論文 / 著書情報
Article / Book Information

Title	Initialization Using Perlin Noise for Training Networks with a Limited Amount of Data
Author	Nakamasa Inoue, Eisuke Yamagata, Hirokatsu Kataoka
Journal/Book name	Proceedings of ICPR 2020 25th International Conference on Pattern Recognition, , , pp. 1023-1028
Pub. date	2021, 5
DOI	https://doi.org/10.1109/ICPR48806.2021.9412955
Copyright	(c)2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Note	This file is author (final) version.

Initialization Using Perlin Noise for Training Networks with a Limited Amount of Data

Nakamasa Inoue[1*], Eisuke Yamagata[1*], Hirokatsu Kataoka[2]
[1]Tokyo Institute of Technology, Tokyo, Japan.
[2]National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan.
E-mail: inoue@c.titech.ac.jp, yamagata.e.ab@m.titech.ac.jp, hirokatsu.kataoka@aist.go.jp

***Abstract*—We propose a novel network initialization method using Perlin noise for training image classification networks with a limited amount of data. Our main idea is to initialize the network parameters by solving an artificial noise classification problem, where the aim is to classify Perlin noise samples into their noise categories. Specifically, the proposed method consists of two steps. First, it generates Perlin noise samples with category labels defined based on noise complexity. Second, it solves a classification problem, in which network parameters are optimized to classify the generated noise samples. This method produces a reasonable set of initial weights (filters) for image classification. To the best of our knowledge, this is the first work to initialize networks by solving an artificial optimization problem without using any real-world images. Our experiments show that the proposed method outperforms conventional initialization methods on four image classification datasets.**

I. Introduction

Image classification is one of the most important topics in the field of pattern recognition and computer vision, with wide applications such as internet search engines, robotics, and security. Over the last 10 years, many studies have shown the effectiveness of deep neural networks for various image classification tasks, including object recognition and action recognition. Most neural networks utilize a large-scale dataset for training because network performance increases with dataset size. For example, deep convolutional networks trained on the ImageNet dataset [1], which consists of 1.2 million images, have been shown to achieve human-level performance in terms of 1,000-class object classification accuracy.

Training networks with a limited amount of data is a challenging problem because network optimization methods often fall into a local solution if the network has many parameters. A recent trend in attempts to solve this problem is to apply fine-tuning, i.e., to resume training from a pre-trained network. For example, for object recognition, networks are often first trained on the ImageNet dataset, and then fine-tuned on another small dataset. This approach works well because ImageNet is large enough to obtain reasonable image representations with some visual filters in hidden layers. However, its application is often limited to non-commercial use such as research and educational purposes. In practice, it is not always easy to collect such a large number of images.

* indicates equal contribution.

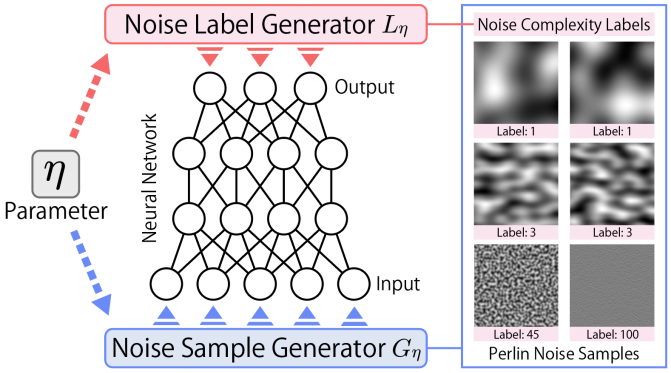

Fig. 1. **Network initialization using Perlin noise**. The proposed method consists of two components, namely a noise sample generator G_η for generating Perlin noise samples and a noise label generator L_η for generating noise category labels. A network is initialized by solving an artificial noise classification problem, where the aim is to classify the generated noise samples into their categories.

This paper proposes a network initialization method that obtains visual filters without using any real-world images. The proposed method utilizes Perlin noise for initializing networks. Perlin noise [2] is a type of noise used in the field of computer graphics to render natural objects such as clouds, fire, and stones. Specifically, the proposed method consists of two steps. First, it generates Perlin noise samples with category labels defined based on noise complexity. Second, it solves a classification problem, in which network parameters are optimized to classify the generated noise samples. Compared with conventional initialization methods such as He initialization [3] based on Gaussian noise, our method provides more complex texture-like visual filters as initial values. We experimentally show that the proposed method outperforms conventional initialization methods utilized in most state-of-the-art image classification systems. Further, we demonstrate that the proposed method contributes to data-efficient pre-training in terms of accuracy improvement using fewer samples for pre-training.

In summary, this paper makes the following contributions:

(1) A novel method is proposed for initializing networks, which solves an artificial classification problem without using real-world images. It consists of two components, namely a noise sample generator and a noise label

generator, as shown in Figure 1.
(2) Category definitions are given for Perlin noise samples based on noise complexity. This provides a specific definition of an artificial classification problem to be solved in the proposed method.
(3) Extensive experiments are conducted on four datasets. In addition to a performance comparison with conventional initialization methods, we show that the proposed method contributes to data-efficient pre-training.

II. Related Work

This section summarizes the network architectures, initialization methods, pre-training methods, and datasets discussed in related studies.

A. Network Architectures

Many previous studies have shown the effectiveness of convolutional neural networks for image classification, which results from their ability to learn visual patterns from a large number of images. The basic idea of convolution and pooling mechanisms was proposed in the 1990s [4]. Since then, various types of network architectures have been proposed. AlexNet [5], which has seven hidden layers, was the first successful application of large-scale training using natural images. VGGNet [6] and InceptionNet [7] stack more layers to explore deeper architectures. ResNet [8] introduces skip connections to avoid the gradient vanishing problem. ResNet also has extensions such as ResNeXt [9], DenseNet [10], and SE-ResNet [11], and is the most widely used architecture for image classification tasks such as object recognition [1], action recognition [12], and scene understanding [13].

To train these networks, a large number of images are needed because the networks typically have more than a million parameters to be optimized. Therefore, training with a limited amount of data is a challenging problem in the field of pattern recognition and computer vision.

B. Initialization Methods

In most state-of-the-art optimization methods, the parameters of neural networks are initialized independently by utilizing probabilistic distributions such as the Gaussian distribution and the uniform distribution.

He initialization [3] is the most popular method, and utilizes a Gaussian distribution with a mean of zero and a variance that is scaled depending on the number of hidden units to initialize the weight parameters at each layer. This method is effective for recent deep convolutional networks with ReLU activations [14]. Xavier initialization [15] utilizes a uniform distribution to initialize weights. This is particularly effective for networks with a smooth activation function such as the sigmoid or tangent functions. Sparse initialization [16] limits the number of non-zero initial weights. It was proposed with a Hessian-free optimization method. Other classic initialization methods include normal initialization, which uses a standard normal distribution, and zero initialization, which assigns zero to all weights. Zero initialization is a good choice for bias parameters or parameters that should be sparse.

Most of these methods can be viewed as initialization using Gaussian noise or uniform noise. In contrast, our idea is to utilize Perlin noise, a type of noise closer to natural visual patterns, for image classification.

C. Pre-Training Methods

Pre-training is a framework for initializing neural networks using a set of collected images. The idea is to first train a network on a large-scale dataset, and then adopt the resulting network as the starting point for training on another dataset.

A recent trend for pre-training is to utilize a large-scale dataset such as ImageNet [1], Places 365 [13], and Kinetics 400-700 [12], [17]. Because these datasets consist of more than a million images with high-quality human-annotated labels, networks pre-trained on them have a reasonable set of visual filters at some hidden layers for recognizing natural objects from images. This is one of the reasons why training from pre-trained parameters outperforms training from scratch in terms of image classification accuracy. However, the use of these datasets is often limited to research and academic purposes. Because it is not always easy to collect such a large number of images, data-efficient pre-training methods are desirable.

Our method is presented here as an initialization method because it does not use any natural images or any manually annotated labels. This also distinguishes our method from semi-supervised [18] [19] [20] [21] and unsupervised [22] learning. Additionally, it can also be viewed as a new type of pre-training method because it has a step that solves a classification problem. Notably, we show experimentally that the proposed method contributes to data-efficient pre-training.

D. Datasets

Various datasets have been created for evaluating image classification methods. Each dataset consists of a set of images and labels for a specific task, such as hand-written character recognition and object recognition.

Examples of hand-written character recognition datasets include MNIST [4] and Omniglot [23]. The MNIST dataset consists of 70,000 binary images of hand-written digits from 0 to 9. The Omniglot dataset consists of about 40,000 grayscale images of 1,623 different hand-written characters from 50 different alphabets. It was originally proposed as a dataset for one-shot learning to explore new learning methods.

Small- and large-scale object recognition datasets have been created. Small-scale datasets are often used for evaluating image classification methods. They include Cifar-10/100 [24], Caltech-101/256 [25], and Pascal VOC [26]. Each of these datasets consists of 10-100k natural images in 10-100 object categories. The Describable Textures Dataset (DTD) [27] focuses on texture categories related to objects. Large-scale datasets such as ImageNet [1] and COCO [28] are often used for pre-training. They consist of more than a million natural images.

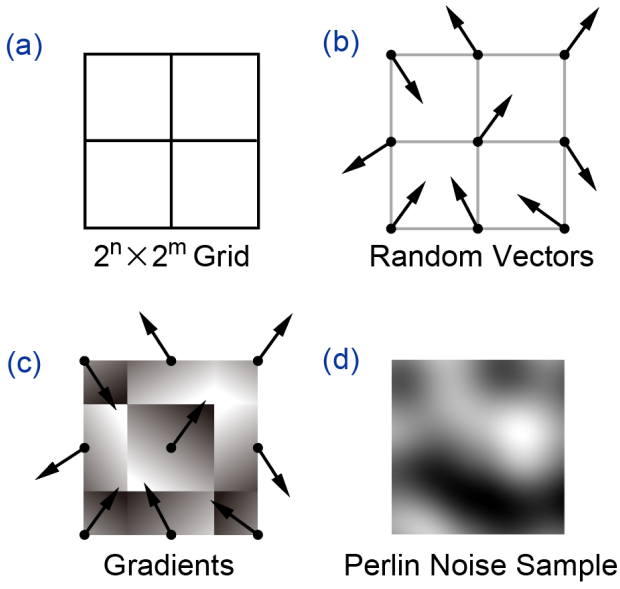

Fig. 2. Four steps used to generate Perlin noise. Noise samples are generated on a $2^n \times 2^m$ grid.

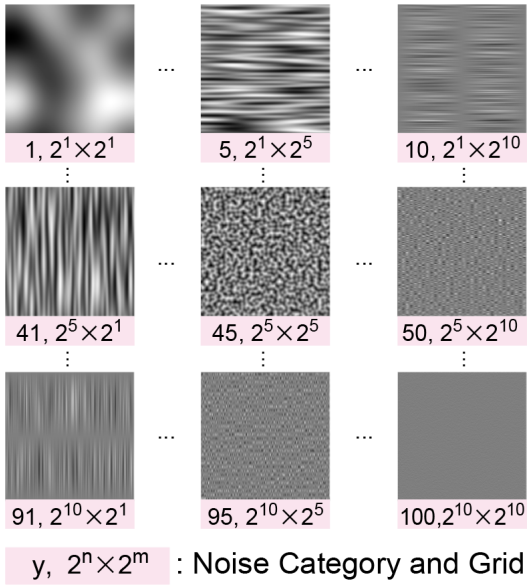

Fig. 3. Examples of noise categories. Labels attached to noise samples correspond to the complexity of Perlin noise.

III. Proposed Method

In this section, we present the proposed network initialization method that uses Perlin noise. Let $\mathcal{N}_\theta$ be a neural network for image classification with a set of parameters θ. The goal is to find reasonable initial values for θ.

Our main idea is to initialize the parameters by solving an artificial noise classification problem. Specifically, the proposed method consists of two steps. First, it generates noise data $\mathcal{D} = \{(\epsilon_i, y_i)\}_{i=1}^T$, where ϵ_i is a Perlin noise sample and y_i is a category label. Second, it solves a classification problem on $\mathcal{D}$, in which network parameters are optimized to maximize the noise classification accuracy. Note that even though this step can also be viewed as pre-training, the entire process is proposed as an initialization method because it does not use any natural images or any human-annotated labels. To the best of our knowledge, this is the first work to initialize networks by solving an artificial classification problem using noise. The rest of this section describes the details of each step.

A. Generation of Noise Data

This subsection describes the generation of noise data $\mathcal{D} = \{(\epsilon_i, y_i)\}_{i=1}^T$ in the first step of the proposed method. As shown in Figure 1, the proposed method has two components, namely a noise sample generator G_η and a noise label generator L_η with shared parameter η. Noise samples and their labels are generated as $\epsilon_i \sim G_\eta$ and $y_i \sim L_\eta$, respectively.

In this work, we utilize Perlin noise [2] to define these two generators. Perlin noise was selected for the following reasons: 1) It can render the textures of some natural objects. For example, in the field of computer graphics, it is used to render clouds, fire, and stones. 2) It has moderate complexity for defining categories (see the visualization of some noise categories in Figure 3). The definitions of the two generators using Perlin noise are given below.

A-1. Noise Sample Generator

Let W, H, and C be the width, height, and number of channels of inputs of the network $\mathcal{N}_\theta$ for image classification, respectively. The noise sample generator G_η generates $\epsilon_i \in \mathbb{R}^{W \times H \times C}$ by following the Perlin noise generation algorithm in [2], summarized below. Note that it has parameter $\eta = (n, m)$ in Step 1 and random values in Step 2.

Step 1: Definition of a grid. This step defines a two-dimensional grid on a blank image whose size is $W \times H$. We define a $2^n \times 2^m$ grid ($n, m \geq 1$) as shown in Figure 2 (a).

Step 2: Placement of random gradient vectors. This step places random gradient vectors $v_{p,q}$ at each grid point, i.e., for $p = 0, 1, 2, \cdots, 2^n$ and $q = 0, 1, 2, \cdots, 2^m$, as shown in Figure 2 (b). The magnitude and angle of each $v_{p,q}$ are uniformly sampled from $[0, R)$ and $[0, 2\pi)$, respectively. Here, we set $R = 0.01 \cdot \max(W, H)$.

Step 3: Generation of gradients. At each pixel, this step computes the dot product of the gradient vectors at the four corners of the cell the pixel belongs to and the distance vectors between the pixel and the corresponding corners. The four dot product values corresponding to the four corners are assigned to each pixel. This step is illustrated in Figure 2 (c). Note that the figure shows only one of the four values at each pixel.

Step 4: Interpolation. Finally, through linear interpolation between the four values computed in the previous step, the final value of each pixel on ϵ_i is determined. An example is shown in Figure 2 (d).

A-2. Noise Label Generator

In the above algorithm, the way that the interval gradient vectors are defined affects the complexity of the generated Per-

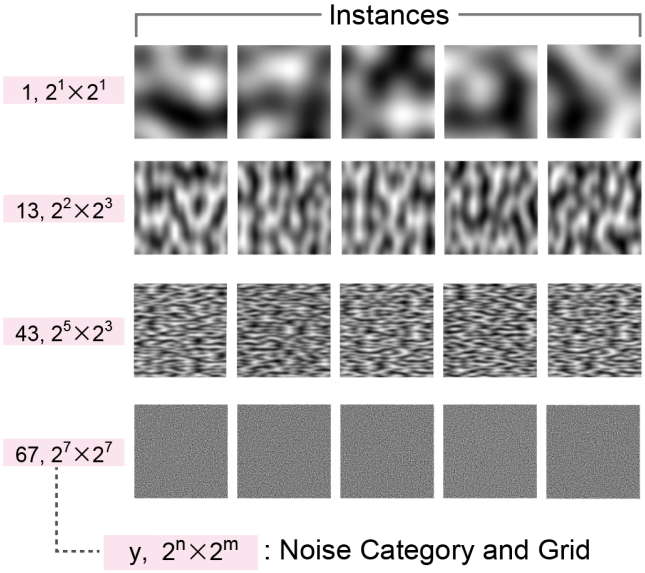

Fig. 4. Intra-category variation of noise samples. Image representations of noise samples for three categories ($y = 1, 13, 43, 67$) are shown.

lin noise. For example, compared to noise samples computed from a dense grid, those computed from a sparser grid will also be sparser in terms of noise complexity.

We use this difference in complexity to define categories of noise samples.

Specifically, to samples $\epsilon_i \sim G_\eta$ generated with parameter $\eta = (n, m)$, the noise label generator L_η attaches labels

$$y_i = (n-1)M + m. \tag{1}$$

Here, we assume that the range of η is $\{(n, m) : 1 \leq n \leq N,\ 1 \leq m \leq M\}$. This means that the range of y_i is $\{1, 2, \cdots, NM\}$ and the number of noise categories is NM.

Figure 3 and Figure 4 show examples of noise categories and intra-category variation of noise samples, respectively. As can be seen, from category to category, noise complexity varies from coarse to fine with changes in gradient direction. This artificial category definition facilitates the acquisition of a reasonable set of visual filters in a portion of the hidden layers of the neural network. For example, with a convolutional neural network, we obtain some texture-like filters at the bottom layer of the network (results are shown in Figure 5 in Sec. 4).

Note that the proposed method has three hyperparameters, namely N, M, and K. N and M are for determining the number of noise categories and K is the number of instances per category. Finally, the generated noise data $\mathcal{D}$ consist of $T = NMK$ noise samples with labels.

B. Optimization

The network parameters are optimized by solving a classification problem on the generated noise data $\mathcal{D} = \{(\epsilon_i, y_i)\}_{i=1}^{T}$. In this step, any type of objective function and optimizer can be introduced. Examples include cross-entropy loss with the SGD or ADAM optimizer, which starts from He initialization. In contrast to standard pre-training, which solves an optimization problem on real-world data, this step solves an artificial optimization problem, and as such this is a kind of closed-form problem. Introducing other types of optimizers to this step to solve this closed-form problem more efficiently, based on the characteristics of Perlin noise, will be considered in future work.

IV. Experiments

In this section, we show the effectiveness of the proposed initialization method on four image classification datasets.

A. Datasets and Evaluation Measures

Cifar-10. This dataset consists of 60,000 color images, each of which has a label from 10 object classes. We follow the standard evaluation procedure, in which 50,000 images are used for training and 10,000 images are used for testing. Classification accuracy over 10 classes is reported.

Cifar-100. This dataset consists of 60,000 color images, each of which has a label from 100 object classes. We follow the standard evaluation procedure, in which 500 and 100 images per class are used for training and testing, respectively. Classification accuracy over 100 classes is reported.

Omniglot. This dataset consists of 38,300 grayscale images of 1,623 different hand-written characters. It was originally proposed as a one-shot learning dataset. We use the data split proposed in [23]. 1,623 and 30,837 images are used for training and testing, respectively. Classification accuracy over 1,623 classes is reported.

Describable Textures Dataset (DTD). This dataset consists of 5640 images of 47 texture categories, such as checkered, striped, meshed, and marbled. We follow the standard evaluation procedure with three equal parts for training, validation, and testing.

ImageNet. This dataset consists of 1.2 million color images of 1,000 object classes. We use this dataset to show that the proposed initialization method helps improve the data efficiency of pre-training.

B. Implementation Details

Our implementation uses ResNet [8] as a backbone network. The results obtained using ResNet50 and ResNet152 are reported. For optimization, we use cross-entropy loss and the SGD optimizer in all experiments. For comparison, we report results obtained using He initialization [3], Xavier initialization [15], sparse initialization [16], and normal initialization. Notably, He initialization is utilized in most state-of-the-art methods for image classification tasks.

C. Results

C-1. Main Results

Table I shows a performance comparison of initialization methods on the four datasets. As shown, the proposed method always outperforms the conventional methods. This shows the effectiveness of initialization using Perlin noise.

TABLE I
PERFORMANCE COMPARISON ON FOUR DATASETS. CLASSIFICATION ACCURACIES (%) FOR EACH DATASET WITH TWO TYPES OF NETWORK ARE SHOWN.

Method	Cifar-10		Cifar-100		Omniglot		DTD	
	ResNet50	ResNet152	ResNet50	ResNet152	ResNet50	ResNet152	ResNet50	ResNet152
Normal initialization	92.62	93.47	75.16	75.59	2.66	2.37	13.68	5.24
Xavier initialization [15]	92.30	93.58	73.85	75.14	5.88	5.57	27.51	24.75
He initialization [3]	93.50	93.43	74.17	75.73	4.61	3.06	24.31	20.14
Proposed method	**93.76**	**94.27**	**77.42**	**78.21**	**17.54**	**18.71**	**55.03**	**54.18**

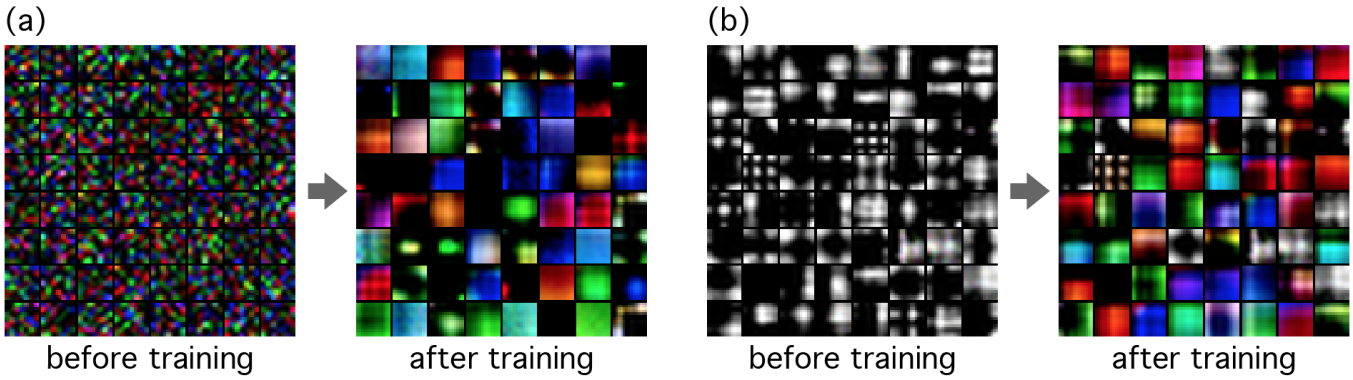

Fig. 5. Visualization of filters of the first convolutional (conv1) layer. Filters before and after training on Cifar-10 are shown. (a) He initialization and (b) proposed method.

To analyze the reason for the superior performance of the proposed method, Figure 5 shows the filters at the bottom layer (conv1 layer) before and after training. As can be seen, the proposed method gives a set of various filters, including some texture-like and edge-like filters. Therefore, after training, it obtains a useful set of colored filters for image classification even if the number of training samples is not very large.

Figure 6 shows validation accuracy curves for the Cifar-100 dataset. Our method not only achieves higher accuracy at the end of training, but its training starts at higher accuracy. This shows that, as an initial value for training, texture and edge filters are a better choice than randomized filters. In this experiment, we used the fixed learning rate schedule in the official PyTorch implementation tuned for training from scratch for a fair comparison. Optimizing the learning rate schedule for our method may lead to faster convergence.

In the present paper, we did not use color Perlin noise because some applications, including hand-written character recognition, use grayscale images. In future work, a noise sample generator that uses the RGB color space can be applied for cases where the dataset consists of color images.

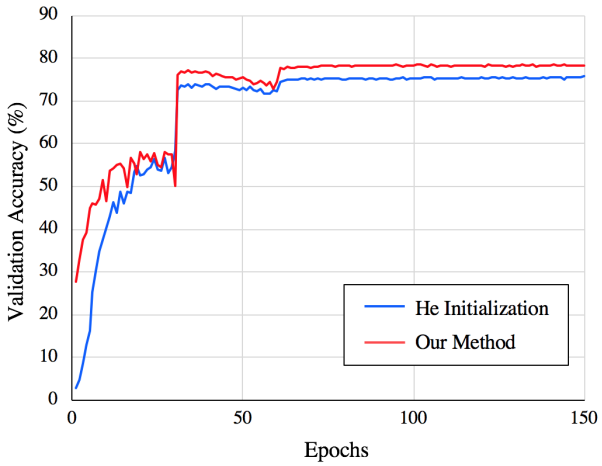

Fig. 6. Validation accuracy curves for the Cifar-100 dataset. He initialization and our method are compared.

C-2. Hyperparameters

To explore the effect of changes in hyperparameters, Table II shows the results obtained using various values for N, M, and K for noise data generation. Note that, as described in Sec. 3, (N, M) determines the number of Perlin noise categories based on noise complexity, and K controls the number of instances per category for generating $T = NMK$ noise samples.

A comparison of two strategies, namely increasing the number of categories and increasing the number of instances, shows that the former is more effective than the latter for improving image classification accuracy. This supports our assumption that noise complexity is an important factor for determining categories.

Performance improvements were observed across both network structures, with ResNet152 always outperforming ResNet50. In general, it is difficult to train a large network with a limited amount of data. However, our method is adaptable

TABLE II

EXPERIMENTAL RESULTS OBTAINED USING VARIOUS VALUES FOR HYPERPARAMETERS IN NOISE DATA GENERATION. ACCURACIES FOR THE CIFAR-100 DATASET ARE SHOWN. NETWORK: BACKBONE NETWORK (RESNET50 OR RESNET152). # CATEGORIES: NUMBER OF PERLIN NOISE CATEGOREIS; $N \times M$ CATEGOREIS ARE GENERATED. # INSTANCES: NUMBER OF INSTANCES PER CATEGORY.

Network	#categories	#instances		
		100	500	1,000
ResNet50	10×10	76.28	75.99	76.44
	18×18	76.95	76.92	75.53
	36×36	75.77	77.42	78.03
ResNet152	10×10	77.09	77.09	77.77
	18×18	77.92	77.69	77.23
	36×36	78.30	78.21	77.51

to networks of different sizes because noise complexity can be adjusted based on the network size. This advantage improves the performance of training large networks with a limited amount of data. To further improve performance, future noise sample generators can be designed using other types of noise.

C-3. Data-Efficient Pre-Training

Finally, we combine our method with pre-training to explore how this combination improves image classification performance. Figure 7 shows the results obtained using various numbers of images from the ImageNet (ILSVRC 2012) dataset for pre-training. According to the results, our method is particularly effective when the number of pre-training images is small. Although this paper focuses on network initialization without using any real-world images, the results imply that combining our method with pre-training methods would lead to more data-efficient pre-training.

V. CONCLUSION

This paper proposed a novel network initialization method that uses Perlin noise. Experiments on four image classification datasets demonstrated that the proposed initialization is effective for training networks for image classification. In particular, our method is effective for training with a limited amount of data because it provides texture-like initial filters that improve image classification accuracy. Our future work will focus on extending this initialization method to other types of noise, including colored noise, and data such as audio and video data, as well as introducing new noise category definitions.

Acknowledgment This work was partially supported by grants JSPS KAKEN 19H01134 and JST ACT-I (JPMJPR16U5). Computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used.

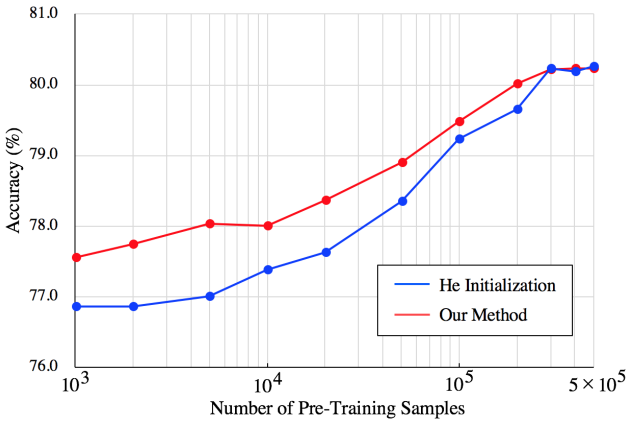

Fig. 7. Performance comparison using various numbers of samples for pre-training. Accuracy for the Cifar-100 dataset is shown. ResNet152 was pre-trained using images from the ImageNet dataset.

REFERENCES

[1] O. Russakovsky, et al., ImageNet Large Scale Visual Recognition Challenge. *In Springer IJCV*, vol. 115, no. 3, pp. 211–252, 2015.
[2] K. Perlin, An Image Synthesizer. *In SIGGRAPH*, pp. 287–296, 1985.
[3] K. He, et al., Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Proc. ICCV*, pp. 1026–1034, 2015.
[4] Y. LeCun, et al., Gradient-based Learning Applied to Document Recognition. *Proc. of the IEEE*, pp. 2278–2324, 1998.
[5] A. Krizhevsky, et al., ImageNet Classification with Deep Convolutional Neural Networks. *Proc. NeurIPS*, pp.1–9, 2012.
[6] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *Proc. ICLR*, 2015.
[7] C. Szegedy, et al., Going Deeper with Convolutions. *Proc. CVPR*, 2015.
[8] K. He, et al., Deep Residual Learning for Image Recognition. *Proc. CVPR*, 2016.
[9] S. Xie, et al., Aggregated Residual Transformations for Deep Neural Networks. *Proc. CVPR*, 2017.
[10] G. Huang, et al., Densely Connected Convolutional Networks. *Proc. CVPR*, 2017.
[11] J. Hu, et al., Squeeze-and-Excitation Networks. *Proc. CVPR*, 2018.
[12] J. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. *Proc. CVPR*, 2017.
[13] B. Zhou, et al., Places: A 10 million Image Database for Scene Recognition. *In IEEE Trans. on PAMI*, vol. 40, no. 6, pp. 1452–1464, 2017.
[14] F. Manessi and A. Rozza. Learning Combinations of Activation Functions. *Proc. ICPR*, pp. 61–66, 2018.
[15] X. Glorot and Y. Bengio. Understanding the Difficulty of Training deep Feedforward Neural Networks. *Proc. AISTAS*, pp. 249–256, 2010.
[16] J. Martens. Deep Learning via Hessian-free Optimization. *Proc. ICML*, 2010.
[17] W. Kay, et al., The Kinetics Human Action Video Dataset. *CoRR 1705.06950*, 2017.
[18] Z. Hailat, et al., Deep Semi-Supervised Learning. *Proc. ICPR*, pp. 2154–2159, 2018.
[19] L. Chen, et al., Semi-Supervised Convolutional Neural Networks with Label Propagation for Image Classification. *Proc. ICPR*, pp. 1319–1324, 2018.
[20] Z. Ling, et al., Semi-Supervised Learning via Convolutional Neural Network for Hyperspectral Image Classification. *Proc. ICPR*, pp. 1–6, 2018.
[21] A. Robles-Kelly and Ran Wei. Semi-Supervised Image Labelling Using Barycentric Graph Embeddings. *Proc. ICPR*, pp. 1518–1523, 2016.
[22] A. Ghaderi and V. Athitsos. Selective Unsupervised Feature Learning with Convolutional Neural Network. *Proc. ICPR*, pp. 2486–2490, 2016.
[23] B. M. Lake, et al., Human-level Concept Learning Through Probabilistic Program Induction. Science, vol. 350(6266), pp. 1332–1338, 2015.
[24] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical Report TR-2009, University of Toronto, 2019.
[25] L. Fei-Fei, et al., One-Shot learning of object categories. *IEEE Trans. Pattern Recognition and Machine Intelligence*, vol. 28 , no. 4, 2006.
[26] M. Everingham, et al., The Pascal Visual Object Classes (VOC) Challenge. *In Springer IJCV*, vol. 88, no. 2, pp. 303–338, 2010.
[27] M. Cimpoi, et al., Describing Textures in the Wild. *Proc. CVPR*, 2014.
[28] T.-Y. Lin, et al., Microsoft COCO: Common Objects in Context. *Proc. ECCV*, 2014.