Supplementary Materials for "Scaling Backwards: Minimal Synthetic Pre-training?"

This document is the supplementary materials for the paper titled "Scaling Backwards: Minimal Synthetic Pre-training?". In Section A, we provide visualizations of our proposed *1p-frac*. The following Section B offers a detailed description of the hyperparameters used in the experiments throughout the main paper. Finally, Section C presents additional experimental results.

A Training data visualization

In this section, we provide visualizations of the *1p-frac* and other data defined in the main text to improve transparency.

A.1 The Effect of Perturbation Degree on Data Distributions

The following visualizations illustrate the effect of varying perturbation degrees (Δ) on the *1p-frac*, with a fixed σ-factor of 3.5 and $L = 1000$. The perturbation degrees explored are $\{0.001, 0.01, 0.05, 0.1, 0.2, 1.0\}$.

$\underline{\Delta = 0.001}$ was used as the noise level for the visualization shown in Figure A. The main text reports that the pre-trained model using this setting achieved a fine-tuning accuracy of 1.2 on CIFAR-100 and 1.9 on ImageNet-100.

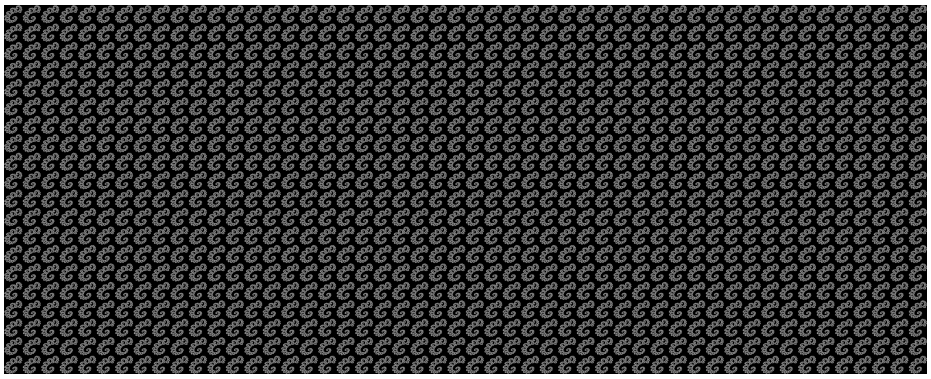

Fig. A: *1p-frac* for $\Delta = 0.001$

$\underline{\Delta = 0.01}$ was used as the noise level for the visualization shown in Figure B. The main text reports that the pre-trained model using this setting achieved a fine-tuning accuracy of 19.8 on CIFAR-100 and 61.8 on ImageNet-100.

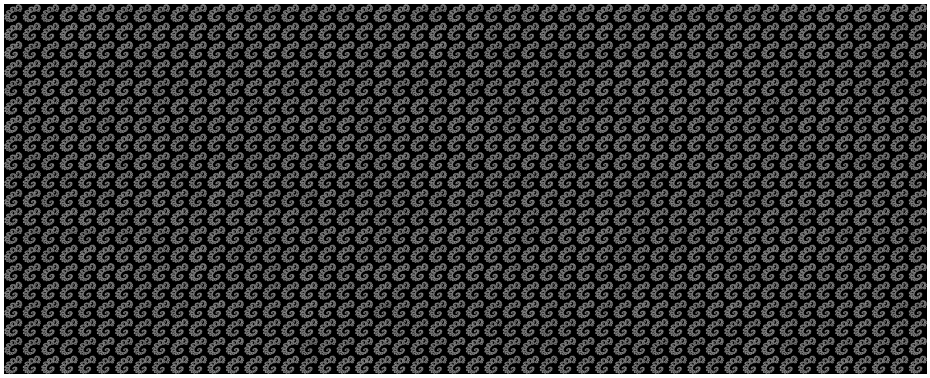

Fig. B: *1p-frac* data for $\Delta = 0.01$

$\underline{\Delta = 0.05}$ was used as the noise level for the visualization shown in Figure C. The main text reports that the pre-trained model using this setting achieved a fine-tuning accuracy of 83.0 on CIFAR-100 and 88.2 on ImageNet-100.

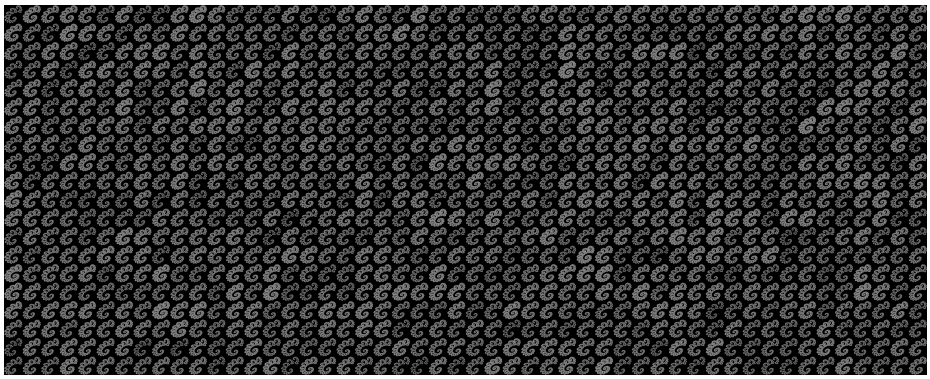

Fig. C: *1p-frac* data for $\Delta = 0.05$

$\underline{\Delta = 0.1}$ was used as the noise level for the visualization shown in Figure D. The main text reports that the pre-trained model using this setting achieved a fine-tuning accuracy of 84.2 on CIFAR-100 and 89.0 on ImageNet-100.
$\underline{\Delta = 0.2}$ was used as the noise level for the visualization shown in Figure E. The main text reports that the pre-trained model using this setting achieved a fine-tuning accuracy of 83.4 on CIFAR-100 and 88.5 on ImageNet-100.
$\underline{\Delta = 1.0}$ was used as the noise level for the visualization shown in Figure F. The main text reports that the pre-trained model using this setting achieved a fine-tuning accuracy of 82.6 on CIFAR-100 and 88.1 on ImageNet-100.

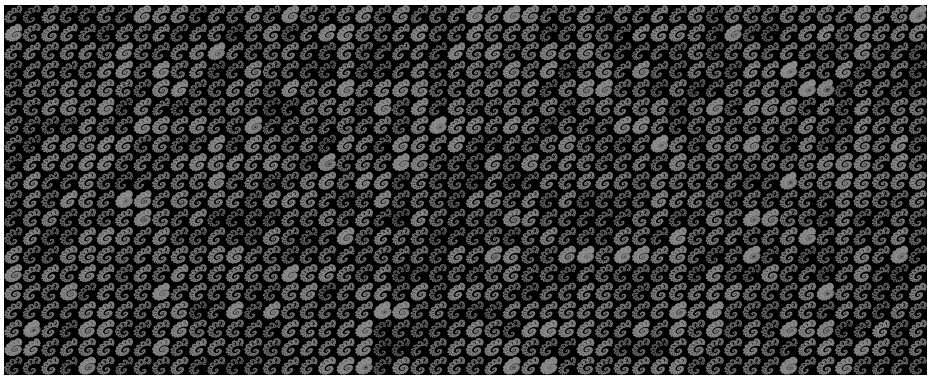

Fig. D: *1p-frac* data for $\Delta = 0.1$

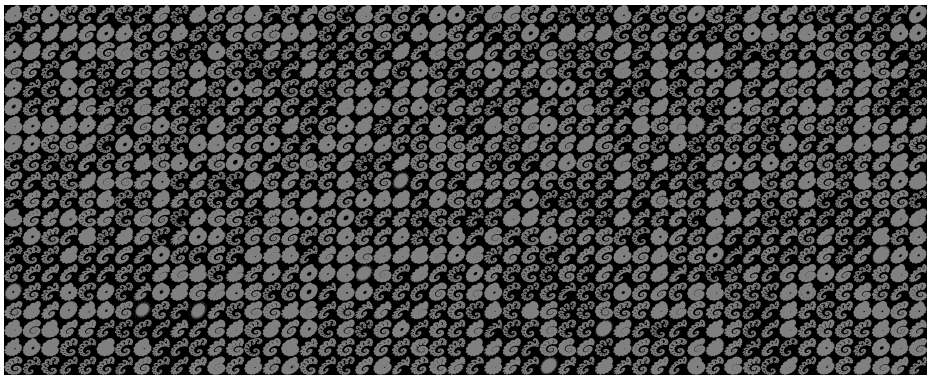

Fig. E: *1p-frac* data for $\Delta = 0.2$

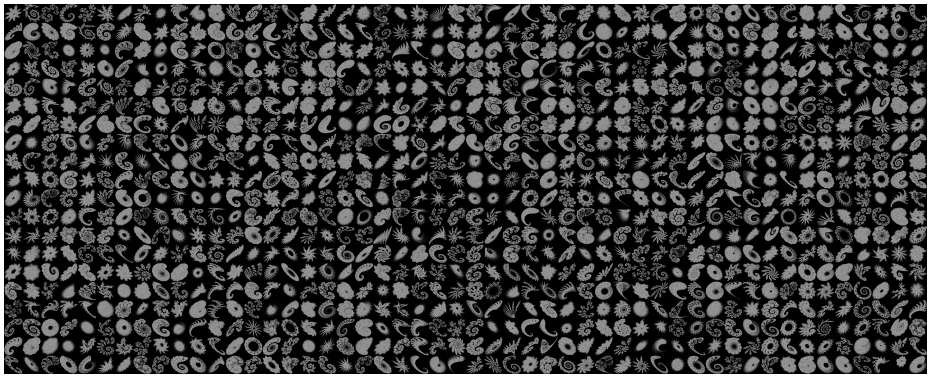

Fig. F: *1p-frac* data for $\Delta = 1.0$

A.2 The Effect of σ-factor on Image Distributions

The following visualizations illustrate the effect of varying σ-factor on the *1p-frac* model, with a fixed Δ of 0.1 and $L = 1000$. The σ-factors explored are $\{4.0, 4.5, 4.0, 6.0\}$.

σ-factor $= 4.0$ was used for the *1p-frac* data visualized in Figure G. The main text reports that the pre-trained model using this *1p-frac* data for σ-factor $= 4.0$ achieved a fine-tuning accuracy of 82.8 on CIFAR-100 and 87.9 on ImageNet-100.

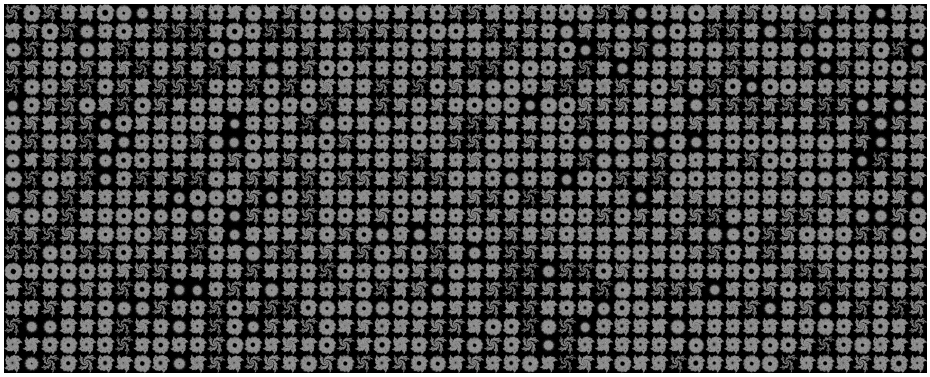

Fig. G: *1p-frac* data for σ-factor $= 4.0$

σ-factor $= 4.5$ was used for the *1p-frac* data visualized in Figure H. The main text reports that the pre-trained model using *1p-frac* data for σ-factor $= 4.5$ achieved a fine-tuning accuracy of 81.9 on CIFAR-100 and 86.9 on ImageNet-100.

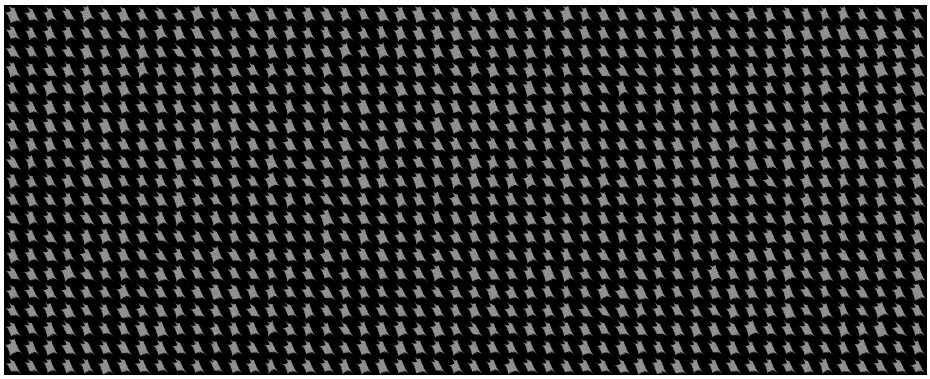

Fig. H: *1p-frac* data for σ-factor $= 4.5$

σ-factor $= 5.0$ was used for the *1p-frac* data visualized in Figure I. The main text reports that the pre-trained model using this *1p-frac* data for σ-factor $= 5.0$ achieved a fine-tuning accuracy of 82.2 on CIFAR-100 and 87.1 on ImageNet-100.

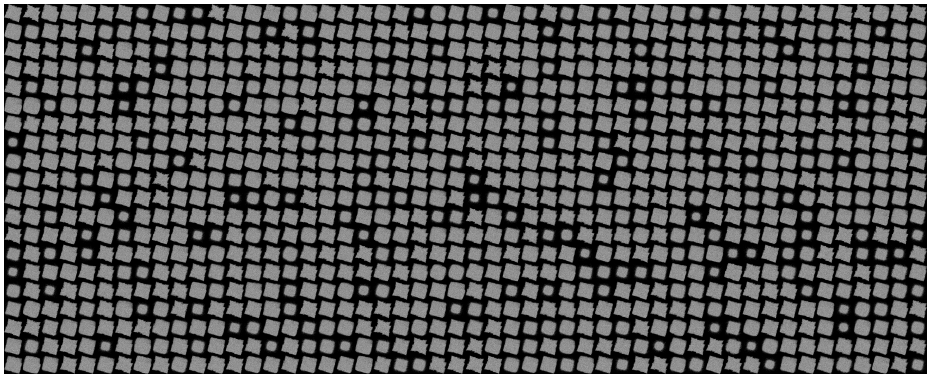

Fig. I: *1p-frac* data for σ-factor $= 5.0$

σ-factor $= 6.0$ was used for the *1p-frac* data visualized in Figure J. The main text reports that the pre-trained model using this *1p-frac* data for σ-factor $= 6.0$ achieved a fine-tuning accuracy of 81.3 on CIFAR-100 and 86.3 on ImageNet-100.

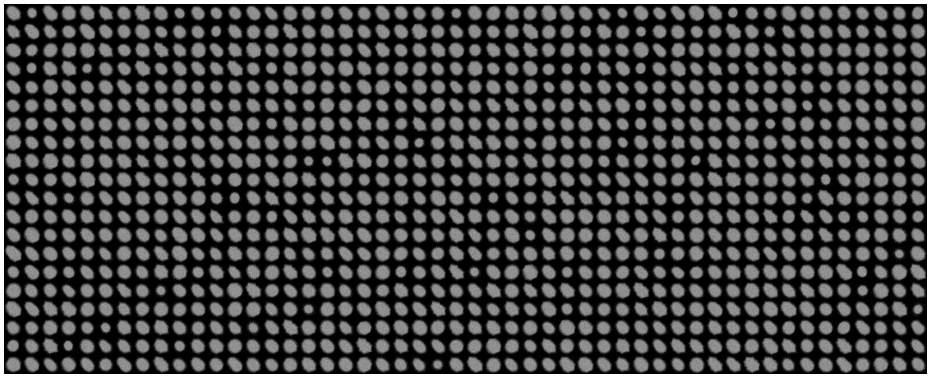

Fig. J: *1p-frac* data for σ-factor $= 6.0$

A.3 Real image with shape augmenatation

For the single real image defined in Figure 5, we applied shape augmentations such as Affine Transformation (Affine T), Elastic Transformation (Elastic T), and Polynomial Transformation (Polynomial T), as shown in Figure 4 in the main text, to perform *1p-frac*-style pre-training using a single real image.

In this section, we focus on visualizing the Canny edge image of Real Img 1 from Figure 5, which yielded the highest fine-tuning accuracy.

Affine T applied to Real Img 1, generates the data shown in Figure K. We used the module defined in PyTorch [2] for transformation. Table B reports that the pre-trained model achieved a fine-tuning accuracy of 82.8 on CIFAR-100.

Fig. K: Data shape augmented by Affine T

Elastic T applied to Real Img 1, generates the data shown in Figure L. We used the module defined in PyTorch [2] for transformation. Table B reports that the pre-trained model achieved a fine-tuning accuracy of 0.8 on CIFAR-100.

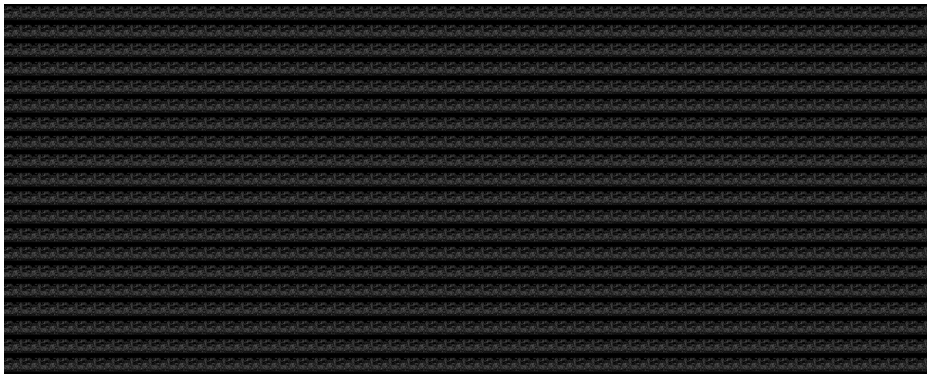

Fig. L: Data shape augmented by Elastic T

Polynomial T applied to Real Img 1, generates the data shown in Figure M. We used the module defined in scikit-image [**?**] for transformation. Table B reports that the pre-trained model achieved a fine-tuning accuracy of 81.9 on CIFAR-100.

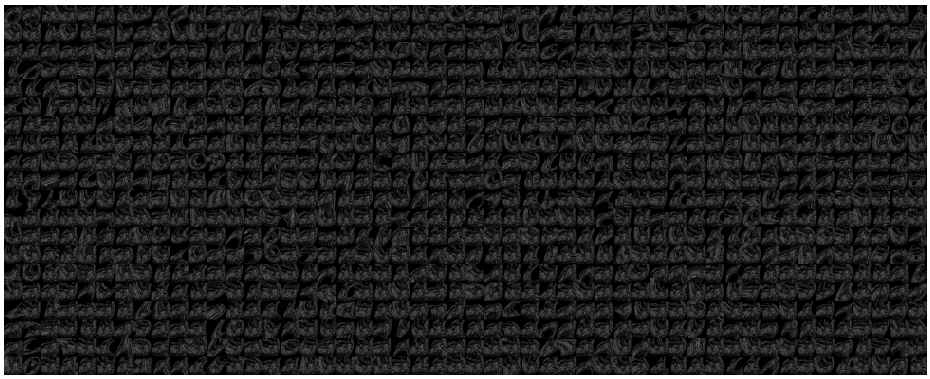

Fig. M: Data shape augmented by Polynomial T

A.4 Visualization for Other Locally perturbed Data

Here, we visualize the four data compared in Tables 3 and 4 of the main text, focusing on the case where $L = 1000$.

RCDB's locally perturbed data is visualized in Figure N. The main text reports that the pre-trained model using this setting achieved a fine-tuning accuracy of 82.5 on CIFAR-100 and 87.9 on ImageNet-100.

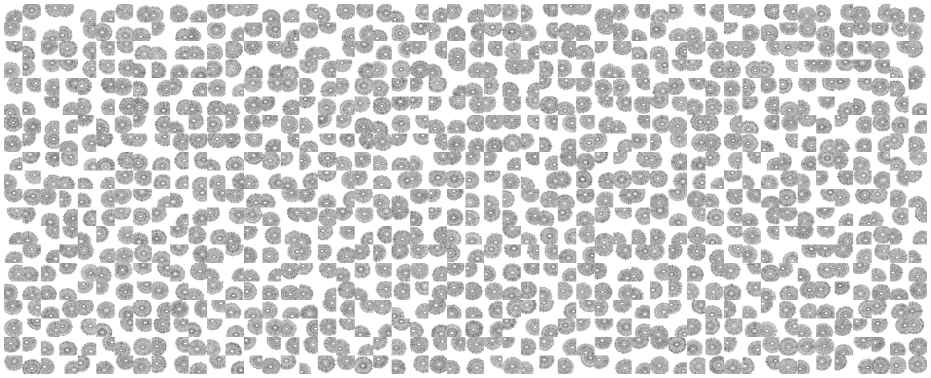

Fig. N: Locally perturbed RCDB

VA's locally perturbed data is visualized in Figure O. The main text reports that the pre-trained model using this Locally perturbed VA data achieved a fine-tuning accuracy of 82.6 on CIFAR-100 and 88.0 on ImageNet-100.

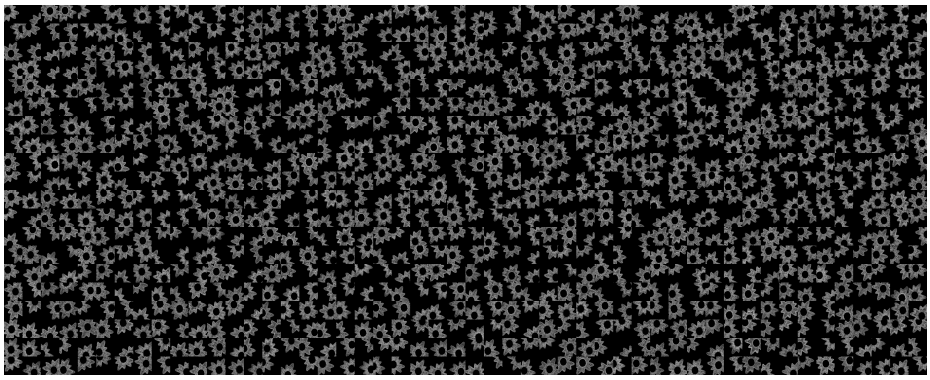

Fig. O: Locally perturbed VA

Gaussian dist.'s locally perturbed data is visualized in Figure P. For the Gaussian distribution image, the following steps were performed to generate the image: (1) Sample 100k points from a two-dimensional normal distribution with mean 0 and standard deviation 44, which will be used for generating the fractal image. (2) When rendering the points onto the image, instead of using points, draw random 3×3 patches on the image, similar to the fractal image. (3) Repeat this operation for the number of perturbations L. The main text reports that the pre-trained model using this setting achieved a fine-tuning accuracy of 1.1 on CIFAR-100 and 5.7 on ImageNet-100.

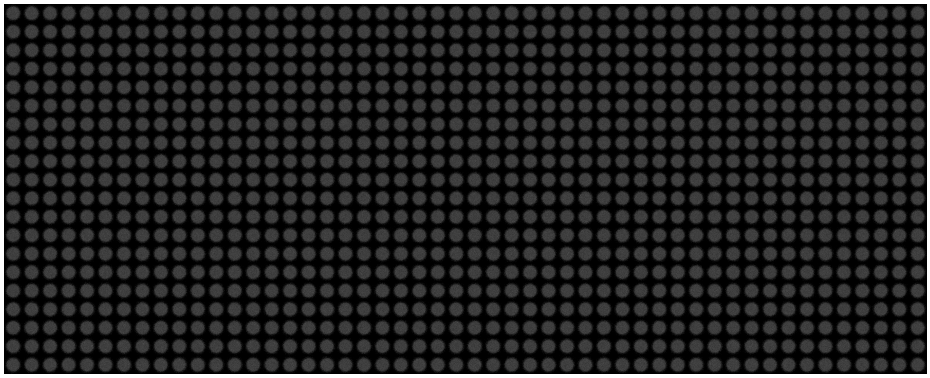

Fig. P: Locally perturbed Gaussian dist.

Uniform dist.'s locally perturbed data is visualized in Figure Q. To generate the Uniform distribution image, the following steps were performed: (1) Sample 100k points from a two-dimensional uniform distribution with a range equal to the image size, which will be used for generating the fractal image. (2) When rendering the points onto the image, instead of using points, draw random 3×3 patches on the image, similar to the fractal image. (3) Repeat this operation for the number of perturbations L. The main text reports that the pre-trained model using this setting achieved a fine-tuning accuracy of 2.0 on CIFAR-100 and 71.1 on ImageNet-100.

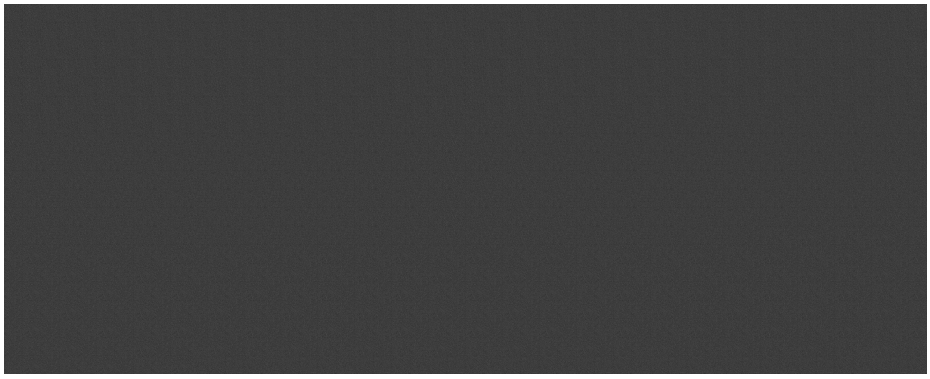

Fig. Q: Locally perturbed Uniform dist.

Table A: Hyper-parameters of pre-training and fine-tuning in *1p-frac* experiments. The basic configuration is essentially the same as that used by Nakamura et al. [1].

Training Step	Pre-training			Fine-tuning	
Model	ViT-T		ViT-B	ViT-T/B	
Sampling points $\mathcal{L}$	1	21k	21k	Full	Full
Epochs	80000	15238	15238	300	1000
Batch Size	256	1024	1024	1024	768
Optimizer	AdamW	AdamW	AdamW	AdamW	SGD
LR	1.0e-3	5.0e-4*	5.0e-4*	1.0e-3	1.0e-2
Weight Decay	0.05	0.05	0.05	0.05	1.0e-4
LR Scheduler	Cosine	Cosine	Cosine	Cosine	Cosine
Warmup Steps	15.238k	15.238k	15.238k	5 (epochs)	10 (epochs)
Resolution	224	224	224	224	224
Label Smoothing	0.1	0.1	0.1	0.1	0.1
Drop Path	0.1	0.1	0.1	0.1	0.1
Rand Augment	(9,0.5)	(9,0.5)	(9,0.5)	(9,0.5)	(9,0.5)
Mixup	0.8	0.8	0.8	0.8	0.8
Cutmix	1.0	1.0	1.0	1.0	1.0
Erasing	0.25	0.25	0.25	0.25	0.25

B Hyper-parameters in our experiments

Table A shows the parameters used when training *1p-frac*. The basic parameter settings were kept consistent with those used by Nakamura et al. [1]. The number of epochs used in training was adjusted to ensure that the number of iterations (i.e., the number of back-propagation steps) was equivalent across experiments

C Additional Experiments

C.1 Pre-training with a single real image and shape augmentation

Table B presents the fine-tuning accuracy on CIFAR-100 when using data generated by applying shape augmentations to the Real/Canny images as shown in Figure 5 of the main text. Here, we use common geometric transformations, Affine T, Elastic T, and Polynomial T, as described in Section A.3. Across all images, Affine T tends to yield the highest accuracy. When using Elastic T, the pre-training results are mixed, with some cases showing moderate success and others not performing well. This suggests that, in pre-training, discriminating global shape differences, as seen in Affine T and Polynomial T, is more important than distinguishing local shape variations, which are characteristic of Elastic T.

C.2 Performance comparison of pre-training models in object detection / instance segmentation

Table C compares the performance of fine-tuning each pre-trained model to object detection and instance segmentation using the COCO dataset [3]. In the

Table B: Relationship between fine-tuning accuracy and pre-training effect from a single real image.

Transformation	Real Img 1	Real Img 2	Real Img 3	Real Img 4	Real Img 5
Affine T	81.8 / 82.8	81.9 / 82.5	81.5 / 82.3	81.7 / 81.5	81.2 / 82.1
Elastic T	1.9 / 0.8	78.8 / 68.3	55.7 / 18.2	73.5 / 20.0	32.8 / 20.5
Polynomial T	81.8 / 81.9	81.6 / 82.1	81.0 / 81.6	79.7 / 81.5	81.2 / 81.3

Table C: Object detection and instance segmentation performance

Pre-training	COCO Det	COCO Inst Seg
	AP_{50} / AP / AP_{75}	AP_{50} / AP / AP_{75}
Scratch	63.7 / 42.2 / 46.1	60.7 / 38.5 / 41.3
ImageNet-1k	69.2 / 48.2 / 53.0	66.6 / 43.1 / 46.5
ImageNet-21k	70.7 / 48.8 / 53.2	67.7 / 43.6 / 47.0
2D-OFDB-21k [35]	67.6 / 46.4 / 51.0	64.6 / 41.6 / 44.7
`1p-frac`	68.1 / 47.3 / 51.9	65.3 / 42.0 / 45.2

result, our proposed `1p-frac` achieves performance comparable to ImageNet. Additionally, it outperforms OFDB, which includes broader distribution of shapes, indicating that our findings hold on tasks beyond classification.

References

1. Nakamura, R., Kataoka, H., Takashima, S., Noriega, E.J.M., Yokota, R., Inoue, N.: Pre-training vision transformers with very limited synthesized images. In: Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 20303–20312. IEEE (2023)
2. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`
3. Tsung-Yi Lin, Michael Maire, S.B.L.B.R.G.J.H.P.P.D.R.C.L.Z.P.D.: Microsoft COCO: Common Objects in Context. In: European Conference on Computer Vision (ECCV). pp. 740–755 (2014)