ParNeC 模式识别与神经计算研究组
PAttern Recognition and NEural Computing

Adaptive Fourier Neural Operators: Efficient Token Mixers for Transformers

John Guibas[3*], **Morteza Mardani**[1,3*], **Zongyi Li**[1,2], **Andrew Tao**[1],
Anima Aanandkumar[1,2], **Bryan Catanzaro**[1]
NVIDIA[1], California Institute of Technology[2], Stanford University[3]

ICLR 2022

Compared with the sota transformers

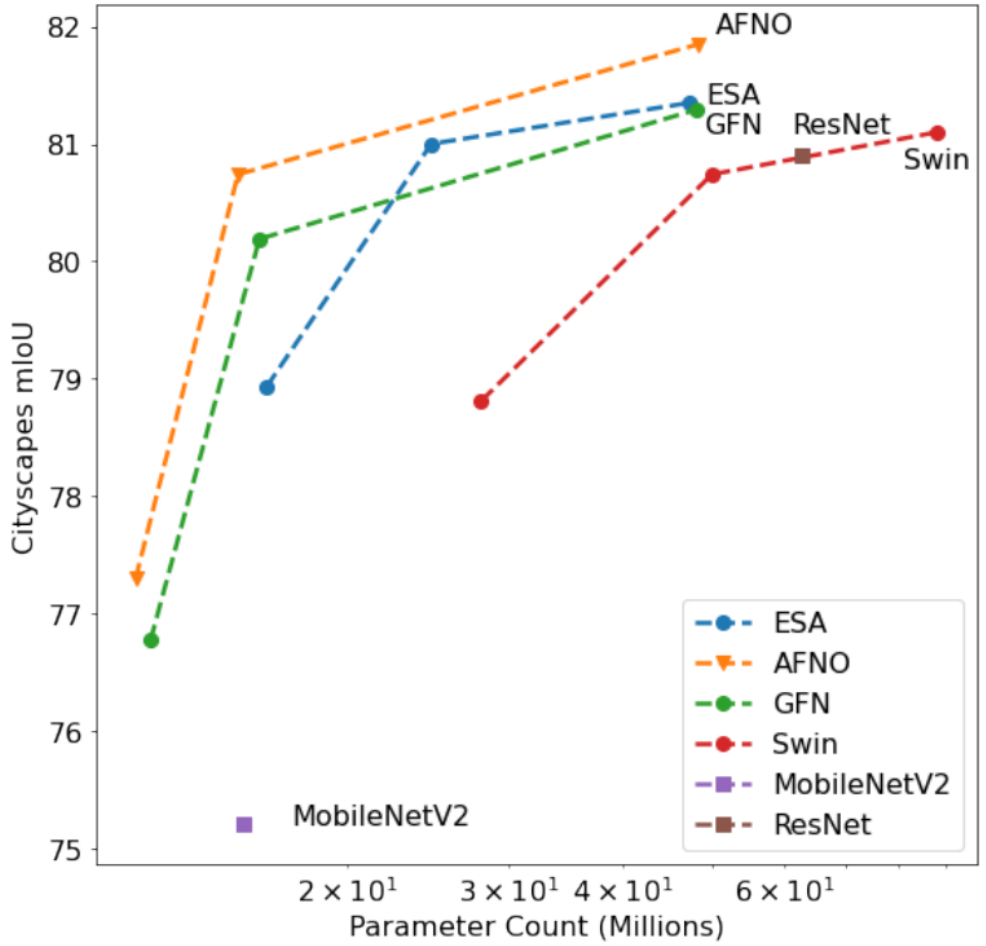
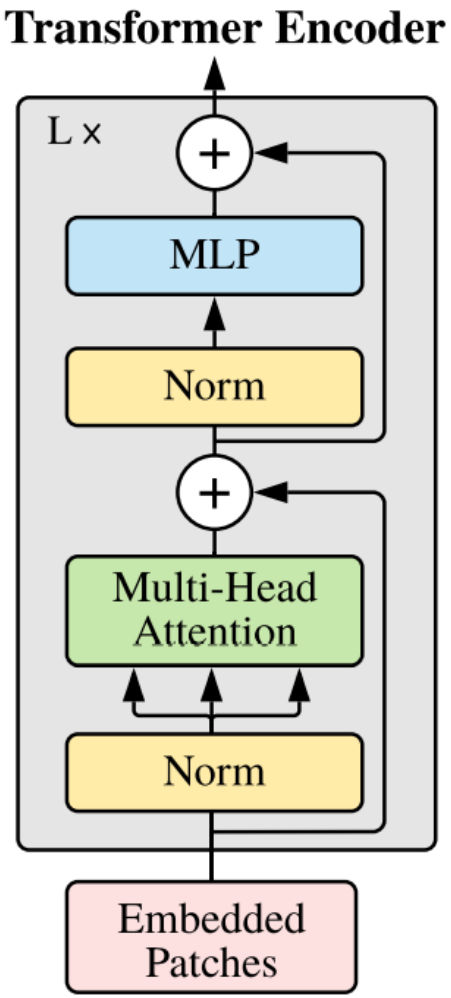
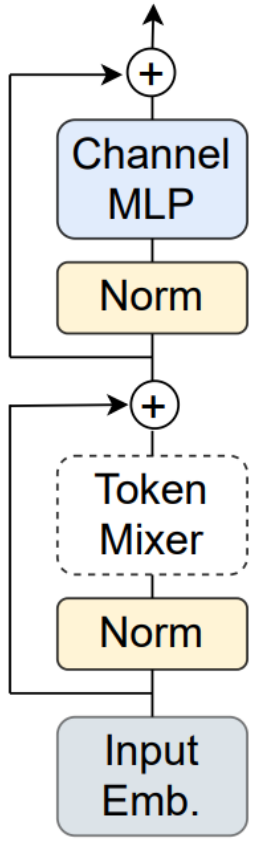


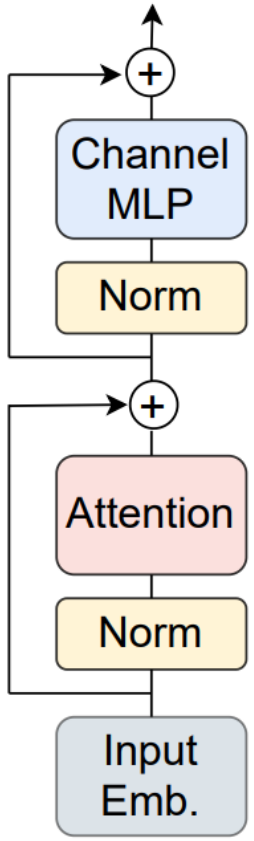
Figure 1: Parameter count and mIoU for Segformer, Swin, and other models at different scales. AFNO consistently outperforms other mixers (see Section 5.7).

MetaFormer and Token Mixer

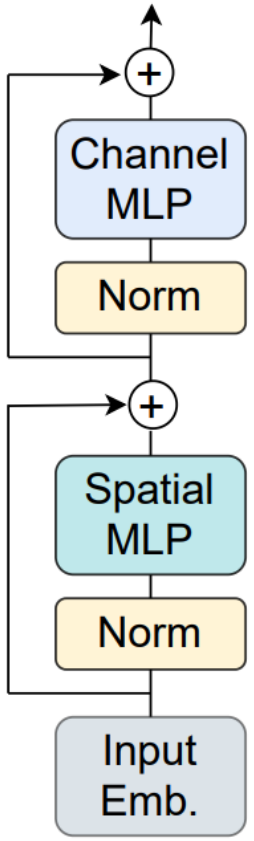




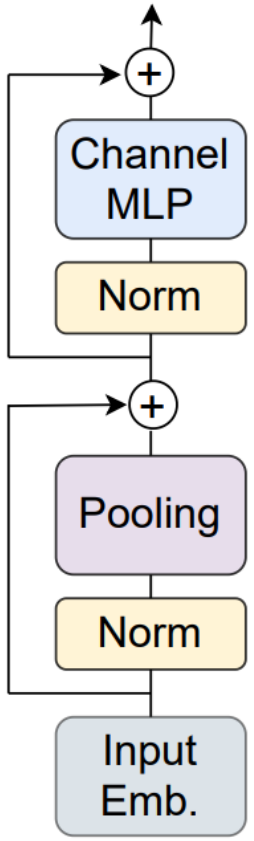
MetaFormer
(General Arch.)



Transformer
(e.g. DeiT)

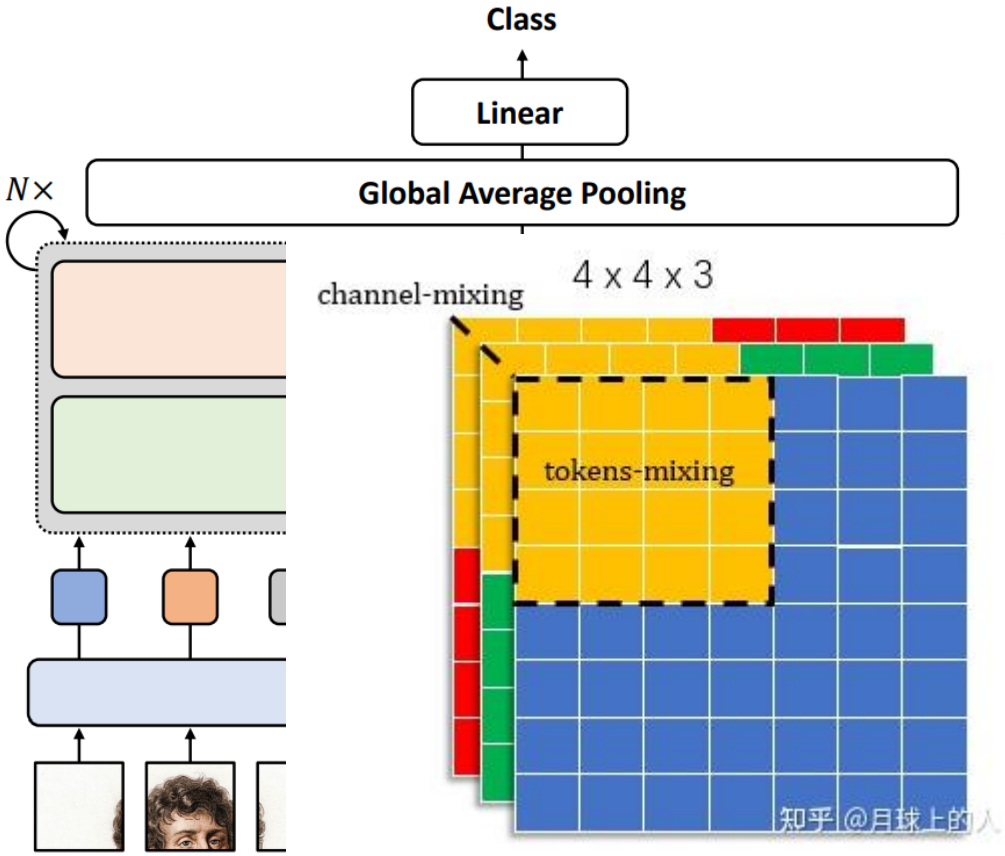


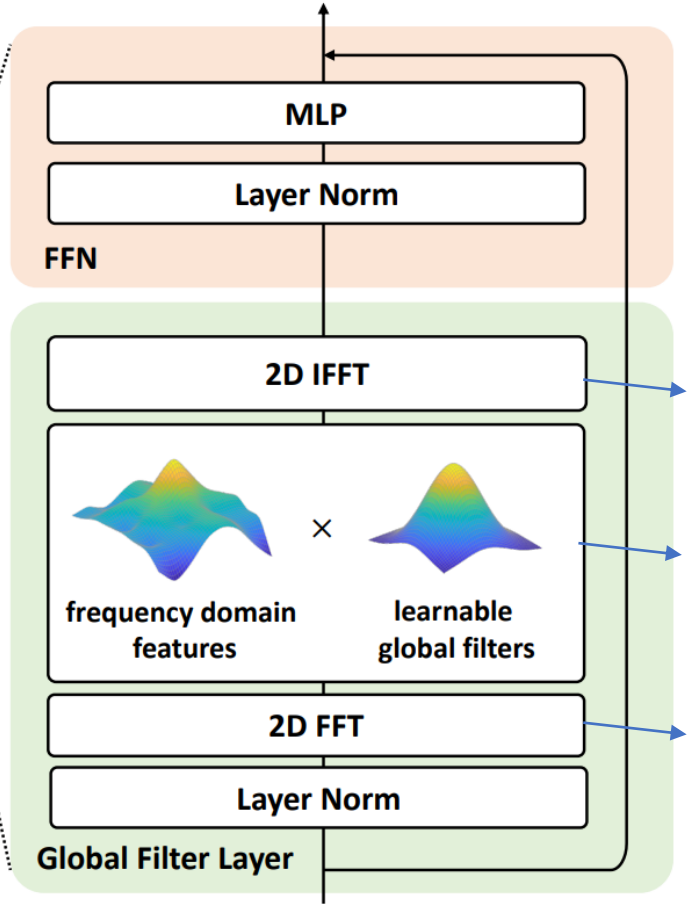
MLP-like model
(e.g. ResMLP)



PoolFormer
(Ours)

Global Filter Networks (GFN)





③ inverse FFT for token demixing

② frequency gating

① spatial token mixing via fast Fourier transform (FFT)

Shortcomings:

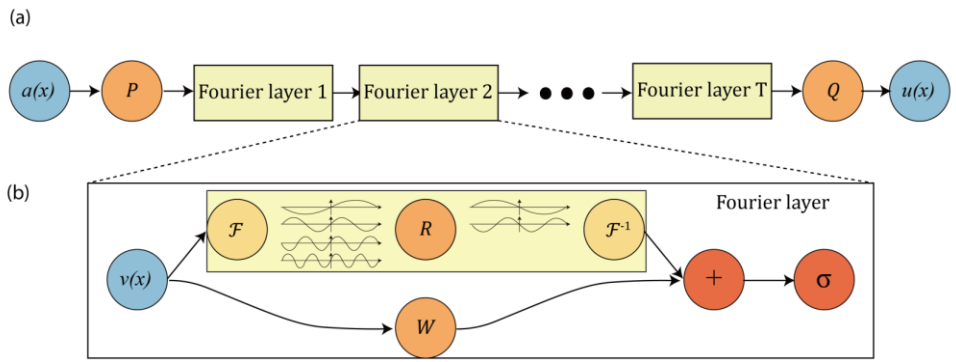
1. lacks adaptivity and expressiveness at high resolutions since the parameter count grows with the sequence size
2. no channel mixing is involved in ②

Approach

Token mixing → Operator learning

continuous elements in the function space

continuous global convolution



Fourier neural operator

Definition 4 (Fourier Neural Operator). For the continuous input $X \in D$ and kernel κ, the kernel integral at token s is found as

$$\mathcal{K}(X)(s) = \mathcal{F}^{-1}\big(\mathcal{F}(\kappa) \cdot \mathcal{F}(X)\big)(s) \qquad \forall s \in D,$$

Modification

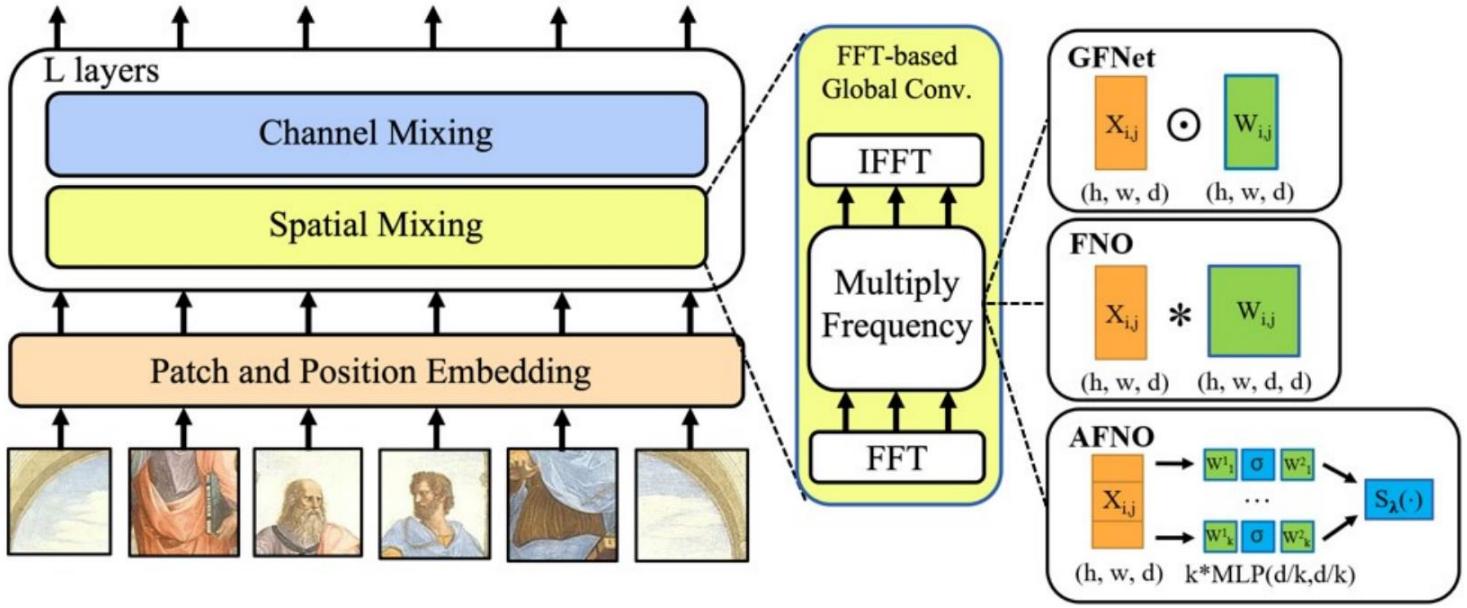


Figure 2: The multi-layer transformer network with FNO, GFN, and AFNO mixers. GFNet performs element-wise matrix multiplication with separate weights across channels (k). FNO performs full matrix multiplication that mixes all the channels. AFNO performs block-wise channel mixing using MLP along with soft-thresholding. The symbols h, w, d, and k refer to the height, width, channel size, and block count, respectively.

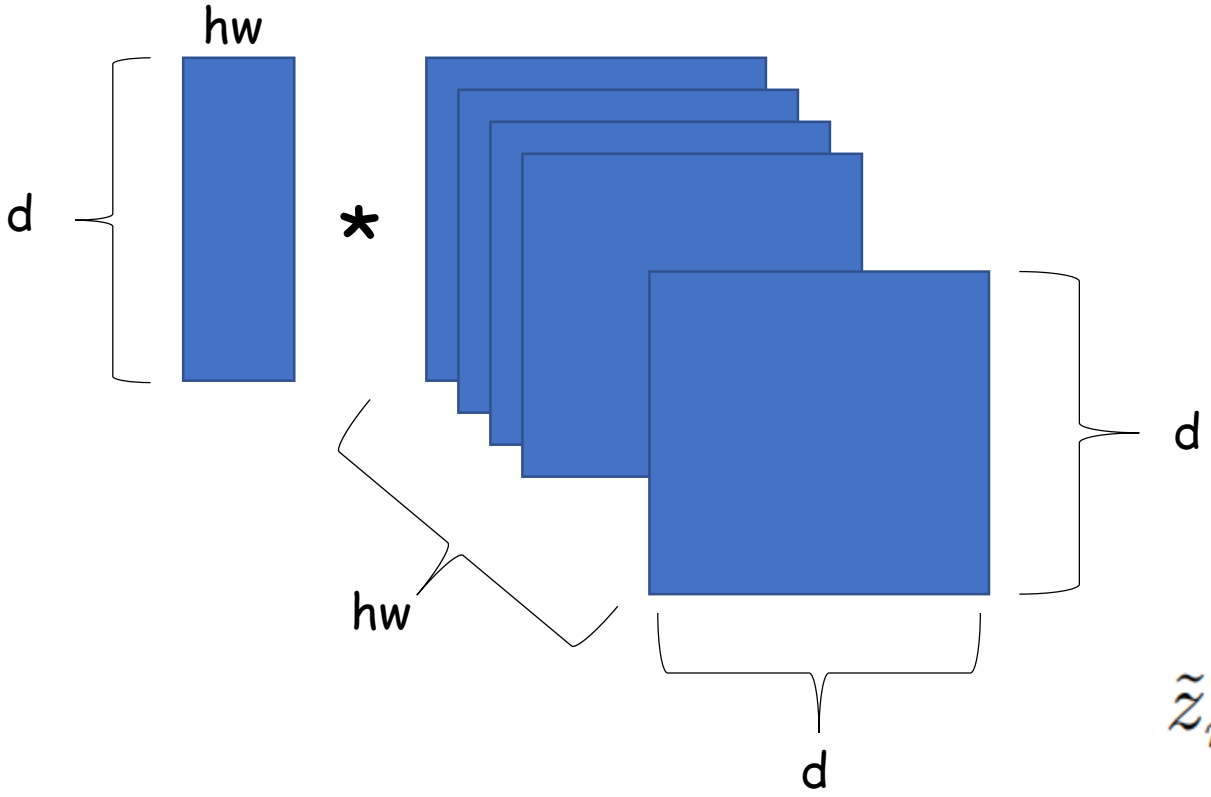
```
def AFNO(x)
   bias = x
   x = RFFT2(x)
   x = x.reshape(b, h, w//2+1, k, d/k)
   x = BlockMLP(x)
   x = x.reshape(b, h, w//2+1, d)
   x = SoftShrink(x)
   x = IRFFT2(x)
   return x + bias
```

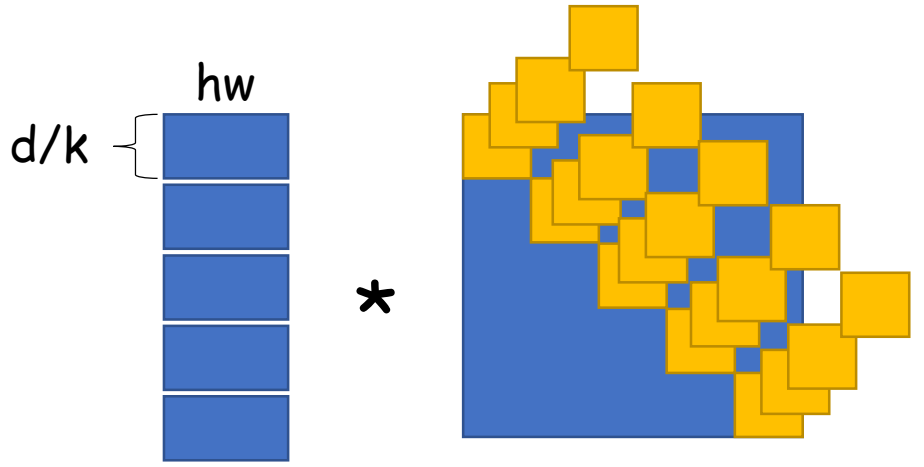
```
x = Tensor[b, h, w, d]
W_1, W_2 = ComplexTensor[k, d/k, d/k]
b_1, b_2 = ComplexTensor[k, d/k]


def BlockMLP(x):
   x = MatMul(x, W_1) + b_1
   x = ReLU(x)
   return MatMul(x, W_2) + b_2
```

Figure 3: Pseudocode for AFNO with adaptive weight sharing and adaptive masking.

Modification: Block-Diagonal Structure and Weight Sharing





$$\tilde{z}_{m,n}^{(\ell)} = W_{m,n}^{(\ell)} z_{m,n}^{(\ell)}, \quad \ell = 1, \ldots, k$$

$$\tilde{z}_{m,n} = \mathrm{MLP}(z_{m,n}) = W_2 \sigma(W_1 z_{m,n}) + b$$

Modification: Soft-Thresholding

Images are inherently sparse in the Fourier domain, and most of the energy is concentrated around low frequency modes.

Thus, one can adaptively mask the tokens according to their importance towards the end task.

$$\tilde{z}_{m,n}^{(\ell)} = W_{m,n}^{(\ell)} z_{m,n}^{(\ell)}, \quad \ell = 1, \ldots, k$$

$$\min \|\tilde{z}_{m,n} - W_{m,n} z_{m,n}\|^2 + \lambda \|\tilde{z}_{m,n}\|_1$$

$$\tilde{z}_{m,n} = S_\lambda(W_{m,n} z_{m,n}) \qquad S_\lambda(x) = \operatorname{sign}(x) \max\{|x| - \lambda, 0\}$$

$$\tilde{z}_{m,n} = \mathrm{MLP}(z_{m,n}) = W_2 \sigma(W_1 z_{m,n}) + b$$

Experiments

南京航空航天大學 NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

ImageNet-1k inpainting

Backbone	Mixer	Params	GFLOPs	Latency(sec)	SSIM	PSNR(dB)
ViT-B/4	Self-Attention	87M	357.2	1.2	**0.931**	**27.06**
ViT-B/4	LS	87M	274.2	1.4	0.920	26.18
ViT-B/4	GFN	87M	177.8	0.7	0.928	26.76
ViT-B/4	AFNO (ours)	87M	257.2	0.8	**0.931**	27.05

Table 2: Inpainting PSNR and SSIM for ImageNet-1k validation data. AFNO matches the performance of Self-Attention despite using significantly less FLOPs.

Few-shot segmentation

Backbone	Mixer	Params	GFLOPs	LSUN-Cats	ADE-Cars	CelebA-Faces
ViT-B/4	Self-Attention	87M	357.2	35.57	49.26	**56.91**
ViT-B/4	LS	87M	274.2	20.29	29.66	41.36
ViT-B/4	GFN	87M	177.8	34.52	47.84	55.21
ViT-B/4	AFNO (ours)	87M	257.2	**35.73**	**49.60**	55.75

Table 3: Few-shot segmentation mIoU for AFNO versus alternative mixers. AFNO surpasses Self-Attention for 2/3 datasets while using less flops.

Experiments

南京航空航天大學 NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

Cityscapes segmentation

Backbone	Mixer	Params	Total GFLOPs	Mixer GFLOPs	mIoU
Segformer-B3/4	SA	45M	N/A	825.7	N/A
Segformer-B3/4	Efficient SA	45M	380.7	129.9	79.7
Segformer-B3/4	LS	45M	409.1	85.0	80.5
Segformer-B3/4	GFN	45M	363.4	2.6	80.4
Segformer-B3/4	AFNO-100% (ours)	45M	440.0	23.7	**80.9**
Segformer-B3/4	AFNO-25% (ours)	45M	429.0	12.4	80.4

Table 4: mIoU and FLOPs for Cityscapes segmentation at 1024×1024 resolution. Note, both the mixer and total FLOPs are included. For GFN and AFNO, the MLP layers are the bottleneck for the complexity. Also, AFNO-25% only keeps 25% of the low frequency modes, while AFNO-100% keeps all the modes. Results for self-attention cannot be obtained due to the long sequence length in the first few layers.

ImageNet-1k classification

Backbone	Mixer	Params	GFLOPs	Top-1 Accuracy	Top-5 Accuracy
ViT-S/4	LS	16M	15.8	80.87	95.31
ViT-S/4	GFN	16M	6.1	78.77	94.4
ViT-S/4	AFNO (ours)	16M	15.3	**80.89**	**95.39**

Table 5: ImageNet-1K classification efficiencyy-accuracy trade-off when the input resolution is 224×224.

Experiments

南京航空航天大學 NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

Ablation studies

Backbone	Mixer	Parameter Count	PSNR	CelebA-Faces mIoU
ViT-XS/4	FNO	16M	24.8	39.27
ViT-XS/4	AFNO [Non-Adaptive Weights]	16M	25.1	44.04
ViT-XS/4	AFNO [Hard Thresholding 35%]	16M	23.58	34.17
ViT-XS/4	AFNO	16M	**25.69**	**49.49**

Table 6: Ablations for AFNO versus FNO, AFNO without adaptive weights, and hard thresholding. Results are on inpainting pretraining with 10% of ImageNet along with few-show segmentation mIoU on CelebA-Faces. Hard thresholding only keeps 35% of low frequency modes. AFNO demonstrates superior performance for the same parameter count in both tasks.

Visualization

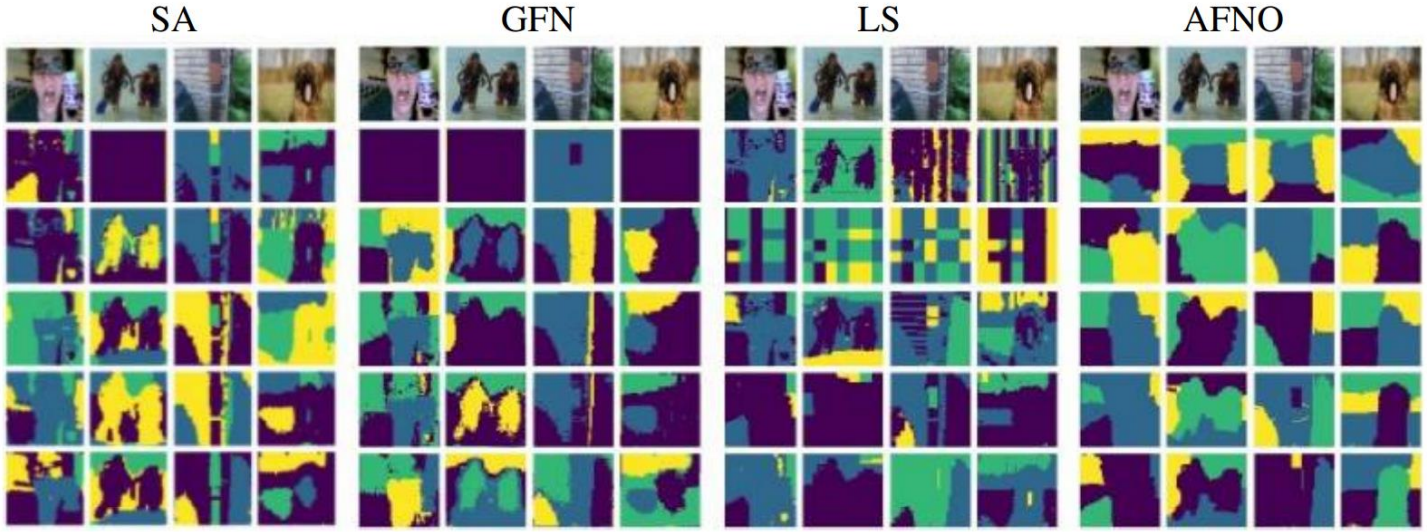


Figure 5: Spectral clustering of tokens for different token mixers. From top to bottom, it shows the input and the layers $2, 4, 6, 8, 10$ for the inpainting pretrained model.

Thanks