Replacing Labeled Real-image Datasets with Auto-generated Contours –Supplementary Material–

Anonymous CVPR submission

Paper ID 436

1. Dataset details in FDSL

Restricted FractalDB. This corresponds to Table 1 in the main paper, where a subset of the parameters in θ of IFS are fixed. The FDSL (Fractal, restricted) varies only three parameters (a_i, c_i, e_i) and the other three (b_i, d_i, f_i) are fixed. Table 1 shows the example of restricted parameters.

Extended FractalDB (ExFractalDB). We list example images of randomly selected 500 classes with 100 instances per class in Figure 1. The figures show images of both the FractalDB and ExFractalDB. We also describe the definition of classes and instances on FractalDB, MV-FractalDB, and ExFractalDB for the preliminary study and experiments in Table 2. The execution code is available in 'FDSL_Datasets/README.md' and 'FDSL_Datasets/ExFractalDB/ExFractalDB.py'.

RadialContourDB (RCDB). We list example images of randomly selected 500 classes with 100 instances per class in Figure 3. The figures show RCDB with the class definition of only #vertices and parameter set η. We also describe the definition of classes and instances of both RCDB for the experiments shown in Table 2. The execution code is available in 'FDSL_Datasets/README.md' and 'FDSL_Datasets/RCDB.py'.

Dead Leaves. Originally, Dead Leaves [5] are the combination of simple patterns such as circles, triangles, and squares. Baradad *et al.* [1] assigned labels through SSL. However, in our case we label the dataset using the FDSL framework. We simply use the ratio of circles, triangles, and squares in the images, and use those as labels. The accuracy of {C10, C100, Cars, Flowers} datasets change from {89.8, 71.4, 32.0, 91.6} when training with SimCLRV2, to {95.9, 79.6, 72.8, 96.9} when pre-trained with FDSL. The definition of classes and instances is shown in Table 2.

Bezier Curves. The previous Bezier curves have only #Lines and #Dots as formula-supervision parameters. However, as a consequence of Hypothesis 2, we improve the Bezier curves by adding a restriction to the control point coordinates to increase the number of formula-supervision parameters. In addition to the conventional instance augmentations, we augment the instances further by randomly selecting vectors with connecting control points. The definition of classes and instances is shown in Table 2.

Table 1. Parameters in restricted FractalDB.

a	b	c	c	e	f
0.33	0.44	0.63	-0.39	-0.7	-0.81
0.011	0.44	-0.91	-0.39	0.97	-0.81
-0.007	0.44	-0.64	-0.39	0.72	-0.81
0.29	0.44	-0.13	-0.39	0.097	-0.81
0.23	0.44	-0.81	-0.39	-0.32	-0.81
-0.2	0.44	0.019	-0.39	-0.23	-0.81
0.4	0.44	-0.44	-0.39	-0.16	-0.81

LineDB. We list example images of randomly selected 500 classes with 5 instances per class in Figure 3. The execution code is available in 'FDSL_Datasets/README.md' and 'FDSL_Datasets/LineDB.py'. The code was written in Python3 and OpenCV. The definition of classes and instances is shown in Table 2.

2. Additional experimental results

ImageNet-1k fine-tuning (Table 3; the additional results of Table 7 in the main paper). We additionally listed the results of ViT-Small architecture and pre-training on 10k-scale FDSL datasets in ImageNet-1k fine-tuning.

For the experiments on larger datasets with more than 10k classes, we use the same hyper-parameters as the DeiT paper [6], except for the global mini-batch size which we changed from 1024 to 8192. Due to this change in the global mini-batch size, we also adjusted the learning rate and number of epochs accordingly. For example, we set the learning rate for ViT-{Tiny, Small, Base} to {8.0e-3, 4.0e-3, 1.0e-3}, and set the number of epochs for datasets with {10k, 21k, 50k} classes to {300, 90, 40}, respectively. AdamW is used as the optimizer, and weight decay is set to 5e-2. Other hyper-parameters and data augmentations are the same as the DeiT paper [6]. Also, for the pre-training on

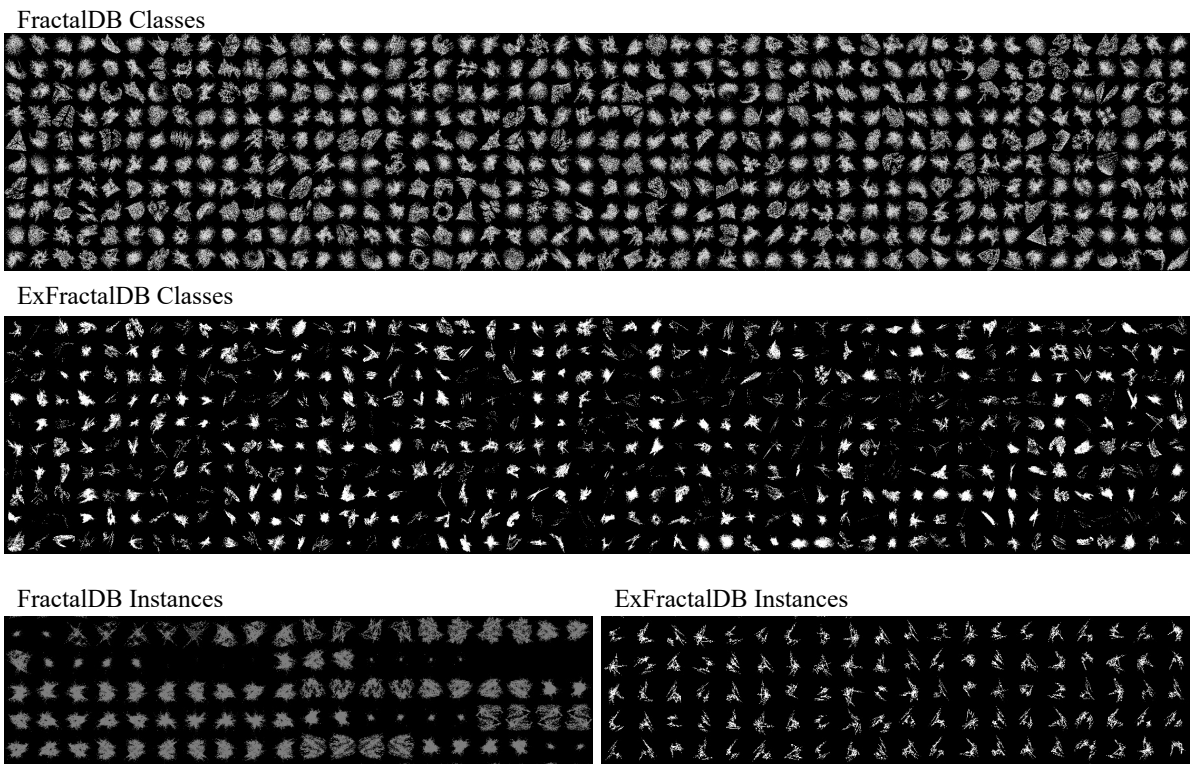


Figure 1. Classes and Instances in FractalDB and ExFractalDB.

Table 2. Definition of class and instance in Fractals (FractalDB, MV-FractalDB and ExFractalDB), Radial Contours (#Vertices and Parameter set η).

DB	Class	Instance
FractalDB [4]	Random params from 2D-IFS	Image rotation, 3×3 patch pattern, Weighting values
MV-FractalDB [7]	Random params from 3D-IFS	Fixed viewpoints
ExFractalDB*	Random params from 3D-IFS	Random viewpoints
RCDB	#Vertices	#polygon, radius, line width, resizing factor, perlin noise
RCDB*	Parameter set η	Image translation
Dead Leaves [1]	Ratio of circle, triangle, and square	Randomly located circle, triangle, and square
Bezier Curves [4]	#Lines & #Dots	Randomly located lines and dots
Bezier Curves*	#Control points, radius, resizing factor	#Lines, and vectors with connecting control points
LineDB	#Lines	Randomly located lines

* The FDSL datasets have been enhanced based on the Hypothesis 2 in this proposal.

larger datasets, we use WebDataset[1] to accelerate IO when loading data. When fine-tuning on ImageNet-1k, we use the same hyper-parameters as the DeiT paper [6], which uses the AdamW optimizer, a 1024 batch size, 1.0e-3 learning rate, 5e-4 weight decay, and 300 epochs.

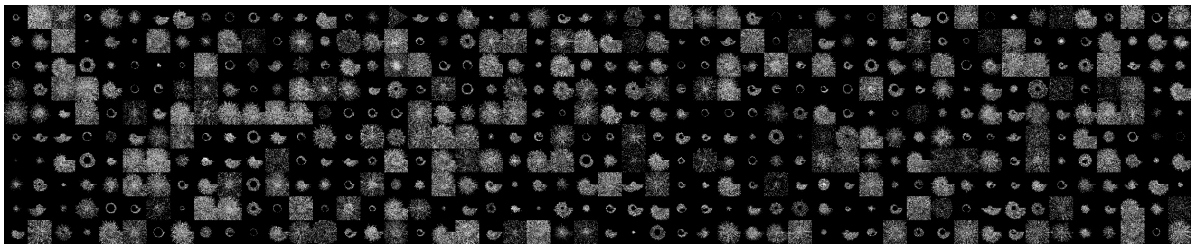
According to the results of ImageNet-1k fine-tuning, the accuracy improves as the model gets larger.

FDSL versus SSL/SL (Table 4; the additional results of Table 9 in the main paper). We additionally listed the results of 10k/21k/50k-scale pre-training on the seven datasets, in addition to the 1k-scale pre-training.

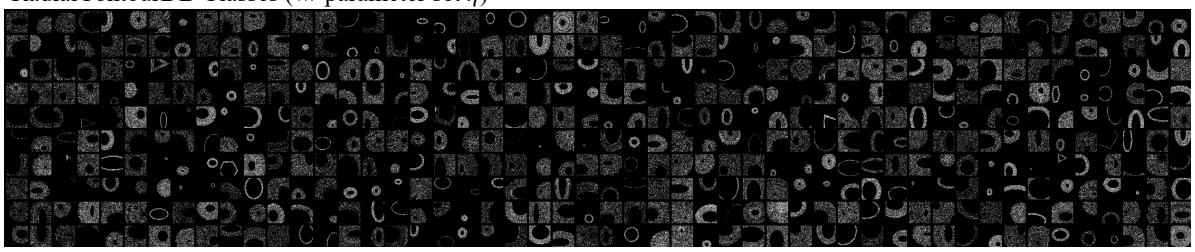
When fine-tuning on datasets other than ImageNet, we use the same hyper-parameter as the DeiT paper [6]. In detail, image size is resized to 224, and we use SGD opti-

[1]WebDataset library https://github.com/tmbdev/webdataset

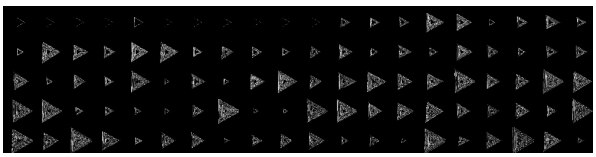
RadialContourDB Classes (w/ only #vertices)



RadialContourDB Classes (w/ parameter set η)



FractalDB Instances (w/ only #vertices)



RadialContourDB Instances (w/ parameter set η)

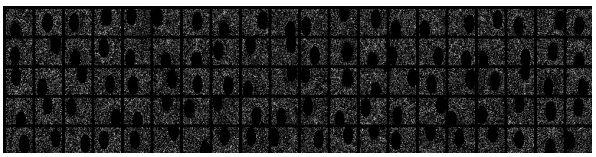
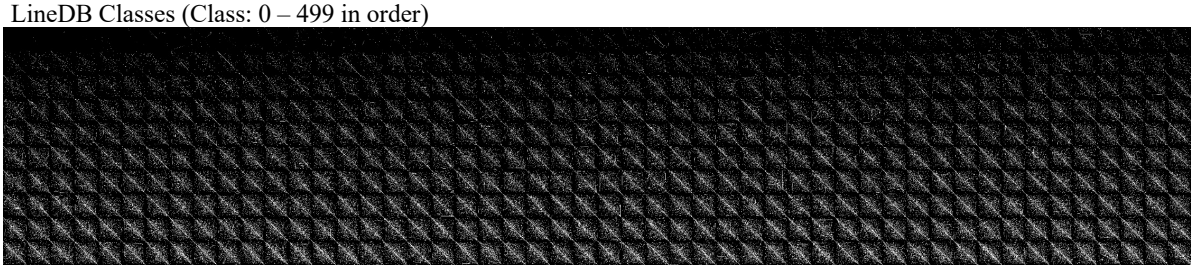


Figure 2. Classes and Instances in RCDB with only #vertices and parameter set η.

LineDB Classes (Class: 0 – 499 in order)



LineDB Instances (5 x 1 instances)

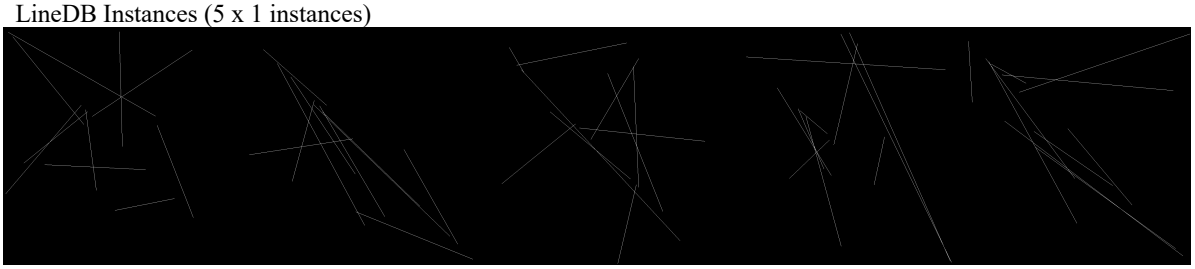


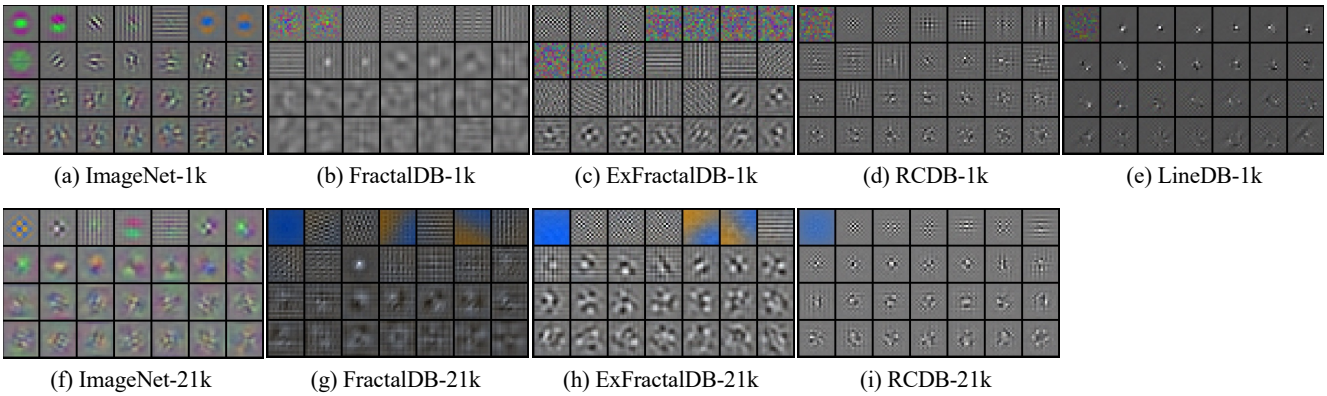
Figure 3. Classes and Instances in LineDB.

mizer, 768 batch size, 1.0e-2 learning rate, 1000 epoch, 1e-4 weight decay. Other hyper-parameters are also the same as previous work [6] when fine-tuning on CIFAR10/100, Cars.

As shown in the FractalDB-10k results between the batch size 1024 and 8192, the smaller batch size may perform better than the listed performance rates.

Other datasets (Table 5). We additionally listed the other datasets including CUB, ImageNet-1k with 1%/10% (IN1%/10%), KMNIST, Kuzushiji-Kanji (Kuzushiji), and ObjectPI.

We use the same hyper-parameter as ImageNet-1k fine-tuning which mentioned above when fine-tuning on IN1%,



(a) ImageNet-1k (b) FractalDB-1k (c) ExFractalDB-1k (d) RCDB-1k (e) LineDB-1k

(f) ImageNet-21k (g) FractalDB-21k (h) ExFractalDB-21k (i) RCDB-21k

Figure 4. Embedding filters in ImageNet-1k/21k, FractalDB-1k/21k, ExFractalDB-1k/21k, RCDB-1k/21k, LineDB-1k.

IN10%. However, we tuned epochs as {300, 1000} and chose the higher one. For fine-tuning on other datasets, as same as the experiments in Table 4, the hyper-parameter is based on fine-tuning on CIFAR10/100, Cars in previous work [6].

From the results of ImageNet-1k with 1%, 10%, and 100% label, the accuracy was dramatically improved as the number of instances is increased. The performance was increased {30.3, 66.2, 82.7} on ImageNet-1k with {1%, 10%, 100%} labels in ExFractalDB-21k pre-training.

Effect of batch size (Table 6). In the work of SimSiam [2], the use of a small batch size had a positive effect on SSL. We show the effect of batch size for FDSL in Table 6. As shown, the optimal batch size for FDSL is around 512 or 1024. However, even for a batch size of 64, which is smaller than the optimal batch size of SimSiam, the scores do not significantly drop from their highest values. This suggests that FDSL has a smaller optimal batch size compared to that of SSL with real images (e.g., ImageNet-1k).

3. Visualization

Visualization of attention maps.

We show the embedding filters for ImageNet-1k/21k, FractalDB-1k/21k, ExFractalDB-1k/21k, RCDB-1k/21k, and LineDB-1k in Figure 4.

References

[1] Manel Baradad, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise. *arXiv preprint arXiv:2106.05963*, 2021. 1, 2

[2] Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. In *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, 2021. 4

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representation (ICLR)*, 2021. 5

[4] Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. Pre-training without Natural Images. In *IEEE/CVF Asian Conference on Computer Vision (ACCV)*, 2020. 2

[5] Daniel L. Ruderman. Origins of scaling in natural images. *Vision Research*, 37(23):3385—-3398, 1997. 1

[6] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, pages 10347–10357, 2021. 1, 2, 3, 4

[7] Ryosuke Yamada, Ryo Takahashi, Ryota Suzuki, Akio Nakamura, Yusuke Yoshiyasu, Ryusuke Sagawa, and Hirokatsu Kataoka. MV-FractalDB: Formula-driven Supervised Learning for Multi-view Image Recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. 2

Table 3. Comparison of ImageNet-1k fine-tuning. Accuracies obtained with ViT-Tiny(Ti)/Small(S)/Base(B) architectures are listed. 10k/21k/50k indicates the number of classes in the pre-training phase.

Pre-training	Img	Type	ViT-Ti	ViT-S	ViT-B
Scratch	–	–	72.6	80.0	79.8
ImageNet-21k	Real	SL	74.1	81.4	81.8
JFT-300M	Real	SL	–	–	84.1*
FractalDB-10k	Synth	FDSL	73.1	80.9	81.4
FractalDB-21k	Synth	FDSL	73.1	81.0	81.8
FractalDB-50k	Synth	FDSL	73.4	80.9	82.1
ExFractalDB-10k	Synth	FDSL	73.6	81.5	82.5
ExFractalDB-21k	Synth	FDSL	73.6	–	82.7
ExFractalDB-50k	Synth	FDSL	73.7	–	82.5
RCDB-10k	Synth	FDSL	73.7	81.3	82.5
RCDB-21k	Synth	FDSL	72.8	–	82.4
RCDB-50k	Synth	FDSL	73.1	–	82.6

* Rate reported in original ViT paper [3].

Table 4. Additional experimental results of ExFractalDB and RCDB in 10k/21k/50k datasets.

Pre-training	Img	Type	Arch.	C10	C100	Cars	Flowers	VOC12	P30	IN100
FractalDB-10k*	Synth	FDSL	Ti	97.8	83.1	89.1	98.8	82.6	80.8	88.0
FractalDB-10k	Synth	FDSL	Ti	96.3	81.6	82.8	98.0	80.6	79.4	88.0
FractalDB-21k	Synth	FDSL	Ti	95.9	81.3	80.2	97.7	78.6	80.9	88.1
FractalDB-50k	Synth	FDSL	Ti	96.3	81.6	81.5	96.8	78.8	81.1	87.7
ExFractalDB-10k	Synth	FDSL	Ti	97.3	83.6	86.7	99.4	82.0	81.1	89.5
ExFractalDB-21k	Synth	FDSL	Ti	97.1	83.5	86.0	98.9	81.4	81.4	89.7
ExFractalDB-50k	Synth	FDSL	Ti	97.1	83.8	85.9	99.2	81.6	81.3	89.9
RCDB-10k	Synth	FDSL	Ti	96.6	81.3	84.8	98.6	80.3	81.3	89.1
RCDB-21k	Synth	FDSL	Ti	96.3	81.2	82.6	98.1	79.5	80.8	88.9
RCDB-50k	Synth	FDSL	Ti	96.2	80.3	82.4	98.8	79.2	81.2	88.4
ExFractalDB-10k	Synth	FDSL	B	97.6	84.3	89.2	99.4	82.4	81.3	89.9
ExFractalDB-21k	Synth	FDSL	B	97.8	85.2	88.1	99.5	82.7	81.6	90.1
ExFractalDB-50k	Synth	FDSL	B	97.9	85.4	89.5	99.4	83.7	81.6	91.0
RCDB-10k	Synth	FDSL	B	97.0	83.5	87.4	99.2	81.6	81.5	89.8
RCDB-21k	Synth	FDSL	B	96.8	82.9	85.9	99.0	81.2	81.2	90.2
RCDB-50k	Synth	FDSL	B	97.0	83.5	87.6	99.0	81.7	81.3	90.7

* Only the rate calculated by batch size 1,024. Due to the larger-scale pre-training, we have conducted these experiments on the batch size 8,192, however, the smaller batch size (e.g., 1,024) may perform better than the listed performance rates.

Table 5. Additional experimental results on ImageNet 1%/10% labels (IN1%/10%), CUB, KMNIST, Kuzushiji-Kanji (Kuzushiji), ObjectPI (ObjPI).

Pre-training	Img	Type	Arch.	CUB	IN1%	IN10%	KMNIST	Kuzushiji	ObjPI
FractalDB-21k	Synth	FDSL	Ti	19.1	22.4	59.7	99.0	82.3	43.1
ExFractalDB-21k	Synth	FDSL	Ti	31.9	27.0	62.1	99.2	82.8	48.3
RCDB-21k	Synth	FDSL	Ti	28.2	24.6	59.4	99.0	82.4	43.1
ImageNet-1k	Real	SL	B	62.5	52.5	61.9	98.9	83.8	62.6
FractalDB-21k	Synth	FDSL	B	23.4	24.1	60.8	99.2	83.0	47.1
ExFractalDB-21k	Synth	FDSL	B	37.4	30.3	66.2	99.2	83.4	53.8
RCDB-21k	Synth	FDSL	B	32.4	26.2	64.9	98.9	83.6	49.7

* Only the rate calculated by batch size 1,024. Due to the larger-scale pre-training, we have conducted these experiments on the batch size 8,192, however, the smaller batch size (e.g., 1,024) may perform better than the listed performance rates.

Table 6. Effect of batch size on ExFractalDB.

Batch size	C10	C100	Cars	Flowers
32	96.5	79.2	82.8	96.4
64	97.2	81.2	87.7	98.2
128	**97.4**	81.9	87.8	97.8
256	97.1	80.9	89.0	98.2
512	97.2	81.8	**90.0**	**99.4**
1024	97.3	**82.6**	89.9	**99.4**
2048	96.9	81.8	88.2	99.1
4096	96.6	81.4	86.7	98.6