-1k, we also set the number of categories/instances to 1k/1k, and generate a total of 1 million videos for network initialization. Tables 9 and 10 list the scores for using the VPN and VF datasets. The recognition accuracy with VF initialization is lower than VPN initialization and is comparable to the performance rates with I3D initialization in Table 10. Note that the VPN intialization achieved the rates with relatively fewer videos (28,800 videos in 144 categories and 200 instances) against to 1M videos in the VF initialization.

Visualization of first convolutional layer. Figure 5 illustrates the weight of each convolutional kernel at the first layer in 3D-ResNet-50 with various initialization and pre-training conditions. Figures 5(a) and 5(b) are convolutional filters initialized with high-freq and low-freq VPN datasets respectively. These panels indicate that the acquired weight pattern in the model trained with a low-spatial-freq VPN is coarser than those in the model trained with a high-spatial-freq VPN, as expected. In our experimental results, the low-freq VPN dataset performed well. Moreover, the three different initialization and pre-training combinations, Kinetics-400 (K4) (Figure 5(d)), I3D→K4 (Figure 5(e)), and VPN→K4 (Figure 5(f)) show different feature representations at the first convolutional layer. Though I3D→K4 maintains ImageNet-like feature representations (Figure 5(c)), our VPN→K4 inherits the K4 filters while showing a more well-organized structure and clearer modulation in both the space and time dimensions.

6. Conclusion and discussion

The paper proposes a network initialization method for improving the video recognition performance of spatiotemporal 3D CNNs. Our proposed initialization method with the Video Perlin Noise (VPN) dataset surpasses ImageNet inflation with I3D in several aspects. We designed the dataset containing motion patterns of various directions and randomly generated frequencies. According to our experimental results, our VPN dataset performs better when we



(a) VPN (High-freq) (b) VPN (Low-freq) (c) ImageNet (d) K4 (e) I3D→K4 (f) VPN→K4

Figure 5. Visualization of the first convolutional layer on 3D-ResNet-50.

apply low-frequency categories rather than high-frequency categories. Also, the performance rates with VPN initialization tend to be better than the rate with I3D initialization. We verified that the VPN initialization is also effective for a bigger video-clip input (e.g. 64/128 [frame], 224/256 [pixel]) and a larger pre-training datasets. Finally, the VPN initialized 3D-ResNet-50 achieved a top-level performance rate (96.5 on UCF-101, 80.3 on HMDB-51 in Table 8) without additional modalities like optical flow images.

According to the experimental results (e.g. Figure 1(b) and Table 7), the VPN initialization especially improve the performance of spatiotemporal models which consist of 3D kernels but not the other architectures including (2+1)D CNN and SlowFast, whose convolution kernels are separated in space and time dimensions. We can also visually confirm that the VPN initialization let the 3D CNN learn good spatiotemporal video representations for movie recognition tasks (see Figure 5 and supplementary video).

Our method is intended as an complementary framework to self-supervised learning (see 'comparison with self-supervised learning' in Section 5.1). In our method, we automatically generate videos and training labels (motion categories) as pretext task training datasets for network initialization, while ordinary self-supervised learning methods requires training datasets beforehand and need to define pretext task based on the datasets. It would be interesting to combine a method generating pretext task datasets as our approach with a self-supervised learning method to improve the performance of the neural networks mainly consists of 3D kernels in the future.

The properties of neurons in area V1 and other areas related to visual motion perception show selective activation in response to a certain orientation, direction or the specific integration/combination of the spatiotemporal frequency distribution [24, 29]. Thus, leveraging the initial state of the network toward the same elementary properties as observed in the human motion system would be beneficial for acquiring a feature representation useful for further complex video recognition. In the paper, we discovered the following findings in terms of biological brain:

Freezing layers (see Section 5.1 (iv)). These results are consistent with the findings of psychophysical experiments, i.e. temporal frequency tuning for human motion perception exhibits the same properties as band-pass filtering and the temporal frequency channels involved are estimated to be much fewer than the spatial frequency channels [39, 12].

Low- and high-freq (see Table 6). These results are consistent with our expectation that Perlin noise is adequate for 3D CNNs to learn a similar feature representation as area V1, which handles the initial stage of motion processing in the cortex.

Visualization (see Figure 5). VPN→K4 illustrates an well-organized structure and clearer modulation in spatiotemporal dimensions, showing similar orientation and direction selectivity as neurons in area V1.

Instance-wise, a single Perlin noise consists of motions patterns in various speed with horizontal or vertical direction, which, in turn, helps to investigate the property of the trained model in terms of spatiotemporal frequency in detail. To prevent overfitting to the Perlin noise classification task at the phase of learning initialization parameters, we add multiple noises of different frequencies to the generated Perlin noise. By training 3D CNNs to classify the category defined by the spatiotemporal frequency configuration of Perlin noise, the networks are expected to acquire a representation similar to the visual system of the human brain. Although VPN initialization is shown to be effective to initialize spatiotemporal 3D kernels, Perlin noise is limited in the variation of orientations (either horizontal or vertical axes) and is not useful to learn more complex motion features. We would like to explore an initialization method using more complex textures such as those generated by Gabor functions that describe better the property of the neurons in V1 [1, 2] in the future work.

Acknowledgement

This paper is based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This work was supported by JSPS KAKENHI Grant Number JP19H01134, 20H04597, 19H04200, and JST Moonshot RD JPMJMS2012. Computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used. We want to thank Seitaro Shinagawa and So Uchida for their helpful comments in research discussions.

References

[1] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *J Opt Soc Am A*, 2(284-299), 1985.

[2] Watson A. B. and Ahumada A. J. Jr. Model of human visual-motion sensing. *J Opt Soc Am A*, 2, 1985.

[3] Manel Baradad, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise, 2021.

[4] M. F. Barnsley. Fractals Everywhere. *Academic Press. New York*, 1988.

[5] A. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6299–6308, 2017.

[6] J. Carreira, E. Noland, C. Hillier, and A. Zisserman. A Short Note on the Kinetics-700 Human Action Dataset. In *arXiv pre-print arXiv:1907.06987*, 2019.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[8] G. Farin. Curves and surfaces for computer aided geometric design: A practical guide. *Academic Press*, 1993.

[9] C. Feichtenhofer, H. Fan, J. Malik, and K. He. SlowFast Networks for Video Recognition. In *International Conference on Computer Vision (ICCV)*, 2019.

[10] M. A. Goodale and A. D. Milner. Separate visual pathways for perception and action. *Trends Neurosci.*, 15(1), 1992.

[11] K. Hara, H. Kataoka, and Y. Satoh. Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet? In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6546–6555, 2018.

[12] R. Hayashi, Y. Sugita, S. Nishida, and K. Kawano. How motion signals are integrated across frequencies: Study on motion perception and ocular following responses using multiple-slit stimuli. *Journal of Neurophysiology*, 103(1), 2010.

[13] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, 2015.

[14] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *J Physiol*, 195, 1968.

[15] 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 (TPAMI)*, 35(1):221–231, 2013.

[16] H. Kataoka, K. Okayasu, A. Matsumoto, E. Yamagata, R. Yamada, N. Inoue, A. Nakamura, and Y. Satoh. Pre-training without Natural Images. In *Asian Conference on Computer Vision (ACCV)*, 2020.

[17] H. Kataoka, T. Wakamiya, K. Hara, and Y. Satoh. Would Mega-scale Datasets Further Enhance Spatiotemporal 3D CNNs? In *arXiv pre-print:2004.04968*, 2020.

[18] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, Suleyman M., and A. Zisserman. The Kinetics Human Action Video Dataset. In *arXiv pre-print arXiv:1705.06950*, 2017.

[19] D. Kim, D. Cho, and I. S. Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *AAAI*, 2019.

[20] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2556–2563, 2011.

[21] I. Laptev and T. Lindeberg. Space-time Interest Points. In *International Conference on Comuter Vision (ICCV)*, pages 432–439, 2003.

[22] B. Mandelbrot. The fractal geometry of nature. *American Journal of Physics*, 51(3), 1983.

[23] M. Marszałek, I. Laptev, and C. Schmid. Actions in Context. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2929–2936, 2009.

[24] Shinji Nishimoto and Jack L. Gallant. A three-dimensional spatiotemporal receptive field model explains responses of area mt neurons to naturalistic movies. *Journal of Neuroscience*, 31(41), 2011.

[25] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381, 1996.

[26] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[27] K. Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, 1985.

[28] K. Perlin. Improving noise. In *Computer Graphics and Interactive Techniques*, 2002.

[29] E. P. Simoncelli and D. J. Heeger. A model of neuronal responses in visual area mt. *Vision Res*, 38, 1998.

[30] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014.

[31] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild. CRCV-TR-12-01, 2012.

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *IEEE International Conference on Comuter Vision (ICCV)*, pages 4489–4497, 2015.

[33] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6459, 2018.

[34] L. G. Ungerleider and M. Mishkin. Two mechanisms of vision in primates. *Psychologische Forschung*, 31, 1982.

[35] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176, 2011.

[36] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *International Conference on Comuter Vision (ICCV)*, pages 3551–3558, 2013.

[37] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *European Conference on Computer Vision (ECCV)*, pages 20–36, 2016.

[38] Z. Wang, Y. Yang, A. Shrivastava, V. Rawal, and Z. Ding. Towards Frequency-Based Explanatin for Robust CNN. In *arXiv pre-print:2005.03141*, 2020.

[39] A. B. Watson and J. G. Robson. *Discrimination at threshold: Labelled detectors in human vision*, Vision Research(21), 1981.

[40] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, and X.-S. Hua. Spatio-temporal autoencoder for video anomaly detection. In *Multimedia Conference*, 2017.