Global Filter Networks for Image Classification

Yongming Rao Wenliang Zhao Zheng Zhu Jiwen Lu Jie Zhou
Department of Automation, Tsinghua University

Abstract

Recent advances in self-attention and pure multi-layer perceptrons (MLP) models for vision have shown great potential in achieving promising performance with fewer inductive biases. These models are generally based on learning interaction among spatial locations from raw data. The complexity of self-attention and MLP grows quadratically as the image size increases, which makes these models hard to scale up when high-resolution features are required. In this paper, we present the Global Filter Network (GFNet), a conceptually simple yet computationally efficient architecture, that learns long-term spatial dependencies in the frequency domain with log-linear complexity. Our architecture replaces the self-attention layer in vision transformers with three key operations: a 2D discrete Fourier transform, an element-wise multiplication between frequency-domain features and learnable global filters, and a 2D inverse Fourier transform. We exhibit favorable accuracy/complexity trade-offs of our models on both ImageNet and downstream tasks. Our results demonstrate that GFNet can be a very competitive alternative to transformer-style models and CNNs in efficiency, generalization ability and robustness. Code is available at `https://github.com/raoyongming/GFNet`.

1 Introduction

The transformer architecture, originally designed for the natural language processing (NLP) tasks [43], has shown promising performance on various vision problems recently [9, 40, 27, 49, 4]. Different from convolutional neural networks (CNNs), vision transformer models use self-attention layers to capture long-term dependencies, which are able to learn more diverse interactions between spatial locations. The pure multi-layer perceptrons (MLP) models [38, 39] further simplify the vision transformers by replacing the self-attention layers with MLPs that are applied across spatial locations. Since fewer inductive biases are introduced, these two kinds of models have the potential to learn more generic and flexible interactions among spatial locations from raw data.

One primary challenge of applying self-attention and pure MLP models to vision tasks is the considerable computational complexity that grows quadratically as the number of tokens increases. Therefore, typical vision transformer style models usually consider a relatively small resolution for the intermediate features (*e.g*. 14×14 tokens are extracted from the input images in both ViT [9] and MLP-Mixer [38]). This design may limit the applications of downstream dense prediction tasks like detection and segmentation. A possible solution is to replace the global self-attention with several local self-attention like Swin transformer [27]. Despite the effectiveness in practice, local self-attention brings quite a few hand-made choices (*e.g*., window size, padding strategy, *etc*.) and limits the receptive field of each layer.

In this paper, we present a new conceptually simple yet computationally efficient architecture called Global Filter Network (*GFNet*), which follows the trend of removing inductive biases from vision models while enjoying the log-linear complexity in computation. The basic idea behind our architecture is to learn the interactions among spatial locations in the frequency domain. Different from the self-attention mechanism in vision transformers and the fully connected layers in MLP models, the interactions among tokens are modeled as a set of learnable *global filters* that are applied

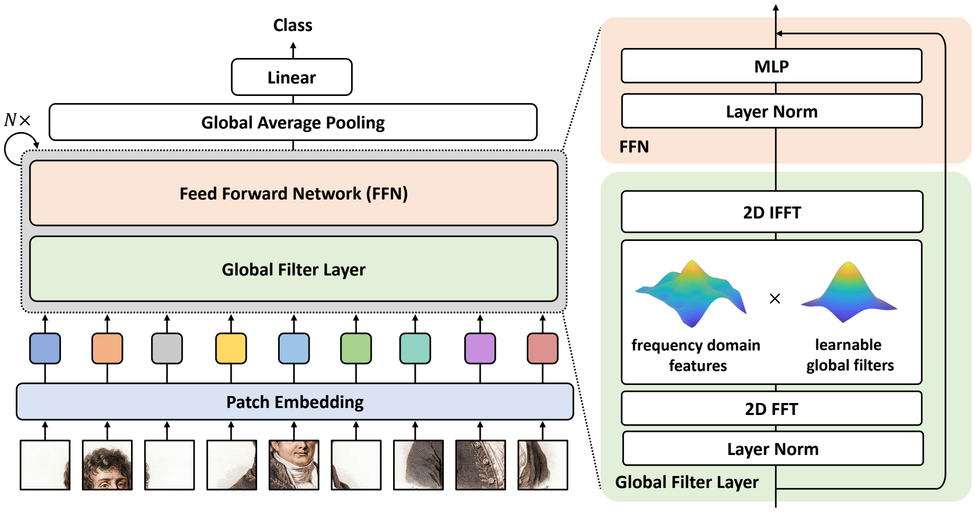


Figure 1: **The overall architecture of the Global Filter Network**. Our architecture is based on Vision Transformer (ViT) models with some minimal modifications. We replace the self-attention sub-layer with the proposed *global filter layer*, which consists of three key operations: a 2D discrete Fourier transform to convert the input spatial features to the frequency domain, an element-wise multiplication between frequency-domain features and the global filters, and a 2D inverse Fourier transform to map the features back to the spatial domain. The efficient fast Fourier transform (FFT) enables us to learn arbitrary interactions among spatial locations with log-linear complexity.

	Complexity (FLOPs)	# Parameters
Depthwise Convolution	k^2HWD	k^2D
Self-Attention	$4HWD^2 + 2H^2W^2D$	$4D^2$
Spatial MLP	H^2W^2D	H^2W^2
Global Filter	$HWD\lceil\log_2(HW)\rceil + HWD$	HWD

Table 1: Comparisons of the proposed *Global Filter* with prevalent operations in deep vision models. H, W and D are the height, width and the number of channels of the feature maps. k is the kernel size of the convolution operation. The proposed global filter is much more efficient than self-attention and spatial MLP.

to the spectrum of the input features. Since the global filters are able to cover all the frequencies, our model can capture both long-term and short-term interactions. The filters are directly learned from the raw data without introducing human priors. Our architecture is largely based on the vision transformers only with some minimal modifications. We replace the self-attention sub-layer in vision transformers with three key operations: a 2D discrete Fourier transform to convert the input spatial features to the frequency domain, an element-wise multiplication between frequency-domain features and the global filters, and a 2D inverse Fourier transform to map the features back to the spatial domain. Since the Fourier transform is used to mix the information of different tokens, the global filter is much more efficient compared to the self-attention and MLP thanks to the $\mathcal{O}(L\log L)$ complexity of the fast Fourier transform algorithm (FFT) [6]. Benefiting from this, the proposed global filter layer is less sensitive to the token length L and thus is compatible with larger feature maps and CNN-style hierarchical architectures *without modifications*. The overall architecture of GFNet is illustrated in Figure 1. We also compare our global filter with prevalent operations in deep vision models in Table 1.

Our experiments on ImageNet verify the effectiveness of GFNet. With a similar architecture, our model outperform the recent vision transformer and MLP models including DeiT [40], ResMLP [39] and gMLP [26]. When using the hierarchical architecture, GFNet can further enlarge the gap. GFNet also works well on downstream transfer learning and semantic segmentation tasks. Our results demonstrate that GFNet can be a very competitive alternative to transformer-style models and CNNs in efficiency, generalization ability and robustness.

2 Related works

Vision transformers. Since Dosovitskiy *et al.* [9] introduce transformers to the image classification and achieve a competitive performance compared to CNNs, transformers begin to exhibit their potential in various vision tasks [3, 4, 49]. Recently, there are a large number of works which aim to improve the transformers [40, 41, 27, 45, 18, 10, 48]. These works either seek for better training strategies [40, 10] or design better architectures [27, 45, 48] or both [41, 10]. However, most of the architecture modification of the transformers [45, 18, 27, 48] introduces additional inductive biases similar to CNNs. In this work, we only focus on the standard transformer architecture [9, 40] and our goal is to replace the heavy self-attention layer ($\mathcal{O}(L^2)$) to an more efficient operation which can still model the interactions among different spatial locations without introducing the inductive biases associated with CNNs.

MLP-like models. More recently, there are several works that question the importance of self-attention in the vision transformers and propose to use MLP to replace the self-attention layer in the transformers [38, 39, 26]. The MLP-Mixer [38] employs MLPs to perform token mixing and channel mixing alternatively in each block. ResMLP [39] adopts a similar idea but substitutes the Layer Normalization with an Affine transformation for acceleration. The recently proposed gMLP [26] uses a spatial gating unit to re-weight tokens in the spatial dimension. However, all of the above models include MLPs to mix the tokens spatially, which brings two drawbacks: (1) like the self-attention in the transformers, the spatial MLP still requires computational complexity quadratic to the length of tokens. (2) unlike transformers, MLP models are hard to scale up to higher resolution since the weights of the spatial MLPs have fixed sizes. Our work follows this trend and successfully resolves the above issues in MLP-like models. The proposed GFNet enjoys log-linear complexity and can be easily scaled up to any resolution.

Applications of Fourier transform in vision. Fourier transform has been an important tool in digital image processing for decades [33, 1]. With the breakthroughs of CNNs in vision [13, 12], there are a variety of works that start to incorporate Fourier transform in some deep learning method [25, 47, 8, 23]. Some of these works employ discrete Fourier transform to convert the images to the frequency domain and leverage the frequency information to improve the performance in certain tasks [23, 47], while others utilize the convolution theorem to accelerate the CNNs via fast Fourier transform (FFT) [25, 8]. In this work, we propose to use learnable filters to interchange information among the tokens in the Fourier domain, inspired by the frequency filters in the digital image processing [33]. We also take advantage of some properties of FFT to reduce the computational costs and the number of parameters.

3 Method

3.1 Preliminaries: discrete Fourier transform

We start by introducing the discrete Fourier transform (DFT), which plays an important role in the area of digital signal processing and is a crucial component in our GFNet. For clarity, We first consider the 1D DFT. Given a sequence of N complex numbers $x[n], 0 \leq n \leq N-1$, the 1D DFT converts the sequence into the frequency domain by:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn} := \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{3.1}$$

where j is the imaginary unit and $W_N = e^{-j(2\pi/N)}$. The formulation of DFT in Equation (3.1) can be derived from the Fourier transform for continuous signal by sampling in both the time domain and the frequency domain (see Appendix A for details). Since $X[k]$ repeats on intervals of length N, it is suffice to take the value of $X[k]$ at N consecutive points $k = 0, 1, \ldots, N-1$. Specifically, $X[k]$ represents to the spectrum of the sequence $x[n]$ at the frequency $\omega_k = 2\pi k/N$.

It is also worth noting that DFT is a one-to-one transformation. Given the DFT $X[k]$, we can recover the original signal $x[n]$ by the inverse DFT (IDFT):

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j(2\pi/N)kn}. \tag{3.2}$$

Algorithm 1 Pseudocode of Global Filter Layer.

```
# x: the token features, B x H x W x D (where N = H * W)
# K: the frequency-domain filter, H x W_hat x D (where W_hat = W // 2 + 1, see Section 3.2 for details)

X = rfft2(x, dim=(1, 2))
X_tilde = X * K
x = irfft2(X_tilde, dim=(1, 2))
```

`rfft2/irfft2`: 2D FFT/IFFT for real signal

For real input $x[n]$, it can be proved that (see Appendix A) its DFT is conjugate symmetric, i.e., $X[N-k] = X^*[k]$. The reverse is true as well: if we perform IDFT to $X[k]$ which is conjugate symmetric, a real discrete signal can be recovered. This property implies that the half of the DFT $\{X[k] : 0 \le k \le \lceil N/2 \rceil\}$ contains the full information about the frequency characteristics of $x[n]$.

DFT is widely used in modern signal processing algorithms for mainly two reasons: (1) the input and output of DFT are both discrete thus can be easily processed by computers; (2) there exist efficient algorithms for computing the DFT. The *fast Fourier transform* (FFT) algorithms take advantage of the symmetry and periodicity properties of W_N^{kn} and reduce the complexity to compute DFT from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. The inverse DFT (3.2), which has a similar form to the DFT, can also be computed efficiently using the inverse fast Fourier transform (IFFT).

The DFT described above can be extend to 2D signals. Given the 2D signal $X[m,n], 0 \le m \le M-1, 0 \le n \le N-1$, the 2D DFT of $x[m,n]$ is given by:

$$X[u,v] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x[m,n] e^{-j2\pi\left(\frac{um}{M} + \frac{vn}{N}\right)}. \tag{3.3}$$

The 2D DFT can be viewed as performing 1D DFT on the two dimensions alternatively. Similar to 1D DFT, 2D DFT of real input $x[m,n]$ satisfied the conjugate symmetry property $X[M-u, N-v] = X^*[u,v]$. The FFT algorithms can also be applied to 2D DFT to improve computational efficiency.

3.2 Global Filter Networks

Overall architecture. Recent advances in vision transformers [9, 40] demonstrate that models based on self-attention can achieve competitive performance even without the inductive biases associated with the convolutions. Henceforth, there are several works [39, 38] that exploit approaches (*e.g.*, MLPs) other than self-attention to mix the information among the tokens. The proposed Global Filter Networks (GFNet) follows this line of work and aims to replace the heavy self-attention layer ($\mathcal{O}(N^2)$) with a simpler and more efficient one.

The overall architecture of our model is depicted in Figure 1. Our model takes as an input $H \times W$ non-overlapping patches and projects the flattened patches into $L = HW$ tokens with dimension D. The basic building block of GFNet consists of: 1) a *global filter layer* that can exchange spatial information efficiently ($\mathcal{O}(L \log L)$); 2) a feedforward network (FFN) as in [9, 40]. The output tokens of the last block are fed into a global average pooling layer followed by a linear classifier.

Global filter layer. We propose global filter layer as an alternative to the self-attention layer which can mix tokens representing different spatial locations. Given the tokens $\boldsymbol{x} \in \mathbb{R}^{H \times W \times D}$, we first perform 2D FFT (see Section 3.1) along the spatial dimensions to convert $\boldsymbol{x}$ to the frequency domain:

$$\boldsymbol{X} = \mathcal{F}[\boldsymbol{x}] \in \mathbb{C}^{H \times W \times D}, \tag{3.4}$$

where $\mathcal{F}[\cdot]$ denotes the 2D FFT. Note that $\boldsymbol{X}$ is a complex tensor and represents the spectrum of $\boldsymbol{x}$. We can then modulate the spectrum by multiplying a learnable filter $\boldsymbol{K} \in \mathbb{C}^{H \times W \times D}$ to the $\boldsymbol{X}$:

$$\tilde{\boldsymbol{X}} = \boldsymbol{K} \odot \boldsymbol{X}, \tag{3.5}$$

where $\odot$ is the element-wise multiplication (also known as the Hadamard product). The filter $\boldsymbol{K}$ is called the *global filter* since it has the same dimension with $\boldsymbol{X}$, which can represent an arbitrary filter in the frequency domain. Finally, we adopt the inverse FFT to transform the modulated spectrum $\tilde{X}$ back to the spatial domain and update the tokens:

$$\boldsymbol{x} \leftarrow \mathcal{F}^{-1}[\tilde{\boldsymbol{X}}]. \tag{3.6}$$

The formulation of the global filter layer is motivated by the frequency filters in the digital image processing [33], where the global filter $\boldsymbol{K}$ can be regarded as a set of learnable frequency filters for different hidden dimensions. It can be proved (see Appendix A) that the global filter layer is equivalent to a depthwise *global circular convolution* with the filter size $H \times W$. Therefore, the global filter layer is different from the standard convolutional layer which adopts a relatively small filter size to enforce the inductive biases of the locality. We also find although the proposed global filter can also be interpreted as a spatial domain operation, the filters learned in our networks exhibit more clear patterns in the frequency domain than the spatial domain, which indicates our models tend to capture relation in the frequency domain instead of spatial domain (see Figure 4). Note that the global filter implemented in the frequency domain is also much more efficient compared to the spatial domain, which enjoys a complexity of $\mathcal{O}(DL \log L)$ while the vanilla depthwise global circular convolution in the spatial domain has $\mathcal{O}(DL^2)$ complexity. We will also show that the global filter layer is better than its local convolution counterparts in the experiments.

It is also worth noting that in the implementation, we make use of the property of DFT to reduce the redundant computation. Since $\boldsymbol{x}$ is a real tensor, its DFT $\boldsymbol{X}$ is conjugate symmetric, *i.e.* $\boldsymbol{X}[H-u, W-v, :] = \boldsymbol{X}^*[H, W, :]$. Therefore, we can take only the half of the values in the $\boldsymbol{X}$ but preserve the full information at the same time:

$$\boldsymbol{X}_r = \boldsymbol{X}[:, 0:\widehat{W}] := \mathcal{F}_r[\boldsymbol{x}], \quad , \widehat{W} = \lceil W/2 \rceil, \tag{3.7}$$

Where $\mathcal{F}_r$ denotes the 2D FFT for real input. In this way, we can implement the global filter as $\boldsymbol{K}_r \in \mathbb{C}^{H \times \widehat{W} \times D}$, which can reduce half the parameters. This can also ensure $\mathcal{F}_r^{-1}[\boldsymbol{K}_r \odot \boldsymbol{X}_r]$ is a real tensor, thus it can be added directly to the input $\boldsymbol{x}$. The global filter layer can be easily in modern deep learning frameworks (*e.g.*, PyTorch [32]), as is shown in Algorithm 1. The FFT and ITTF are well supported by GPU and CPU thanks to the acceleration libraries like `cuFFT` and `mkl-fft`, which makes our models perform well on hardware.

Relationship to other transformer-style models. The GFNet follows the line of research about the exploration of approaches to mix the tokens. Compared to existing architectures like vision transformers and pure MLP models, we exhibit that GFNet has several favorable properties: 1) GFNet is more efficient. The complexity of both the vision transformers [9, 40, 41] and the MLP models [38, 39] is $\mathcal{O}(L^2)$. Different from them, global filter layer only consists an FFT ($\mathcal{O}(L \log L)$), an element-wise multiplication ($\mathcal{O}(L)$) and an IFFT ($\mathcal{O}(L)$), which means the total computational complexity is $\mathcal{O}(L \log L)$. 2) Although pure MLP models are simpler compared to transformers, it is hard to fine-tune them on higher resolution (*e.g.*, from 224 × 224 resolution to 384 × 384 resolution) since they can only process a fixed number of tokens. As opposed to pure MLP models, we will show that our GFNet can be easily scaled up to higher resolution. Our model is more flexible since both the FFT and the IFFT have no learnable parameters and can process sequences with arbitrary length. We can simply interpolate the global filter $\boldsymbol{K}$ to $\boldsymbol{K}' \in \mathbb{C}^{H' \times W' \times D}$ for different inputs, where $H' \times W'$ is the target size. The interpolation is reasonable due to the property of DFT. Each element of the global filter $\boldsymbol{K}[u, v]$ corresponds to the spectrum of the filter at $\omega_u = 2\pi u/H, \omega_v = 2\pi v/W$ and thus, the global filter $\boldsymbol{K}$ can be viewed as a sampling of a continuous spectrum $\boldsymbol{K}(\omega_u, \omega_v)$, where $\omega_u, \omega_v \in [0, 2\pi]$. Hence, changing the resolution is equivalent to changing the sampling interval of $\boldsymbol{K}(\omega_u, \omega_v)$. Therefore, we only need to perform interpolation to shift from one resolution to another.

We also notice recently a concurrent work FNet [24] leverages Fourier transform to mix tokens. Our work is distinct from FNet in three aspects: (1) FNet performs FFT to the input and directly adds the real part of the spectrum to the input tokens, which blends the information from different domains (spatial/frequency) together. On the other hand, GFNet draws motivation from the frequency filters, which is more reasonable. (2) FNet only keeps the real part of the spectrum. Note that the spectrum of real input is conjugate symmetric, which means the real part is exactly symmetric and thus contains redundant information. Our GFNet, however, utilizes this property to simplify the computation. (3) FNet is designed for NLP tasks, while our GFNet focuses on vision tasks. In our experiments, we also implement the FNet and show that our model outperforms it.

Architecture variants. Due to the limitation from the quadratic complexity in the self-attention, vision transformers [9, 40] are usually designed to process a relatively small feature map (*e.g.*, 14×14). However, our GFNet, which enjoys log-linear complexity, avoids that problem. Since in our GFNet the computational costs do not grow such significantly when the feature map size increases, we can adopt a hierarchical architecture inspired by the success of CNNs [22, 13]. Generally speaking, we can start from a large feature map (*e.g.*, 56 × 56) and gradually perform downsampling after a few

Table 2: **Comparisons with transformer-style architectures on ImageNet.** We compare different transformer-style architectures for image classification including vision transformers [40], MLP-like models [39, 26] and our models that have comparable FLOPs and the number of parameters. We report the top-1 accuracy on the validation set of ImageNet as well as the number of parameters and FLOPs. All of our models are trained with 224×224 images. We use "↑384" to represent models finetuned on 384×384 images for 30 epochs.

Model	Params (M)	FLOPs (G)	Resolution	Top-1 Acc. (%)	Top-5 Acc. (%)
DeiT-Ti [40]	5	1.2	224	72.2	91.1
gMLP-Ti [26]	6	1.4	224	72.0	-
GFNet-Ti	7	1.3	224	74.6	92.2
ResMLP-12 [39]	15	3.0	224	76.6	-
GFNet-XS	16	2.8	224	78.6	94.2
DeiT-S [40]	22	4.6	224	79.8	95.0
gMLP-S [26]	20	4.5	224	79.4	-
GFNet-S	25	4.5	224	80.0	94.9
ResMLP-36 [39]	45	8.9	224	79.7	-
GFNet-B	43	7.9	224	80.7	95.1
GFNet-XS↑384	18	8.4	384	80.6	95.4
DeiT-B [40]	86	17.5	224	81.8	95.6
gMLP-B [26]	73	15.8	224	81.6	-
GFNet-S↑384	28	13.2	384	81.7	95.8
GFNet-B↑384	47	23.3	384	82.1	95.8

blocks. In this paper, we mainly investigate two kinds of variants of GFNet: transformer-style models with a fixed number of tokens in each block and CNN-style hierarchical models with gradually downsampled tokens. For transformer-style models, we begin with a 12-layer model (*GFNet-XS*) with a similar architecture with DeiT-S and ResMLP-12. Then, we obtain 3 variants of the model (*GFNet-Ti*, *GFNet-S* and *GFNet-B*) by simply adjusting the depth and embedding dimension, which have similar computational costs with ResNet-18, 50 and 101 [13]. For hierarchical models, we also design three models (*GFNet-H-Ti*, *GFNet-H-S* and *GFNet-H-B*) that have these three levels of complexity following the design of PVT [44]. We use 4×4 patch embedding to form the input tokens and use a non-overlapping convolution layer to downsample tokens following [44, 27]. Unlike PVT [44] and Swin [27], we directly apply our building block on different stages without any modifications. The detailed architectures can be found in Appendix B.

4 Experiments

We conduct extensive experiments to verify the effectiveness of our GFNet. We present the main results on ImageNet [7] and compare them with various architectures. We also test our models on the downstream transfer learning datasets including CIFAR-10/100 [21], Stanford Cars [20] and Flowers-102 [31], and show the potential of our models on dense prediction tasks on commonly used semantic segmentation benchmark ADE20k [51]. Lastly, we investigate the efficiency and robustness of the proposed models and provide visualization to have an intuitive understanding of our method.

4.1 ImageNet results

Setups. We conduct our main experiments on ImageNet [7], which is a widely used large-scale benchmark for image classification. ImageNet contains roughly 1.2M images from 1,000 categories. Following common practice [13, 40], we train our models on the training set of ImageNet and report the single-crop top-1 accuracy on 50,000 validation images. To fairly compare with previous works [40, 39], we follow the most training details for our models and do not add extra regularization methods like [18]. Different from [40], we does not use EMA model [34], RandomEarse [50] and repeated augmentation [16], which are important to train DeiT while sightly hurting the performance

Table 3: **Comparisons with hierarchical architectures on ImageNet.** We compare different hierarchical architectures for image classification including convolutional neural networks [13, 35], hierarchical vision transformers [44, 27] and our hierarchical models that have comparable FLOPs and number of parameters. We report the top-1 accuracy on the validation set of ImageNet as well as the number of parameters and FLOPs. All models are trained and tested with 224 × 224 images.

Model	Params (M)	FLOPs (G)	Top-1 Acc. (%)	Top-5 Acc. (%)
ResNet-18 [13]	12	1.8	69.8	89.1
RegNetY-1.6GF [35]	11	1.6	78.0	-
PVT-Ti [26]	13	1.9	75.1	-
GFNet-H-Ti	15	2.0	80.1	95.1
ResNet-50 [40]	26	4.1	76.1	92.9
RegNetY-4.0GF [35]	21	4.0	80.0	-
PVT-S [26]	25	3.8	79.8	-
Swin-Ti [27]	29	4.5	81.3	-
GFNet-H-S	32	4.5	81.5	95.6
ResNet-101 [40]	45	7.9	77.4	93.5
RegNetY-8.0GF [35]	39	8.0	81.7	-
PVT-M [26]	44	6.7	81.2	-
Swin-S [27]	50	8.7	83.0	-
GFNet-H-B	54	8.4	82.9	96.2

of our models. We set the gradient clipping norm to 1 for all of our models. During finetuning at the higher resolution, we use the hyper-parameters suggested by the implementation of [40] and train the model for 30 epochs. All of our models are trained on a single machine with 8 GPUs. More details can be found in Appendix B.

Comparisons with transformer-style architectures. The results are presented in Table 2. We compare our method with different transformer-style architectures for image classification including vision transformers (DeiT [40] and MLP-like models (ResMLP [39] and gMLP [26] that have similar complexity and number of parameters. We see that our method can clearly outperform recent MLP-like models like ResMLP [39] and gMLP [26], and show similar performance with DeiT. Specifically, GFNet-XS outperforms ResMLP-12 by 2.0% while having slightly fewer FLOPs. GFNet-S also achieves better top-1 accuracy compared to gMLP-S and DeiT-S. Our tiny model is significantly better compared to both DeiT-Ti (+2.4%) and gMLP-Ti (+2.6%) with the similar level of complexity.

Comparisons with hierarchical architectures. We compare different kinds of hierarchical models in Figure 3. ResNet [13] is the most widely used convolutional model while RegNet [35] is a family of carefully designed CNN models. We also compare with recent hierarchical vision transformers PVT [44] and Swin [27]. Benefiting from the log-linear complexity, GFNet-H models show significantly better performance than ResNet, RegNet and PVT and achieve similar performance with Swin while having a much simpler and more generic design.

Fine-tuning at higher resolution. One prominent problem of MLP-like models is that the feature resolution is not adjustable. On the contrary, the proposed global filter is more flexible. We demonstrate the advantage of GFNet by finetuning the model trained at 224 × 224 resolution to higher resolution following the practice in vision transformers [40]. As shown in Table 2, our model can easily adapt to higher resolution with only 30 epoch finetuning and achieve better performance.

4.2 Downstream tasks

Transfer learning. To test the generality of our architecture and the learned representation, we evaluate GFNet on a set of commonly used transfer learning benchmark datasets including CIFAR-10 [21], CIFAR-100 [21], Stanford Cars [20] and Flowers-102 [31]. We follow the setting of previous works [37, 9, 40, 39], where the model is initialized by the ImageNet pre-trained weights and finetuned on the new datasets. We evaluate the transfer learning performance of our basic model and best model. The results are presented in Table 4. The proposed models generally work well on

Table 4: **Results on transfer learning datasets**. We report the top-1 accuracy on the four datasets as well as the number of parameters and FLOPs.

Model	FLOPs	Params	CIFAR-10	CIFAR-100	Flowers-102	Cars-196
ResNet50 [13]	4.1G	26M	-	-	96.2	90.0
EfficientNet-B7 [37]	37G	66M	98.9	91.7	98.8	94.7
ViT-B/16 [9]	55.4G	86M	98.1	87.1	89.5	-
ViT-L/16 [9]	190.7G	307M	97.9	86.4	89.7	-
Deit-B/16 [40]	17.5G	86M	99.1	90.8	98.4	92.1
ResMLP-12 [39]	3.0G	15M	98.1	87.0	97.4	84.6
ResMLP-24 [39]	6.0G	30M	98.7	89.5	97.9	89.5
GFNet-XS	2.8G	16M	98.6	89.1	98.1	92.8
GFNet-H-B	8.4G	54M	99.0	90.3	98.8	93.2

Table 5: **Semantic segmentation results on ADE20K.** We report the mIoU on the validation set. All models are equipped with Semantic FPN [19] and trained for 80K iterations following [44]. The FLOPs are tested with 1024 × 1024 input. We compare the models that have similar computational costs and divide the models into three groups: 1) tiny models using ResNet-18, PVT-Ti and GFNet-H-Ti; 2) small models using ResNet-50, PVT-S, Swin-Ti and GFNet-H-S and 3) base models using ResNet-101, PVT-M, Swin-S and GFNet-H-B.

Backbone	*Tiny*			*Small*			*Base*		
	FLOPs	Params	mIoU	FLOPs	Params	mIoU	FLOPs	Params	mIoU
ResNet [13]	127	15.5	32.9	183	28.5	36.7	260	47.5	38.8
PVT [44]	123	17.0	35.7	161	28.2	39.8	219	48.0	41.6
Swin [27]	-	-	-	182	31.9	41.5	274	53.2	45.2
GFNet-H	126	26.6	41.0	179	47.5	42.5	261	74.7	44.8

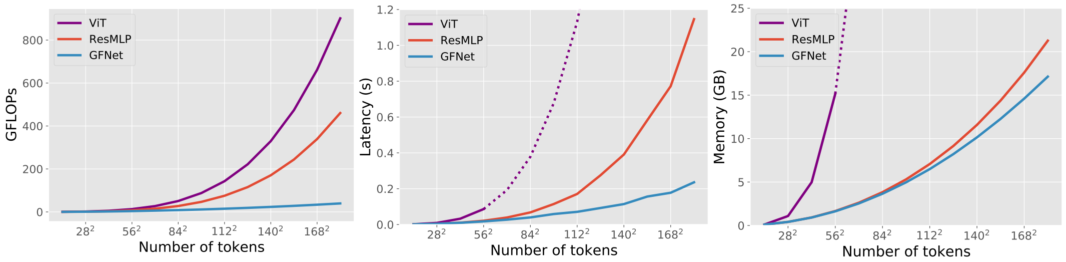
downstream datasets. GFNet models outperform ResMLP models by a large margin and achieve very competitive performance with state-of-the-art EfficientNet-B7. Our models also show competitive performance compared to state-of-the-art CNNs and vision transformers.

Semantic segmentation. We evaluate our GFNet on ADE20K [51], a challenging semantic segmentation dataset that is commonly used to test vision transformers. We use the Semantic FPN framework [19] and follow the experiment settings in PVT [44]. We train our model for 80K steps with a batch size of 16 on the training set and report the mIoU on the validation set following common practice. We compare the performance and the computational costs of the GFNet series and other commonly used baselines in Table 5. To produce hierarchical feature maps, we adopt the GFNet-H series in the semantic segmentation experiments. We observe that our GFNet works well on the dense prediction task and can achieve very competitive performance in different levels of complexity.

4.3 Analysis and visualization

Efficiency of GFNet. We demonstrate the efficiency of our GFNet in Figure 2, where the models are compared in theoretical FLOPs, actual latency and peak memory usage on GPU. We test a single building block of each model (including one token mixing layer and one FFN) with respect to the different numbers of tokens and set the feature dimension and batch size to 384 and 32 respectively. The self-attention model quickly runs out of memory when feature resolution exceeds 56^2, which is also the feature resolution of our hierarchical model. The advantage of the proposed architecture becomes larger as the resolution increases, which strongly shows the potential of our model in vision tasks requiring high-resolution feature maps.

Complexity/accuracy trade-offs. We show the computational complexity and accuracy trade-offs of various transformer-style architectures in Figure 3. It is clear that GFNet achieves the best trade-off among all kinds of models.



(a) (b) (c)

Figure 2: Comparisons among GFNet, ViT [9] and ResMLP [39] in **(a)** FLOPs **(b)** latency and **(c)** GPU memory with respect to the number of tokens (feature resolution). The dotted lines indicate the estimated values when the GPU memory has run out. The latency and GPU memory is measured using a single NVIDIA RTX 3090 GPU with batch size 32 and feature dimension 384.

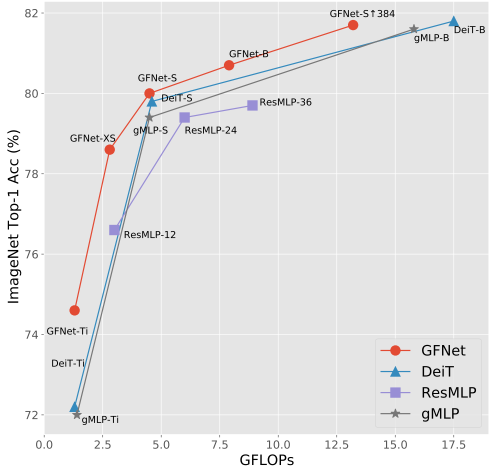


Figure 3: ImageNet acc. *vs* model complexity.

Table 6: Comparisons among the GFNet and other variants based on the transformer-like architecture on ImageNet. We show that GFNet outperforms the ResMLP [39], FNet [24] and models with local depthwise convolutions. We also report the number of parameters and theoretical complexity in FLOPs.

Model	Acc (%)	Param (M)	FLOPs (G)
DeiT-S [40]	79.8	22	4.6
Local Conv (3×3)	77.7	15	2.8
Local Conv (5×5)	78.1	15	2.9
Local Conv (7×7)	78.2	15	2.9
ResMLP [39]	76.6	15	3.0
FNet [24]	71.2	15	2.9
GFNet-XS	78.6	16	2.8

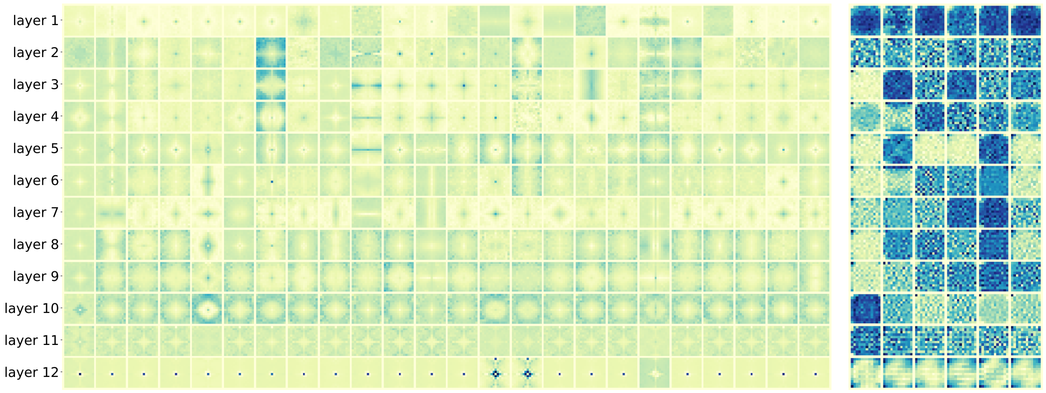
Ablation study on the global filter. To more clearly show the effectiveness of the proposed global filters, we compare GFNet-XS with several baseline models that are equipped with different token mixing operations. The results are presented in Table 6. All models have a similar building block (token mixing layer + FFN) and the same feature dimension of $D = 384$. We also implement the recent FNet [24] for comparison, where a 1D FFT on feature dimension and a 2D FFT on spatial dimensions are used to mix tokens. As shown in Table 6, our method outperforms all baseline methods except DeiT-S that has 64% higher FLOPs.

Robustness & generalization ability. Inspired by the [30], we further conduct experiments to evaluate the robustness and the generalization ability of the GFNet. For robustness, we consider ImageNet-A, ImageNet-C, FGSM and PGD. ImageNet-A [15] (IN-A) is a challenging dataset that contains natural adversarial examples. ImageNet-C [14] (IN-C) is used to validate the robustness of the model under various types of corruption. We use the mean corruption error (mCE, lower is better) on ImageNet-C as the evaluation metric. FGSM [11] and PGD [29] are two widely used algorithms that are targeted to evaluate the adversarial robustness of the model by single-step attack and multi-step attack, respectively. For generalization ability, we adopt two variants of ImageNet validation set: ImageNet-V2 [36] (IN-V2) and ImageNet-Real [2] (IN-Real). ImageNet-V2 is a re-collected version of ImageNet validation set following the same data collection procedure of ImageNet, while ImageNet-Real contains the same images as ImageNet validation set but has reassessed labels. We compare GFNet-S with various baselines in Table 7 including CNNs, Transformers and MLP-like architectures and find the GFNet enjoys both favorable robustness and generalization ability.

Visualization. The core operation in GFNet is the element-wise multiplication between frequency-domain features and the global filter. Therefore, it is easy to visualize and interpret. We visualize the frequency domain filters as well as their corresponding spatial domain filters in Figure 4. The learned

Table 7: **Evaluation of robustness and generalization ability**. We measure the robustness from different aspects, including the adversarial robustness by adopting adversarial attack algorithms including FGSM and PGD and the performance on corrupted/out-of-distribution datasets including ImageNet-A [15] (top-1 accuracy) and ImageNet-C [14] (mCE, lower is better). The generalization ability is evaluated on ImageNet-V2 [36] and ImageNet-Real [2].

Model	FLOPs	Params	ImageNet		Generalization		Robustness			
	(G)	(M)	Top-1↑	Top-5↑	IN-V2↑	IN-Real↑	FGSM↑	PGD↑	IN-C↓	IN-A↑
ResNet-50 [13]	4.1	26	76.1	92.9	67.4	85.8	12.2	0.9	76.7	0.0
ResNeXt50-32x4d [46]	4.3	25	79.8	94.6	68.2	85.2	34.7	13.5	64.7	10.7
DeiT-S [40]	4.6	22	79.8	95.0	68.4	85.6	40.7	16.7	54.6	18.9
ResMLP-12 [39]	3.0	15	76.6	93.2	64.4	83.3	23.9	8.5	66.0	7.1
GFNet-S	4.5	25	80.1	94.9	68.5	85.8	42.6	21.0	53.8	14.3



(a) Frequency domain

(b) Spatial domain

Figure 4: Visualization of the learned *global filters* in GFNet-12. We visualize the original frequency domain global filters in (a) and show the corresponding spatial domain filters for the first 6 columns in (b). There are more clear patterns in the frequency domain than the spatial domain.

global filters have more clear patterns in the frequency domain, where different layers have different characteristics. Interestingly, the filters in the last layer particularly focus on the low-frequency component. The corresponding filters in the spatial domain are less interpretable for humans.

5 Conclusion

We have presented the Global Filter Network (*GFNet*), which is a conceptually simple yet computationally efficient architecture for image classification. Our model replaces the self-attention sub-layer in vision transformer with 2D FFT/IFFT and a set of learnable *global filters* in the frequency domain. Benefiting from the token mixing operation with log-linear complexity, our architecture is highly efficient. Our experimental results demonstrated that GFNet can be a very competitive alternative to vision transformers, MLP-like models and CNNs in accuracy/complexity trade-offs.

A Discrete Fourier transform

In this section, we will elaborate on the derivation and the properties of the discrete Fourier transform.

A.1 From Fourier transform to discrete Fourier transform

Discrete Fourier transform (DFT) can be derived in many ways. Here we will introduce the formulation of DFT from the standard Fourier transform (FT), which is originally designed for continuous signals. The FT converts a continuous signal from the time domain to the frequency domain and can be viewed as an extension of the Fourier series. Specifically, the Fourier transform of the signal $x(t)$

is given by

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt := \mathcal{F}[x(t)]. \tag{A.1}$$

The inverse Fourier transform (IFT) has a similar form to the Fourier transform:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\omega. \tag{A.2}$$

From the formulas of the FT and the IFT we can have a glimpse of the duality property of the FT between the time domain and the frequency domain. The duality indicates that the properties in the time domain always have their counterparts in the frequency domain. There are a variety of properties of Fourier transform. To name a few basic ones, the FT of a unit impulse function (a.k.a. Dirac delta function) is

$$\mathcal{F}[\delta(t)] = \int_{-\infty}^{\infty} \delta(t)e^{-j\omega t}dt = \int_{0-}^{0+} \delta(t)dt = 1, \tag{A.3}$$

and the time shifting property:

$$\mathcal{F}[\delta(t-t_0)] = \int_{-\infty}^{\infty} x(t-t_0)e^{-j\omega t}dt = e^{-j\omega t_0}\int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt = e^{-j\omega t_0}X(j\omega). \tag{A.4}$$

However, we rarely deal with continuous signal in the real application. A general practice is to perform *sampling* to the continuous signal to obtain a sequence of discrete signal. The sampling can be achieved using a sequence of unit impulse functions,

$$x_s(t) = x(t)\sum_{n=-\infty}^{\infty} \delta(t-nT_s) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t-nT_s), \tag{A.5}$$

where T_s is the sampling interval. Taking the FT of the sampled signal $x_s(t)$ and applying Equation (A.3) and Equation (A.4), we have

$$X_s(j\omega) = \sum_{n=-\infty}^{\infty} x(nT_s)e^{-j\omega nT_s}. \tag{A.6}$$

In the above equation, it is direct to show that $X_s(j\omega)$ is a *periodic* function with the fundamental period as $2\pi/T_s$. Actually, there is always a correspondence between the discrete signal in one domain and the periodic signal in the other domain. Usually, we prefer a normalized frequency $\omega \leftarrow \omega T_s$ such that the period of $X_s(j\omega)$ is exact 2π. We can further denote $x[n] = x(nT_s)$ as the sequence of discrete signal and derive the discrete-time Fourier transform (DTFT):

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}. \tag{A.7}$$

If the discrete signal $x[n]$ has finite length N (which is common in digital signal processing), the DTFT becomes

$$X(e^{j\omega}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega n}, \tag{A.8}$$

where we assume the non-zero terms lie in $[0, N-1]$ without loss of generality. Note that the DTFT is a continuous function of ω and we can obtain a sequence of $X[k]$ by sampling $X(e^{j\omega})$ at frequencies $\omega_k = 2\pi k/N$:

$$X[k] = X(e^{j\omega})|_{\omega=2\pi k/N} = \sum_{n=0}^{N-1} x[n]e^{-j(2\pi/N)kn}, \tag{A.9}$$

which is exactly the formulation of DFT. The extension from 1D DFT to 2D DFT is straightforward. In fact, The 2D DFT can be viewed as performing 1D DFT on the two dimensions alternatively, *i.e.*, the 2D DFT of $x[m,n]$ is given by:

$$X[u,v] = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x[m,n]e^{-j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}. \tag{A.10}$$

A.2 Some properties of DFT

DFT of real signals. Given a real signal $x[n]$, the DFT of it is *conjugate symmetric*, which can be proved as follows:

$$X[N-k] = \sum_{n=0}^{N-1} x[n]e^{-j(2\pi/N)(N-k)n} = \sum_{n=0}^{N-1} x[n]e^{j(2\pi/N)kn} = X^*[k]. \tag{A.11}$$

For 2D signals, we have a similar result:

$$\begin{aligned} X[M-u, N-v] &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x[m,n]e^{-j2\pi\left(\frac{(M-u)m}{M}+\frac{(N-v)n}{N}\right)} \\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x[m,n]e^{j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)} = X^*[u,v]. \end{aligned} \tag{A.12}$$

In our GFNet, we leverage this property to reduce the number of learnable parameters and redundant computation.

The convolution theorem. One of the most important property of Fourier transform is the convolution theorem. Specifically, for the DFT, the convolutional theorem states that the *multiplication* in the frequency domain is equivalent to the *circular convolution* in the time domain. The circular convolution of a signal $x[n]$ and a filter $h[n]$ can be defined as

$$y[n] = \sum_{m=0}^{N-1} h[m]x[((n-m))_N], \tag{A.13}$$

where we use $((n))_N$ to denote n modulo N. Consider the DFT of $y[n]$, we have

$$\begin{aligned} Y[k] &= \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} h[m]x[((n-m))_N]e^{-j(2\pi/N)kn} \\ &= \sum_{m=0}^{N-1} h[m]e^{-j(2\pi/N)km} \sum_{n=0}^{N-1} x[((n-m))_N]e^{-j(2\pi/N)k(n-m)} \\ &= H[k]\left(\sum_{n=m}^{N-1} x[n-m]e^{-j(2\pi/N)k(n-m)} + \sum_{n=0}^{m-1} x[n-m+N]e^{-j(2\pi/N)k(n-m)}\right) \\ &= H[k]\left(\sum_{n=0}^{N-m-1} x[n]e^{-j(2\pi/N)kn} + \sum_{n=N-m}^{N-1} x[n]e^{-j(2\pi/N)kn}\right) \\ &= H[k]\sum_{n=0}^{N-1} x[n]e^{-j(2\pi/N)kn} = H[k]X[k], \end{aligned} \tag{A.14}$$

where the right hand is exactly the multiplication of the signal and the filter in the frequency domain. The convolution theorem in 2D scenario can be derived similarly. Therefore, our global filter layer is equivalent to a depth-wise circular convolution, where the filter has the same size as the feature map.

B Implementation Details

The detailed architectures. To better compare with previous methods, we use the identical overall architecture to DeiT Samll [40] and ResMLP-12 [39] for GFNet-XS, where only the self-attention/MLP sub-layers, the final classifier and the residual connection are modified (using a single residual connection in each block will lead to 0.2% top-1 accuracy improvement on ImageNet for GFNet-XS). We set the number of layers and embedding dimension to $\{12, 19, 19\}$ and $\{256, 384, 512\}$ for GFNet-{Ti, S, B}, respectively. The architectures of our hierarchical models are shown in Table 8. We use the similar strategy as ResNet [13] to increase network depth where we fix

Table 8: **The detailed architectures of hierarchical GFNet variants.** We adopt hierarchical architectures where the we use patch embedding layer to perform downsampling. "$\downarrow n$" indicates the stride of the downsampling is n. "GFBlock(D)" represents one building block of GFNet with embedding dimension D. We set the MLP expansion ratio to 4 for all the feedforward networks.

	Output Size	GFNet-H-Ti	GFNet-H-S	GFNet-H-B
Stage1	$\frac{H}{4} \times \frac{W}{4}$	Patch Embedding$\downarrow$4 GFBlock(64) $\times$ 3	Patch Embedding$\downarrow$4 GFBlock(96) $\times$ 3	Patch Embedding$\downarrow$4 GFBlock(96) $\times$ 3
Stage2	$\frac{H}{8} \times \frac{W}{8}$	Patch Embedding$\downarrow$2 GFBlock(128) $\times$ 3	Patch Embedding$\downarrow$2 GFBlock(192) $\times$ 3	Patch Embedding$\downarrow$2 GFBlock(192) $\times$ 3
Stage3	$\frac{H}{16} \times \frac{W}{16}$	Patch Embedding$\downarrow$2 GFBlock(256) $\times$ 10	Patch Embedding$\downarrow$2 GFBlock(384) $\times$ 10	Patch Embedding$\downarrow$2 GFBlock(384) $\times$ 27
Stage4	$\frac{H}{32} \times \frac{W}{32}$	Patch Embedding$\downarrow$2 GFBlock(512) $\times$ 3	Patch Embedding$\downarrow$2 GFBlock(768) $\times$ 3	Patch Embedding$\downarrow$2 GFBlock(768) $\times$ 3
Classifier		Global Average Pooling, Linear		

the number of blocks for the stage 1,2,4 to 3 and adjust the number of blocks in stage 3. For small and base hierarchical models, we adopt the LayerScale normalization [42] for more stable training. The high efficiency of our GFNet makes it possible to *directly* process a large feature map in the early stages (*e.g.*, $H/4 \times W/4$) without introducing any handcraft structures like Swin [27].

Details about ImageNet experiments. We train our models for 300 epochs using the AdamW optimizer [28]. We set the initial learning rate as $\frac{\text{batch size}}{1024} \times 0.001$ and decay the learning rate to $1e^{-5}$ using the cosine schedule. We use a linear warm-up learning rate in the first 5 epochs and apply gradient clipping to stabilize the training process. We set the stochastic depth coefficient [17] to 0, 0, 0.15 and 0.25 for GFNet-Ti, GFNet-XS, GFNet-S and GFNet-B. For hierarchical models, we use the stochastic depth coefficient of 0.1, 0.2, and 0.4 for GFNet-H-Ti, GFNet-H-S, and GFNet-H-B. During finetuning at the higher resolution, we use the hyper-parameters suggested by the implementation of [40] and train the model for 30 epochs with a learning rate of $5e^{-6}$ and set the weight decay to $1e^{-6}$. We set the stochastic depth coefficient to 0.1 for GFNet-S and GFNet-B during finetuning.

Details about transfer learning experiments. We evaluate generality of learned representation of GFNet on a set of commonly used transfer learning benchmark datasets including CIFAR-10 [21], CIFAR-100 [21], Stanford Cars [20] and Flowers-102 [31]. We follow the setting of previous works [37, 9, 40, 39], where the model is initialized by the ImageNet pre-trained weights and finetuned on the new datasets. During finetuning, we use the AdamW optimizer and set the weight decay to $1e^{-4}$. We use batch size 512 and a smaller initial learning rate of 0.0001 with cosine decay. Linear learning rate warm-up in the first 5 epochs and gradient clipping with a max norm of 1 are also applied to stabilize the training. We keep most of the regularization methods unchanged except for removing stochastic depth following [40]. For relatively larger datasets including CIFAR-10 and CIFAR-100, we train the model for 200 epochs. For other datasets, the model is trained for 1000 epoch. Our models are trained and evaluated on commonly used splits following [37]. The detailed splits are provided in Table 9.

Details about semantic segmentation experiments. We conduct the semantic segmentation experiments using MMSegmentation toolbox [5]. We follow the experiment settings in PVT [44]. We train our model for 80K steps with a batch size of 16 where we use 8 GPUs with 2 images on each GPU. We set the stochastic depth coefficient to 0.1, 0.2 and 0.2 for GFNet-H-Ti, GFNet-H-S and GFNet-H-B respectively.

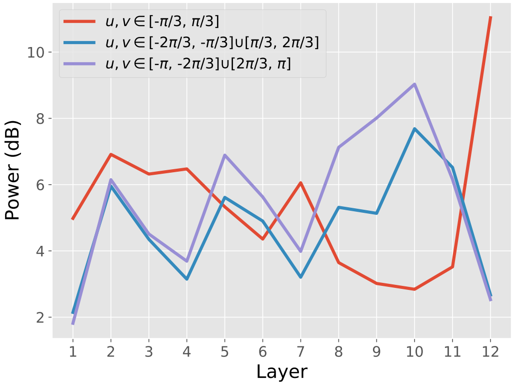


Figure 5: The average power on different frequency ranges of each layer. We can observe that the global filters of different layers focus on different frequencies.

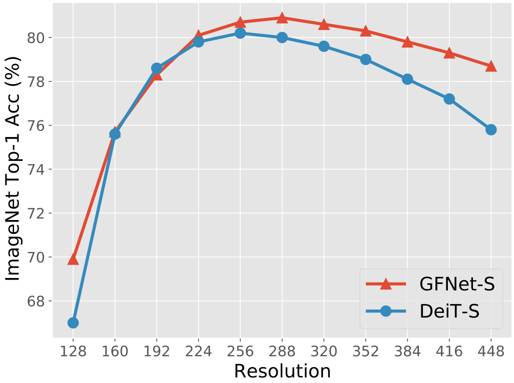


Figure 6: ImageNet accuracy of GFNet and DeiT [40] when directly evaluated on different resolutions without fine-tuning. The GFNet can better adapt to various resolutions.

Table 9: **Transfer learning datasets.** We provide the training set size, test set size and the number of categories as references.

Dataset	Train Size	Test size	#Categories
CIFAR-10 [21]	50,000	10,000	10
CIFAR-100 [21]	50,000	10,000	100
Stanford Cars [20]	8,144	8,041	196
Flowers-102 [31]	2,040	6,149	102

C More Analysis

Power distribution. We plot the power of the global filters on different frequency ranges of each layer in Figure 5, where we can have a clearer picture of how the global filters of different layers capture the information of different frequencies.

Directly adapting to other resolutions. As is discussed in Section 3.2, one of the advantages of GFNet is the ability to deal with arbitrary resolutions. To verify this, we *directly* evaluate GFNet-S trained with 224 × 224 images on different resolutions (from 128 to 448). We plot the accuracy of GFNet-S and DeiT-S in Figure 6 and find our GFNet can adapt to different resolutions with less performance drop than DeiT-S.

Filter visualization for hierarchical models. We also provide the visualization of the frequency-domain global filters for the hierarchical model GFNet-H-B in Figure 7.

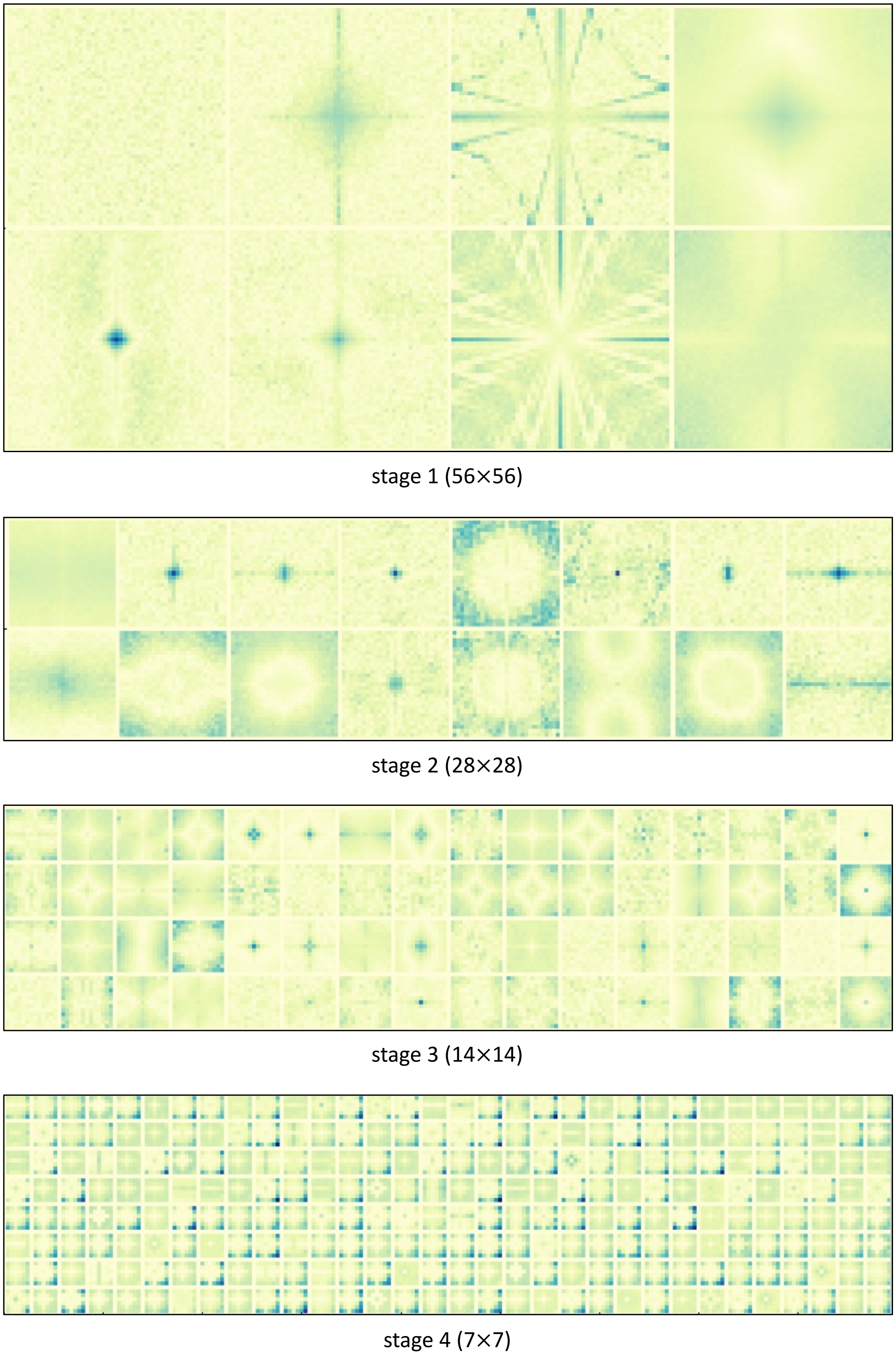
References

[1] Gregory A Baxes. *Digital image processing: principles and applications*. John Wiley & Sons, Inc., 1994. 3

[2] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020. 9, 10

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229. Springer, 2020. 3

[4] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, 2021. 1, 3



stage 1 (56×56)

stage 2 (28×28)

stage 3 (14×14)

stage 4 (7×7)

Figure 7: **Visualization of the learned global filters in GFNet-H-B.** We visualize the frequency domain global filters from different stages with different sizes.

[5] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 13

[6] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965. 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 6

[8] Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai Qian, Yu Bai, Geng Yuan, et al. Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices. In *MICRO*, pages 395–408, 2017. 3

[9] 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. *arXiv preprint arXiv:2010.11929*, 2020. 1, 3, 4, 5, 7, 8, 9, 13

[10] Chengyue Gong, Dilin Wang, Meng Li, Vikas Chandra, and Qiang Liu. Improve vision transformers training by suppressing over-smoothing. *arXiv preprint arXiv:2104.12753*, 2021. 3

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 9

[12] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 3

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3, 5, 6, 7, 8, 10, 12

[14] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 9, 10

[15] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, pages 15262–15271, 2021. 9, 10

[16] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *CVPR*, pages 8129–8138, 2020. 6

[17] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661, 2016. 13

[18] Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Xiaojie Jin, Anran Wang, and Jiashi Feng. Token labeling: Training a 85.5% top-1 accuracy vision transformer with 56m parameters on imagenet. *arXiv preprint arXiv:2104.10858*, 2021. 3, 6

[19] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, pages 6399–6408, 2019. 8

[20] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCVW*, pages 554–561, 2013. 6, 7, 13, 14

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6, 7, 13, 14

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25:1097–1105, 2012. 5

[23] Jae-Han Lee, Minhyeok Heo, Kyung-Rae Kim, and Chang-Su Kim. Single-image depth estimation based on fourier domain analysis. In *CVPR*, pages 330–339, 2018. 3

[24] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021. 5, 9

[25] Shaohua Li, Kaiping Xue, Bin Zhu, Chenkai Ding, Xindi Gao, David Wei, and Tao Wan. Falcon: A fourier transform based approach for fast and secure convolutional neural network predictions. In *CVPR*, pages 8705–8714, 2020. 3

[26] Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021. 2, 3, 6, 7

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 1, 3, 6, 7, 8, 13

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 13

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 9

[30] Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Shaokai Ye, Yuan He, and Hui Xue. Rethinking the design principles of robust vision transformer. *arXiv preprint arXiv:2105.07926*, 2021. 9

[31] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, pages 722–729, 2008. 6, 7, 13, 14

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 5

[33] Ioannis Pitas. *Digital image processing algorithms and applications*. John Wiley & Sons, 2000. 3, 5

[34] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 6

[35] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020. 7

[36] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, pages 5389–5400. PMLR, 2019. 9, 10

[37] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114. PMLR, 2019. 7, 8, 13

[38] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021. 1, 3, 4, 5

[39] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13

[40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14

[41] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 3, 5

[42] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 13

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 1

[44] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, 2021. 6, 7, 8, 13

[45] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 3

[46] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 10

[47] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *CVPR*, pages 4085–4095, 2020. 3

[48] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 3

[49] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *arXiv preprint arXiv:2012.15840*, 2020. 1, 3

[50] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, volume 34, pages 13001–13008, 2020. 6

[51] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, pages 633–641, 2017. 6, 8