APPENDIX

A VARIOUS RESOLUTION IMAGES

There are no image datasets prepared for various resolutions. Therefore, creating images in various resolutions through an appropriate resizing method is important in our task. Chrabaszcz et al. (2017) showed that low-resolution images resized by some interpolations have the characteristics of the original image. Among various interpolation methods, the bilinear interpolation method retains the most characteristics of the original images. Therefore, we constructed low-resolution image datasets applying the same bilinear interpolation method to the original dataset.

After constructing the low-resolution datasets, we examined the performance decrease with various pre-trained models. To check the performance decrease, the low-resolution images were resized back to the target resolution (224) with various interpolation methods. Table 5 shows that the bicubic interpolation has the smallest performance decrease. As mentioned in Section 1, images resized by the bicubic interpolation were used as input to various pre-trained models.

Table 5: The top-1 accuracy of pre-trained ConvNeXt-Tiny in various resolutions when resized up with each interpolating method.

Method	Acc. (%)	Resolution			
		32	64	128	**224**
Nearest	Top-1	0.9	18.17	58.27	**82.52**
Bilinear	Top-1	29.47	60.36	76.98	**82.52**
Bicubic	Top-1	**34.51**	**64.88**	**77.27**	**82.52**
Area	Top-1	0.9	34.92	74.42	**82.52**
Nearest-exact	Top-1	0.9	18.27	58.27	**82.52**

Table 6: Classification accuracy drops when scaling up low-resolution images, *i.e.*, 32, 64, 128, to target resolution, *i.e.*, 224 or 299, using existing image interpolation methods. We report the best interpolation method's results.

Model	Acc. (%)	Resolution					# Param.
		32	64	128	**224**	**299**	
ResNet-18	Top-1	22.09	50.53	65.52	69.76	-	11.69M
	Relative	31.67	72.43	93.91	100	-	
Inception-V3	Top-1	22.05	53.92	71.77	-	77.29	27.16M
	Relative	33.69	72.39	93.51	-	100	
ViT-B16	Top-1	47.39	68.91	78.13	81.07	-	86.57M
	Relative	58.46	85.00	69.37	100	-	
ConvNeXt-Tiny	Top-1	34.51	64.88	77.27	82.52	-	28.59M
	Relative	42.02	79.00	94.08	100	-	

B DETAILED DESCRIPTION OF DATASETS

The following are detailed descriptions of the datasets used in our experiments. The number of classes, train images, and test images are organized in Table 7.

B.1 IMAGENET-1K

ImageNet-1k (Russakovsky et al., 2015) is the most widely used dataset in image classification along with iNaturalist. In particular, the most widely used dataset, ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012), contains 1000 categories of 1.2 million images. ImageNet is built with the WordNet hierarchy which means each category is explained by several words or phrases. Thus, its goal is to offer a set of qualified images in the same hierarchical words or phrases.

B.2 FINE-GRAINED DATASETS

Fine-grained Datasets contain classes from a single category, such as cars, birds, flowers, aircraft, or food, and these are more difficult to classify where detailed elements must be considered.

Stanford Cars Stanford Cars (Krause et al., 2013) is a dataset containing car images. There are 196 classes with information on make, model, and year (*e.g.*, 2012 BMW M3 coupe).

Oxford-IIIT Pets Oxford-IIIT Pets (Parkhi et al., 2012) contains images of cats and dogs with 37 categories. The label consisted of species and breeds. This dataset also provides a bounding box for segmentation tasks.

Flowers Flowers (Nilsback & Zisserman, 2008) includes 102 categories of flowers that are mainly found in the United Kingdom. Even flowers belonging to the same categories have large variations in pose, size, and light.

FGVC Aircraft FGVC Aircraft (Maji et al., 2013) is used in the fine-grained recognition challenge 2013 (FGComp). Aircraft with various sizes, purposes, driving forces, and other features are classified by those fine-grained features. The 100 categories of aircraft include the make or name of the model series.

Food-101 Food-101 (Bossard et al., 2014) contains 101 kinds of food images, which were originally provided to be used for RandomForest. The number of images is much larger compared to other fine-grained datasets. The images in this dataset are already rescaled and contain noises ranging from color intensity to wrong labels.

Table 7: The number of classes, train images and test images.

Datasets	# of classes	# of train images	# of test images
ImageNet-1k	1000	1,281,167	50,000
Stanford Cars	196	8,144	8,041
Oxford-IIIT Pets	37	3,680	3,669
Flowers	102	1,020	6,149
FGVC Aircraft	100	3,334	3,333
Food-101	101	75,750	25,250

C TRAINING ALGORITHM

We describe a two-stage training algorithm in Algorithm. 1. We first stage for training with target resolution in the first while loop, followed by the second while loop for training with various resolutions.

Algorithm 1: 2-stage Training of PAC-FNO

Input: Target resolution data D_{target}, Number of low-resolutions N, Low resolution data $\{D_{low_1}, \cdots, D_{low_N}\}$, Class label y, Predicted class label $\hat{y}$, First phase iteration number K_{first}, Second phase iteration number K_{second}, PAC-FNO parameters $\boldsymbol{\theta}_o$, Pre-trained backbone model parameters $\boldsymbol{\theta}_p$, cross-entropy loss CE

Initialize $\boldsymbol{\theta}_o$;
$k \leftarrow 0$;
/* First training stage */
while $k < K_{first}$ **do**
  for *each mini-batch* $B \subseteq D_{target}$ **do**
    $\{\hat{\mathbf{y}}_i\}_{i=1}^{|B|} \leftarrow$ Backbone(PAC-FNO($B; \boldsymbol{\theta}_o$);$\boldsymbol{\theta}_p$);
    Train $\boldsymbol{\theta}_o$ and $\boldsymbol{\theta}_p$ with $CE(\{\hat{\mathbf{y}}_i\}_{i=1}^{|B|}, \{\mathbf{y}_i\}_{i=1}^{|B|})$;
  $k \leftarrow k + 1$;
end
$k \leftarrow 0$;
/* Second training stage */
while $k < K_{second}$ **do**
  for $i \leftarrow 1$ *to* N **do**
    for *each mini-batch* $B \subseteq D_{low_i}$ **do**
      $\{\hat{\mathbf{y}}_i\}_{i=1}^{|B|} \leftarrow$ Backbone(PAC-FNO($B; \boldsymbol{\theta}_o$);$\boldsymbol{\theta}_p$);
      Train $\boldsymbol{\theta}_o$ with $CE(\{\hat{\mathbf{y}}_i\}_{i=1}^{|B|}, \{\mathbf{y}_i\}_{i=1}^{|B|})$;
  for *each mini-batch* $B \subseteq D_{target}$ **do**
    $\{\hat{\mathbf{y}}_i\}_{i=1}^{|B|} \leftarrow$ Backbone(PAC-FNO($B; \boldsymbol{\theta}_o$);$\boldsymbol{\theta}_p$);
    Train $\boldsymbol{\theta}_o$ with $CE(\{\hat{\mathbf{y}}_i\}_{i=1}^{|B|}, \{\mathbf{y}_i\}_{i=1}^{|B|})$;
  $k \leftarrow k + 1$;
end

D BACKBONE MODELS

We propose a plug-in module for multi-scale classification. Therefore, we applied various pre-trained classification models from CNN-based to ViT-based classification models. All classification models were trained with ImageNet-1k from scratch with the settings in Table 8. TORCHVISION provides all such pre-trained models.

ResNet-18 Residual network (ResNet) is a model that applies the concept of residual connection to CNN architectures. ResNet-18 (He et al., 2016) consists of 18 convolutional blocks and 8 residual layers. It is the most fundamental model in image classification.

Table 8: Recipe for the pre-trained models provided in TORCHVISION.

Model	ResNet-18	ViT-B16	ConvNeXt-Tiny
Epochs	90	300	600
Batch size	32	512	128
Optimizer	sgd	adamw	adamw
Learning rate (lr.)	0.1	3e-3	1e-3
Weight decay	1e-4	0.3	0.05
lr. scheduler	steplr	cosineannealinglr	cosineannealinglr
lr. warmup method	-	linear	linear
lr. warmup epochs	-	30	5
lr. warmup decay	-	0.033	0.01
Amp	-	○	-
Random erase	-	-	0.1
Label smoothing	-	0.11	0.1
Mixup alpha	-	0.2	0.2
Auto augment	-	ra	ta wide
Clip grad norm	-	1	-
Ra sampler	-	○	○
Cutmix alpha	-	1.0	1.0
Model-ema	-	○	○
Norm weight decay	-	-	0.0
Train crop size	224	224	176
Test resize size	256	256	232
Test crop size	224	224	224
Ra reps	-	3	4

Inception-V3 Inception networks (Szegedy et al., 2016) are one of the most popular classification models, which stacks deep convolutional layers in an efficient way. They proposed techniques such as the concatenation of convolutional layers with different kernel sizes, kernel decomposition, and backpropagation with an auxiliary classifier for efficient training. We used the pre-trained Inception-V3 model provided by TORCHVISION, and there is no training recipe for training from scratch.

ViT-B16 Vision Transformer (Dosovitskiy et al., 2020) (ViT) is a model that uses transformers based on a self-attention architecture in the image domain. It divides an image into patches and calculates all patch-by-patch relationships through self-attention. Because of these calculations, much memory and training time are required. Among those ViT-based models, ViT-B16 has the smallest model size and divides an image into 16 patches.

ConvNeXt-Tiny ConvNeXt (Liu et al., 2022) is one of the most recent models with good performance using only convolutional networks. It shows better performance with fewer parameters compared to ViT. To achieve good performance, it utilizes several advanced training schemes for convolutional networks. ConvNeXt-Tiny refers to the smallest model size in the ConvNeXt family.

E HYPERPARAMETERS

In Tables 9 and 10, we list all the key hyperparameters in our experiments for each dataset. Our Appendix accompanies some trained checkpoints and one can easily reproduce.

Table 9: The best hyperparameter of our main experiments (ImageNet-1k).

ImageNet-1k	ResNet-18	Inception-V3	Vision Transformer	ConvNeXt-Tiny
# of parallel AC-FNO blocks (m)	2	2	2	4
# of stages (n)	1	1	1	2
First phase training lr.	1e-3	1e-3	1e-3	2e-4
Second phase training lr.	1e-6	1e-6	1e-6	2e-6

Table 10: The best hyperparameter of our experiments on the fine-grained datasets.

ConvNeXt-Tiny	StanfordCars	Oxford-IIIT Pets	Flowers	FGVC Aircraft	Food-101
# of parallel AC-FNO blocks (m)	4	2	2	2	2
# of stages (n)	2	2	1	1	1
First phase training lr.	1e-3	1e-3	1e-3	1e-3	1e-3
Second phase training lr.	1e-6	1e-5	1e-6	1e-6	1e-6

F EVALUATION

F.1 EXPERIMENTAL SETUP

We run our experiments on a machine equipped with Intel i9 CPUs and Nvidia RTX A5000/A6000 GPUs. We implement PAC-FNO using Python v3.8 and PyTorch v1.12.

F.2 NUMBER OF PARAMETERS OF PAC-FNO

We report the number of parameters of PAC-FNO according to backbone models and datasets.

Table 11: The number of parameters according to backbone models.

ImageNet-1k	ResNet-18	Inception-V3	Vision Transformer	ConvNeXt-Tiny
# of PAC-FNO parameters	+0.91M	+1.61M	+0.91M	+3.65M

Table 12: The number of parameters according to datasets

ConvNeXt-Tiny	StanfordCars	Oxford-IIIT Pets	Flowers	FGVC Aircraft	Food-101
# of PAC-FNO parameters	+3.65M	+3.65M	+3.65M	+3.65M	+3.65M

F.3 ADDITIONAL SUPER-RESOLUTION METHODS

In this section, we report experimental results with the additional latest super-resolution baseline. We fine-tune a pre-trained classification model with low-resolution images that are upscaled by the super-resolution model. We note that we used the latest super-resolution model.

In Table 13, combining the super-resolution and fine-tune methods shows better performance than simply upscaling low-resolution images to the target resolution using the super-resolution model. Even at 32 resolution, it shows slightly better performance than PAC-FNO. However, this super-resolution method has two major drawbacks. The first drawback is the model size. DRPN's $\times 8$ upscaling model has a similar number of parameters the the pre-trained model size, i.e., 23.21M vs.28.59M. Second, a model is needed for each resolution. In other words, upscaling models for $\times 8$, $\times 4$, and $\times 2$ are needed to handle 28, 56, and 112 resolutions, respectively. In contrast, our proposed PAC-FNO can handle images of all resolutions with an additional 3.65M network and shows good performance.

Additionally, we verify the superiority of PAC-FNO by providing a comparison with a method equipped with the latest super-resolution model. OSRT (Yu et al., 2023) is a state-of-the-art model in the super-resolution domain but it does not support $\times 8$ upscale. Therefore, we only use the $\times 2$ and $\times 4$ upscale models of OSRT. As a result, PAC-FNO shows better performance than OSRT and OSRT (fine-tune) methods.

Table 13: **Results with the additional latest super-resolution method.** For the experiment, we used ConvNeXt-Tiny as a pre-trained model. (Left: ImageNet-1k, Right: ImageNet-C/P Fog)

Model	Method	Metric	Resolution						
			28	32	56	64	112	128	224
ConvNeXt-Tiny	Resize	Top1-Acc (%)	27.5	34.5	60.7	64.9	75.0	77.3	82.5
	Fine-tune	Top1-Acc (%)	40.2	62.3	65.8	76.0	76.4	80.7	81.8
	DRPN	Top1-Acc (%)	40.7	-	68.2	-	79.4	-	82.5
		# of Parameters (M)	23.21	-	10.43	-	5.95	-	-
	DRPN	Top1-Acc (%)	**60.8**	-	72.5	-	76.7	-	82.5
	(Fine-tune)	# of Parameters (M)	23.21	-	10.43	-	5.95	-	-
	OSRT	Top1-Acc (%)	-	-	61.4	-	75.4	-	82.5
		# of Parameters (M)	-	-	11.93	-	11.79	-	-
	OSRT	Top1-Acc (%)	-	-	71.2	-	78.4	-	81.2
	(Fine-tune)	# of Parameters (M)	-	-	11.93	-	11.79	-	-
	PAC-FNO	Top1-Acc (%)	58.9	**63.2**	**77.6**	**76.2**	**80.2**	**80.7**	81.8
		# of Parameters (M)	3.65						

Model	Method	Metric	Resolution						
			28	32	56	64	112	128	224
ConvNeXt-Tiny	Resize	Top1-Acc (%)	8.12	10.8	27.2	31.9	48.5	52.3	58.4
	Fine-tune	Top1-Acc (%)	23.2	28.2	47.5	51.4	61.0	62.2	63.0
	DRPN	Top1-Acc (%)	0.67	-	0.99	-	1.32	-	58.4
		# of Parameters (M)	23.21	-	10.43	-	5.95	-	-
	DRPN	Top1-Acc (%)	21.8	-	42.3	-	56.8	-	61.0
	(Fine-tune)	# of Parameters (M)	23.21	-	10.43	-	5.95	-	-
	OSRT	Top1-Acc (%)	-	-	19.4	-	37.9	-	58.4
		# of Parameters (M)	-	-	11.93	-	11.79	-	-
	OSRT	Top1-Acc (%)	-	-	42.3	-	56.4	-	59.4
	(Fine-tune)	# of Parameters (M)	-	-	11.93	-	11.79	-	-
	PAC-FNO	Top1-Acc (%)	25.4	30.4	48.2	51.7	60.1	61.4	62.8
		# of Parameters (M)	3.65						

F.4 ADDITIONAL FINE-GRAINED DATASETS

Tables 14 and 15 show the performance of the remaining fine-grained dataset, Stanford Cars. PAC-FNO shows good performance in most cases, especially at low-resolution compared to other models. Tables 16 and 17 show the performance of the Oxford-IIIT Pets with the ViT model. PAC-FNO shows good performance than FNO at all resolutions.

Table 14: Top-1 accuracy on low-resolution.

Dataset	Method	Resolution						
		28	32	56	64	112	128	**224**
Standford Cars	Resize	14.7	22.5	66.3	73.6	88.2	89.6	91.5
	Fine-tune	12.3	66.5	70.5	88.5	91.2	92.4	92.6
	DRLN	1.98	-	36.1	-	87.0	-	91.5
	DRPN	41.5	-	84.2	-	90.9	-	91.5
	FNO	11.9	19.2	67.8	75.3	91.0	92.6	**93.9**
	UNO	57.4	66.2	85.9	87.8	91.6	92.1	92.3
	A-FNO	67.6	74.4	88.0	89.7	92.1	92.3	92.7
	PAC-FNO	**70.4**	**76.6**	**89.0**	**90.0**	**92.6**	**92.8**	93.5

Table 15: Relative accuracy on low-resolution.

Dataset	Method	Resolution						
		28	32	56	64	112	128	**224**
Standford Cars	Resize	16.1	24.5	72.4	80.4	96.4	97.9	100
	Fine-tune	13.6	71.9	76.1	95.6	98.5	**99.8**	100
	DRLN	2.16	-	39.4	-	95.1	-	100
	DRPN	51.9	-	92.1	-	99.3	-	100
	FNO	12.7	20.5	72.2	80.2	96.9	98.6	100
	UNO	62.2	71.7	93.1	95.1	99.2	**99.8**	100
	A-FNO	72.9	80.2	94.9	**96.7**	**99.4**	99.5	100
	PAC-FNO	**75.3**	**81.9**	**95.2**	96.3	99.0	99.3	100

Table 16: Top-1 accuracy on low-resolution.

Dataset	Method	Resolution						
		28	32	56	64	112	128	**224**
Oxford-IIIT Pets	FNO	26.3	33.0	58.1	64.8	82.9	85.8	91.3
	PAC-FNO	**40.3**	**46.2**	**69.0**	**72.5**	**86.8**	**89.2**	**92.2**

Table 17: Relative accuracy on low-resolution.

Model	Method	Resolution						
		28	32	56	64	112	128	**224**
Oxford-IIIT Pets	FNO	28.8	36.1	63.6	71.0	90.8	94.0	100
	PAC-FNO	**43.7**	**50.1**	**74.8**	**78.6**	**94.1**	**96.7**	**100**

F.5 ADDITIONAL METRIC FOR IMAGENET-1K AND FINE-GRAINED DATASETS

Table 18 and 19 show the relative accuracy of low-resolution images generated by ImageNet-1k and fine-grained datasets. In ImageNet-1k, the performance of PAC-FNO is the best in most of the datasets, and especially in lower resolution. In fine-grained datastes, PAC-FNO shows good performance in all cases. In other words, PAC-FNO works very well for low-resolution image classification.

Table 18: Relative accuracy on low-resolution images generated from ImageNet-1k

Model	Method	Resolution							
		28	32	56	64	112	128	**224**	**299**
ResNet-18	Resize	23.9	31.7	65.5	72.3	91.3	93.8	100	-
	Fine-tune	1.6	3.3	15.7	25.1	53.1	77.8	100	-
	DRLN	0.3	-	24.5	-	90.0	-	100	
	DRPN	45.3	-	79.7	-	96.7	-	100	
	FNO	57.3	64.5	84.3	87.7	96.4	97.7	100	-
	UNO	57.5	65.3	84.9	88.3	96.7	98.1	100	-
	A-FNO	**63.2**	**72.3**	**87.9**	**91.2**	**97.7**	**98.5**	100	-
	PAC-FNO	60.8	67.9	86.2	89.5	97.3	98.4	100	-
Inception-V3	Resize	21.6	28.5	62.9	69.7	89.9	92.9	-	100
	Fine-tune	51.1	60.9	82.2	90.1	94.1	94.6	-	100
	FNO	62.4	68.9	87.4	89.5	95.5	97.6	-	100
	UNO	54.6	62.2	84.6	88.0	96.6	97.7	-	100
	A-FNO	44.6	52.9	79.6	83.7	94.8	96.3	-	100
	PAC-FNO	**62.9**	**70.0**	**87.8**	**90.2**	**97.1**	**98.1**	-	100
ViT-B16	Resize	51.3	58.4	81.4	85.0	94.7	96.3	100	-
	Fine-tune	49.1	59.8	82.1	85.8	95.3	96.7	100	-
	DRLN	4.6	-	51.8	-	94.9	-	100	
	DRPN	64.4	-	90.0	-	**98.6**	-	100	-
	FNO	49.1	58.5	83.1	86.8	96.7	97.9	100	-
	UNO	54.5	63.1	84.7	87.7	97.1	98.3	100	-
	A-FNO	**69.1**	**74.8**	**90.9**	**93.2**	98.5	**99.2**	100	-
	PAC-FNO	58.5	65.5	86.7	89.9	97.6	98.7	100	-
ConvNeXt-Tiny	Resize	33.3	41.8	73.6	78.7	90.9	93.7	100	-
	Fine-tune	49.1	76.2	80.4	92.9	93.4	98.7	100	-
	DRLN	0.4	-	30.4	-	87.0	-	100	-
	DRPN	49.3	-	82.7	-	96.2	-	100	-
	FNO	54.0	61.7	85.1	88.1	96.6	97.7	100	-
	UNO	72.7	**78.0**	**91.4**	93.7	98.1	98.9	100	-
	A-FNO	61.8	68.5	87.2	90.3	97.2	98.1	100	-
	PAC-FNO	**72.3**	77.5	**91.4**	**93.5**	**98.4**	99.0	100	-

Table 19: Relative accuracy on low-resolution images generated from Fine-grained datasets

Dataset	Method	Resolution						
		28	32	56	64	112	128	**224**
Oxford-IIIT Pets	Resize	31.4	38.8	74.8	82.2	95.9	97.3	100
	Fine-tune	34.8	43.8	78.0	84.5	97.0	97.5	100
	DRLN	3.6	-	39.3	-	94.0	-	100
	DRPN	44.3	-	89.9	-	98.8	-	100
	FNO	20.9	59.1	66.3	76.8	94.9	98.9	100
	UNO	12.3	17.4	47.8	55.9	88.7	92.8	100
	A-FNO	30.3	37.4	71.3	78.8	95.6	97.5	100
	PAC-FNO	**80.0**	**84.3**	**93.8**	**95.5**	**98.7**	**99.3**	100
Flowers	Resize	41.2	49.6	78.5	83.7	96.0	97.7	100
	Fine-tune	46.2	53.3	82.2	85.4	96.5	98.0	100
	DRLN	10.3	-	55.7	-	95.2	-	100
	DRPN	67.3	-	**92.8**	-	99.1	-	100
	FNO	26.2	34.0	68.2	75.6	94.9	97.0	100
	UNO	24.1	31.4	67.1	74.6	94.5	96.4	100
	A-FNO	29.2	36.6	69.3	76.2	93.8	96.6	100
	PAC-FNO	**78.5**	**82.6**	**92.8**	**94.6**	**99.2**	**99.8**	100
FGVC Aircraft	Resize	3.1	3.5	34.3	52.4	89.1	93.2	100
	Fine-tune	10.7	20.3	50.4	73.1	**96.7**	95.2	100
	DRLN	1.4	-	16.4	-	86.4	-	100
	DRPN	23.1	-	74.8	-	96.6	-	100
	FNO	32.6	41.1	73.6	79.3	92.4	93.8	100
	UNO	2.0	2.1	11.2	27.8	89.5	93.6	100
	A-FNO	8.4	14.2	62.9	73.3	92.9	95.3	100
	PAC-FNO	**46.5**	**55.6**	**82.6**	**85.6**	94.9	**96.9**	100
Food-101	Resize	44.0	52.8	83.6	88.0	96.7	97.9	100
	Fine-tune	55.0	59.3	88.0	89.9	96.8	98.2	100
	DRLN	7.7	-	42.9	-	95.3	-	100
	DRPN	60.2	-	89.6	-	65.3	-	100
	FNO	48.0	56.5	85.8	89.2	97.1	98.0	100
	UNO	57.5	64.5	85.9	89.1	96.9	97.8	100
	A-FNO	52.2	58.6	81.9	85.9	95.0	96.1	100
	PAC-FNO	**82.9**	**86.1**	**94.6**	**96.1**	**98.7**	**99.2**	100

F.6 ADDITIONAL NATURAL INPUT VARIATIONS

Table 20 shows the remaining input variation results of ImageNet-C/P Hendrycks & Dietterich (2019). For the remaining input variations, we report results at 32, 64, 128, and 224 resolution, excluding SR models whose performance appears to be meaningless. In most cases, PAC-FNO shows good performance at 32 and 64 resolution, and Fine-tune shows good performance at 128×128 and 224×224 resolution. However, at 128 and 224, there is a performance difference of up to 5% (Glass noise 44.2% vs. 39.5% at 128 × 128 resolution), but at 32 and 63, there is a performance difference of up to 47% (Pixelate 44.2% vs. 39.5% at 32 × 32 resolution). In other words, PAC-FNO shows good performance by a large margin at low-resolution.

Table 20: **Performance of PAC-FNO on the input variation tasks.** We show the top-1 accuracy of the ConvNext-Tiny model on remaining input variations, chosen from ImageNet-C/P (Hendrycks & Dietterich, 2019).

Variation	Model	Resolution			
		32	64	128	224
Gaussian Noise	Resize	11.3	37.9	55.3	47.9
	Fine-tune	11.6	39.5	**57.4**	51.5
	FNO	38.4	47.1	46.5	43.4
	UNO	42.8	54.6	56.5	52.1
	A-FNO	37.8	43.1	44.0	42.8
	PAC-FNO	**48.0**	**54.8**	56.5	**52.3**
Shot Noise	Resize	11.3	37.3	53.8	45.4
	Fine-tune	11.6	39.0	**56.2**	49.5
	FNO	38.2	45.9	43.9	41.1
	UNO	41.2	53.6	55.2	**50.5**
	A-FNO	38.1	42.9	42.8	41.2
	PAC-FNO	**47.7**	**54.0**	55.0	50.0
Impulse Noise	Resize	10.8	36.9	53.2	44.6
	Fine-tune	11.3	38.5	**55.5**	48.5
	FNO	37.2	45.2	41.8	38.5
	UNO	37.2	53.5	54.3	48.9
	A-FNO	36.5	41.3	39.3	37.2
	PAC-FNO	**46.7**	**53.5**	54.1	**50.2**
Defocus Noise	Resize	10.0	32.2	48.3	43.2
	Fine-tune	11.1	36.9	56.9	55.9
	FNO	35.8	44.9	47.4	47.6
	UNO	47.2	51.5	50.5	47.3
	A-FNO	38.6	47.2	47.4	47.2
	PAC-FNO	**51.3**	**57.7**	**57.5**	**56.2**
Glass Noise	Resize	11.7	33.5	38.7	31.5
	Fine-tune	12.7	36.9	**44.2**	**38.8**
	FNO	39.8	43.3	34.9	32.4
	UNO	26.1	43.4	33.4	30.7
	A-FNO	42.8	39.6	28.7	26.1
	PAC-FNO	**54.4**	**50.9**	39.5	35.1
Motion Noise	Resize	10.6	36.5	54.0	48.1
	Fine-tune	11.5	39.6	**58.8**	**55.9**
	FNO	36.3	46.9	48.7	47.4
	UNO	41.3	50.6	49.3	47.9
	A-FNO	37.6	44.9	42.6	41.3
	PAC-FNO	**50.4**	**56.5**	55.3	54.1
Zoom Noise	Resize	10.3	25.7	43.0	43.6
	Fine-tune	10.9	26.8	46.4	**48.9**
	FNO	35.2	38.5	37.3	37.0
	UNO	33.5	40.2	39.4	39.5
	A-FNO	36.0	35.9	33.8	33.5
	PAC-FNO	**49.6**	**48.8**	**46.5**	44.6
Snow	Resize	4.9	20.2	46.6	49.6
	Fine-tune	5.2	21.8	**49.0**	**52.5**
	FNO	20.3	31.1	41.6	45.9
	UNO	40.1	36.5	44.0	43.6
	A-FNO	18.1	27.4	37.5	40.4
	PAC-FNO	**31.9**	**41.1**	48.3	49.4
Frost	Resize	6.5	30.8	54.9	54.1
	Fine-tune	6.9	32.3	56.7	**57.0**
	FNO	27.8	42.3	49.0	50.9
	UNO	46.0	48.8	54.0	53.6
	A-FNO	22.6	37.7	45.3	46.0
	PAC-FNO	**40.3**	**52.4**	**56.9**	56.9
Contrast	Resize	8.1	40.4	62.9	59.5
	Fine-tune	8.5	41.7	**65.8**	**64.7**
	FNO	31.8	53.5	63.3	63.5
	UNO	59.2	59.6	65.2	62.8
	A-FNO	34.8	55.6	62.1	59.2
	PAC-FNO	**44.9**	**59.9**	65.1	62.3
Elastic Transform	Resize	12.2	40.0	56.8	53.2
	Fine-tune	12.6	41.5	**59.1**	**56.6**
	FNO	38.3	49.7	52.1	51.5
	UNO	45.4	52.8	50.5	48.4
	A-FNO	41.3	48.2	47.8	45.4
	PAC-FNO	**51.8**	**57.3**	55.8	53.5
Pixelate	Resize	14.5	81.0	63.8	53.2
	Fine-tune	15.1	52.9	66.8	**56.6**
	FNO	48.6	66.7	62.5	42.9
	UNO	40.8	70.9	64.6	52.2
	A-FNO	53.2	67.8	58.9	40.8
	PAC-FNO	**62.6**	**73.2**	**67.9**	54.4
Jpeg Compression	Resize	12.9	44.4	64.3	62.4
	Fine-tune	13.3	45.9	66.3	65.5
	FNO	45.2	62.5	**65.2**	**63.7**
	UNO	**61.1**	61.2	64.4	62.2
	A-FNO	50.9	**64.8**	63.9	61.1
	PAC-FNO	52.3	63.2	64.8	63.0
Speckle Noise	Resize	12.1	41.2	59.2	53.5
	Fine-tune	12.5	42.6	**66.2**	57.1
	FNO	41.6	52.7	51.3	48.8
	UNO	48.3	58.6	59.9	56.2
	A-FNO	42.7	49.6	49.8	48.3
	PAC-FNO	**52.1**	**60.3**	61.4	**57.8**
Gaussian Blur	Resize	10.4	34.8	51.1	46.9
	Fine-tune	11.5	39.1	59.0	58.4
	FNO	37.3	47.8	50.6	50.8
	UNO	51.2	54.0	53.3	50.2
	A-FNO	40.1	50.2	51.3	51.2
	PAC-FNO	**52.5**	**60.0**	**60.7**	**59.4**

Table 21: **Performance of PAC-FNO according to low and high-frequency filter.** We report top-1 accuracy on low-resolution images generated from ImageNet-1k in ConvNeXt-Tiny.

ImageNet-1k	32	64	128	224
PAC-FNO (low pass filter)	53.5	71.4	78.7	79.0
PAC-FNO (high pass filter)	21.6	49.4	68.2	74.8
PAC-FNO	**58.9**	**74.5**	**80.2**	**81.5**

ImageNet-C/P Fog	32	64	128	224
PAC-FNO (low pass filter)	18.0	41.7	52.4	54.4
PAC-FNO (high pass filter)	5.92	23.0	43.2	50.2
PAC-FNO	**25.4**	**48.2**	**60.1**	**62.8**

F.7 ADDITIONAL ABLATION STUDIES

We provide an analysis of the impact of low and high-frequency information on accuracy/generalization through ablation experiments. Table 21 shows thah compared to PAC-FNO, using low-pass and high-pass filters results in lower accuracy and generalization. When using a high-pass filter, it is expected to show good performance in ImageNet-C/P Fog, but since the perfor-

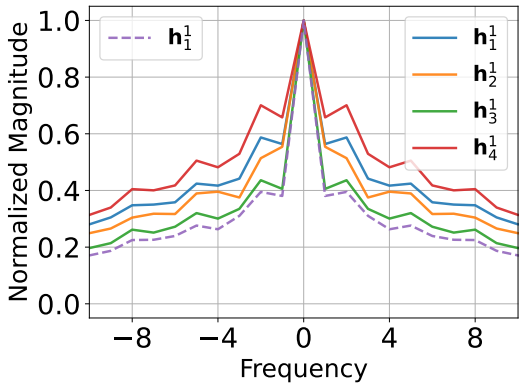


Figure 7: **Comparison of spectral responses according to the configuration of the AC-FNO block.** We test the ConvNeXT-Tiny backbone model on ImageNet-1k and visualize only the hidden vector of the first layer ($m = 1$) for the hidden vectors $\mathbf{h}_n^m$. In the case of parallel (solid line), there are four hidden vectors ($\mathbf{h}_n^1, n \in \{1, 2, 3, 4\}$), and in the case of serial (dashed line), there is one hidden vector ($\mathbf{h}_n^1, n \in \{1\}$).

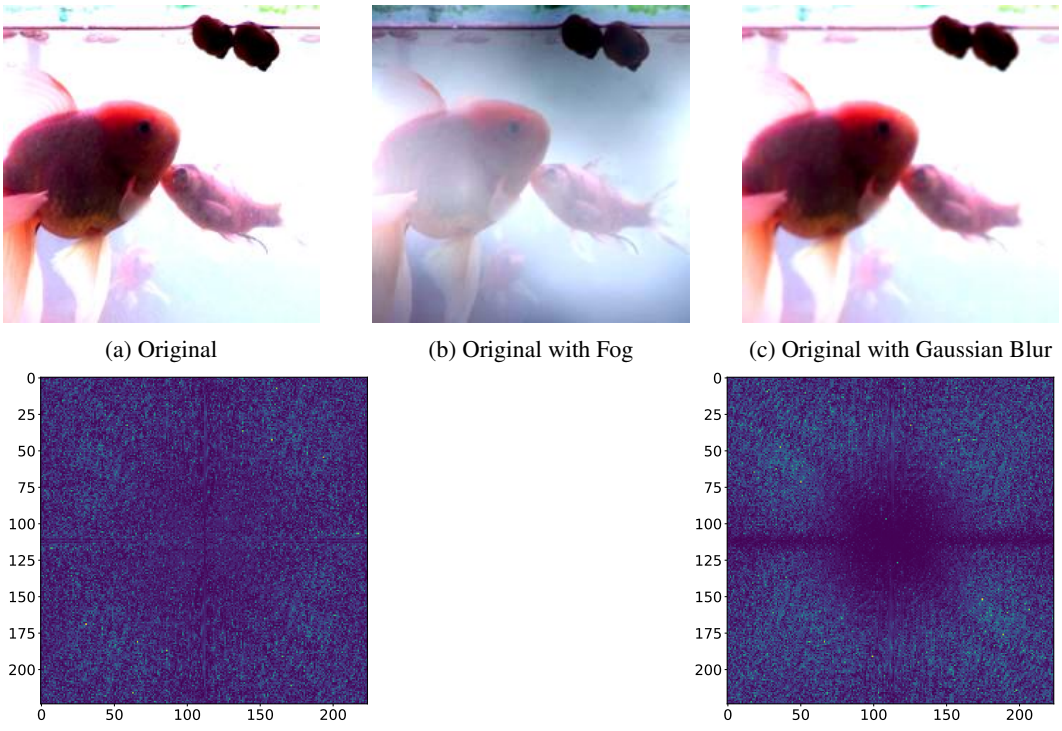
mance even on clean images is not good, it does not show good performance in terms of generalization. Therefore, only PAC-FNO, which uses both low and high-frequency components, shows good performance in terms of accuracy/generalization.

F.8 Effectiveness of parallel architecture

In this section, we show the efficacy of the parallel configuration of the AC-FNO block by visualizing which frequencies are captured. For fair comparison, we visualize the first layer output, which contains the most information of an original input sample, for the following two settings: AC-FNO in our proposed parallel configuration and AC-FNO in a serial configuration. Figure 7 shows spectral responses according to the configuration of the AC-FNO block. The farther it is from the center, the higher its frequency is.

In Figure 7, We show that in the parallel configuration, each hidden vector not only captures the low-frequency components but also captures the high-frequency components. Moreover, their frequencies are sometimes complementary to each other. In particular, $\mathbf{h}_4^1$ has large normalized magnitudes at high frequency ranges, which means that $\mathbf{h}_3^1$ captures high-frequency components well. On the other hand, the hidden vector of the serial configuration (dashed line) is concentrated at low-frequency (center).

Additionally, since the parallel configuration of the AC-FNO block captures both high and low-frequency components, it also shows good performance for image degradations as shown in Figure 8. Figures 8d and 8e are visualizations in the frequency domain. In other words, PAC-FNO consider high-frequency information well in those cases.



(a) Original (b) Original with Fog (c) Original with Gaussian Blur

(d) Frequency domain of fog (e) Frequency domain of Gaussian blur

Figure 8: **Visualization of image degradation.** (d) and (e) are visualizations of degradation without clean images e.g., (b)-(a) and (c)-(a) in the frequency domain.

F.9 FLOPs AND RUNTIME

We report the FLOPs and runtime on data at different resolutions

Table 22: Results of FLOPs and runtime on ImageNet-1k in ConvNeXt-Tine.

ImageNet-1k	Method	Metrics	Resolution			
			28	56	112	224
ConvNeXt-Tiny	Resize	GFLOPs	8.96	8.96	8.96	8.96
		Runtimes (s)	0.006	0.006	0.006	0.006
	Fine-tune	GFLOPs	8.96	8.96	8.96	8.96
		Runtimes (s)	0.006	0.006	0.006	0.006
	DRLN	GFLOPs	180.66	412.05	1200.50	8.96
		Runtimes (s)	0.378	0.498	0.913	0.007
	DRPN	GFLOPs	1220.42	576.94	387.88	8.96
		Runtimes (s)	0.1532	0.164	0.171	0.007
	FNO	GFLOPs	9.78	9.78	9.78	9.78
		Runtimes (s)	0.016	0.016	0.016	0.016
	UNO	GFLOPs	9.10	9.10	9.10	9.10
		Runtimes (s)	0.018	0.018	0.018	0.018
	AFNO	GFLOPs	8.96	8.96	8.96	8.96
		Runtimes (s)	0.010	0.010	0.010	0.010
	PAC-FNO	GFLOPs	8.98	8.98	8.98	8.98
		Runtimes (s)	0.013	0.013	0.013	0.013