2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
June 28-30, 2023. Seattle, Washington, USA

Deformable Fractional Filters

Julio Zamora-Esquivel, Anthony Rhodes, Edgar Macias-Garcia and Lama Nachman
Intel Labs
Email: {julio.zamora.esquivel, anthony.rhodes, edgar.macias.garcia, lama.nachman}@intel.com

Abstract—This paper introduces Deformable Fractional Filters (DFFs) for Convolutional Neural Networks (CNNs). DFFs enhance the efficiency of conventional Deformable Convolutional Filters by introducing a compression mechanism rooted in techniques from fractional calculus. Concretely, our method reduces the parameter overhead requirement of convolutional filters by replacing the kernel with a fractional approximation, which can be trained using only three parameters – regardless of the kernel size. DFFs present a compelling use case for the compression of networks that require large kernel sizes. To demonstrate the benefits of DFFs, we report experimental results across a diverse set of computer vision problem domains, including classification, semantic segmentation, and medical imaging. Our experiments illustrate the favorable performance and regularization properties presented by DFFs in comparison with other baseline CNNs.

I. Introduction

Computer vision has generated an impressive array of increasingly sophisticated techniques over the last decade, driven chiefly by Convolutional Neural Networks (CNNs). Modern deep learning models have achieved state-of-the-art accuracy in a variety of diverse tasks, including image classification [1], semantic segmentation [2], object detection [3], pose detection [4], among others. Much of the research in this area stems from the development of increasingly deep architectures, comprised of millions (even billions [5]) of trainable parameters to continuously achieve better performance. Due to the memory and computational constraints of these large-scale models, many of these outstanding results have yet to fully translate over to edge computing devices and other compute constrained environments.

As an alternative to training and deploying unwieldy overparameterized models, researchers have recently focused on more sustainable network designs [6] to generate smaller and more efficient models at the cost of a potentially minor trade-off in accuracy. There have been many approaches proposed in this vein, including ShuffleNet [7], MobileNet [8], HENet [9], and SqueezeNet [10].

Motivated by prior research exploring the addition of adaptive parameters to activation functions [11] and convolutional kernels [12], in this work we apply concepts from fractional calculus [13] to enable a neural network to learn a reduced representation of a convolutional kernel in functional form, i.e., as a fractional kernel. In addition, we demonstrate that neural networks utilizing fractional kernels perform comparably to state-of-the-art models on several benchmark data sets (MNIST [14], TCGA-LGG [15], and ADE20K [16]), while having a reduction in the number of kernel parameters in the compressed layers.

We furthermore show that this novel convolutional paradigm facilitates the creation of generalized high-performance architectures that are more efficient in terms of training time and accuracy. In particular, our analysis focuses on larger kernel sizes (e.g., 5×5, 7×7 and above), as our method can leverage the benefits of these larger kernels while yielding appreciable increases in model compression rates. Recent research [17], [18] has demonstrated the superior performance of Convolutional Neural Networks using large kernel sizes on several applications, including high fidelity computer visions tasks including semantic segmentation, super-resolution upsampling, and medical imaging; moreover, the positive benefits of using larger kernels to increase the effective receptive field of a CNN is well-documented [19].

II. Related work

Over the past decade, computer vision research has tended to leverage the benefits of the compositional structure of high-capacity, deep networks [20]. Recent work [21], however, reveals that many deep models suffer from severe inefficiencies due to the presence of gross overparameterization. These discoveries have spurred interest in the development of more efficient CNNs; SqueezeNet [10] and MobileNets [8], for example, compress the convolutional kernels by using $N\times N\times1$ kernels instead of $N\times N\times3$, thus reducing the number of channels processed by the convolutional layer, yielding a reduced model in the number of trainable parameters. In [22], the authors demonstrate a dynamic filter framework in which a network generates a single filter used by all nodes in the first layer of convolutional kernels and is trained with the other network parameters.

Pruning represents a common technique used to reduce the memory and compute overhead required by overparameterized models [23]. Pruning methods do not conflict with the present work; both pruning and fractional filters can further augment model compression results. Similarly, memory-reducing approaches such as quantization [24] can also be applied in concert with our technique to reduce the number of network parameters. As an alternative approach to improving CNN model efficiency, [25] introduced Deformable Convolutions and Deformable ROI pooling based on augmenting the spatial sampling locations in CNN modules using trainable offset parameters for each

filter. These techniques were further improved in [26], introducing a learnable modulation parameter that controls the spatial support of the deformable convolution.

A. Fractional Derivative

The theory of fractional calculus [27] has previously been applied to neural network design. In [11], the authors use fractional calculus to group existing activation functions into families by defining the fractional order of a primitive activation function that is tuned during training. In this way, each neuron in the network learns a bespoke activation function, demonstrating, for instance, that ResNet-18 utilizing this adaptive activation method can outperform a ResNet-100 topology [28]. In a most recent work [29], authors introduced a generalized Fractional Convolutional Filter (FF), which has the flexibility to behave as any novel, customized, or well-known filter by employing only five parameters. This parameter reduction provides a nominal 5X parameter compression per kernel compared to standard (5×5) convolutional kernels while preserving generalization capacity. In recent years, fractional calculus has successfully served as a tool for modeling complex dynamics [30], for understanding wave propagation [31], and for working with quantum physics [32], among other applications.

To understand how a fractional derivative works, we begin with a simple, illustrative example. The natural n-derivatives of the function $f(x) = x^k$ are defined as:

$$\frac{\mathrm{d}^{\mathrm{a}} f(x)}{\mathrm{d}x^a} = k(k-1)(k-2)\cdots(k-a+1)x^{k-a}, \quad (1)$$

$$= \frac{k!}{(k-a)!}x^{k-a}. \quad (2)$$

For the case above, the factorial operation can only be defined for non-negative integer numbers.

$$\Gamma(z) = \int_0^\infty t^{(z-1)}e^{-t}dt, \quad (3)$$

a known efficient method to compute Gamma is [33]:

$$\Gamma(z) = \frac{e^{-\gamma z}}{z}\prod_{k=1}^{\infty}\left(\left(1+\frac{z}{k}\right)^{-1}e^{\frac{z}{k}}\right), \quad (4)$$

where γ is the Euler-Mascheroni constant [34]. Thus, by replacing the factorial in Eq. (2) by the Gamma function, the fractional derivative is then given by [35]:

$$D^a f(x) = \frac{\mathrm{d}^{\mathrm{a}} f(x)}{\mathrm{d}x^a} = \frac{\Gamma(k+1)}{\Gamma(k+1-a)}x^{k-a}. \quad (5)$$

The above definition represents the fractional derivative of a function $f(x) = x^k$ valid for $k, x \geq 0$. In the following sections, we apply these fractional calculus concepts to render a a fractional kernel used in our DFFs modules.

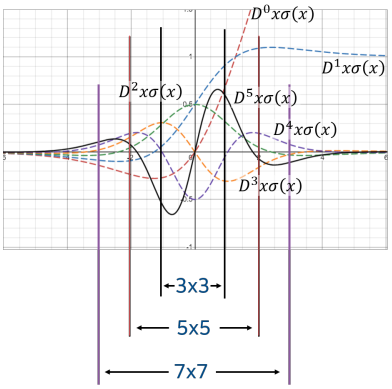

Fig. 1. Plots generated by our fractional filter definition in Eq. 6. The fractional derivative of the swish function $x\sigma(x)$ produces a wide range of function morphologies, as shown. In addition, because the filter has a corresponding explicit functional form, generating filters of varying sizes requires no additional parameter overhead.

B. Swish Fractional Filter

By calculating the fractional derivative of the so-called Swish function $f(x) = x\sigma(x)$ [36], one can generate approximations of many popular filters used in computer vision applications [37], including the Gaussian, Sobel, and Laplacian (see Figure 1). In addition, it is possible to produce an infinite number of novel filters that represent interpolations between these conventional filter types. To this end, we approximate the fractional derivative of a Swish function using truncated series as follows:

$$D^\alpha f(x) = \frac{1}{h^\alpha}\sum_{n=0}^{15}\frac{(-1)^n\Gamma(\alpha+1)(x-nh)\sigma(x-nh)}{\Gamma(n+1)\Gamma(1-n+\alpha)}, \quad (6)$$

where α denotes the fractional derivative order and $\Gamma(\alpha)$ represents the gamma function defined in Eq. (3). By tuning a single trainable parameter (α) in our fractional filter construction, generating a diverse family of filters is possible, as shown in Figure 1. Despite the truncation applied in Eq. (6), a model requiring many such calculations for training can be computationally expensive. In order to alleviate this computational overhead, we formulate a more efficient, recursively-defined approximation of the fractional derivative of the Swish function $f(x) = x\sigma(x)$ as follows: Let $g_1(x)$ denote the Swish function:

$$g_1(x) = f(x) = x\sigma(x), \quad (7)$$

where the first derivative of $g_1(x)$ is given by

$$g_2(x) = \sigma(x) + x\sigma(x)(1-\sigma(x)), \quad (8)$$

Following this approach for the subsequent derivatives and using $g_{(i+1)}(x) = D^i x\sigma(x)$ to represent the i-

derivative of the Swish function, one can similarly derive the following higher-order derivatives:

$$g_1(x) = x\sigma(x), \tag{9}$$
$$g_2(x) = g_1(x) + \sigma(x)(1 - g_1(x)), \tag{10}$$
$$g_3(x) = g_2(x) + \sigma(x)(1 - 2g_2(x)), \tag{11}$$
$$g_4(x) = g_3(x) + \sigma(x)(1 - g_2(x) - 3g_3(x)). \tag{12}$$

From this sequence of derivatives, it is possible to approximate any fractional derivative by adding the α parameter in the range $(0, 1)$.

C. Two Dimensional Filters

To define a two-dimensional filter, we independently compute the fractional derivatives of the swish functions for x and y and calculate the exterior product of the two generated vectors. Concretely, we compute the fractional derivative of the two-dimensional fractional kernel as the realization of $D^\alpha f(x) \times D^\beta f(y)$. The memory benefits of using the fractional filter increase according to the size of the filter. For example, a 3×3 kernel will yield a reduction from 9 to 3 parameters for each standard kernel replaced by a fractional kernel in a neural network layer. During training, every round of backpropagation requires recomputing FF parameters and the per-cell filter values.

During inference, the fractional filter is initially calculated (just once, when loading the network) from the stored parameters and then integrated into a CNN workflow, just as with traditional kernels. This initial step requires the evaluation of the fractional derivative over the $N \times N$ parameters of the kernel (e.g., 25 elements in the case of a 5×5 filter). By contrast, during training, the evaluation of the filters is required once per iteration, but we only need to compute $M \times N$ values per filter. In the 2D case, we also compute the $D^b G(y)$, so that the (i, j) components of a FF are defined: $K(i, j) = D^\alpha f(i) \cdot D^\beta f(j)$ for $1 \leq i, j \leq N$ (see Figure 2).

III. Deformable Fractional Filter

We augment the deformable convolution framework introduced in [25] by: (i) replacing conventional filters with 2D fractional filters, and (ii) incorporating a compression of the number of trainable offset and modulation parameters by using a simple linear projection. Regarding (ii), this compression process helps maintaining a low number of total trainable parameters required for DFFs (including the parameters required for 2D fractional filters and deformable offsets).

Deformable convolutions add 2D learnable offsets to regular grid sampling locations for standard convolutions. Including these offset sampling locations allows for the free-form deformation of the sampling grid, which means that convolutional filters can attend to groups of pixels in a more dynamic and non-rigid fashion than in conventional implementations. Note that deformable convolutions introduce an additional $2N$ trainable offset parameters for

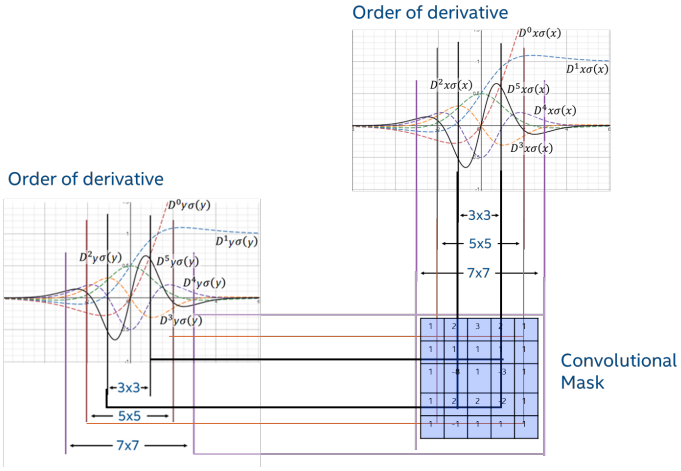

Fig. 2. The computation of the N^2 elements of a filter are executed by the exterior product of the N elements on the x-axis and the N elements on the y-axis. The remaining values are calculated by multiplying these x and y vectors.

each kernel's original N filter values (one for each x, and y filter offset, respectively). In our implementation of DFFs, we require $\frac{2N}{k}$ trainable offset parameters, where k denotes a linear projection factor (we use $k = 2$ in our experiments). [25] demonstrated performance improvements on object localization and segmentation tasks over baseline CNN models. while [26] improved deformable convolutions by adding a trainable modulation parameter that controls the spatial support of the convolution.

Figure 3 provides a summary schematic of our Deformable Fractional Filter construction. As shown, we replace the standard convolution filter employed by [25] with our 2D fractional filter, defined as $D^\alpha(x\sigma(x)) \cdot D^\beta(y\sigma(y))$; we furthermore enable the modulation mechanism introduced in [26]. Finally, to reduce the total parameter overhead required for DFFs, we additionally include trainable linear projection operations (shared per layer) for the offset and modulation parameters in each DFF. Our experiments also utilized a $2X$ compression of both the offset and modulation parameters.

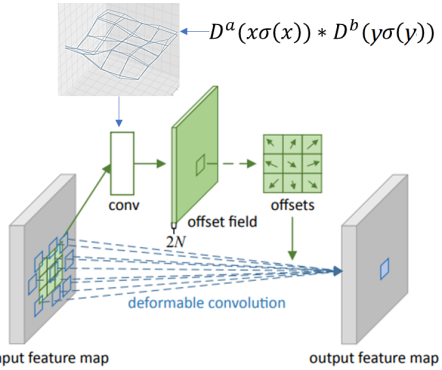

Fig. 3. The Deformable Filter modified by replacing the conventional convolutional filter with our fractional filter.

IV. Experimental Results

To analyze the benefits of the proposed methods, we conducted experiments on a set of standard models employing convolutional and deformable filters, comparing with modified versions using our DFFs filters across four datasets: MNIST [38] and FASHION-MNIST [39] for image classification, ADE20K [16] for semantic segmentation, and TCGA-LGG [15] for medical image segmentation.

A. MNIST

The MNIST dataset was employed to train and test four different models (Table I); A standard five-layer CNN model and three variations of the model by modifying its last Convolutional layer with a Deformable (DF), Deformable Fractional (DFF), and Fractional (FF) filter, respectively. The results of the training and validation steps are presented in Figure 4. As can be seen, the Fractional Filter model achieves a lower accuracy than the standard, as the filter employed approximates the traditional convolution causing a compression in the required convolution parameters (from 9 to 3). On the other hand, DF filters generate greater accuracy, while the DFF filters achieve an intermediate performance between both (preserving performance while requiring fewer number of parameters); For instance, a compression of 5X is achieved using a forced parameter $\sigma = 1$ (Table II). The use of DFFs additionally demonstrates consistently faster convergence and improved generalization properties when compared to baseline models.

TABLE I
MNIST custom topology employed for evaluations

Name	Output Size	std $(w \times h, ch, s)$	dsf_5
Conv1	$28 \times 28 \times 32$	$3 \times 3, 32, 1$	$3 \times 3, 32, 1$
Conv2	$26 \times 26 \times 32$	$3 \times 3, 32, 1$	$3 \times 3, 32, 1$
Conv3	$16 \times 16 \times 32$	$3 \times 3, 32, 1$	$3 \times 3, 32, 1$
Conv4	$8 \times 8 \times 32$	$3 \times 3, 32, 1$	$3 \times 3, 32, 1$
Conv5	$1 \times 1 \times 32$	$3 \times 3, 32, 1$	DSF $3 \times 3, 32, 1$
FC	10	32×10	32×10

TABLE II
Parameter's compression after applying Fractional filters

Model	Parameters	Model	Parameters
5-CNN model	37912	DF model	53248
FF model	31466	DFF model	40613

B. FASHION-MNIST

As a second experiment, we evaluated the aforementioned model architectures on the FASHION-MNIST dataset [39]; these results are shown in Figure 5. Our

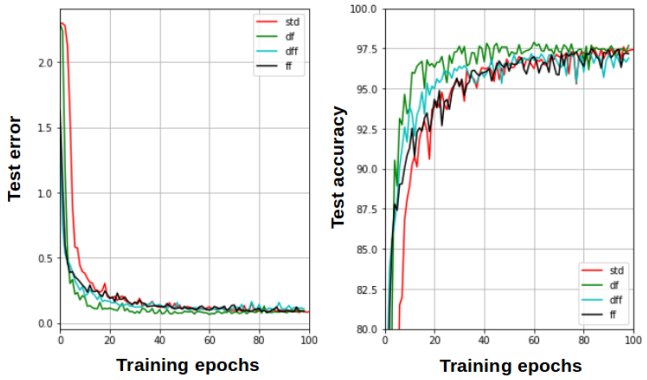

Fig. 4. Test loss and accuracy for the MNIST Dataset. Std: Standard 5-layer model, DF: Deformable filter, DFF: Deformable Fractional filter, and FF: Fractional filter.

experimental results show that both the FF and DFF-based CNNs outperform the baseline model for classification accuracy and training efficiency. Furthermore, the FF model outperforms the baseline CNN despite being more compact; similarly, the DFF model performs comparably with the larger baseline DF model.

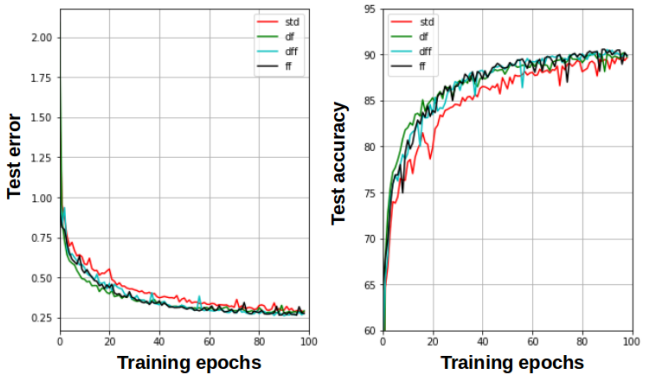

Fig. 5. Test loss and accuracy for experimental models using the FASHION-MNIST Dataset. std: Standard 5-layer model, DF: Deformable filter, DFF: Deformable Fractional filter, and FF: Fractional filter.

C. ADE20K:Semantic segmentation

Next, we tested our method on a semantic segmentation task using the ADE20K [16] dataset. In our experiments, we employed the dilated ResNet50 architecture provided by [40] as a baseline model. In our implementation, we replaced the decoder block's final convolution (the layer with the largest parameter overhead) with a DFF layer, then we trained the model from scratch during 30 epochs.

Once trained, the modified model achieves a general accuracy of 80.27 % on the test dataset. Furthermore, as presented in Table III, our model using a DFF layer was able to achieve comparable performance with the standard CNNs model while employing much fewer trainable parameters (approximately 24 % fewer parameters than

UperNet50 [41]), even surpassing the performance of larger models.

TABLE III
Validation accuracy and size of semantic segmentation models employing the ADE20K dataset [40]

Model (Encoder + Decoder)	Parameters	Accuracy
MobileNetV2dilated + C1	2181932	76.8
MobileNetV2dilated + PPM	13588396	78.26
ResNet18dilated + C1	12248940	77.41
ResNet18dilated + PPM	24566636	79.29
ResNet50dilated + PPM	51579116	80.13
UperNet50	64222294	80.23
DFF-ResNet50dilated + PPM (Ours)	39106871	80.27
ResNet101dilated +PPM	70571244	80.91

In order to help differentiate the qualitative performance of the filters, we compared this modified model with a standard baseline (ResNet50 Dilated) for semantic segmentation on a testing video. We show representative examples of this experiment in Figure 6. The model using DFFs consistently renders smoother, achieving contiguous segmentation frames compared with the standard network architecture while requiring fewer model parameters.

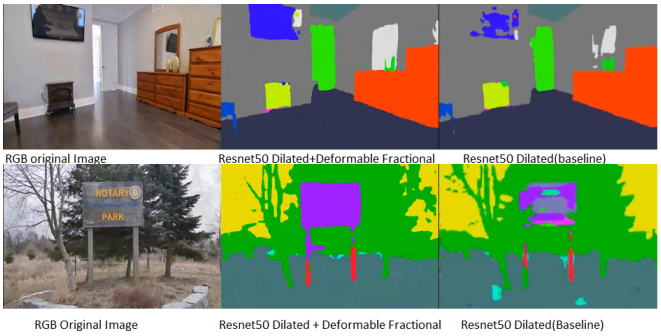

Fig. 6. Segmentation performance of the Deformable Fractional model against the standard ResNet50 model [40] using a random indoor video.

D. TCGA-LGGrepre: Segmentation for Medical Images

We additionally tested DFF-based architectures on a challenging medical segmentation task on the TCGA-LGG dataset [15]. In this experiment, we applied the U-Net [42] model implemented in [43] as a baseline architecture and replaced the first convolutional layer in the first and last resNet blocks with a DF, FF, and DFF layer, respectively, using the same layer parameters. Next, we trained the model from scratch during 20 epochs (Figure 7).

As can be seen in Figure 7, the model with the Deformable Fractional Filter layer achieves better convergence speed in both training and validation datasets compared with the standard model (Figure 8) while rendering a comparable accuracy performance, despite requiring significantly fewer parameters (Table IV).

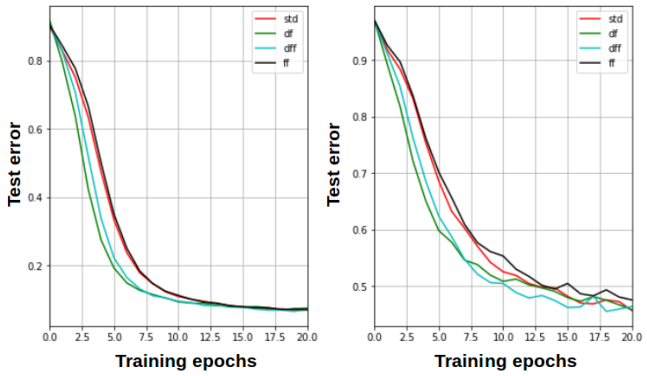

Fig. 7. Training and *validation* loss reached for the models using a subset of the TCGA-LGG dataset for a standard (std) and a Deformable Fractional filter (dff) model.

TABLE IV
TCGA-LGG U-Net evaluation results

Model	Parameters	T. Loss	V. Loss
U-Net	7763041	0.06917	0.45715
DF-U-Net	7779376	0.06998	0.45847
DFF-U-Net	7769281	0.06516	0.45563
FF-U-Net	7750177	0.07018	0.47596

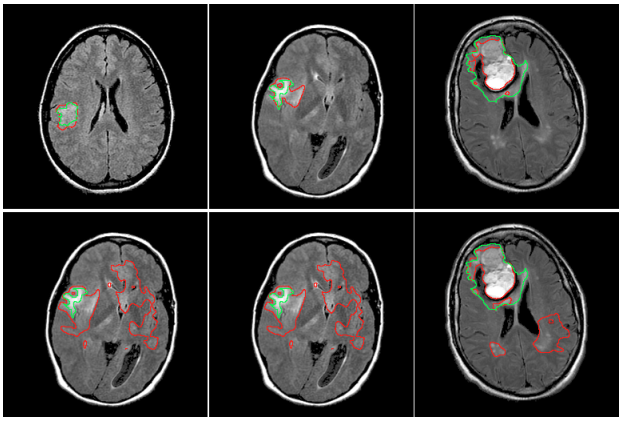

Fig. 8. Segmentation results on several representative test images from the TCGA-LGG dataset, after training for only two epochs. The DFF model demonstrates strong generalization performance relative to the baseline model, even at the early stages of training. Top: U-Net model with a DFF layer on the first and last Resnet block. Bottom: Standard U-Net model.

V. Conclusions

In this paper, we introduced Deformable Fractional Filters for CNNs. DFFs represent a novel convolutional network module that offers compression benefits and improved model expressivity. Leveraging fractional calculus, DFFs reduce the parameter overhead requirement of convolutional filters by replacing traditional filters with a fractional approximation. Through experiments, we demonstrated the favorable performance and regularization properties of DFFs compared with baseline CNNs across a wide variety of popular datasets.

References

[1] J. Deng, W. Dong, and et al., "ImageNet: A Large-Scale Hierarchical Image Database," in CVPR09, 2009.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[3] J. Dai, Y. Li, and et al., "R-fcn: Object detection via region-based fully convolutional networks," in Proceedings of the 30th International Conference on Neural Information Processing Systems, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 379–387.

[4] G. Moon, J. Y. Chang, and K. M. Lee, "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[5] A. Radford, "Improving language understanding by generative pre-training," 2018.

[6] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," ArXiv, vol. abs/1906.02243, 2019.

[7] X. Zhang, X. Zhou, and et al., "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848–6856.

[8] A. G. Howard, M. Zhu, and et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[9] R. Z. Qiuyu Zhu, "Henet: A highly efficient convolutional neural networks optimized for accuracy, speed and storage," arXiv:1803.02742, 2018.

[10] F. Iandola, S. Han, and et al., "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," ICLR2017, 2017.

[11] J. Zamora-Esquivel, A. Cruz-Vargas, and et al., "Adaptive activation functions using fractional calculus," in Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019, pp. 0–0.

[12] J. Zamora-Esquivel and A. Cruz-Vargas, "Adaptive convolutional kernels," in Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019, pp. 0–0.

[13] I. Podlubny, Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications, ser. Mathematics in Science and Engineering.

[14] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[15] J. Kirby and K. Smith, "The Cancer Genome Atlas Low Grade Glioma (TCGA-LGG) data collection," The Cancer Imaging Archive (TCIA) Public Access, 2022, accessed on: Jan. 10, 2022. [Online]. Available: https://wiki.cancerimagingarchive.net/display/Public/TCGA-LGG.

[16] B. Zhou, H. Zhao, and et al, "Scene parsing through ade20k dataset," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[17] D. Liu, D. Zhang, and et al., "3d large kernel anisotropic network for brain tumor segmentation," in Neural Information Processing - 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part VII, ser. Lecture Notes in Computer Science, vol. 11307. Springer, 2018, pp. 444–454.

[18] B. Wang, S. Qiu, and H. He, "Dual encoding u-net for retinal vessel segmentation." Berlin, Heidelberg: Springer-Verlag, p. 84–92.

[19] W. Luo, Y. Li, and et al.

[20] H. Mhaskar and T. Poggio, "Deep vs. shallow networks : An approximation theory perspective," ArXiv, vol. abs/1608.03287, 2016.

[21] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," arXiv: Learning, 2019.

[22] B. D. Brabandere, X. Jia, and et al., "Dynamic filter networks," CoRR, vol. abs/1605.09673, 2016. [Online]. Available: http://arxiv.org/abs/1605.09673

[23] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Training pruned neural networks," 2018, cite arxiv:1803.03635. [Online]. Available: http://arxiv.org/abs/1803.03635

[24] I. Hubara, M. Courbariaux, and et al., "Quantized neural networks: Training neural networks with low precision weights and activations," in Journal of Machine Learning Research, 2018.

[25] J. Dai, H. Qi, and et al., "Deformable convolutional networks," in Proceedings of the IEEE international conference on computer vision, 2017, pp. 764–773.

[26] X. Zhu, H. Hu, and et al., "Deformable convnets v2: More deformable, better results," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9308–9316.

[27] S. Das, Functional Fractional Calculus, 2nd ed. Springer Publishing Company, Incorporated, 2014.

[28] K. He, X. Zhang, and et al., "Deep residual learning for image recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[29] J. Zamora, A. Cruz, and et al., "Convolutional filter approximation using fractional calculus," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 383–392.

[30] S. W. Wheatcraft and M. M. Meerschaert, "Fractional conservation of mass," Advances in Water Resources, vol. 31, no. 10, pp. 1377–1381, 2008.

[31] S. Holm and S. P. Näsholm, "A causal and fractional all-frequency wave equation for lossy media," The Journal of the Acoustical Society of America, vol. 130, no. 4, pp. 2195–2202, 2011.

[32] N. Laskin, "Fractional schrödinger equation," Physical Review E, vol. 66, no. 5, p. 056108, 2002.

[33] P. J. Davis, ""leonhard euler's integral: A historical profile of the gamma function"," American Mathematical/ doi:10.2307/2309786, vol. Monthly. 66 (10), p. 849–869, 2016.

[34] S. I. Abramowitz M., "Handbook of mathematical functions with formulas, graphs and mathematical tables," vol. 10th ed, pp. 258–259, 1972.

[35] R. Herrmann, Fractional Calculus. World Scientific Publishing, 2011.

[36] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," arXiv: Neural and Evolutionary Computing, 2017.

[37] R. Gonzalez and R. Woods, Digital image processing. Prentice Hall, 2008. [Online]. Available: https://www.pearson.com/us/higher-education/program/Gonzalez-Digital-Image-Processing-4th-Edition/PGM241219.html

[38] Y. LeCun, "The mnist database of handwritten digits," http://yann. lecun. com/exdb/mnist/, 1998.

[39] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," arXiv preprint arXiv:1708.07747, 2017.

[40] B. Zhou, H. Zhao, and et al., "Semantic understanding of scenes through the ade20k dataset," International Journal on Computer Vision, 2018.

[41] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[42] M. Buda, A. Saha, and M. A. Mazurowski, "Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm," Computers in Biology and Medicine, vol. 109, 2019.

[43] mateuszbuda, "U-Net implementation in PyTorch for FLAIR abnormality segmentation in brain MRI ," github, 2022, accessed on: Jan. 5, 2022. [Online]. Available: https://github.com/mateuszbuda/brain-segmentation-pytorch.