Formula-Supervised Visual-Geometric Pre-training – Supplementary Material –

Ryosuke Yamada*[1,2], Kensho Hara*[1], Hirokatsu Kataoka[1], Koshi Makihara[1], Nakamasa Inoue[1,3], Rio Yokota[1,3], and Yutaka Satoh[1,2]

[1] AIST, [2] University of Tsukuba, [3] Tokyo Institute of Technology
* equal contribution

Project Page; https://ryosuke-yamada.github.io/fdsl-fsvgp/

A FSVGP details

This section describes the details of our Formula-Supervised Visual-Geometric Pre-training (FSVGP). Section A.1 details the Visual Geometric Fractal Database (VG-FractalDB). Section A.2 details a unified model for pre-training VG-FractalDB.

A.1 VG-FractalDB construction details

This section delineates the methodology employed in constructing the VG-FractalDB, focusing on using 3D Iterated Function Systems (3D-IFS) [1] and our dataset diversity and consistency between visual and geometric modalities.

3D-IFS is a mathematical framework for generating fractal geometry. It is central to defining the categories and variations in VG-FractalDB. Formula-supervised consistency labels in VG-FractalDB are linked to the 3D-IFS parameters. In certain 3D-IFS parameter cases, the 3D fractal point cloud is concentrated in a part of the 3D space. Therefore, the quality of the 3D fractal point cloud is checked based on the variance threshold to exclude such 3D fractal point clouds. Only the 3D fractal point clouds whose variance value exceeds the variance threshold value in all axes are defined as the categories of VG-FractalDB. The variance threshold ensures a wide variety of fractal shapes. For augmenting within each category, we used FractalNoiseMix proposed by Yamada et al. [8]. This augmentation technique enriches the dataset with a broader range of fractal geometries by augmenting 3D fractal models by mixing other 3D fractal models.

The 3D fractal models are then projected onto 2D planes to generate fractal images. This process randomly selects a camera viewpoint in 3D space. A perspective projection transformation maps point clouds onto a 2D plane. This particular transformation is chosen to accurately maintain the relative size and shape of 3D objects in the 2D rendering. Each parameter must be defined to achieve a realistic projection, such as the viewing angle (focal length), aspect ratio, and near and far planes. We set the focal length to 45 degrees, the aspect ratio to 1.0, and the near and far planes to 1.0 and 100, respectively. The camera

viewpoint setting is also an integral part of the projection process. This involves determining the camera's position, the point it is looking at, and its upward direction. These elements are used to compute a view matrix, which transforms the 3D objects from the world coordinate system to the camera coordinate system.

For each 3D fractal model, a corresponding fractal image is generated from a randomly selected viewpoint. This approach ensures that each pair of 3D fractal point clouds and fractal images uniquely represents a particular viewpoint. The resulting VG-FractalDB provides a rich 2D-3D fractal data representation for classification pre-training.

A.2 Pre-training transformer model details

We designed a single transformer model for learning VG-FractalDB. Our transformer model is built upon the standard Vision Transformer (ViT) [2] and Point Transformer (PointT) [9] structure, comprising transformer blocks. Each block includes a multi-head self-attention mechanism and a Multi-Layer Perceptron (MLP) block integrated with LayerNorm for normalization.

The property of our single transformer model is to process both fractal images and 3D fractal point clouds through different embedding procedures tailored to the nature of each data type. For images, the image is then divided into patches of size 16×16, with each patch undergoing a linear projection to transform it into an embedding. For point cloud data, we start by downsampling a point cloud to a specific number of points. The downsampling point cloud is then clustered using a K-nearest neighbor, ensuring that local geometries within the cloud are preserved. These clustered points are passed through an MLP, generating embeddings.

Our transformer model is designed to be simple, learning visual-geometric representation from VG-FractalDB. Using distinct embedding processes for different data types showcases our transformer model's flexibility and potential to adapt diverse downstream tasks.

B Experimental setting details

This section describes the experimental setup in detail. First, Section B.1 describes the training setup in FSVGP. Sections B.2 and Section B.3 describe the experimental setup for image recognition and 3D object recognition, respectively. Finally, Section B.4 explains in detail the setup of the ablation study.

B.1 Pre-training

Our experiments set the hyperparameters based on the Data-efficient image Transformers (DeiT) model [7], as detailed in Table A. The training scripts were adapted from previous studies [6], providing a foundational framework for our approach.

Table A: Pre-training setting.

Config	Value	
	VG-FractalDB-1k	VG-FractalDB-21k
Epochs	200	100
Batch Size	1024	8192
Optimizer	AdamW	AdamW
LR	5e-4	5e-4
Weight Decay	0.05	0.05
LR Scheduler	Cosine decay	Cosine decay
Warmup Steps	5k	5k
Resolution	224×224	224×224
Label Smoothing	0.1	0.1
Drop Path	0.1	0.1
Rand Augment	9 / 0.5	9 / 0.5
Mixup	0.8	0.8
Cutmix	1.0	1.0
Erasing	0.25	0.25

Table B: Image classification setting.

Config	Value	
	ImageNet-1k	Others
Epochs	300	300
Batch Size	1024	1024
Optimizer	AdamW	AdamW
LR	5e-4	5e-4
Weight Decay	0.05	0.05
LR Scheduler	Cosine decay	Cosine decay
Warmup Steps	5 (epoch)	5 (epoch)
Resolution	224×224/384 × 384	224×224
Label Smoothing	0.1	0.1
Drop Path	0.1	0.1
Rand Augment	9 / 0.5	9 / 0.5
Mixup	0.8	0.8
Cutmix	1.0	1.0
Erasing	0.25	0.25

Table C: Image object detection and instance segmentation setting.

Config	Value	
	From Scratch	Pre-train
Epochs	30	30
Batch Size	16	16
Optimizer	AdamW	AdamW
LR	1.6e-4	4e-1
Weight Decay	0.2	0.1
Warmup Steps	1k	1k
Resolution	1024×1024	1024×1024
Drop Path	0.1/0.4	0.1/0.4
Large Scale Jitter	[0.1, 2.0]	[0.1, 2.0]
Rand Flip	0.5	0.5

B.2 Image recognition

Image classification. Our experiments validated our results using the image classification dataset that previous studies evaluated. We compare the top-1 accuracy during fine-tuning in 300 epochs as an evaluation metric. Hyperparameters at additional learning are shown in Table B. These are the same conditions as in the previous experimental setup in FDSL [6].

Image object detection and instance segmentation. This experiment was validated at MS COCO2017 using the official ViTDet [5] GitHub. We used the hyperparameters of ViTDet as they are. The specific hyperparameters for the fine-tuning are shown in Table C.

Table D: 3D object classification setting.

Config	Value	
	VG-FractalDB	Others
Epochs	300	300
Batch Size	32	32
Optimizer	AdamW	AdamW
LR	5e-4	5e-4
Weight Decay	0.05	0.05
LR Scheduler	Cosine decay	Cosine decay
Warmup Steps	10 (epoch)	10 (epoch)
Num. of Points	1024(M)/2048(S)	1024(M)/2048(S)
Num of Patches	64	64
Patch Size	32	32
Augmentation	ScaleAndTranslate	ScaleAndTranslate

Table E: 3D object detection and parts segmentation setting.

Config	Value	
	ScanNet	ShapeNet-parts
Epochs	1080	300
Batch Size	32	32
Optimizer	AdamW	AdamW
LR	4e-4	5e-4
Weight Decay	0.1	0.05
LR Scheduler	Linear warmup	Cosine decay
Warmup Steps	20 (epoch)	10 (epoch)
Num. of Points	40000	2048
Num of Query/Patches	256	64
Patch Size	–	32
Augmentation	RandomCuboid	ScaleAndTranslate

B.3 3D object recognition

3D object classification. We used ModelNet40 and ScanObjectNN. The evaluation was conducted on ModelNet40 and three ScanObjectNN subsets: OBJ-BG (including object surroundings), OBJ-ONLY (objects without background), and PB-T50-RS (a challenging subset with translated, rotated, and scaled objects). We employed the AdamW optimizer for fine-tuning and adjusted over 300 epochs using a cosine decay schedule. Models were fine-tuned on point clouds with 1024 points for ModelNet40 and 2048 points for ScanObjectNN, and performance was measured using overall accuracy, focusing on the highest accuracy achieved within 300 epochs. The specific hyperparameters for the fine-tuning are shown in Table D.

Few-shot learning. We conducted experiments by selecting K classes from the ModelNet40 dataset and sampling $N + 20$ objects from each class. These classes formed the basis for K-way, N-shot training subsets, with K and N varying between $\{5, 10\}$ and $\{10, 20\}$, respectively. We created ten different subsets for these experiments and evaluated the model's performance by computing the mean and standard deviation of the highest accuracy obtained across these subsets. The AdamW optimizer was used during fine-tuning, adjusting it according to a cosine decay schedule over 150 epochs. We fine-tuned the model on ModelNet40 using point clouds of 1024 points each.

3D object detection. In our 3D object detection experiment, the ScanNet was used as a benchmark. We adopted the 3DETR model to fine-tune our 3D object detection approach, using its PointT-Small backbone network. The hyperparameters were tuned to those used in the original 3DETR. Our evaluation metrics were based on mean average precision (mAP) at 25% and 50% intersection over union (IoU). The specific fine-tuning hyperparameters are shown in Table E.

Parts segmentatoin. We employed the ShapeNetPart dataset to evaluate part segmentation, which involves identifying detailed class labels for each point of a 3D model. We assessed performance using the mean IoU (mIoU_{ins}) across all instances and IoU for each category. Furthermore, we reported the Mean IoU across all categories (mIoU_{cat}), ensuring equal treatment of each category in the dataset, irrespective of its frequency. This approach provides a comprehensive overview of the model's segmentation performance. The specific hyperparameters for the fine-tuning are shown in Table E.

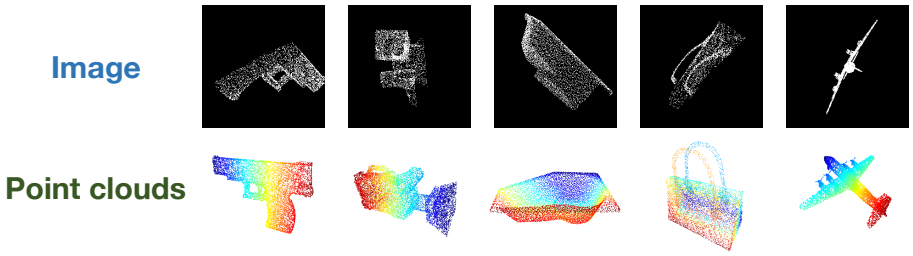

Fig. A: The examples of image and point cloud pair data in ShapeNet.

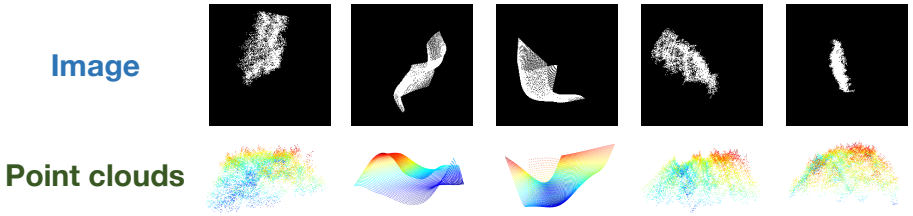

Fig. B: The examples of image and point cloud pair data in the Visual-Geometric Perlin Noise dataset.

B.4 Ablation study

(i) Which is more effective, fractal point clouds or CAD models in FSVGP? In this experiment, we tested the pre-training effect of FSVGP by applying it to an existing 3D dataset, ShapeNet. We generated images and point clouds for ShapeNet based on the VG-FractalDB construction procedure. Specifically, we project each 3D model of a ShapeNet onto an image from a random viewpoint position. An example of a generated image and point cloud is shown in Figure A.

(ii) Can other generation rules be effective in FSVGP? In this experiment, we verified the pre-training effect of the generation rules by comparing fractal and Perlin noise in terms of the mathematical formula regularity that generates the data. Perlin noise is a gradient noise function for generating natural-looking textures and shapes, and previous studies [3,4] have reported its effectiveness in generating pre-trained datasets for image and video recognition. Therefore, we employed Perlin noise as the generating function to be compared in this experiment, considering its extensibility to 3D models.

We first generate 2D Perlin noise. Next, we lift the 2D Perlin noise to a point cloud. We then construct the Visual-Geometric Perlin Noise (VG-PN) dataset by projecting the point cloud onto an image. The 2D Perlin noise is pre-defined as a 100×100 grid. Random coordinates are determined within each grid, and a gradient vector is generated from the vertices of each grid based on these coordinates. The values in the grid are determined by linearly complementing the gradient vectors. The key parameters for generating the Perlin noise, the

Table F: Effect of formula supervision.

Shuffle type	CIFAR100	ModelNe40
w/o shuffle	**85.9**	**92.9**
category	84.4	92.7
instance + category	83.5	92.5

Table G: Effect of loss functions.

Loss function	CIFAR100	ModelNet40
CE	**85.9**	**92.9**
VGC	8.4	92.5
CE + VGC	85.4	92.2

frequency, and scale, are varied within a specific range to ensure the diversity of the shape of the 3D Perlin noise. The VG-PN dataset defines these parameters as categories. The 2D Perlin noise is converted to a 3D Perlin noise as a point cloud by taking the values of each grid of the 2D Perlin noise as the Z-coordinate, finally, by projecting the 3D Perlin noise onto a image under the same conditions as VG-FractalDB. Finally, the 3D Perlin noise is projected onto the image under the same conditions as VG-FractalDB to generate the image/point cloud pair data, as shown in Figure B.

C Additional experiments

C.1 What is the pre-training effect of collapsing the pair labels in VG-FractalDB?

This experiment verifies the pre-training effect based on the formula-supervised consistency label. We shuffled each pair of fractal data in VG-FractalDB to make it inconsistent. Specifically, we implement two shuffle methods, named *category* and *instance + category*, which shuffle the categories of 3D fractal point clouds for each category and instance, respectively. Let $\mathbf{I} = \{\mathbf{I}^1, \mathbf{I}^2, \ldots, \mathbf{I}^C\}$ and $\mathbf{X} = \{\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^C\}$ denote the image and pointcloud data, respectively, where C is the number of categories. The instances of images and point clouds in category c are denoted as $\mathbf{I}^c = \{\mathbf{I}^c_1, \mathbf{I}^c_2, \ldots, \mathbf{I}^c_M\}$ and $\mathbf{X}^c = \{\mathbf{X}^c_1, \mathbf{X}^c_2, \ldots, \mathbf{X}^c_M\}$, respectively, where M is the number of instances in each category.

The *category* shuffle randomizes the category indices of point clouds to destroy the consistency of categories for images and point clouds. After *category* shuffle, the instances of point clouds in category c are denoted as $\mathbf{X}^c_{\text{cs}} = \{\mathbf{X}^{c'}_1, \mathbf{X}^{c'}_2, \ldots, \mathbf{X}^{c'}_M\}$, where c' is the shuffled category index. Therefore, the category labels for point cloud data are different from those for image data in the pre-training step, though the labels in each category are consistent for both images and point clouds.

The *instance + category* shuffle randomizes both instance and category indices of point clouds to disrupt the consistency of instances for images and point clouds. After *instance + category* shuffle, the instances of point clouds in category c are denoted as $\mathbf{X}^c_{\text{ics}} = \{\mathbf{X}^{c'_1}_{i'_1}, \mathbf{X}^{c'_2}_{i'_2}, \ldots, \mathbf{X}^{c'_M}_{i'_M}\}$, where c'_j and i'_j are the shuffled category and instance indices for the j-th instance, respectively. Therefore, even the labels in each category are not consistent in point clouds. Note that the shuffling methods exclusively randomize the labels for point cloud data to disrupt the consistency between image and point cloud data. In other words, the labels associated with image data remain unaffected by the shuffling.

Table F shows that FSVGP without shuffling was more effective than *category shuffle* and *instance + category* shuffle in CIFAR100 and ModelNet40. This result shows that the formula-supervised consistency labels used in FSVGP improve the performance of pre-training. The pre-training using the data shuffled by *instance + category* still achieved reasonable results. We believe pre-training on such data optimizes the model towards near-optimal parameters based on consistent image data, even though the shuffled point cloud data may impede convergence. To validate the hypothesis, we examined the loss values both with and without the shuffling. The values for image data were similar (2.45 vs 2.48), whereas the values for point cloud data differed significantly (6.90 vs 1.02). In addition, the shuffling of both visual and geometric modalities disrupted the pre-training, causing a divergence in the loss values.

C.2 Does the standard cross-entropy loss function alone suffice for pre-training in FSVGP?

We contrast two scenarios: one employing CE loss based on the formula-supervised consistency label and another employing cross-entropy loss with a constraint term derived from visual-geometric correspondence (VGC). We developed VGC as consistency labels, representing whether the pair of images and point cloud represent the same instance. We shuffle the point cloud data instances in each category to generate a non-consistent pair. In each epoch of pre-training, we utilize both non-shuffled and shuffled data equally, randomly splitting the dataset in half. VGC calculates the loss values using cross-entropy loss with consistency labels. Table G shows that FSVGP with only CE loss is better than the fine-tuning accuracy with VGC + CE loss. This result finds that FSVGP learns visual-geometric representation with only CE loss rather than explicit visual-geometric correspondence terms such as VGC.

C.3 Evaluation of the performance of pre-training models by linear probing

Our experiment of this paper basically followed the evaluation protocols of previous FDSL studies. However, we believe that it is important to know about the feature representation that the pre-trained models learn through linear probing. Therefore, we investigate the feature representations learned by FSVGP (VG-FractalDB-1k) and MAE (ImageNet). Specifically, we stop the gradient update of some transformer blocks in ViT during fine-tuning and evaluate which transformer block feature representations in ViT contribute to fine-tuning.

We froze the first m blocks of ViT-B during the fine-tuning ($m = 0$ and 12 indicate full fine-tuning and linear probing, respectively). As shown in Figure C, although the difference in data domain between real images and fractal data degenerates the performance of FSVGP in linear probing, the fine-tuning from pre-trained representations significantly improves the performance. This result indicates the meaningful representation learned from FSVGP, especially in early layers.

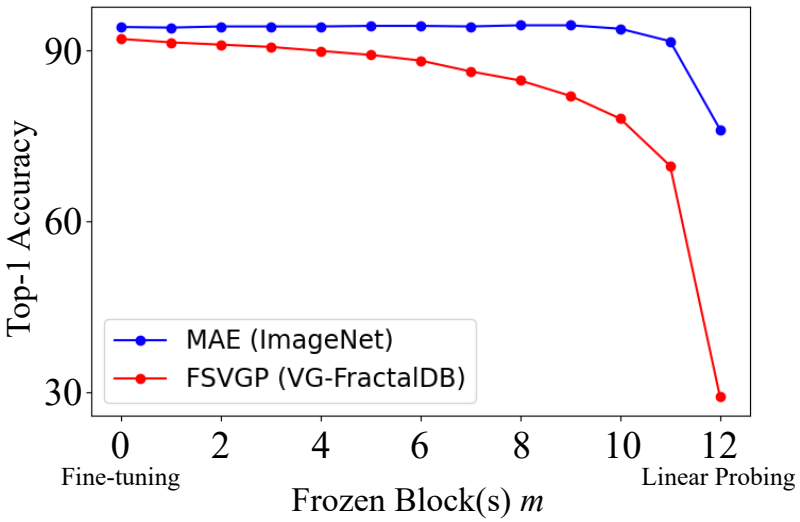

Fig. C: Comparison of classification accuracy when parameter update of each transformer block is frozen during fine-tuning of SVGP (VG-FractalDB-1k) and MAE (ImageNet). We use ViT-B on ImageNet100.

C.4 Multi-modal evaluations in 3D object classification

We consider multi-modal evaluation important for showing the use case of FSVGP. Therefore, We conducted an initial experiment of 3D object classification using images and point clouds on ModelNet40. We confirmed that VG-FractalDB (V + G) outperforms VG-FractalDB (V or G) by +0.2 points and +0.6 points, respectively, when fine-tuning images and point clouds on ModelNet40. This result suggests the potential applications of FSVGP, such as autonomous driving with point clouds and bird's-eye view images.

D Qualitative examples

The visualized predictions of the MS COCO underscore the ability of our FSVGP model to identify and delineate objects with high accuracy in complex scenes. Figure D demonstrates the FSVGP's accuracy in pinpointing object locations and discriminating between overlapping entities in detail-rich images. For example, in Figure D, one can observe the FSVGP's acute precision in detecting and separating a cluster of beans on a plate, demonstrating its ability to locate and distinguish even the smallest objects. In addition, the figure highlights the model's ability to detect overlapping objects, such as a book partially obscured by a houseplant, demonstrating the nuanced recognition capabilities of FSVGP across a wide range of object categories.

Fig. D: FSVGP Success Cases: compare ground truth with training from scratch, MAE, VisualAtom, and FSVGP output results. We use VitDet (ViT-B) on MS COCO 2017 Val.

References

1. Barnsley, M.F.: Fractals everywhere. Academic press (2014) 1
2. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: Proceedings of the International Conference on Learning Representation (ICLR) (2021) 2
3. Inoue, N., Yamagata, E., Kataoka, H.: Initialization using perlin noise for training networks with a limited amount of data. In: Proceedings of the International Conference on Pattern Recognition (ICPR). pp. 1023–1028 (2021) 5
4. Kataoka, H., Hara, K., Hayashi, R., Yamagata, E., Inoue, N.: Spatiotemporal initialization for 3d cnns with generated motion patterns. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 1279–1288 (2022) 5
5. Li, Y., Mao, H., Girshick, R., He, K.: Exploring plain vision transformer backbones for object detection. In: European Conference on Computer Vision (ECCV). pp. 280–296. Springer (2022) 3
6. Takashima, S., Hayamizu, R., Inoue, N., Kataoka, H., Yokota, R.: Visual atoms: Pre-training vision transformers with sinusoidal waves. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 18579–18588 (2023) 2, 3
7. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: Proceedings of the International Conference on Machine Learning (ICML). pp. 10347–10357 (2021) 2
8. Yamada, R., Kataoka, H., Chiba, N., Domae, Y., Ogata, T.: Point cloud pre-training with natural 3d structures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 21283–21293 (2022) 1

9. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 16259–16268 (2021) 2