Pre-Training Without Natural Images

Hirokatsu Kataoka¹ · Kazushige Okayasu¹ · Asato Matsumoto¹ · Eisuke Yamagata² · Ryosuke Yamada¹ · Nakamasa Inoue² · Akio Nakamura³ · Yutaka Satoh¹

Received: 29 March 2021 / Accepted: 25 November 2021 / Published online: 24 February 2022
© The Author(s) 2022

Abstract

Is it possible to use convolutional neural networks pre-trained without any natural images to assist natural image understanding? The paper proposes a novel concept, Formula-driven Supervised Learning (FDSL). We automatically generate image patterns and their category labels by assigning fractals, which are based on a natural law. Theoretically, the use of automatically generated images instead of natural images in the pre-training phase allows us to generate an infinitely large dataset of labeled images. The proposed framework is similar yet different from Self-Supervised Learning because the FDSL framework enables the creation of image patterns based on any mathematical formulas in addition to self-generated labels. Further, unlike pre-training with a synthetic image dataset, a dataset under the framework of FDSL is not required to define object categories, surface texture, lighting conditions, and camera viewpoint. In the experimental section, we find a better dataset configuration through an exploratory study, e.g., increase of #category/#instance, patch rendering, image coloring, and training epoch. Although models pre-trained with the proposed Fractal DataBase (FractalDB), a database without natural images, do not necessarily outperform models pre-trained with human annotated datasets in all settings, we are able to partially surpass the accuracy of ImageNet/Places pre-trained models. The FractalDB pre-trained CNN also outperforms other pre-trained models on auto-generated datasets based on FDSL such as Bezier curves and Perlin noise. This is reasonable since natural objects and scenes existing around us are constructed according to fractal geometry. Image representation with the proposed FractalDB captures a unique feature in the visualization of convolutional layers and attentions.

Keywords Formula-driven supervised learning · Image recognition · Representation learning

1 Introduction

The introduction of sophisticated pre-training image representation has led to a great expansion of the potential of image recognition. Image representations with e.g., the ImageNet/Places pre-trained convolutional neural networks (CNNs), have without doubt become the most important

breakthrough in recent years (Deng et al. 2008; Zhou et al. 2017). Specifically, we had a lot to learn from the ImageNet project, such as a huge number of annotations by tens of thousands of participants accomplished by crowdsourcing and a well-organized categorization based on WordNet Fellbaum (1998). There are several important steps including category definition, image collection, labeling, image selection, and cross-checking. Due to the scale and labeling quality, the construction of this dataset became the baseline for subsequent projects. Thanks to ImageNet and other salient projects, the concept has changed from model-driven to data-driven methods in the era of deep neural networks. However, due to the fact that the annotation was carried out by a large number of unspecified people, most of whom are not experts in image classification and the corresponding areas, the dataset contains some labels which are incorrect and/or violate rules and norms concerning privacy and ethics (Yang et al. 2020). This limits ImageNet to only non-commercial usage. Moreover, in 2020, access rights to the 80M Tiny Images dataset were

Communicated by Hiroshi Ishikawa.

The code, datasets, and pre-trained models are publicly available:
<https://github.com/hirokatsukataoka16/FractalDB-Pretrained-ResNet-PyTorch>.

✉ Hirokatsu Kataoka
hirokatsu.kataoka@aist.go.jp

¹ National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan

² Tokyo Institute of Technology, Tokyo, Japan

³ Tokyo Denki University, Tokyo, Japan

withdrawn (Torralba et al. 2008) on the basis of a technical report (Birhane and Prabhu 2021). In this way, several large-scale image datasets are no longer publicly available due to privacy and ethical issues. From another perspective, though models trained on massive-scale datasets such as JFT-300M (Sun et al. 2017) and Instagram-3.5B (Mahajan et al. 2018) have been shown to exhibit superior performance in terms of image recognition, these datasets are limited to use inside of a company and are not currently publicly available. Note that YFCC-100M was made publicly available in the machine learning community but access rights of the Flickr-based dataset were apparently withdrawn¹. We believe that these occurrences concerning large-scale image datasets and their pre-trained CNN models significantly impedes the prospects of vision-based recognition.

We begin by considering a pre-trained CNN model with a million natural images. In most cases, representative image datasets consist of natural images taken by a camera that express a projection of the real world. Although the space of image representation is enormous (a 300k-pixel grayscale image has $256^{300,000}$ space), a CNN model has been shown to be capable of recognizing natural images from among around one million natural images from the ImageNet dataset. We believe that labeled images on the order of millions have a great potential to improve image representation as a pre-trained model. However, we suggest that it is pertinent to consider the following question: *Can we accomplish pre-training without any natural images for parameter fine-tuning on a dataset including natural images?* To the best of our knowledge, the ImageNet/Places pre-trained models have not been replaced by a model trained without natural images. Here, we consider pre-training without natural images. To replace the models pre-trained with natural images, we attempt to find a method for automatically generating images. Automatically generating a large-scale labeled image dataset is challenging. However, a model pre-trained without natural images makes it possible to solve problems related to privacy, copyright, and ethics, as well as issues related to the cost of image collection and labeling.

Our problem setting is similar in some respects to self-supervised learning (SSL) which automatically generates pseudo labels in natural images. The representative SSL methods contain e.g., contrastive labels (CPC Oord et al. 2018, MoCo He et al. 2020, SimCLR Chen et al. 2020), context-based labels (Jigsaw puzzle Noroozi and Favaro 2016, Rotation Gidaris et al. 2018, DeepCluster Caron et al. 2018), and generation-based labels (colorization Zhang et al. 2016, BigBiGAN Donahue and Simonyan 2019). Our goal is to automatically create both self-generating images and their labels for constructing a pre-training CNN model. Therefore, it is different from SSL in terms of natural images usage. The

SSL framework is still subject to some concerns regarding the above-mentioned dataset-related problems.

Unlike a synthetic image dataset, can we automatically make image patterns and their labels with image projection from a mathematical formula? Regarding synthetic datasets, the SURREAL dataset (Varol et al. 2017) has successfully made training samples of estimating human poses with human-based motion capture (mocap) and background. In this context, Domain Randomization (e.g., Tobin et al. 2017; Sundermeyer et al. 2018) and Cut-and-Paste Learn (e.g., Dwibedi et al. 2017; Remez et al. 2018) are also successful approaches for automatically synthesizing from defined object models by considering, e.g., object posture, foreground-background boundary, background, lighting conditions, and camera viewpoint. In contrast, our Formula-driven Supervised Learning and the generated formula-driven image dataset has significant potential to automatically generate an image pattern and a label. For example, we consider using *fractals*, a sophisticated natural formula (Mandelbrot 1983). Generated fractals can differ drastically following a slight change in the parameters, and can often be distinguished in the real-world. Most natural objects appear to be composed of complex patterns, but fractals allow us to understand and reproduce these patterns.

We believe that the concept of pre-training without natural images can simplify large-scale DB construction, and that models pre-trained on formula-driven images can be effective. The advantage of using a formula-driven image dataset comprised of automatically generated image patterns and labels is that it enables us to efficiently solve some of the current issues surrounding using a CNN, namely, large-scale image database construction without human annotation and image downloading. Fundamentally, construction of the dataset does not rely on any natural images (e.g. ImageNet Deng et al. 2008 or Places Zhou et al. 2017) or closely resembling synthetic images (e.g., SURREAL Varol et al. 2017). The present paper makes the following contributions.

The concept of pre-training without natural images provides a method by which to automatically generate a large-scale image dataset complete with image patterns and their labels. In order to construct such a database, through exploratory research, we experimentally disclose ways to automatically generate categories using fractals. In what follows, two sets of randomly searched fractal databases are generated in the following manner: FractalDB-1k/10k, which consists of 1000/10,000 categories (see Fig. 1 for all FractalDB-1k categories). See Fig. 2a for Formula-driven Supervised Learning from categories of FractalDB-1k. Regarding the proposed database, the FractalDB pre-trained model outperforms some models pre-trained by human annotated datasets (see Table 8 for details). Furthermore, Fig. 2b shows that FractalDB pre-training accelerated

¹ <http://mmcommons.org/>.

Fig. 1 All categories in the FractalDB-1k dataset. 1,000 fractal categories are listed, rendered by Iterated Function Systems (IFS). Surprisingly, a CNN architecture classifies the image patterns with close to 100% training accuracy

(a) The pre-training framework with fractal geometry for feature representation learning. We can enhance natural image recognition by pre-training without natural images.

(b) Accuracy transition on CIFAR-10 dataset among ImageNet-1k, FractalDB-1k and training from scratch.

Fig. 2 Proposed *pre-training without natural images* based on fractals, which represent natural phenomena existing in the real world (Formula-driven Supervised Learning). We automatically generate a large-scale labeled image dataset based on an iterated function system (IFS)

the convergence speed, which was much better than training from scratch and similar to ImageNet pre-training.

2 Related Work

2.1 Pre-Training on Large-Scale Datasets

A number of large-scale datasets have been made publically available for exploring how to extract image representations. ImageNet (Deng et al. 2008), which consists of more than 14 million images, is the most widely-used dataset for pre-training networks. Because it comprises images of 20k natural object categories, the obtained image representation is often effective for various visual recognition tasks

in the real world. COCO (Lin et al. 2014) and OpenImages (Krasin et al. 2017) provide a large number of images with ground-truth bounding boxes for object detection and segmentation masks for instance segmentation. In terms of scene recognition, Places (Zhou et al. 2017) provides more than 10 million images comprising 434 scene categories such as “restaurant”, “dining hall”, and “forest”. To capture human actions, video datasets such as Kinetics (Kay et al. 2017) and Moments-in-Time (Monfort et al. 2019) often improve image and video representations. These datasets have contributed to improving the accuracy of DNNs. Some in-house datasets, e.g., JFT-300M (Sun et al. 2017) and IG-3.5B (Mahajan et al. 2018), are known to be useful for further improving pre-training performance. Historically, in terms of multiple evaluation metrics, pre-training on ImageNet has

been proved to be one of the most promising and reasonable approaches. This is because image representations can be adapted to each target task by applying transfer learning techniques (Donahue et al. 2014; Huh et al. 2016; Kornblith et al. 2019) including simple fine-tuning.

2.2 Learning Frameworks

Supervised learning with manually and precisely annotated images is currently the most promising framework for obtaining strong image representations, and thus reducing the annotation cost is an important research topic. Recently, the research community has been considering how to decrease the volume of labeled data required for training. Example approaches include weakly-supervised learning, semi-supervised learning, unsupervised learning, and self-supervised learning. Among these approaches, self-supervised learning has attracted significant attention due to its performance in terms of both accuracy and cost efficiency. The idea is to configure a simple but suitable task, called a pre-text task (Doersch et al. 2015; Noroozi and Favaro 2016; Noroozi et al. 2018; Zhang et al. 2016; Noroozi et al. 2017; Gidaris et al. 2018), in which networks learn to predict obvious labels on unlabeled images. For example, relative positions and/or rotations of image patches are used as obvious labels in some conventional methods such as jigsaw puzzle (Noroozi and Favaro 2016), image rotation (Gidaris et al. 2018), and colorization (Zhang et al. 2016). They are far from being a fully suitable alternative to human annotation, but the idea has proven to be effective for learning representations. More recent approaches including DeepCluster (Caron et al. 2018), MoCo (He et al. 2020), and SimCLR (Chen et al. 2020) are closer to the performance pre-trained by human-annotated datasets like ImageNet. More recent studies discussed SSL with single images (Asano et al. 2020) and self-labeling (Asano et al. 2020); therefore, we believe that the pre-training (pre-text task in SSL) can be done without any natural images. In addition to the self-generated labels that SSL creates, our training on FDSL enables the automatic rendering of training images based on a mathematical formula.

2.3 Network Architectures

In many visual recognition tasks, neural networks have achieved state-of-the-art performance. In particular, CNNs having several tens to hundreds of hidden layers, each of which performs convolutional or pooling operations, are often utilized with the above learning frameworks. The first success in large-scale image classification was achieved in 2012 with AlexNet (Krizhevsky et al. 2012), which is a network with eight layers. Subsequently, deeper network architectures were proposed such as VGGNet (Simonyan and

Zisserman 2015) with 16 to 19 layers and the Inception network (GoogLeNet) (Szegedy et al. 2015) with more than 20 layers. ResNet (He et al. 2016) further explored architectures with 100+ layers by introducing skip connections. Among these network architectures, ResNet is the most widely-used due to its training stability. It also has various extensions such as ResNeXt (Xie et al. 2017), MobileNet (Howard et al. 2017; Sandler et al. 2018; Howard et al. 2019), SENet (Hu et al. 2020), and DenseNet (Huang et al. 2017).

2.4 Synthetic Image Pre-Training

There exists a similar setting with synthetic image pre-training in visual representation learning. In this context, we introduce the usage of 2D and 3D data configuration.

One of the most promising frameworks in 2D images is ‘Cut-and-Paste Learn’, which enabled to train a CNN from segmented real image and background (Dwibedi et al. 2017; Remez et al. 2018). Dwibedi *et al.* discovered that a CNN can be trained with only synthetic images (Dwibedi et al. 2017). They gave an image label from a segmented image and added a bounding box when a synthetic image is created. A segmented image must be embedded with Poisson blending (Perez et al. 2003) into a synthetic image while considering a boundary between object and background. The Cut-and-Paste Learn can be applied in semantic segmentation tasks (Remez et al. 2018). Shrivastava *et al.* proposed an image transformation from synthetic to photo-realistic image based on a generative model (Shrivastava et al. 2017). We witnessed how the approach is effective in simple real-world patterns.

In synthetic datasets from 3D data, we have assigned mocap humans (Varol et al. 2017) and CAD/scanned objects (Tobin et al. 2017; Sundermeyer et al. 2018; Movshovitz-Attias et al. 2016). Although these synthetic approaches with 3D data successfully increased the number of training datasets, detailed definitions are required such as object posture, background texture, lighting condition, and camera viewpoint.

2.5 Mathematical Formula for Image Projection

One of the best-known formula-driven image projections is fractals. Fractal theory has been discussed for many years (e.g., Mandelbrot 1983; Landini et al. 1995; Smith et al. 1996). Fractal theory has been applied to rendering a graphical pattern in a simple equation (Barnsley 1988; Monro and Budbridge 1995; Chen and Bi 1997) and constructing visual recognition models (Pentland 1984; Varma and Garg 2007; Xu et al. 2009; Larsson et al. 2017). Although a rendered fractal pattern loses its infinite potential for representation by projection to a 2D-surface, a human can recognize the rendered fractal patterns as natural objects.

Since the success of these studies relies on the fractal geometry of naturally occurring phenomena (Mandelbrot 1983; Falconer 2004), our assumption that fractals can assist learning image representations for recognizing natural scenes and objects is supported. Other methods, namely those involving Bezier curves (Farin 1993) or Perlin noise (Perlin 2002), have also been discussed in terms of computational rendering. We also implement and compare these methods in the experimental section (see Table 12).

3 Automatically Generated Large-Scale Dataset

Figure 3 presents an overview of the Fractal DataBase (FractalDB), which consists of an infinite number of pairs of fractal images I and their fractal categories c with an iterated function system (IFS) (Barnsley 1988). We chose fractal geometry because this means that a simple equation can be used to render complex patterns that are closely related to natural objects. All fractal categories are randomly searched (see Fig. 2a), and the intra-category instances are expansively generated by considering category configurations such as rotation and patch. (The augmentation is shown as $\theta \rightarrow \theta'$ in Fig. 3.)

In order to construct a pre-trained CNN model, the FractalDB is applied to each training of the parameter optimization as follows. (i) Fractal images with paired labels are randomly sampled by a mini batch $B = \{(I_j, c_j)\}_{j=1}^b$. (ii) Calculate the gradient of B to reduce the loss. (iii) Update the parameters. Note that we replace the pre-training step, such as using the ImageNet pre-trained model. We also conduct a fine-tuning step as well as plain transfer learning (e.g., ImageNet pre-training and CIFAR-10 fine-tuning).

3.1 Fractal Image Generation

In order to construct fractals, we use IFS (Barnsley 1988). In fractal analysis, an IFS is defined on a complete metric space $\mathcal{X}$ by

$$\text{IFS} = \{\mathcal{X}; w_1, w_2, \dots, w_N; p_1, p_2, \dots, p_N\}, \quad (1)$$

where $w_i : \mathcal{X} \rightarrow \mathcal{X}$ are transformation functions, p_i are probabilities which sum to 1, and N is the number of transformations.

Using the IFS, a fractal $S = \{\mathbf{x}_t\}_{t=0}^{\infty} \in \mathcal{X}$ is constructed by the random iteration algorithm (Barnsley 1988), which repeats the following two steps for $t = 0, 1, 2, \dots$ from an initial point $\mathbf{x}_0$. (i) Select a transformation w^* from $\{w_1, \dots, w_N\}$ with pre-defined probabilities $p_i = p(w^* = w_i)$ to determine the i -th transformation. (ii) Produce a new point $\mathbf{x}_{t+1} = w^*(\mathbf{x}_t)$.

Since the focus herein is on representation learning for image recognition, we construct fractals in the 2D Euclidean space $\mathcal{X} = \mathbb{R}^2$. In this case, each transformation is assumed in practice to be an affine transformation (Barnsley 1988), which has a set of six parameters $\theta_i = (a_i, b_i, c_i, d_i, e_i, f_i)$ for rotation and shifting:

$$w_i(\mathbf{x}; \theta_i) = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \mathbf{x} + \begin{bmatrix} e_i \\ f_i \end{bmatrix}. \quad (2)$$

An image representation of the fractal S is obtained by drawing dots on a black background. The details of this step with its adaptable parameters are explained in Sect. 3.3.

3.2 Fractal Categories

Undoubtedly, automatically generating categories for pre-training of image classification is a challenging task. Here, we associate the categories with fractal parameters $a-f$. As shown in the experimental section, we successfully generate a number of pre-trained categories on FractalDB (see Fig. 6) through formula-driven image projection by an IFS.

Since an IFS is characterized by a set of parameters and their corresponding probabilities, i.e., $\Theta = \{(\theta_i, p_i)\}_{i=1}^N$, we assume that a fractal category has a fixed Θ and propose 1,000 or 10,000 randomly searched fractal categories (FractalDB-1k/10k). The reason for using 1,000 categories is closely related to the experimental results for various #categories in Fig. 5.

3.2.1 FractalDB-1k/10k

consists of 1000/10,000 different fractals (examples shown in Fig. 2a), the parameters of which are automatically generated by repeating the following procedure. First, N is sampled from a discrete uniform distribution, $\mathbb{N} = \{2, 3, 4, 5, 6, 7, 8\}$. Second, the parameter θ_i for the affine transformation is sampled from the uniform distribution on $[-1, 1]^6$ for $i = 1, 2, \dots, N$. Third, p_i is set to

$$p_i = \frac{\det A_i}{\sum_{i=1}^N \det A_i}, \quad (3)$$

where $A_i = (a_i, b_i; c_i, d_i)$ is a rotation matrix of the affine transformation. Finally, $\Theta_i = \{(\theta_i, p_i)\}_{i=1}^N$ is accepted as a new category if the filling rate r of the representative image of its fractal S is investigated in the experiment (see Table 2). The filling rate r is calculated as the number of pixels of the fractal with respect to the total number of pixels in the image.

Fig. 3 Overview of the proposed framework. Generating FractalDB: Pairs of an image I_j and its fractal category c_j are generated without human labeling and image downloading. Application to transfer

learning: A FractalDB pre-trained convolutional network is assigned to conduct transfer learning for other datasets

Fig. 4 Intra-category augmentation of a *leaf* fractal. Here, a_i , b_i , c_i , and d_i are for rotation, and e_i and f_i are for shifting

3.3 Adaptable Parameters for FractalDB

As described in the experimental section, we investigated several parameters related to fractal parameters and image rendering. The types of parameters are listed as follows.

3.3.1 #Category and #Instance

We believe that the effects of #category and #instance are the most significant in the pre-training task. We change the two parameters from 16 to 1,000 as $\{16, 32, 64, 128, 256, 512, 1,000\}$.

3.3.2 Patch versus Point

We apply a 3×3 [pixel] patch filter to generate fractal images in addition to the rendering at each 1×1 [pixel] point. The patch rendering means that 3×3 [pixel] patch is drawn instead of 1×1 [pixel] point at each iteration to generate a fractal image. These rendering methods make a difference as the ‘original (point)’ and ‘patch’ in Fig. 3. The patch filter creates variation in the pre-training phase. We repeat the following process t times. We set a pixel (u, v) , and then a random dot(s) with a 3×3 patch is inserted in the sampled area.

3.3.3 Filling Rate r

We set the filling rate from 0.05 (5%) to 0.25 (25% at 5% intervals, namely, $\{0.05, 0.10, 0.15, 0.20, 0.25\}$). Note that we could not yield any randomized category at a filling rate of over 30%.

3.3.4 Weight of Intra-Category Fractals (w)

In order to generate an intra-category image, the parameters for an image representation are varied. Intra-category images are generated by changing one of the parameters a_i , b_i , c_i , d_i , e_i and f_i with weighting parameter w . The basic parameter is from $\times 0.8$ to $\times 1.2$ at intervals of 0.1, i.e., $\{0.8, 0.9, 1.0, 1.1, 1.2\}$. Figure 4 shows an example of the

intra-category variation in fractal images. We believe that the intra-class diversity based on the weighting parameter helps to improve the performance for image classification.

3.3.5 #Dot (t) and Image Size (W, H)

The #Dot parameter means the number of drawing iterations with 3×3 [pixel] patch or 1×1 [pixel] point in a fractal image. We vary the parameters t as {100K, 200K, 400K, 800K} and (W and H) as {256, 362, 512, 764, 1024}. The averaged parameter in grayscale has a pixel value of $(r, g, b) = (127, 127, 127)$ (for pixel values from 0 to 255).

3.3.6 Grayscale/Color Configuration

The renderer plots dots with fixed grayscale pixels $(r, g, b) = (127, 127, 127)$ (range of pixel value: 0–255). In the color configuration, we plot randomly colored dots with discrete uniform distributions at each pixel.

3.3.7 Training Epoch

We set a longer training epoch in FractalDB. Due to the computational resource, the computing is limited up to 200 epochs when we consider a larger computational task such as FractalDB-10k. In the present experiment, we take checkpoints during pre-training by setting the number of epochs to 90, 120, or 200 and then fine-tune. Other Self-Supervised Learning methods like SimCLR (Chen et al. 2020) explore a longer training strategy. We also plan to carry out a longer training strategy in future.

3.4 Other Formula-Driven Image Datasets

We list and describe how to construct other formula-driven image databases with Perlin noise (PerlinNoiseDB) (Perlin 2002) and Bezier curves (BezierCurveDB) (Farin 1993).

3.4.1 PerlinNoiseDB

Perlin Noise is a widely used method for generating textures in computer graphics. Just like fractals, it is formula driven and capable of constructing a database without human annotation. It matches the concept of *Pre-training without Natural Images*, and thus we implemented a PerlinNoiseDB as a comparator to the FractalDB.

Generating Perlin noise can be divided into three steps: definition of a 2D-grid, calculation based on an argument point, and interpolation. First, a 2D-grid is defined with a random gradient vector given at each grid point. Next, the value of each argument point is computed by a dot product between gradient vectors at the four corners of the cell the point belongs to, and distance vectors between the argument

point and the corresponding grid points. Finally, through an interpolation between the four values computed in step 2, the final value of the argument point is determined. Through this simple process, Perlin noise can be generated.

The interval gradient vectors affect the complexity of the generated noise. For example, compared to noise computed from a grid with a gradient vector at every grid point, noise computed from a grid with a gradient vector at every two grid points will be rougher in terms of complexity of the noise. In the implementation of the PerlinNoiseDB, we used this difference in complexity to generate categories. For example, in PerlinNoiseDB-100, we defined a 1024×1024 grid with gradient vectors 2^{10-n} grid points vertically and 2^{10-m} grid points horizontally ($n, m = 1, 2, \dots, 10$), which makes category n_m . As a result, we created 100 categories: 01_01, 01_02, ..., 10_09, 10_10. 01_01 is the category with the roughest noise, whereas 10_10 is the one with the most detailed noise.

As for the instances within each of the categories, we changed the gradient vectors. The angles at which the gradient vectors are defined at each grid point are determined randomly. Therefore, redefining the gradient vectors would result in different gradient vectors, and thus different noise. Using this simple method, there are 10,000 instances per category in PerlinNoiseDB-100.

Comparing several datasets, PerlinNoiseDB-100, Perlin NoiseDB-1296, it seemed that the more categories there are, the better the accuracy. Further, between datasets with the same number of categories but with a different number of instances, datasets with more instances performed better. This tendency is the same as that of FractalDB.

3.4.2 BezierCurveDB

Just like PerlinNoiseDB, we also implemented a Bezier-CurveDB in comparison to the FractalDB. The Bezier-CurveDB consists of images of Bezier curves. It is a method for generating smooth curves in computer graphics. Bezier curves are also formula driven and can construct a dataset without human annotation.

Bezier curves are $n-1$ -dimensional curves generated from n points. De Casteljau's algorithm is a widely used method for drawing the curves. Bezier curves are generated by the following procedure:

1. Plot n dots
2. Form lines between those dots.
3. Plot dots dividing each line into $t : 1 - t$
4. Repeat (2) to (3) until only one dot is left

We implemented a BezierCurveDB for pre-training, and describe the dataset categories and instances. Note that generated images are composed of lines formed to render the

curves. The image categories are defined by a pair of n and s representing the number of dots plotted first in generating the Bezier curves, and the number of line division steps, respectively. For example, in the BezierCurvesDB-1024 dataset, we defined category n_s by combining 32 numbers ($n, s = 3, 4, \dots, 33, 34$). As a result, we created 1024 categories: 03_03, 03_04, 03_05, ..., 34_32, 34_33, 34_34. 03_03 is the category of 2D curves generated with dividing lines into 3 equal parts. Next, in terms of the instances within each of the categories, the location of the first dot was varied. By plotting these first dots randomly, we created 1,000 instances per category in BezierCurveDB.

We compared several datasets and BezierCurveDB (BezierCurveDB-144 and BezierCurveDB-1024), as per the approach taken with PerlinNoiseDB. As a result, it was found that BezierCurveDB has the same tendency as FractalDB and PerlinNoiseDB i.e., the higher the number of categories or instances, the higher the accuracy.

4 Experiments

Through a set of experiments, we investigated the effectiveness of FractalDB and how to construct categories with the effects of configuration, as mentioned in Sect. 3.3. We then quantitatively evaluated and compared the proposed framework with Supervised Learning (ImageNet-1k and Places-365, namely ImageNet (Deng et al. 2008) and Places (Zhou et al. 2017) pre-trained models) and SSL (Deep Cluster-10k (Caron et al. 2018)) on several datasets (Krizhevsky 2009; Deng et al. 2008; Zhou et al. 2017; Everingham et al. 2015; Lake et al. 2015). In SSL, we used DeepCluster-10k because this method is the most similar to the proposed method from the perspective of pseudo labels based on the specific function. In DeepCluster-10k, k-means clustering is applied to create labels from convolutional features.

4.1 Implementation Details

To confirm the properties of FractalDB and compare our pre-trained feature with previous studies, we principally used ResNet-50. Several architectures such as AlexNet and the ResNet-family are investigated in Table 9; however, the other experiments are conducted using only ResNet-50. We simply replaced the pre-training phase with our FractalDB (e.g., FractalDB-1k/10k), without changing the fine-tuning step. Moreover, in using the fine-tuning datasets, we conducted a standard training/validation. For pre-training and fine-tuning, we used the momentum stochastic gradient descent (SGD) (Bottou 2010) optimization algorithm with a momentum value of 0.9, a basic batch size of 256, and an initial learning rate of 0.01. The learning rate was multiplied by 0.1 when the learning epoch reached 30 and then again at epoch 60.

Training was performed up to epoch 90. Moreover, the input images were cropped to a size of 224×224 [pixel] from a 256×256 [pixel] input image. We implemented only random cropping as a data augmentation method, since our goal is to evaluate the potential of FractalDB pre-training in a simple manner.

4.2 Tunings and Comparisons

We explored the configuration of formula-driven image datasets regarding fractal generation by comparing the models trained on variously configured FractalDBs. We evaluate their performance on CIFAR-10/100 (C10, C100), ImageNet-100 (IN100), and Places-30 (P30) datasets. Considering the computational resource, we assigned IN100 and P30 as a replacement for the ImageNet-1k and Places-365 datasets. We randomly selected 100/30 categories from ImageNet-1k and Places-365 datasets. The parameters correspond to those mentioned in Sect. 3.3. Additionally, we compared the best practice in FractalDB pre-training to the related pre-training on representative datasets.

4.2.1 #Category and #Instance

In Figs. 5a–d, we plot the performance of FractalDBs, configured with various numbers of category and instance, to investigate their effects. We investigate the parameters with {16, 32, 64, 128, 256, 512, 1000} on both properties. Here, we find that the larger values tend to be better. At the beginning, a larger parameter in pre-training tends to improve the accuracy in fine-tuning on all the datasets. With C10/100, we can see +7.9/+16.0 increases in performance as #category increases from 16 to 1,000. Performance improvements are also discernable as #instance per category increases, albeit to a lower extent: +5.2/+8.9 on C10/100.

Hereafter, we assigned 1,000 [category] $\times$ 1,000 [instance] as a basic dataset size and tried to train 10k categories since the #category parameter is more effective in improving performance.

4.2.2 Patch versus Point

In Table 1, we investigate effects of the different sized filters in the generation process. Table 1 shows the difference between 3×3 [pixel] patch rendering and 1×1 [pixel] point rendering. Here, we find that Patch with 3×3 [pixel] is better. We can confirm that the 3×3 [pixel] patch rendering is better for pre-training with 92.1 vs. 87.4 (+4.7) on C10 and 72.0 vs. 66.1 (+5.9) on C100. Moreover, when comparing random patch patterns to fixed patch in image rendering, performance rates increased by {+0.8, +1.6, +1.1, +1.8} on {C10, C100, IN100, P30}.

Fig. 5 Effects of #category and #instance on the CIFAR-10/100, ImageNet-100 and Places-30 datasets. The other parameter is fixed at 1,000, e.g. #Category is fixed at 1,000 when #Instance changed by {16, 32, 64, 128, 256, 512, 1,000}

4.2.3 Filling Rate

In Table 2, we investigate the effects of the different filling rates. The top scores for each dataset and the parameter are 92.0, 80.5 and 75.5 with a filling rate of 0.10 on C10, IN100 and P30, respectively. Based on these results, although there are no significant changes between {0.05, 0.10, 0.15}, a filling rate of 0.10 appears to be better.

4.2.4 Weight of Intra-Category Fractals

In Table 3, we investigate the effects of intra-category variance by changing the intervals as follows. Starting from the basic parameter at intervals of 0.1 with {0.8, 0.9, 1.0, 1.1, 1.2} (see Fig. 4), we varied the intervals as 0.1, 0.2, 0.3, 0.4, and 0.5. For the case in which the interval is 0.5, we set {0.01, 0.5, 1.0, 1.5, 2.0} in order to avoid the weighting value being set as zero. A higher intra-category variance tends to provide higher accuracy. We confirm that the accuracies varied as {92.1, 92.4, 92.4, **92.7**, 91.8} on C10, where 0.4 is the highest performance rate (92.7), but 0.5 decreases the recognition rate (91.8). We conclude an interval of 0.4 to be the best. We used the weight value with a 0.4 interval, i.e., {0.2, 0.6, 1.0, 1.4, 1.8}.

4.2.5 #Dot

In Table 4, we investigate the effects of the different numbers of dots by comparing 100k, 200k, and 400k dots. The best parameters for each configuration are 100K on C10 (91.3), 200k on C100/P30 (71.0 / 74.8), and 400k on IN100 (80.0). Although a larger value is suitable on IN100, a lower value tends to be better on C10, C100, and P30. For the #dot parameter, we select 200k considering the balance in terms of rendering speed and accuracy.

4.2.6 Image Size

In Table 5, we investigate the effects of the different image sizes. In terms of image size, 256×256 [pixel] and 362×362

[pixel] perform similarly, e.g., 73.6 (256) vs. 73.2 (362) on C100. A larger size, such as 1024×1024 , is sparse in the image plane. Therefore, the fractal image projection produces better results in the cases of 256×256 [pixel] and 362×362 [pixel]. Here, a larger image size with a large amount of #dot can clearly represent the fractal geometry. However, due to the limitation of computational resources and pixel characteristics, we set the image size in rendering time as 362×362 .

4.2.7 Grayscale/Color Configuration

In Table 6, we investigate the difference between two configurations with grayscale and color FractalDB. In pre-training on the FractalDB, the two configurations were compared, and the results for color were found to be slightly better. The effect of the color property does not appear to be strong in the pre-training phase, e.g., 93.1 (w/ color) vs. 92.9 (w/o color) on C10.

4.2.8 Training Epoch

In Table 7, we explore the three types of training terms in FractalDB-1k: 90, 120, and 200 epochs in the pre-training phase. According to the results, we can confirm that the effect of longer-term training (200 epochs) is relatively higher than shorter term training using 90 or 120 epochs.

4.2.9 Best Practice in FractalDB Pre-trained Model

We further explored the set of parameters in the FractalDB pre-trained model. According to the results of the explorative study and additional tuning with parameter combinations, the highest accuracies occurred in #category (1000/10,000), #instance (1,000), patch (fixed 3×3 patch in an image), filling rate (0.2), weight of intra-category fractals (0.4), #dot (200k), image size (362×362), color configuration (random color), and training epoch (200 epochs). The performance rates are shown in Table 8.

Table 1 Exploration: patch versus point

	C10	C100	IN100	P30
Point	87.4	66.1	73.9	73.0
Patch (random)	92.1	72.0	78.9	73.2
Patch (fix)	92.9	73.6	80.0	75.0

Bold values indicate top-ranked scores in the experimental results

Table 2 Exploration: filling rate

	C10	C100	IN100	P30
.05	91.8	72.4	80.2	74.6
.10	92.0	72.3	80.5	75.5
.15	91.7	71.6	80.2	74.3
.20	91.3	70.8	78.8	74.7
.25	91.1	63.2	72.4	74.1

Bold values indicate top-ranked scores in the experimental results

Table 3 Exploration: weights

	C10	C100	IN100	P30
.1	92.1	72.0	78.9	73.2
.2	92.4	72.7	79.2	73.9
.3	92.4	72.6	79.2	74.3
.4	92.7	73.1	79.6	74.9
.5	91.8	72.1	78.9	73.5

Bold values indicate top-ranked scores in the experimental results

Table 4 Exploration: #Dot

	C10	C100	IN100	P30
100k	91.3	70.8	78.8	74.7
200k	90.9	71.0	79.2	74.8
400k	90.4	70.3	80.0	74.5

Bold values indicate top-ranked scores in the experimental results

Table 5 Exploration: image size

	C10	C100	IN100	P30
256	92.9	73.6	80.0	75.0
362	92.2	73.2	80.5	75.1
512	90.9	71.0	79.2	73.0
724	90.8	71.0	79.2	73.0
1024	89.6	68.6	77.5	71.9

Bold values indicate top-ranked scores in the experimental results

Table 6 Grayscale versus color for the pre-training model

	C10	C100	IN100	P30
Grayscale	92.9	73.6	80.5	75.5
Color	93.1	74.7	80.6	75.4

Bold values indicate top-ranked scores in the experimental results

Table 7 Training epoch

	C10	C100	IN100	P30
90 ep	93.1	74.7	80.6	75.4
120 ep	93.1	74.7	81.4	75.6
200 ep	93.2	75.7	82.7	75.9

Bold values indicate top-ranked scores in the experimental results

Table 8 Classification accuracies of Ours (FractalDB-1k / 10k), Scratch, DeepCluster-10k (DC-10k), ImageNet-100/1k and Places-30/365 pre-trained models on representative pre-training datasets

Method	Pre-train Img	Type	C10	C100	IN1k	P365	VOC12	OG
Scratch	–	–	87.6	62.7	76.1	49.9	58.9	1.1
DC-10k	Natural	Self-supervision	89.9	66.9	66.2	51.5	67.5	15.2
Places-30	Natural	Supervision	90.1	67.8	69.1	–	69.5	6.4
Places-365	Natural	Supervision	94.2	76.9	71.4	–	78.6	10.5
ImageNet-100	Natural	Supervision	91.3	70.6	–	49.7	72.0	12.3
ImageNet-1k	Natural	Supervision	96.8	84.6	–	50.3	85.8	17.5
FractalDB-1k	Formula	Formula-supervision	93.4	75.7	70.3	49.5	58.9	20.9
FractalDB-10k	Formula	Formula-supervision	94.1	77.3	71.5	50.8	73.6	29.2

We show the types of pre-trained image (Pre-train Img; which includes {Natural Image (Natural), Formula-driven Image (Formula)}) and Supervision types (Type; which includes {Self-supervision, Supervision, Formula-supervision}). We employed CIFAR-10 (C10), CIFAR-100 (C100), ImageNet-1k (IN1k), Places-365 (P365), classification set of Pascal VOC 2012 (VOC12) and Omniglot (OG) datasets. The **bold and underlined** values show the best scores, and **bold** values indicate the second best scores

4.2.10 Comparison to Other Pre-trained Datasets

We compared **Scratch** from random parameters, **Places-30 / 365** (Zhou et al. 2017), **ImageNet-100/1k** (ILSVRC'12) (Deng et al. 2008), and **FractalDB-1k/10k** in Table 8. Since the hyperparameters of representative learning configuration are different depending on the publication, we implemented all frameworks fairly with the same parameters and compared the method (FractalDB-1k/10k) to the baselines (Scratch, DeepCluster-10k, Places-30/365, and ImageNet-100/1k). The hyperparameters are already shown in the implementation details.

The proposed FractalDB pre-trained model recorded several good performance rates. We respectively describe them by comparing our Formula-driven Supervised Learning with Scratch, Self-supervised and Supervised Learning.

4.2.11 Comparison to Training from Scratch

FractalDB-1k/10k pre-trained models recorded much higher accuracies than models trained from scratch on relatively small-scale datasets (C10/100, VOC12 and OG). In case of fine-tuning on large-scale datasets (ImageNet-1k/Places-365), the effect of pre-training was relatively small. However, in fine-tuning on Places-365, the FractalDB-10k pre-trained model helped to improve the performance rate which was also higher than ImageNet-1k pre-training (FractalDB-10k 50.8 vs. ImageNet-1k 50.3).

4.2.12 Comparison to Self-Supervised Learning

We assigned DeepCluster-10k (Caron et al. 2018) to compare the automatically generated image categories. The 10k denotes pre-training with 10k categories. We believe that the auto-annotation with DeepCluster is the most similar method to our formula-driven image dataset. DeepCluster-10k also assigns the same category to images that have similar image patterns based on K-means clustering. Our FractalDB-1k/10k pre-trained models outperformed DeepCluster-10k on five different datasets, e.g., FractalDB-10k 94.1 versus DeepCluster 89.9 (C10), 77.3 versus DeepCluster-10k 66.9 (C100). Our method is thus superior to DeepCluster-10k which is a self-supervised learning method to learn feature representations in image recognition.

4.2.13 Comparison to Supervised Learning

We compared four types of supervised pre-training (e.g., ImageNet-1k and Places-365 datasets and their limited categories ImageNet-100 and Places-30 datasets). ImageNet-100 and Places-30 are subsets of ImageNet-1k and Places-365. The numbers correspond to the number of categories. At the beginning, our FractalDB-10k surpassed the ImageNet-100/Places-30 pre-trained models on all fine-tuning datasets. The results show that our framework is more effective than pre-training with subsets from ImageNet-1k and Places-365.

Table 9 Other architectures

Arch.	PT	PTimg	Type	C10	C100	IN1k	P365	VOC12	OG
AlexNet	Scratch	–	–	88.2	64.5	56.5	49.7	61.5	15.7
	P365	Nat. Img	Supervision	91.9	72.5	46.3	–	N/A	22.0
	IN1k	Nat. Img	Supervision	93.6	76.0	–	50.5	77.9	23.3
	F1k	Formula	Formula-supervision	89.4	65.91	59.0	50.5	65.5	25.2
	F10k	Formula	Formula-supervision	N/A	N/A	57.6	49.6	52.8	21.5
ResNet-18	Scratch	–	–	89.3	67.5	69.7	51.6	64.2	0.5
	P365	Nat. Img	Supervision	93.2	73.4	65.5	–	76.7	14.7
	IN1k	Nat. Img	Supervision	95.7	81.6	–	50.2	83.4	20.9
	F1k	Formula	Formula-supervision	92.1	72.4	65.2	50.7	64.2	33.0
	F10k	Formula	Formula-supervision	91.9	72.7	64.8	51.7	70.7	30.9
ResNet-50	Scratch	–	–	87.6	62.7	76.1	49.9	58.9	1.1
	P365	Nat. Img	Supervision	94.2	76.9	71.4	–	78.6	10.5
	IN1k	Nat. Img	Supervision	96.8	84.6	–	50.3	85.8	17.5
	F1k	Formula	Formula-supervision	93.4	75.7	70.3	49.5	58.9	20.9
	F10k	Formula	Formula-supervision	94.1	77.3	71.5	50.8	73.6	29.2
ResNet-152	Scratch	–	–	85.2	59.5	78.3	48.2	55.1	1.6
	P365	Nat. Img	Supervision	94.9	77.8	71.4	–	55.4	15.8
	IN1k	Nat. Img	Supervision	97.7	86.8	–	49.6	87.4	20.1
	F1k	Formula	Formula-supervision	93.7	75.7	71.8	54.3	55.1	23.4
	F10k	Formula	Formula-supervision	94.3	78.0	72.1	54.4	73.2	30.6
ResNeXt-101	Scratch	–	–	85.3	59.7	68.7	46.9	55.9	2.4
	P365	Nat. Img	Supervision	94.8	78.3	72.3	–	55.0	22.6
	IN1k	Nat. Img	Supervision	96.0	81.2	–	48.1	81.6	25.5
	F1k	Formula	Formula-supervision	94.0	77.1	72.7	48.9	55.9	24.0
	F10k	Formula	Formula-supervision	94.3	78.3	72.3	48.6	73.4	27.6
DenseNet-161	Scratch	–	–	90.9	69.4	77.1	51.0	63.8	2.7
	P365	Nat. Img	Supervision	95.3	78.8	73.6	–	79.3	25.6
	IN1k	Nat. Img	Supervision	97.6	86.7	–	51.8	86.4	28.5
	F1k	Formula	Formula-supervision	94.0	77.0	72.3	51.1	74.1	23.5
	F10k	Formula	Formula-supervision	94.5	78.8	72.6	52.0	73.1	28.9

We list AlexNet, ResNet-{18, 50, 152}, ResNeXt-101, and DenseNet-161 architectures. We compare the FractalDB-1k/10k (F1k/F10k) with training from scratch (Scratch), Places-365/ImageNet-1k pre-trained models (P365 / IN1k). The datasets used are as per Table 8. The **bold and underlined** values show the best scores, and **bold** values indicate the second best scores

We compare the supervised pre-training methods that currently represent the most promising pre-training approach. Although our FractalDB-1k/10k is not superior to them in all settings, our method partially outperformed the ImageNet-1k pre-trained model on Places-365 (FractalDB-10k 50.8 vs. ImageNet-1k 50.3) and Omniglot (FractalDB-10k 29.2 vs. ImageNet-1k 17.5) and Places-365 pre-trained model on CIFAR-100 (FractalDB-10k 77.3 vs. Places-365 76.9) and ImageNet (FractalDB-10k 71.5 vs. Places-365 71.4). The ImageNet-1k pre-trained model is much better than our proposed method on fine-tuning datasets such as C100 and VOC12 since these datasets contain similar categories such as animals and tools.

4.2.14 Comparison with Other Architecture Ablations

We further compare the proposed pre-trained models in several architectures. We assigned eight representative architectures, namely, AlexNet, VGGNet-{16, 19}, ResNet-{18, 50, 152}, ResNeXt-101, and DenseNet-161. The results are shown in Table 9. However, during the experiment, we could not optimize FractalDB pre-trained VGGNet-{16, 19}. Therefore, accuracies with VGGNet-{16, 19} are not included in the table.

In the ResNet-family architectures on ResNets, ResNeXt-101, and DenseNet-161, we confirmed a similar tendency. The FractalDB pre-trained models achieved the top accuracies on OG and Places-365, better results on C100. From

the results on C10, the FractalDB pre-trained models seem to increase the performance rates depending on the deeper layers, from 18 to 152. On the other hand, the FractalDB pre-trained AlexNet also assists with fine-tuning on the ImageNet-1k dataset. The gap between scratch and FractalDB pre-training was +2.5 pt (FractalDB-10k, 59.0, vs. Scratch, 56.5). According to the experiments on several CNN architectures, the proposed FractalDB is effective in the pre-training phase.

Fig. 6 The relationship between label noise and accuracy. In this experiment, noise 0% and 100% respectively mean normal FractalDB pre-training and fully randomized FractalDB pre-training. We show the transitions up-to 1000 iterations, therefore, the maximum accuracy is around 80% in the case of ‘Noise 0%’

4.3 Explorative Study

We also validated the proposed framework in terms of (i) category assignment, (ii) convergence speed, (iii) freezing parameters in fine-tuning, (iv) comparison to other formula-driven image datasets, (v) model ensemble, (vi) recognized category analysis, and (vii) visualization of first convolutional filters and attention maps.

4.3.1 Category assignment (see Fig. 6 and Table 10)

At the beginning, we validated whether the optimization can be successfully performed using the proposed FractalDB. Figure 6 shows how the pre-training accuracy varies as a function of label noise. We randomly replaced the category labels. Here, 0% and 100% noise indicate normal training and fully randomized training, respectively. According to the results on FractalDB-1k, a CNN model can successfully classify fractal images, which are defined by iterated functions. Moreover, well-defined categories with a balanced pixel rate allow optimization on FractalDB. When fully randomized labels were assigned in FractalDB training, the architecture could not classify any images and the loss value was static (the accuracies are 0% at most). According to the result, we confirmed that the effect of the fractal category is reliable enough to train the image patterns.

Moreover, we used DeepCluster-10k to automatically assign categories to the FractalDB. Table 10 shows the comparison between category assignment with DeepCluster-10k

Table 10 The classification accuracies of the FractalDB-1k/10k (F1k/F10k) and DeepCluster-10k (DC-10k)

Dataset	Category ^(classification) %						
Mtd	PT Img	C10	C100	IN1k	P365	VOC12	OG
DC-10k	Natural	89.9	66.9	66.2	51.2	67.5	15.2
DC-10k	Formula	83.1	57.0	65.3	53.4	60.4	15.3
F1k	Formula	93.4	75.7	70.3	49.5	58.9	20.9
F10k	Formula	94.1	77.3	71.5	50.8	73.6	29.2

Bold values indicate top-ranked scores in the experimental results
Mtd/PT Img means Method and Pre-trained images

Table 11 Freezing parameters

Freezing layer(s)	C10	C100	IN100	P30
w/o freezing	93.4	75.7	82.7	75.9
Conv1	92.3	72.2	77.9	74.3
Conv1, Layer1	92.0	72.0	77.5	72.9
Conv1, Layer1–2	89.3	68.0	71.0	68.5
Conv1, Layer1–3	82.7	56.2	55.0	58.3
Conv1, Layer1–4	49.4	24.7	21.2	31.4

We fixed the first convolutional filter (Conv1) and layers (Layer 1–4) in ResNet-50 architecture. The w/o freezing means that the full-layer update which is the same as fine-tuning

Table 12 Other formula-driven image datasets with Bezier Curves DataBase (BCDB) and Perlin Noise DataBase (PNDB) in addition to FractalDB (FDB).

Pre-training	#img	C10	C100	IN100	P30
Scratch	0	87.6	60.6	75.3	70.3
BCDB-144	144k	87.6	62.5	72.7	73.5
BCDB-1024	1.024M	89.7	68.1	73.0	73.6
PNDB-100	100k	90.9	70.2	73.0	73.3
PNDB-1296	1.296M	90.4	71.1	79.7	74.2
FDB-1k	1.0M	93.4	75.7	82.7	75.9

We also list the number of pre-training image (#img)

Bold values indicate top-ranked scores in the experimental results

(k-means) and FractalDB-1k/10k (IFS). We confirm that DeepCluster-10k cannot successfully assign a category to fractal images. The gaps between IFS and k-means assignments are {11.0, 20.3, 13.2} on {C10, C100, VOC12}. This clearly indicates that our category assignments in FDSL, through the principle of IFS and the parameters in equation (2), work well compared to DeepCluster-10k.

4.3.2 Convergence Speed (see Fig. 2b)

The transitioned pre-training accuracies values in FractalDB are similar to those of ImageNet pre-trained model and much faster than scratch from random parameters (Fig. 2b). We validated the convergence speed in fine-tuning on C10. As a result of pre-training with FractalDB-1k, we accelerated the convergence speed in fine-tuning which is similar to the ImageNet pre-trained model. According to the findings on pre-training in He *et al.* He et al. (2019), the FractalDB pre-training can also promotes faster transfer learning on additional datasets.

4.3.3 Freezing Parameters in Fine-Tuning (see Table 11)

Although full-parameter fine-tuning is better, conv1 and 2 acquired a highly accurate image representation (Table 11). Freezing the conv1 layer provided a -1.1 (92.3 vs. 93.4) or -3.5 (72.2 vs. 75.7) decrease from fine-tuning on C10 and C100, respectively. Compared to the other results, such as those for conv1–4/5 freezing, the bottom layer tended to train a better representation. The FractalDB pre-training did not learn from natural images; therefore, the fixed layers fine-tuning is not effective. The FractalDB pre-trained model must train middle layers to acquire natural image representations in the fine-tuning phase.

Fig. 7 Twenty-model ensemble with FractalDB-1k

4.3.4 Comparison to other Formula-Driven Image Datasets (see Table 12)

Thus far, the proposed FractalDB-1k/10k are better than other formula-driven image datasets. We used Perlin noise (Perlin 2002) and Bezier curves (Farin 1993) to generate image patterns and their categories just as FractalDB dataset.

We confirmed that both Perlin noise and Bezier curves are also beneficial in terms of making a pre-trained model which can achieve better rates than training from scratch. However, the proposed FractalDB is better than these approaches (Table 12). For a fairer comparison, we cite a similar #category in the formula-driven image datasets, namely FractalDB-1k (total #image: 1M), Bezier-1024 (1.024M) and Perlin-1296 (1.296M). The significantly improved rates are +3.0 (FractalDB-1k 93.4 vs. Perlin-1296 90.4) on C10, +4.6 (FractalDB-10k 75.7 vs. Perlin-1296 71.1) on C100, +3.0 (FractalDB-1k 82.7 vs. Perlin-1296 79.7) on IN100, and +1.7 (FractalDB-1k 75.9 vs. Perlin-1296 74.2) on P30.

4.3.5 Ensemble Model (see Fig. 7)

The FractalDB pre-trained model helps to improve accuracy with a model ensemble in addition to a single model. Figure 7 shows the results for a 20-model ensemble with FractalDB-1k. The final accuracy reaches 94.7/79.3 on C10/C100 datasets.

4.3.6 Recognized Category Analysis (see Table 13)

We investigated which categories are better recognized by the FractalDB pre-trained model compared to the ImageNet pre-trained model. Table 13 shows the category names and classification rates. The FractalDB pre-trained model tends to be better when an image contains recursive patterns (e.g., a keyboard, maple trees).

4.3.7 Visualization of First Convolutional Filters (see Fig. 8a–e) and Attention Maps (see Fig. 8f)

We visualized first convolutional filters and Grad-CAM (Selvaraju et al. 2017) with pre-trained ResNet-50. As seen in ImageNet-1k/Places-365 / DeepCluster-10k (Fig. 8a, b, e) and FractalDB-1k/10k pre-training (Fig. 8c, d), our pre-trained models clearly generate different feature representations from conventional natural image datasets. Based on the experimental results, we confirmed that the proposed FractalDB successfully pre-trained a CNN model without any natural images even though the convolutional basis filters are different from the natural image pre-training with ImageNet-1k/DeepCluster-10k.

The pre-trained models with Grad-CAM can generate heatmaps fine-tuned on the C10 dataset. According to the

center-right and right in Fig. 8f, the FractalDB-1k/10k also look at the objects.

5 Discussion and Conclusion

We achieved *pre-training without natural images* through Formula-Driven Supervised Learning (FDSL) based on fractals. We successfully pre-trained models on FractalDB and fine-tuned the models on several representative datasets, including CIFAR-10/100, ImageNet, Places-365 and Pascal VOC. The performance rates were higher than those of models trained from scratch and some supervised / self-supervised learning methods.

5.1 Towards a Better Pre-Trained Dataset

The proposed FractalDB pre-trained model partially outperformed ImageNet-1k/Places-365 pre-trained models, e.g., FractalDB-10k 77.3 vs. Places-365 76.9 on CIFAR-100, FractalDB-10k 50.8 vs. ImageNet-1k 50.3 on Places-365. If we could improve the transfer accuracy of the pre-training without natural images, then the ImageNet dataset and the pre-trained model may be replaced so as to protect fairness, preserve privacy, and decrease annotation labor. Recently, for example, 80M Tiny Images² and ImageNet (human-related categories)³ withdrew public access. We complementarily update our framework with similar research topics such as self-supervised learning and unsupervised feature representation learning. We generated surprising results from FDSL without any natural images in the sense of natural-image-like data representations, such as geometric viewpoint changes and smoothly jointed pixels in images.

5.2 A Different Image Representation From Human Annotated Datasets

The visual patterns pre-trained by FractalDB acquire a unique feature in a different way from ImageNet-1k (see Fig. 8). The FractalDB pre-trained model can acquire a good representation to understand natural images even though there are no natural images in the pre-training phase. In the future, steerable pre-training may be available depending on the fine-tuning task. Due to the characteristics of automatically generated datasets, we can create any labels, e.g., geometric representations, centroids/bounding boxes, and area segments. Through our experiments, we confirm that the parameter tuning and configuration search approaches are effective to enhance the performance for fine-tuning on natural image datasets. We hope that the proposed pre-training

² <https://groups.csail.mit.edu/vision/TinyImages/>.

³ <http://image-net.org/update-sep-17-2019>.

Table 13 Performance rates in which FractalDB was better than the ImageNet pre-trained model on C10/C100/IN100/P30 fine-tuning

Dataset	Category ^{(rates)%}
C10	–
C100	Bee ^(89versus87) , chair ^(92versus89) , keyboard ^(95versus93) , maple tree ^(72versus71) , motorcycle ^(99versus95) , orchid ^(92versus90) , pine tree ^(70versus69)
IN100	Kerry blue terrier ^(88versus87) , marmot ^(92versus90) , giant panda ^(92versus91) , television ^(80versus79) , dough ^(64versus62) , valley ^(94versus93)
P30	cliff ^(64versus62) , mountain ^(40versus27) , skyscraper ^(85versus84) , tundra ^(79versus77)

We listed the performance rates with FractalDB-1k versus ImageNet pre-trained models

Fig. 8 Visualization results: **a–e** show the activation of the 1st convolutional layer on ResNet-50, and **f** illustrates attentions with Grad-CAM (Selvaraju et al. 2017)

framework will be amenable to a broader range of tasks, e.g., object detection and semantic segmentation, and will become a flexibly generated pre-training dataset.

5.3 Are Fractals a Good Rendering Formula?

We are looking for better mathematically generated image patterns and their categories. We confirmed that FractalDB is

better than datasets based on Bezier curves or Perlin noise in the context of the pre-trained model (see Table 12). Moreover, the proposed FractalDB can generate a good set of categories, e.g., the fact that the training accuracy decreased depending on the label noise (see Fig. 6) and the formula-driven image generation is better than DeepCluster-10k in most cases, as a method for category assignment (see Table 10) show how the fractal categories worked well. According to the experi-

ments conducted herein, the FractalDB pre-trained model is the most effective method by comparing with PerlinNoiseDB and BezierCurveDB. However, there is scope to improve the image representation and use a better rendering engine. We believe that the framework has great potential. The FDSL does not require natural images taken by a camera, manual category definition and assignment, or the burden of annotation labor. Moreover, in order to construct a large-scale pre-training dataset, the framework is not limited to use fractal geometry. Any mathematical formulas, natural laws, and rendering functions can be employed to create image patterns and their image labels in the automatically created dataset.

Acknowledgements 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 No. JP19H01134. AI Bridging Cloud Infrastructure (ABCI) provided by the National Institute of Advanced Industrial Science and Technology (AIST) was used.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Asano, Y.M., Rupprecht, C., & Vedaldi, A. (2020). A critical analysis of self-supervision, or what we can learn from a single image. In international conference on learning representation (ICLR).
- Asano, Y.M., Rupprecht, C., & Vedaldi, A. (2020). Self-labelling via simultaneous clustering and representation learning. In international conference on learning representation (ICLR).
- Barnsley, M. F. (1988). *Fractals everywhere*. New York: Academic Press.
- Birhane, A., & Prabhu, V.U. (2021). Large image datasets: A pyrrhic win for computer vision? Winter conference on applications of computer vision (WACV).
- Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In 19th international conference on computational statistics (COMPSTAT), pp. 177–187.
- Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In European conference on computer vision (ECCV), pp. 132–149.
- Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In International conference on machine learning (ICML).
- Chen, Y. Q., & Bi, G. (1997). 3-D IFS fractals as real-time graphics model. *Computers and Graphics*, 21(3), 367–370.
- Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In The IEEE international conference on computer vision and pattern recognition (CVPR), pp. 248–255.
- Doersch, C., Gupta, A., & Efros, A. (2015). Unsupervised visual representation learning by context prediction. In The IEEE international conference on computer vision (ICCV), pp. 1422–1430.
- Donahue, J., Jia, Y., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In International conference on machine learning (ICML) pp. 647–655.
- Donahue, J., & Simonyan, K. (2019). Large scale adversarial representation learning. In arXiv pre-print [arXiv:1907.02544](https://arxiv.org/abs/1907.02544).
- Dwivedi, D., Misra, I., & Hebert, M. (2017). Cut, paste and learn: Surprisingly easy synthesis for instance detection. In International conference on computer vision (ICCV).
- Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2015). The Pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision (IJCV)*, 111(1), 98–136.
- Falconer, K. (2004). *Fractal geometry: Mathematical foundations and applications*. New Jersey: John Wiley and Sons.
- Farin, G. (1993). *Curves and surfaces for computer aided geometric design: A practical guide*. Cambridge: Academic Press.
- Fellbaum, C. (1998). WordNet: An electronic lexical database. BradfordBooks.
- Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In International conference on learning representation (ICLR).
- He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In The IEEE international conference on computer vision and pattern recognition (CVPR).
- He, K., Girshick, R., & Dollár, P. (2019). Rethinking ImageNet pre-training. In The IEEE international conference on computer vision (ICCV), pp. 4918–4927.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In The IEEE international conference on computer vision and pattern recognition (CVPR), pp. 770–778.
- Howard, A.G., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan M. and Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., & Adam, H. (2019). Searching for MobileNetV3. In The IEEE international conference on computer vision (ICCV), pp. 1314–1324.
- Howard, A.G., Zhu M., C.B., Kalenichenko, D., Wang, W., Weyand, T., An-dreetto, M., & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv pre-print [arXiv:1704.04861](https://arxiv.org/abs/1704.04861).
- Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2020). Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42, 2011–2023.
- Huang, G., Liu, Z., Maaten, L.V.d., & Weinberger, K.Q. (2017). Densely connected convolutional networks. In The IEEE international conference on computer vision and pattern recognition (CVPR), pp. 4700–4708.
- Huh, M., Agrawal, P., & Efros, A.A. (2016). What makes ImageNet good for transfer learning? In Advances in neural information processing systems NIPS 2016 Workshop.
- Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., M., S., & Zisserman, A. (2017). The kinetics human action video dataset. arXiv pre-print [arXiv:1705.06950](https://arxiv.org/abs/1705.06950).
- Kornblith, S., Shlens, J., & Le, Q.V. (2019). Do better imagenet models transfer better? In The IEEE international conference on computer vision and pattern recognition (CVPR), pp. 2661–2671.
- Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Hajja, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Kamali, S., Mallochi, M., Pont-Tuset, J., Veit, A., Belongie, S., Gomes, V., Gupta, A., Sun, C., Chechik, G., Cai, D., Feng, Z., Narayanan, D., &

- Murphy, K. (2017). OpenImages: A public dataset for large-scale multi-label and multi-class image classification.
- Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.
- Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (eds.) *Advances in neural information processing systems (NIPS)* 25, pp. 1097–1105.
- Lake, B. M., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266), 1332–1338.
- Landini, G., Murry, P.I., & Misson, G.P. (1995). Local connected fractal dimensions and lacunarity analyses of 60 degree fluorescein angiograms. In: *Investigative Ophthalmology and Visual Science*, pp. 2749–2755.
- Larsson, G., Maire, M., & Shakhnarovich, G. (2017). FractalNet: Ultra-deep neural networks without residuals. In *International conference on learning representation (ICLR)*.
- Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollar, P., & Zitnick, C.L. (2014). Microsoft COCO: Common objects in context. In *European conference on computer vision (ECCV)*, pp. 740–755.
- Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Barambe, A., & Maaten, L.v.d. (2018). Exploring the limits of weakly supervised pretraining. In *European conference on computer vision (ECCV)*, pp. 181–196.
- Mandelbrot, B. (1983). The fractal geometry of nature. *American Journal of Physics*, 51(3)
- Monfort, M., Andonian, A., Zhou, B., Ramakrishnan, K., Adel Bargal, S., Yan, T., Brown, L., Fan, Q., Gutfreund, D., Vondrick, C., & Oliva, A. (2019). Moments in time dataset: One million videos for event understanding. In *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*.
- Monro, D.M., & Budbridge, F. (1995). Rendering algorithms for deterministic fractals. In *IEEE computer graphics and its applications*, pp. 32–41.
- Movshovitz-Attias, Y., Kanade, T., & Sheikh, Y. (2016). How useful is photo-realistic rendering for visual learning? In *European conference on computer vision (ECCV)*.
- Noroozi, M., & Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision (ECCV)*.
- Noroozi, M., Pirsavash, H., & Favaro, P. (2017). Representation learning by learning to count. In *The IEEE international conference on computer vision (ICCV)*, pp. 5898–5906.
- Noroozi, M., Vinjimoor, A., Favaro, P., & Pirsavash, H. (2018). Boosting self-supervised learning via knowledge transfer. In *The IEEE International conference on computer vision and pattern recognition (CVPR)*, pp. 9359–9367.
- Oord, A.v.d., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.
- Pentland, A. P. (1984). Fractal-based description of natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 6(6), 661–674.
- Perez, P., Gangnet, M., & Blake, A. (2003). Poisson image editing. *ACM Transactions on Graphics (TOG)* 22
- Perlin, K. (2002). Improving noise. *ACM Transactions on Graphics (TOG)*, 21(3), 681–682.
- Remez, T., Huang, J., & Brown, M. (2018). Learning to segment via cut-and-paste. In *European conference on computer vision (ECCV)*.
- Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., & Chen, L.C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. Mobile networks for classification, detection and segmentation. *arXiv pre-print arXiv:1801.04381*.
- Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *The IEEE international conference on computer vision (ICCV)*, pp. 618–626.
- Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., & Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. In *IEEE international conference on computer vision and pattern recognition (CVPR)*.
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International conference on learning representations (ICLR)*.
- Smith, T. G. J., & Marks, W. B. (1996). Fractal methods and results in cellular morphology - dimensions, lacunarity and multifractals. *Journal of Neuroscience Methods*, 69(2), 123–136.
- Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *The IEEE international conference on computer vision (ICCV)*, pp. 843–852.
- Sundermeyer, M., Marton, Z.C., Durner, M., Brucker, M., & Triebel, R. (2018). Implicit 3D orientation learning for 6D object detection from RGB images. In *European conference on computer vision (ECCV)*.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *The IEEE international conference on computer vision and pattern recognition (CVPR)*, pp. 1–9.
- Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *International conference on intelligent robots and systems (IROS)*.
- Torralba, A., Fergus, R., & Freeman, W.T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*.
- Varma, M., & Garg, R. (2007). Locally invariant fractal features for statistical texture classification. In *The IEEE international conference on computer vision (ICCV)*, pp. 1–8.
- Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M.J., Laptev, I., & Schmid, C. (2017). Learning from synthetic humans. In *The IEEE international conference on computer vision and pattern recognition (CVPR)*, pp. 109–117.
- Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *The IEEE international conference on computer vision and pattern recognition (CVPR)*, pp. 1492–1500.
- Xu, Y., Ji, H., & Fermuller, C. (2009). Viewpoint invariant texture description using fractal analysis. *International Journal of Computer Vision (IJCV)*, 83(1), 85–100.
- Yang, K., Qinami, K., Fei-Fei, L., Deng, J., & Russakovsky, O. (2020). Towards fairer datasets: Filtering and balancing the distribution of the people subtree in the ImageNet hierarchy. In *Conference on fairness, accountability and transparency (FAT)*.
- Zhang, R., Isola, P., & Efros, A.A. (2016). Colorful image colorization. In *European conference on computer vision (ECCV)*.
- Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2017). Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40, 1452–1464.