FGFP: A Fractional Gaussian Filter and Pruning for Deep Neural Networks Compression

Kuan-Ting Tu [*1] **Po-Hsien Yu** [*2] **Yu-Syuan Tseng** [2] **Shao-Yi Chien** [2]

Abstract

Network compression techniques have become increasingly important in recent years because the loads of Deep Neural Networks (DNNs) are heavy for edge devices in real-world applications. While many methods compress neural network parameters, deploying these models on edge devices remains challenging. To address this, we propose the fractional Gaussian filter and pruning (FGFP) framework, which integrates fractional-order differential calculus and the Gaussian function to construct fractional Gaussian filters (FGFs). To reduce the computational complexity of fractional-order differential operations, we introduce Grünwald-Letnikov fractional derivatives to approximate the fractional-order differential equation. The number of parameters for each kernel in FGF is minimized to only seven. Beyond the architecture of Fractional Gaussian Filters, our FGFP framework also incorporates Adaptive Unstructured Pruning (AUP) to achieve higher compression ratios. Experiments on various architectures and benchmarks show that our FGFP framework outperforms recent methods in accuracy and compression. On CIFAR-10, ResNet-20 achieves only a 1.52% drop in accuracy while reducing the model size by 85.2%. On ImageNet2012, ResNet-50 achieves only a 1.63% drop in accuracy while reducing the model size by 69.1%.

[*]Equal contribution [1]Graduate School of Advanced Technology, National Taiwan University, Taipei, Taiwan [2]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan. Correspondence to: Kuan-Ting Tu <kttu@media.ee.ntu.edu.tw>, Po-Hsien Yu <michaelyu@media.ee.ntu.edu.tw>, Yu-Syuan Tseng <ystseng@media.ee.ntu.edu.tw>, Shao-Yi Chien <sychien@ntu.edu.tw>.

Accepted to *Workshop on Tiny Titans: The next wave of On-Device Learning for Foundational Models at 42nd International Conference on Machine Learning (TTODLer-FM@ICML 2025)*, Vancouver, Canada. Copyright 2025 by the author(s).

1. Introduction

Over the past decade, computer vision has experienced rapid advancements, primarily driven by the emergence of deep neural networks (DNNs). Modern deep learning models now achieve state-of-the-art performance across various tasks, including image classification, object detection, and pose estimation. To improve accuracy, many tasks leverage an increased number of trainable parameters by expanding the depth or width of models. Although larger models can deliver higher precision, they also have significant drawbacks, such as higher memory storage and access costs. This poses substantial challenges for deploying models on edge devices in real-world applications, such as smartphones, laptops, or resource-constrained embedded systems.

Many studies have explored efficient techniques to deploy large models on edge devices. Unstructured pruning (Zhang et al., 2018; Han et al., 2015; Frankle & Carbin, 2019) preserves accuracy, but remains high complexity of computing. Structured pruning (Tang et al., 2020; Yuan et al., 2021; Zhou et al., 2021; Wang et al., 2024; Yuan et al., 2024; Pham et al., 2024b; Yang et al., 2024) reduces computation by removing kernels or channels, though it often sacrifices important features and leads to significant accuracy loss. Low-rank compression (Denton et al., 2014; Phan et al., 2020; Chu & Lee, 2021; Li et al., 2022; Guo et al., 2024; Liu et al., 2024; Sui et al., 2024) provides another solution by decomposing the kernels into lower-rank forms, reducing both parameters and computation. Recent hybrid methods (Li et al., 2020; 2021; Ruan et al., 2024; Pham et al., 2024a) combine pruning and low-rank techniques to achieve higher compression with low accuracy degradation.

The motivation for FGFP comes from previous work (Zamora et al., 2021), which successfully integrated the traditional computer vision filters, such as the Gaussian, Laplacian (Gonzalez & Woods, 2008), and Sobel (Kanopoulos et al., 1988) filter with CNNs through fractional-order differentiation. FGFP shares the same fractional Gaussian filter (FGF) parameters across all channels in each kernel to reduce parameters. However, applying the same filter can hurt accuracy, so we apply the channel-attention mechanism to minimize the accuracy degradation and design two types of FGFs: the channel-attention fractional

Gaussian filter (CA-FGF) and three-dimensional fractional Gaussian filter (3D-FGF). Furthermore, we leverage the Grünwald–Letnikov fractional derivative (Jalalinejad et al., 2018) to approximate fractional differential equations to simplify the computational complexity of fractional-order differential operations. To maximize the compression ratio, we also explore incorporating the adaptive unstructured pruning (AUP) strategy into our FGFP.

In summary, the main contributions of this work are as follows:

- We propose the fractional Gaussian filter and pruning (FGFP) framework, which combines the fractional Gaussian filter (FGF) and adaptive unstructured pruning (AUP) to reduce the number of parameters significantly.
- We use the channel-attention mechanism to design two forms of the fractional Gaussian filter (FGF): the channel-attention fractional Gaussian filter (CA-FGF) and the three-dimensional fractional Gaussian filter (3D-FGF). Both the FGFs perform excellently in removing the redundant parameters with low degradation of model accuracy. Moreover, we introduce the Grünwald–Letnikov fractional derivative into the FGF and simplify the fractional derivative to the trinomial polynomial.
- We conducted comprehensive experiments with state-of-the-art methods on two benchmarks, CIFAR-10 and ImageNet2012. The experimental results illustrate our FGFP's effectiveness, and the FGFP outperforms several recent works. For instance, when evaluating the FGFP with ResNet-20 on CIFAR-10, our method achieves only a 1.58% drop in accuracy while reducing the model size by 85.1%. Also, on ImageNet2012, our method achieves only a 1.63% drop in accuracy while reducing the model size by 69.1% with ResNet-50.

2. Proposed Methods

In this section, we will introduce the detailed Fractional Gaussian Filter and Pruning (FGFP), and the overall methods are shown as Fig. 1.

2.1. Grünwald-Letnikov Fractional Derivatives

Ordinarily, the definition of nth-order derivative of function f is (Jalalinejad et al., 2018):

$$f^{(n)}(x) = \frac{d^n f}{dx^n} = \lim_{h\to 0} \frac{1}{h^n} \sum_{r=0}^{n} (-1)^r \binom{n}{r} f(x - rh). \tag{1}$$

According to the Grünwald-Letnikov fractional derivatives for univariate function f which mentioned in (Scherer et al., 2011; Jalalinejad et al., 2018), we can redefined eq. (1) as belows:

$$D^{\alpha}_{G-L} f(x) = \lim_{h\to 0} \frac{1}{h^\alpha} \sum_{r=0}^{\left[\frac{x-a}{h}\right]} (-1)^r \binom{\alpha}{r} f(x - rh), \tag{2}$$

where

$$\binom{\alpha}{r} = \frac{\Gamma(\alpha+1)}{\Gamma(r+1)\Gamma(\alpha-r+1)}$$

and Γ is the gamma function. We can also approximate and expand eq. (2) as follows:

$$\begin{aligned} D^{\alpha}_{G-L} f(x) \approx{}& f(x) + (-\alpha) f(x-1) + \frac{(-\alpha)(-\alpha+1)}{2} f(x-2) \\ &+ \cdots + \frac{\Gamma(-\alpha+1) f(x-n)}{n!\Gamma(-\alpha+n+1)}. \end{aligned} \tag{3}$$

$$\begin{aligned} D^{\alpha}_{G-L} f(x) \approx{}& f(x) - \alpha f(x-1) + \frac{\alpha(\alpha-1)}{2} f(x-2) \\ &+ \epsilon^{\alpha}_{x} f(x), \end{aligned} \tag{4}$$

where $\epsilon^{\alpha}_{x} f(x)$ is the corresponding approximate error, and the error can be ignored in eq. (4). Thus, the Grüwald-Letnikov derivative can be simplified as follows:

$$D^{\alpha}_{G-L} f(x) \approx f(x) - \alpha f(x-1) + \frac{\alpha(\alpha-1)}{2} f(x-2), \tag{5}$$

2.2. Fractional Gaussian Filter (FGF)

The Gaussian filter is widely used in image processing and signal processing, primarily to reduce high-frequency noise and detail in images or signals while preserving crucial structural information. The Gaussian filter applies a convolution operation using a kernel derived from the Gaussian function, and the 2D Gaussian function is defined as:

$$G(x, y) = e^{-\frac{(x-x_0)^2 + (y-y_0)^2}{\sigma^2}}, \tag{6}$$

where σ represents the standard deviation of the Gaussian distribution and x_0 and y_0 are the positions of the center of the peak. Notably, the 2D Gaussian function can be decomposed into the product of two 1D Gaussian filters, and we can redefined eq. (6) as below:

$$\begin{aligned} G(x, y) &= e^{-\frac{(x-x_0)^2 + (y-y_0)^2}{\sigma^2}} \\ &= e^{-\frac{(x-x_0)^2}{\sigma^2}} \cdot e^{-\frac{(y-y_0)^2}{\sigma^2}} = G(x)G(y), \end{aligned} \tag{7}$$

where $G(x)$ and $G(y)$ are the 1D Gaussian function in x- and y-direction. Furthermore, the first and second derivatives of the Gaussian filter are the Sobel filter and the Laplacian filter, commonly used to detect edges in images by removing the low-frequency features. Since different order derivatives of the Gaussian filter can extract features from various frequencies, general frequency-adjustable filters can be obtained by applying the fractional derivatives to the Gaussian filter (Zamora et al., 2021). Hence, the fractional Gaussian filters (FGF), can be defined as follows:

$$\begin{aligned} F_{fg} &= D^a_x D^b_y G(x, y) = D^a_x D^b_y G(x)G(y) \\ &= D^a_x G(x) \times D^b_y G(y), \end{aligned} \tag{8}$$

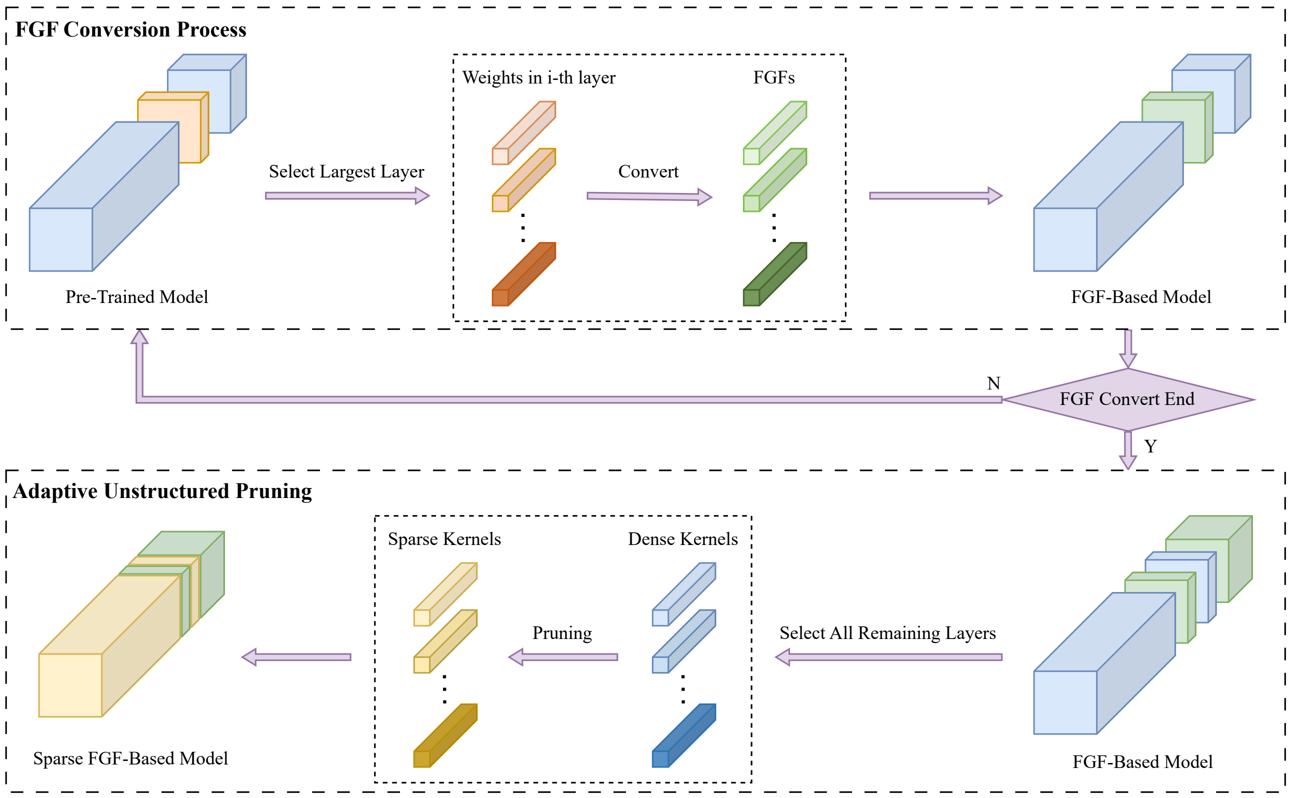

Figure 1. Overview of the FGFS methodology. The FGF conversion process begins with selecting the largest layer in the pre-trained model and then converting weights into FGF representations to generate the FGF-based model. This process repeats until all selected layers are converted. The AUP process, where the remaining layers of the FGF-based model undergo adaptive unstructured pruning, transforms dense kernels into sparse kernels. The final result is a sparse FGF-based model optimized for performance and efficiency.

where F_{fg} denotes the fractional Gaussian filters, D is the fractional derivatives, a and b are the order of the fractional derivatives. Moreover, to simplify the computation of the fractional derivatives, we introduce the Grünwald-Letnikov Fractional Derivatives. According to eq. (5), we can simplify $D^a_x G(x)$ and $D^b_y G(y)$ as below:

$$D^a_x G(x) = G(x) - aG(x-1) + \frac{a(a-1)}{2} G(x-2); \quad (9)$$

$$D^b_y G(y) = G(y) - bG(y-1) + \frac{b(b-1)}{2} G(y-2). \quad (10)$$

In the original fractional Gaussian filter, there are five parameters for each channel, a, b, x_0, y_0, and σ. Also, the range of them would be limited as follows: $a \in [0, 2]$, $b \in [0, 2]$, $x_0 \in (-\infty, \infty)$, $y_0 \in (-\infty, \infty)$, and $\sigma \in (0, \infty)$. The fractional orders a and b are constrained to the range $[0, 2]$ to ensure that the learned functions remain within the domain of traditional filters, facilitating the identification of suitable filter forms. The remaining parameters follow the common settings of the classical Gaussian function. However, generating the FGF for each channel results in poor model compression efficiency. Specifically, in the case of a 3×3 filter, the number of trainable parameters is reduced from nine to five, achieving only a compression ratio of 44.4%. To address this problem, we propose two types of FGF: the channel-attention fractional Gaussian filter (CA-FGF) and the three-dimensional fractional Gaussian filter (3D-FGF).

As shown in Fig. 2, we share the five parameters for all channels to achieve the maximum compression ratio. However, sharing these parameters would cause the model accuracy degradation since all FGFs for each channel are the same. To address this, we would introduce the weights for each channel in our CA-FGF. The mechanism of the weighted channel can emphasize the essential features and suppress the insignificant ones to improve the model's accuracy. Furthermore, the mechanism of the weighted channel in the CA-FGF can reduce the parameters from $5 \times ch$ to $5 + ch$, where ch is the channel number of the FGF. In the 3D-FGF, we utilize the fractional derivatives of the Gaussian function to constrain the weights of channels to improve model generalizability. Therefore, the 3D-FGF can be defined as follows:

$$\begin{aligned} F_{3d} &= D^a_x D^b_y D^c_{ch} G(x, y, ch) = D^a_x D^b_y D^c_{ch} G(x) G(y) G(ch) \\ &= D^a_x G(x) \times D^b_y G(y) \times D^c_{ch} G(ch), \end{aligned} \quad (11)$$

where c is the order of the fractional derivatives in the channel direction. By using the fractional derivatives of the Gaussian function as the weights of channels, the 3D-FGF can achieve the maximum compression ratio. The trainable parameters can be reduced from $5 \times ch$ to seven, including the fractional order (a, b, c), the offset of the center(x_0, y_0, ch_0), and the standard deviation of the Gaussian distribution (σ). Same as the original FGF, the ranges of parameters are defined as follows: $a \in [0, 2]$, $b \in [0, 2]$, $c \in [0, 2]$, $x_0 \in (-\infty, \infty)$, $y_0 \in (-\infty, \infty)$, $ch_0 \in (-\infty, \infty)$, and $\sigma \in (0, \infty)$.

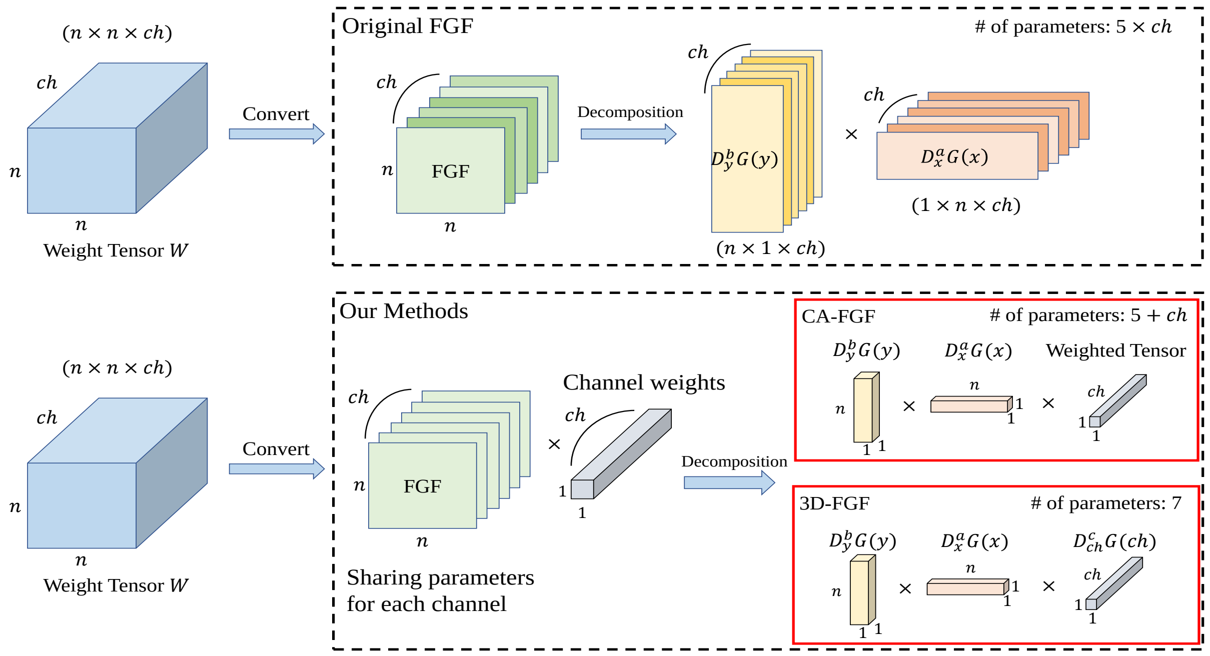

Figure 2. The details of FGF Transformation in the FGFS. The composition of the original FGF for a single kernel, where independent FGFs are applied to each channel, reduces the number of parameters from $n \times n \times ch$ to $5 \times ch$. On the other hand, the FGFS contains two FGF forms, the CA-FGF and the 3D-FGF. The FGFS shares the parameters of the FGF across all channels. The CA-FGF can reduce the parameter count to $5 + ch$ with the channel-attention mechanism. Additionally, to achieve further compression, the 3D-FGF constraints the channel weights and reduces the parameter count to just 7.

2.3. Adaptive Unstructured Pruning (AUP)

Adaptive Unstructured Pruning (AUP) eliminates the redundant parameters by removing those with absolute values smaller than the pruning threshold during each round. The pruning threshold is determined by the predefined percentage p_r of non-zero parameters to be removed in each round. However, removing the numerous parameters in the model causes a decrease in its performance. To prevent a significant drop in accuracy after pruning, the model is fine-tuned after each round of pruning to recover its accuracy. Suppose the fine-tuning model after a pruning round cannot reach the accuracy threshold θ_{acc}. In that case, this pruning round will be abandoned, and the model will reload the previous round's model for additional fine-tuning before continuing pruning. When the model can not reach the accuracy threshold θ_{acc} after several pruning rounds, we will reduce the predefined percentage p_r and continue attempting to prune. Once the sparsity ratio reaches the target, we fine-tune the model to restore its accuracy to a higher level.

2.4. Fractional Gaussian Filter and Pruning (FGFP)

The Fig. 1 demonstrates the fractional Gaussian filter and pruning (FGFP) framework. First, we convert the filters of the pre-trained model into the FGF. Previous studies (Zamora et al., 2021; Llanza et al., 2023) have demonstrated that applying fractional Gaussian filters (FGF) in deeper layers with larger input channels yields higher compression ratios with minimal accuracy degradation. Hence, we apply FGF to the deeper layers with larger input channels. After replacing the large layer with the FGF, we apply adaptive unstructured pruning (AUP) methods to the remaining layers. In this step, we define a layer-wise pruning ratio p_r, typically between 3% and 6%, to control the proportion of parameters removed in each pruning round. Moreover, we independently perform the adaptive unstructured pruning for each layer to prevent concentrating the pruned parameters in a single layer due to the collectively smaller parameter values. After all layers are processed with the FGF transform and AUP, we can obtain the compressed model with the FGFP framework.

3. Experimental Results

3.1. Experimental Settings

Datasets. To evaluate the performance of our proposed method, we employed two widely used benchmarks in image classification: CIFAR-10 and ImageNet2012. CIFAR-10 is a classic small-scale image dataset consisting of 10 classes, with 50K training images and 10K test images, each of size 32×32. We further partitioned the CIFAR-10 training set during training into a training subset of 45K images and a validation subset of 5K images. ImageNet2012, on the other hand, is a large-scale dataset for image classification, comprising approximately 1.28M training images, 50K validation images, and 100K test images.

Networks. On the CIFAR-10 dataset, we evaluated the FGFP with CA-FGF and 3D-FGF using ResNet-20, ResNet-32 (He et al., 2016), and WRN-28-10 (Zagoruyko & Komodakis, 2016). In addition, since ResNet-18 and ResNet-50 (He et al., 2016) are larger networks, we apply the FGFP with the 3D-FGF in ResNet-18 and ResNet-50 for a higher compression ratio on the ImageNet2012

Table 1. Results for ResNet-20, ResNet-32 and WRN-28-10 on CIFAR-10 dataset. "$*$" denotes validation accuracy since the corresponding work does not provide test accuracy.

Method	Post-Trained Model Type	Top-1 Accuracy (%)			Parameter CR (%)
		Baseline	Compressed	$\Delta\downarrow$	
ResNet-20					
SCOP (Tang et al., 2020)	Sparse	92.22	90.75	1.47	56.3
Hinge (Li et al., 2020)	Low-Rank + Sparse	92.54	91.84	0.70	55.5
FGFP(CA-FGF) (ours)	Fractional Filter + Sparse	91.34	90.77	**0.57**	59.3
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	91.34	90.34	1.00	**66.7**
PSTRN-M (Li et al., 2022)	Low-Rank	91.25	89.30	1.95	85.2
ELRT (Sui et al., 2024)	Low-Rank	91.25	89.64	1.61	83.4
TDLC (Liu et al., 2024)	Low-Rank	91.25	88.58	2.65	80.5
FGFP(CA-FGF) (ours)	Fractional Filter + Sparse	91.34	90.20	**1.14**	81.5
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	91.34	89.82	1.52	**85.2**
ResNet-32					
SCOP (Tang et al., 2020)	Sparse	92.66	92.13	0.53	56.2
PSTRN-S (Li et al., 2022)	Low-Rank	92.49	91.43	1.06	60.9
FGFP(CA-FGF) (ours)	Fractional Filter + Sparse	92.64	92.11	**0.53**	**76.1**
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	92.64	91.92	0.72	**76.1**
PSTRN-M (Li et al., 2022)	Low-Rank	92.49	90.59	1.90	80.4
ELRT (Sui et al., 2024)	Low-Rank	92.49	91.21	1.28	80.4
FGFP(CA-FGF) (ours)	Fractional Filter + Sparse	92.64	91.85	**0.79**	**80.4**
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	92.64	91.80	0.84	**80.4**
WRN-28-10					
GrowEfficient (Yuan et al., 2021)	Sparse	96.20	95.30	0.90*	90.7
BackSparse (Zhou et al., 2021)	Sparse	96.20	95.60	0.60*	91.6
FGFP(CA-FGF) (ours)	Fractional Filter + Sparse	94.78	93.68	1.10*	91.6
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	94.78	94.24	**0.54***	**96.8**

dataset.

Evaluation Metrics. The model is evaluated using the accuracy drop and the number of parameters required. Considering the model's performance, top-1 accuracy is utilized on classification tasks. Also, the parameter compression ratio (CR) is defined as the percentage reduction in the number of parameters compared to the original model.

Configurations. All experiments use the stochastic gradient descent (SGD) optimizer. The batch sizes are 128 and 256 for CIFAR-10 and ImageNet2012, respectively. The learning rates in the FGF training stage are 0.1 for all model architectures. The learning rates in the adaptive unstructured pruning stage are set at 0.01 to 0.001 for ResNet-20 and ResNet-32, and 0.0001 for WRN-28-10, ResNet-18, and ResNet-50, respectively.

3.2. Results and Analysis

CIFAR-10. Table 1 presents the comparison results between our FGFP and recent works on ResNet-20, ResNet-32, and WRN-28-10. When we evaluate the performance of ResNet-20 on the test set, the compression ratio of FGFP can reach higher than 80% no matter using both the 3D-FGF and the CA-FGF. Meanwhile, the reduction of accuracy is less than the PSTRN-M (Li et al., 2022), the ELRT (Sui et al., 2024), and the TDLC (Liu et al., 2024). Similarly, on ResNet-32, FGFS also achieves excellent performance, reaching a compression ratio of 80.4% while maintaining higher accuracy than both PSTRN-M and ELRT. For WRN-28-10, the FGFP, especially with the 3D-FGF, can compress the network by 96.8% and significantly outperform the GrowEfficient (Yuan et al., 2021) and BackSparse (Zhou et al., 2021).

ImageNet2012. To evaluate the scalability of the FGFP, we perform experiments on the ImageNet2012 dataset using the ResNet-18 and ResNet-50 architectures, which are shown in Table 2. On ResNet-18, FGFP with 3D-FGF achieves a 74.7% compression ratio with only about 1% accuracy degradation, demonstrating superior performance compared to recent works such as FR (Chu & Lee, 2021) and LRPET (Guo et al., 2024). Similarly, on ResNet-50, the FGFP with the 3D-FGF also achieves a higher compression ratio, 69.1%, than other compression methods, while the accuracy of the FGFP only drops 1.63%. Also, the FGFP can maintain better performance when the network model is compressed by various compression ratios.

Table 2. Results for ResNet-18 and ResNet-50 on ImageNet2012 dataset.

Method	Post-Trained Model Type	Top-1 Accuracy (%)			Parameter CR (%)
		Baseline	Compressed	Δ ↓	
ResNet-18					
FR (Chu & Lee, 2021)	Low-Rank	69.76	69.04	0.72	57.0
LRPET (Guo et al., 2024)	Low-Rank	69.76	67.87	1.89	50.3
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	69.30	68.61	**0.69**	60.1
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	69.30	68.28	1.02	**74.7**
ResNet-50					
EDP (Ruan et al., 2024)	Low-Rank + Sparse	75.90	75.34	0.56	43.9
ARPruning (Yuan et al., 2024)	Sparse	76.15	72.31	3.84	56.8
SFI-FP (Yang et al., 2024)	Sparse	76.15	75.21	0.94	57.3
CORING (Pham et al., 2024b)	Sparse	76.15	75.55	0.60	56.7
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	76.16	75.64	**0.52**	**57.4**
Stable (Phan et al., 2020)	Low-Rank	76.15	74.68	1.47	60.2
CC (Li et al., 2021)	Low-Rank + Sparse	76.15	74.54	1.61	58.6
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	76.16	75.42	**0.74**	**62.7**
AHC-A (Wang et al., 2024)	Sparse	76.20	74.70	1.50	63.4
LRPET-S (Guo et al., 2024)	Low-Rank	76.15	73.72	2.43	64.0
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	76.16	74.82	**1.34**	**66.8**
NORTON (Pham et al., 2024a)	Low-Rank + Sparse	76.15	74.00	2.15	68.8
FGFP(3D-FGF) (ours)	Fractional Filter + Sparse	76.16	74.53	**1.63**	**69.1**

3.3. Ablation Study

Comparison of FGFP and AUP. We conducted an ablation study on CIFAR-10 using ResNet-20 to demonstrate that combining fractional filters and pruning outperforms using adaptive unstructured pruning alone. The experimental results are shown in Table 3. We aim to compress the network model to the specified number of parameters with adaptive unstructured pruning (AUP) and the FGFP, which combines the AUP and the CA-FGF. In Table 3, the FGFP shows superior performance compared to using AUP alone, achieving a 0.33% accuracy improvement when the remaining parameters are reduced to 0.07M.

Table 3. Performance comparison between FGFP and AUP for ResNet-20 on CIFAR-10.

Method	ΔAcc. ↓(%)	#Parameter
AUP	1.19	0.07M
FGFP(CA-FGF + AUP)	**0.86**	0.07M

Comparison of FGFP(CA-FGF) and FGFP(3D-FGF). To analyze the relationship between CA-FGF, 3D-FGF, and adaptive unstructured pruning (AUP), we converted the 28th, 29th, 30th, and 31st layers of ResNet-32 into FGF layers while performing AUP on the remaining layers. As shown in Table 4, when only FGF conversion was applied, the accuracy of CA-FGF was 0.25% higher than that of 3D-FGF. After applying AUP, when the total number of parameters was reduced to 0.07M, the accuracy of CA-FGF remained 0.16% higher than that of 3D-FGF. However, as parameters dropped to 0.05M, excessive AUP pruning caused significant feature loss, resulting in an additional 0.31% accuracy drop in CA-FGF compared with 3D-FGF.

Table 4. Comparison of accuracy and compression ratio between FGFP using CA-FGF and 3D-FGF for ResNet-32 on CIFAR-10.

Method	ΔAcc. ↓(%)	#Parameter
CA-FGF	**0.36**	0.33M
3D-FGF	0.61	0.32M
FGFP(CA-FGF)	**1.11**	0.07M
FGFP(3D-FGF)	1.27	0.07M
FGFP(CA-FGF)	2.26	0.05M
FGFP(3D-FGF)	**1.95**	0.05M

4. Conclusion

In this paper, we present the novel fractional Gaussian filter and Pruning (FGFP) framework for network model compression. There are two principal mechanisms in the FGFP: the fractional Gaussian filter (FGF) and the adaptive unstructured pruning (AUP). We integrate fractional-order differential calculus with the Gaussian function to construct the FGF, incorporating the Grünwald–Letnikov fractional derivatives for simplification. By sharing the parameters of the FGF and utilizing the technique of channel attention, the number of parameters in the network can be reduced to seven. Moreover, we apply the AUP to our FGFP to achieve the maximum compression ratio and maintain the accuracy of the models. We also utilize comprehensive experiments on CIFAR-10 and ImageNet2012 to validate the effectiveness of our proposed method.

According to the experimental results, the 85.2% compression ratio can be achieved with only a 1.52% degradation in accuracy by using the FGFP to compress the ResNet-20. Also, the FGFP can achieve the compression ratio of 69.1% while the accuracy only decreases by 1.63% on ResNet-50 with the ImageNet2012 dataset. In summary, the FGFP, which combines the fractional Gaussian filter and adaptive unstructured pruning, is a promising solution to mitigate parameter redundancy in modern deep neural networks and achieves substantial model compression with minimal accuracy degradation.

References

Chu, B.-S. and Lee, C.-R. Low-rank tensor decomposition for compression of convolutional neural networks using funnel regularization. *arXiv preprint arXiv:2112.03690*, 2021.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, 2014.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Gonzalez, R. and Woods, R. *Digital Image Processing*. Prentice Hall, 2008. ISBN 9780131687288.

Guo, K., Lin, Z., Chen, C., Xing, X., Liu, F., and Xu, X. Compact model training by low-rank projection with energy transfer. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2024. doi: 10.1109/TNNLS.2024.3400928.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Jalalinejad, H., Tavakoli, A., and Zarmehi, F. A simple and flexible modification of grünwald–letnikov fractional derivative in image processing. *Mathematical Sciences*, 12(3):205–210, 2018.

Kanopoulos, N., Vasanthavada, N., and Baker, R. Design of an image edge detection filter using the sobel operator. *IEEE Journal of Solid-State Circuits*, 23(2):358–367, 1988. doi: 10.1109/4.996.

Li, N., Pan, Y., Chen, Y., Ding, Z., Zhao, D., and Xu, Z. Heuristic rank selection with progressively searching tensor ring network. *Complex & Intelligent Systems*, 8(2):771–785, 2022. doi: 10.1007/s40747-021-00308-x.

Li, Y., Gu, S., Mayer, C., Van Gool, L., and Timofte, R. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8015–8024, 2020.

Li, Y., Lin, S., Liu, J., Ye, Q., Wang, M., Chao, F., Yang, F., Ma, J., Tian, Q., and Ji, R. Towards compact cnns via collaborative compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6434–6443, 2021.

Liu, W., Liu, P., Shi, C., Zhang, Z., Li, Z., and Liu, C. Tdlc: Tensor decomposition-based direct learning-compression algorithm for dnn model compression. *Concurrency and Computation: Practice and Experience*, 2024.

Llanza, A., Keddous, F. E., Shvai, N., and Nakib, A. Deep learning models compression based on evolutionary algorithms and digital fractional differentiation. In *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–9, 2023.

Pham, V. T., Zniyed, Y., and Nguyen, T. P. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2024a. doi: 10.1109/TNNLS.2024.3370294.

Pham, V. T., Zniyed, Y., and Nguyen, T. P. Efficient tensor decomposition-based filter pruning. *Neural Networks*, 178:106393, 2024b. ISSN 0893-6080. doi: 10.1016/j.neunet.2024.106393. URL https://www.sciencedirect.com/science/article/pii/S0893608024003174.

Phan, A.-H., Sobolev, K., Sozykin, K., Ermilov, D., Gusak, J., Tichavskỳ, P., Glukhov, V., Oseledets, I., and Cichocki, A. Stable low-rank tensor decomposition for compression of convolutional neural network. In *European Conference on Computer Vision (ECCV)*, pp. 522–539. Springer, 2020.

Ruan, X., Liu, Y., Yuan, C., Li, B., Hu, W., Li, Y., and Maybank, S. Edp: An efficient decomposition and pruning scheme for convolutional neural network compression. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4499–4513, 2024. doi: 10.1109/TNNLS.2020.3018177.

Scherer, R., Kalla, S. L., Tang, Y., and Huang, J. The grünwald–letnikov method for fractional differential equations. *Computers & Mathematics with Applications*, 62(3):902–917, 2011.

Sui, Y., Yin, M., Gong, Y., Xiao, J., Phan, H., and Yuan, B. Elrt: Efficient low-rank training for compact convolutional neural networks. *arXiv preprint arXiv:2401.10341*, 2024.

Tang, Y., Wang, Y., Xu, Y., Tao, D., XU, C., Xu, C., and Xu, C. Scop: Scientific control for reliable neural network pruning. In *Advances in Neural Information Processing Systems*, pp. 10936–10947, 2020.

Wang, H., Ling, P., Fan, X., Tu, T., Zheng, J., Chen, H., Jin, Y., and Chen, E. All-in-one hardware-oriented model compression for efficient multi-hardware deployment. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(12):12345–12359, 2024. doi: 10.1109/TCSVT.2024.3434626.

Yang, L., Gu, S., Shen, C., Zhao, X., and Hu, Q. Soft independence guided filter pruning. *Pattern Recognition*, 153:110488, 2024. ISSN 0031-3203. doi: 10.1016/j.patcog.2024.110488. URL https://www.sciencedirect.com/science/article/pii/S0031320324002395.

Yuan, T., Li, Z., Liu, B., Tang, Y., and Liu, Y. Arpruning: An automatic channel pruning based on attention map ranking. *Neural Networks*, 174:106220, 2024. ISSN 0893-6080. doi: 10.1016/j.neunet.2024.106220. URL https://www.sciencedirect.com/science/article/pii/S0893608024001448.

Yuan, X., Savarese, P., and Maire, M. Growing efficient deep networks by structured continuous sparsification. In *International Conference on Learning Representations*, 2021.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1–87.12, 2016.

Zamora, J., Cruz Vargas, J. A., Rhodes, A., Nachman, L., and Sundararajan, N. Convolutional filter approximation using fractional calculus. In *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 383–392, 2021.

Zhang, T., Ye, S., Zhang, K., Tang, J., Wen, W., Fardad, M., and Wang, Y. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *European Conference on Computer Vision (ECCV)*, pp. 184–199. Springer, 2018.

Zhou, X., Zhang, W., Chen, Z., DIAO, S., and Zhang, T. Efficient neural network training via forward and backward propagation sparsification. In *Advances in Neural Information Processing Systems*, pp. 15216–15229, 2021.