Scaling Continuous Kernels with Sparse Fourier Domain Learning

Clayton Harper[1], Luke Wood[2], Peter Gerstoft[2], Mitchell A. Thornton[1], and Eric C. Larson[1]

[1] Southern Methodist University, Darwin Deason Institute for Cybersecurity, Dallas, TX, USA {caharper,eclarson,mitch@smu.edu}

[2] University of California San Diego, Computer Science, La Jolla, CA, USA lukewoodcs@gmail.com, gerstoft@ucsd.edu

Abstract. Convolution in the spectral domain is a linear-time computation, making it possible to train convolutional neural networks (CNNs) with significantly larger kernels than their spatial counterparts, which typically are limited to 3x3 or 5x5 kernels. To solve this, continuous kernel networks have been proposed. Continuous kernel networks replace fixed-size kernels with generating functions, enabling kernels with global context while maintaining a manageable parameter count. However, learning continuous kernels is prohibitively expensive, even for modest-sized datasets. In this work we propose an optimized method for efficiently training arbitrarily large MLP-backed kernels directly in the spectral domain. Our method addresses three key challenges in learning continuous kernel representations: computational efficiency during training, parameter efficiency in representation, and training speed. We evaluate spectral domain continuous kernel CNNs on medium sizes image classification datasets while achieving comparable accuracy to that of their spatial domain counterparts.

Keywords: Continuous kernels, Spectral learning

1 Introduction

Continuous kernel representations have been suggested for learning large convolutional kernels while maintaining a fixed-parameter budget [11, 10, 5, 12, 13]. Instead of directly parameterizing convolutional kernels as discrete weights, this approach uses a small neural network as a multi-layer perceptron (MLP) to generate kernels by sampling at specified spatial positions. This formulation treats convolutional kernels as continuous vector-valued functions, allowing them to adapt to arbitrary resolutions seamlessly. By querying the MLP at finer spatial intervals, larger kernels can be synthesized without increasing the parameters, offering a scalable and flexible alternative to traditional convolutional kernels.

In continuous kernel generation, the MLP generates a kernel by sampling at spatial intervals corresponding to the dimensions of the input feature map. For instance, if the input feature map has dimensions $H \times W$, the MLP is sampled

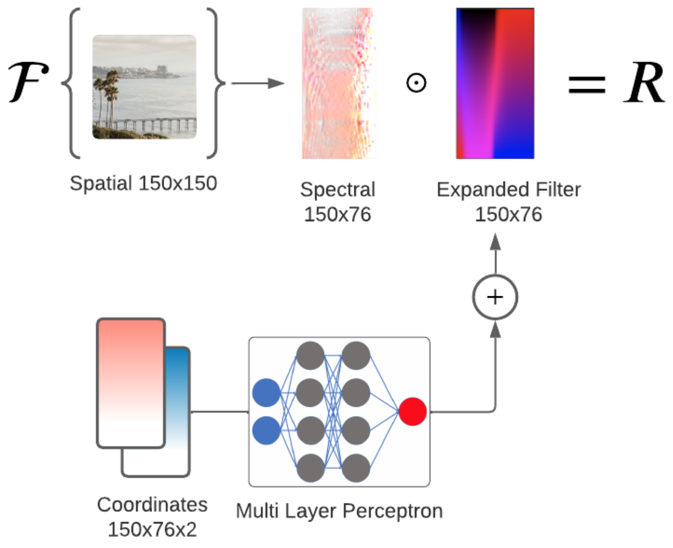

Fig. 1. Continuous Kernels parameterized with an MLP.

at $H \times W$ positions, producing a kernel of the same dimensions. Although the MLP generates a $H \times W$ kernel, the region with non-zero values—the active region—can vary during training. Thus, a notable advantage of this formulation is its ability to dynamically learn the effective kernel size. For example, a gradient update could change the active region from $h_1 \times w_1$ to $h_2 \times w_2$. This dynamic behavior is intrinsic to continuous kernel representations, as the MLP regenerates the kernel at each forward pass. This process enables the effective kernel size to be learned through gradient-based optimization, allowing the network to adapt its receptive fields to the underlying data. This adaptability improves the expressiveness and representational capacity of the network.

This capability distinguishes continuous kernels from prior approaches to dynamic kernel size learning [2, 4, 9], which typically rely on a single static kernel size applied uniformly across all channels and filters in a layer. Such uniformity limits the network's capacity to capture diverse spatial dependencies. In contrast, continuous kernels can generate unique kernel sizes for each channel, providing a more flexible and nuanced mechanism for spatial modeling.

Despite their advantages, the practical deployment of continuous kernels is hindered by several challenges, including (1) high parameter counts, (2) significant computational and memory demands during training, and (3) slow training speeds. These limitations often make it challenging to scale continuous kernels effectively in real-world applications, highlighting the need for methods that can reduce resource requirements while retaining the flexibility and adaptability of these representations.

The computational overhead associated with training continuous kernels is substantial. This is due to the fact that continuous kernel methods, unlike traditional CNNs, must generate the entire convolutional kernel dynamically at each forward pass. This on-the-fly kernel generation incurs significant computational costs. Additionally, automatic differentiation, a technique used by modern deep learning frameworks such as PyTorch and TensorFlow, relies on storing intermediate activations for gradient computation. For continuous kernels, this process is particularly memory-intensive, as it involves backpropagating through the kernel generation process itself. As a result, scaling continuous kernel methods to large datasets or high-resolution inputs becomes prohibitively expensive, historically limiting their practicality in large-scale applications.

In this work, we propose a novel approach, Continuous Fourier Convolutions (CF-Convs), which addresses key challenges in continuous kernel learning by leveraging the Fourier domain and introducing a stochastic update mechanism. Instead of regenerating the entire kernel at every forward pass, CF-Convs stochastically update only a subset of kernel values, significantly reducing computational and memory demands. Additionally, CF-Convs bypass the costly Fourier transforms over the entire generated kernel required for spatial-domain continuous kernels [11, 10] by learning directly in the spectral domain. This integration of Fourier-based learning and stochastic updates offers a scalable and practical framework for continuous kernel representations.

2 Related Work

Continuous convolutional kernels have emerged as an alternative to discrete convolutional filters, enabling flexible, learnable receptive fields [11, 10, 5]. These approaches align with implicit neural representations, where neural networks parameterize continuous functions instead of storing discrete kernel weights [7, 8, 16].

Although the majority of continuous kernel methods operate in the spatial domain, the Fourier domain has also been explored for kernel parameterization. Notably, Wood and Larson introduced a Fourier-based approach that employed 2D Gaussian parameterized kernels to reduce the parameter burden [18]. However, their use of shared weights across input channels constrained the adaptability of the generated kernels, leading to uniform kernel sizes across channels—similar to the limitations of traditional discrete spatial CNNs.

3 Continuous Kernels

Terminology: Let x denote the input feature map and k the convolutional kernel. The spatial dimensions (height and width) are given by H and W, while C_{in} and C_{out} represent the number of input and output channels, respectively. The Fourier transform over the spatial dimensions $H \times W$ is denoted by $\mathcal{F}$, with $\mathcal{F}^{-1}$ as its inverse. The input feature map and convolutional kernel are defined as $x \in \mathbb{R}^{H \times W \times C_{\text{in}}}$ and $k \in \mathbb{R}^{H \times W \times C_{\text{in}} \times C_{\text{out}}}$, respectively.

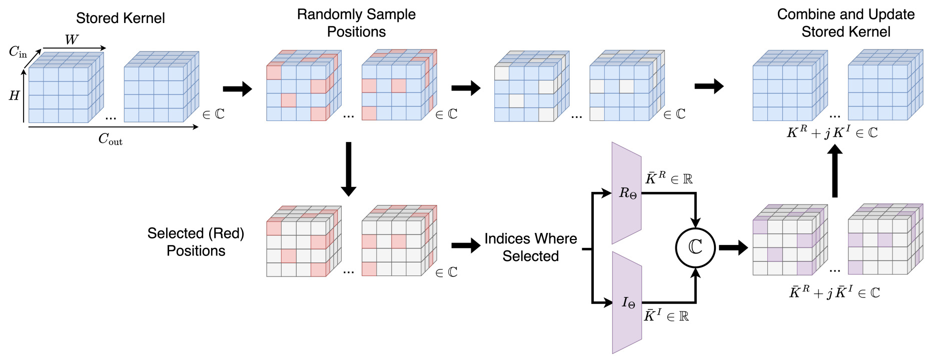

Fig. 2. Visualization of sparse sampling for efficient CF-Conv training. Randomly sampled positions for a single training step are highlighted in red. In the split kernel configuration, R_Θ and I_Θ represent the MLPs that generate the real and imaginary components of the complex-valued Fourier kernel $K \in \mathbb{C}$, respectively.

A convolutional kernel for a given layer is given by the shape $H \times W \times C_{\text{in}} \times C_{\text{out}}$. Instead of learning each value discretely, a continuous convolutional kernel replaces a discrete filter with a learnable function f_Θ (*e.g.*, an MLP). f_Θ dynamically generates kernel, k, values based on positional coordinates: $k(h, w, i, o) = f_\Theta(h, w, i, o)$ where $h \in [0, H-1]$, $w \in [0, W-1]$, $i \in [0, C_{\text{in}}-1]$, and $o \in [0, C_{\text{out}}-1]$ index the spatial and channel dimensions of the kernel. Typically, these indices are normalized to be in the [0, 1] interval, a convention we follow in this work.

Prior methods often transform both the generated kernel k and the input x into the Fourier domain, performing convolution via pointwise multiplication to leverage the computational efficiency of Fourier-based convolutions for large kernels [11, 10, 5]. Learning directly in the Fourier domain is an appealing alternative, as it eliminates the need to compute the Fourier transform of k at each forward pass. However, this approach introduces additional challenges, particularly the need to parameterize both real and imaginary components, effectively doubling the memory and computational requirements of f_Θ. In the next section, we introduce a method to address this limitation and enable learning directly in the Fourier domain.

4 CF-Convs

A convolutional layer in the Fourier domain for a given layer L is defined as:

$$g^{(L)} = x \star k = \mathcal{F}^{-1}\left(\sum_{i=1}^{C_{\text{in}}} X_i \odot K_{i,o}\right), \quad \forall o \in C_{\text{out}} \tag{1}$$

where $g \in \mathbb{R}^{(H \times W \times C_{\text{out}})}$ is the output in the spatial domain. Fourier counterparts are given by $X = \mathcal{F}(x)$ and $K = \mathcal{F}(k)$. Here, $X \in \mathbb{C}^{(H \times W \times C_{\text{in}})}$ and

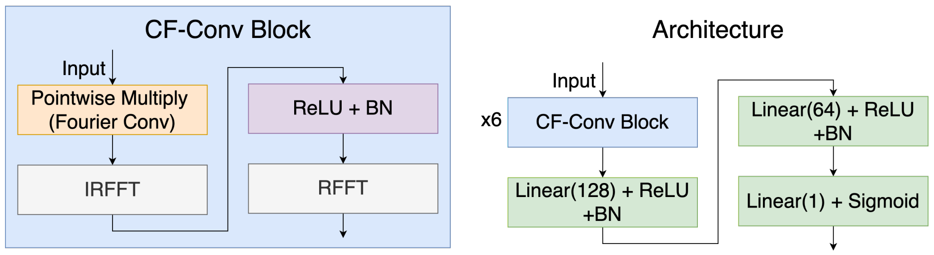

Fig. 3. Architectural overview of Cats vs. Dogs using CF-Convs. The convolutional kernel is generated using our CF-Conv approach and applied through Fourier-based convolution, as described in Equation (1), where pointwise multiplication produces the final convolutional output.

$K \in \mathbb{C}^{(H\times W\times C_{\text{in}}\times C_{\text{out}})}$. The operation $\odot$ denotes element-wise multiplication in the Fourier domain. The spatial output g is then recovered by applying the inverse Fourier transform. Since the Fourier transform produces complex-valued outputs, the convolutional kernel in a continuous kernel setup must separately parameterize its real and imaginary components, denoted as $K^R_{i,o}$ and $K^I_{i,o}$, respectively. This configuration, known as a split kernel configuration [14], is represented as $K_{i,o} = K^R_{i,o} + j\, K^I_{i,o}$, where $K_{i,o} \in \mathbb{C}^{(H\times W)}$.

Naive continuous kernels consume substantial memory due to their reliance on automatic differentiation, which stores intermediate activations for gradient computation—similar. However, Continuous Fourier Convolutions (CF-Convs), unlike spatial continuous kernels, require two MLPs to generate K, effectively doubling the memory overhead. Gradients must be computed and stored for all $H \cdot W \cdot C_{\text{in}} \cdot C_{\text{out}}$ positions, leading to an effective MLP "batch" size of $H \cdot W \cdot C_{\text{in}} \cdot C_{\text{out}}$ per MLP. As a result, even a modest 6-layer CF-Conv CNN with just 32 filters per layer can exhaust 80GB of GPU RAM. To make CF-Convs practical, memory reduction techniques are essential.

5 Memory Reduction

To reduce the memory consumption of CF-Convs, several strategies can be employed. Gradient checkpointing, or rematerialization [1, 6], reduces memory usage by storing only a subset of intermediate activations and recomputing the rest during the backward pass. While this approach saves memory, it incurs additional computational overhead due to repeated computation. Another method is using a *scan* operation, which sequentially applies a function over a collection of elements, accumulating results without storing all intermediate values simultaneously. Although gradient checkpointing and scan operations reduce memory consumption, they can significantly increase training times. These methods act more as analgesics, addressing the symptoms rather than solving the core problem of high memory consumption.

To address both training time and memory efficiency, we propose a sparse sampling strategy that focuses updates on a randomly sampled subset of kernel positions, referred to as *selected positions* (highlighted in red in Figure 2). Uniform sampling ensures that no positional bias is introduced. By reducing the number of evaluations per forward pass, our strategy significantly lowers memory consumption by minimizing the storage of intermediate activations. Furthermore, fewer MLP evaluations decrease computational overhead, resulting in faster forward passes and more efficient training.

During initialization, the kernel, with shape $H \times W \times C_{\text{in}} \times C_{\text{out}}$ is stored as a state variable for both the real and imaginary components of the CF-Conv layer. It is randomly initialized with a uniform distribution in the range [-1,1], ensuring no bias toward specific spatial positions or frequencies. During each forward pass, only the selected positions are re-evaluated and updated. The remaining positions, referred to as *unselected positions* (highlighted in blue in Figure 2), are held constant during that specific iteration. However, because sampling is randomized at every forward pass, unselected positions in one iteration may be selected in future iterations, ensuring that all positions are eventually updated over the course of training.

6 Experiments and Results

To evaluate the performance of our CF-Convs, we employ a 6-layer CNN using 32 filters at each layer (see Figure 3) on the Cats vs. Dogs dataset with image sizes of $150 \times 150 \times 3$ [3]. Additionally, we experiment with different numbers of selected positions—2^{12}, 2^{15}, and 2^{18}—to evaluate the trade-off between memory consumption and training stability.

To maintain parameter parity with a baseline 3×3 spatial CNN, we carefully select the architecture for CF-Convs. In the CF-Convs parameterization, which uses two MLPs, a network with [32, 32, 32, 16, 16, 16, 8, 8, 8, 1] neurons is employed for each MLP. Both configurations use ReLU activations between linear layers. Our training protocol incorporates a straightforward augmentation pipeline involving up to 3 augmentations per image using the *RandAugment* class provided by *KerasCV*, which is exclusively applied to the training set [19].

Early experiments revealed difficulties in learning when the architecture operated solely in the Fourier domain. These challenges may stem from the complexities of differentiating through complex-valued activation functions [14, 17]. To address this, we apply the (real) inverse FFT after each spectral convolutional, over the height and width axes, followed by traditional activation functions in the spatial domain, as illustrated in Figure 3 (left). The outputs from the convolutional layers are average pooled across the channel dimension and fed into linear layers with 128 and 64 neurons, respectively, with ReLU activations applied throughout. A final sigmoid layer is used for binary classification.

As shown in Table 1, sparse updates combined with scan or vmap operations dramatically reduce training time compared to the rematerialization and naive+scan approaches. Using vmap (vectorized map) further improves efficiency

Table 1. Comparison of epoch training time, parameter count, number of selected positions (*e.g.*, 2^{12}), and accuracy across different methods on the Cats vs. Dogs dataset. A 6-layer CNN with 32 filters per layer is used for all experiments. Training times are reported on A100 80GB GPUs.

Method	**Epoch Training Time**	**# Params** $\downarrow$	**Accuracy (%)** $\uparrow$
Spatial 3×3 CNN	$\sim$ 30s	60K	**86.64**
CF-Convs			
Naive	–	59K	Out-of-memory
Rematerialization	> 2 days	–	–
Scan	> 2 days	–	–
Sparse updates			
+Scan			
2^{18}	$\sim$ 18 min	–	–
+Vmap			
2^{12}	$\sim$ 4.5 min	59K	75.26
2^{15}	$\sim$ 4.5 min	59K	79.27
2^{18}	$\sim$ 5 min	59K	85.30

by running parallel computations. While there remains an approximately $10\times$ difference in training speeds compared to spatial CNNs, several important considerations contextualize this disparity. Firstly, the comparison is against spatial CNNs utilizing small 3×3 filters, whereas our layers can learn kernels ranging from spatial equivalents of 1×1 to $H \times W$ in about 5 minutes. Training time for our layers is agnostic to the kernel size, taking the same 5 minutes whether the spatial equivalent is 1×1 or $H \times W$. In contrast, spatial CNNs experience increased training times with larger kernel sizes. In the worst case scenario where the kernel is $H \times W$ the algorithmic complexity of performing the forward pass of a single kernel is $H \times W^2$ whereas with our approach the algorithmic complexity remains $H \times W$ via Fourier-based convolution. As such, while a performance gap continues to exists for small sized kernels our CNNs can efficiently scale to arbitrarily sized kernels.

However, CF-Conv performance still trails 3×3 spatial CNNs. This discrepancy suggests that CF-Convs may require more complex MLP architectures or further optimization to fully realize their potential. Moreover, the learning dynamics of spatial CNNs are well-understood and have been extensively validated empirically, whereas CF-Convs are still relatively new and may benefit from additional refinement and tuning.

We also examine the impact of different numbers of selected points on model performance, experimenting with 2^{12}, 2^{15}, and 2^{18} selected positions. The results indicate that smaller samples introduce more noise and provide less accurate gradient approximations. In contrast, using 2^{18} ($\sim$260,000) selected positions yields the best performance, striking an optimal balance between gradient approximation accuracy and memory usage. This configuration allows for stable and efficient training, while still fitting comfortably into GPU memory.

These results demonstrate that while CF-Convs have promising potential, further refinement is needed to match the performance of traditional spatial CNNs. Nonetheless, CF-Convs with sparse updates highlight the potential of Fourier-based continuous kernel methods, particularly for applications involving larger and more complex architectures.

7 Discussion

Our proposed method for scaling CF-Conv networks via sparse kernel updates addresses key challenges in memory utilization and training speed. By introducing sparse updates, we significantly reduce the memory required during training, making the method feasible for large-scale applications. However, the technique still has limitations. Each CF-Conv layer must store a stateful kernel variable with dimensions $H \times W \times C_{\text{in}} \times C_{\text{out}} \times 2$, accounting for the real and imaginary components. For models with many convolutional filters or large spatial dimensions, this can still become memory-intensive. To mitigate this, model-parallelism could be a viable strategy, where model weights are distributed across multiple GPUs or TPUs to alleviate memory constraints [15].

Another challenge is that applying pointwise activation functions directly in the Fourier domain leads to suboptimal performance. This requires that the inverse transform be utilized and the activation function applied in the spatial domain. Further processing requires transformation back to the Fourier domain. This occurs because both Fourier-domain convolution (pointwise multiplication) and pointwise activation functions operate independently on each frequency component, preventing interactions between different frequencies. However, such inter-frequency interactions may be crucial for capturing complex patterns.

To address this, we apply an inverse Fourier transform after each convolution and perform activations in the spatial domain. This allows pointwise non-linearities to act on spatial signals containing a mixture of frequencies, potentially enabling richer interactions. While this approach introduces additional computational overhead due to repeated FFT and IFFT operations, it preserves the efficiency of Fourier-based learning while leveraging spatial non-linearities effectively. Notably, spatial continuous kernels require an additional Fourier transform over the full generated kernel, which has dimensionality $H \times W \times C_{\text{in}} \times C_{\text{out}}$. In contrast, CF-Convs apply Fourier transforms only to the convolution result, which has a lower dimensionality of $H \times W \times C_{\text{out}}$, reducing computational cost. Therefore, a vital next step to improve the proposed method is to investigate how to approximate spatial activations directly in the Fourier domain.

The development of improved complex-valued activation functions could provide an alternative solution. With improved complex-valued non-linearities, it may become possible to directly apply activations in the Fourier domain without the need for intermediate transforms. This would allow the network to fully utilize phase and amplitude information, making the approach particularly appealing for domains like audio, radar, and sonar image processing, where such information is critical.

8 Conclusion

Our work introduces CF-Convs, a novel approach for learning continuous convolutional kernels in the Fourier domain. CF-Convs address three of the fundamental challenges of continuous convolutional filter learning: parameter efficiency, memory efficiency and training speed. While CNNs using our CF-Convs still lag behind their traditional spatial CNN counterparts, our novel training algorithm allows for CF-Convs to learn convolutional kernels of arbitrary size, making them a promising direction for larger-scale applications. Future works can easily expand upon this work as our code and experiments are fully open source.

References

1. Chen, T., Xu, B., Zhang, C., Guestrin, C.: Training deep nets with sublinear memory cost. arXiv preprint arXiv:1604.06174 (2016)
2. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 764–773 (2017). https://doi.org/10.1109/ICCV.2017.89
3. Elson, J., Douceur, J.J., Howell, J., Saul, J.: Asirra: A captcha that exploits interest-aligned manual image categorization. In: Proceedings of 14th ACM Conference on Computer and Communications Security (CCS). Association for Computing Machinery, Inc. (October 2007), https://www.microsoft.com/en-us/research/publication/
4. Jacobsen, J.H., Van Gemert, J., Lou, Z., Smeulders, A.W.M.: Structured receptive fields in cnns. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2610–2619 (2016). https://doi.org/10.1109/CVPR.2016.286
5. Knigge, D.M., Romero, D.W., Gu, A., Gavves, E., Bekkers, E.J., Tomczak, J.M., Hoogendoorn, M., jakob Sonke, J.: Modelling long range dependencies in nd: From task-specific to a general purpose CNN. In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=ZW5aK4yCRqU
6. Kumar, R., Purohit, M., Svitkina, Z., Vee, E., Wang, J.: Efficient rematerialization for deep networks. Advances in Neural Information Processing Systems **32** (2019)
7. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4460–4470 (2019)
8. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 165–174 (2019)
9. Pintea, S.L., Tömen, N., Goes, S.F., Loog, M., van Gemert, J.C.: Resolution learning in deep convolutional networks using scale-space theory. IEEE Transactions on Image Processing **30**, 8342–8353 (2021). https://doi.org/10.1109/TIP.2021.3115001
10. Romero, D.W., Bruintjes, R.J., Tomczak, J.M., Bekkers, E.J., Hoogendoorn, M., van Gemert, J.: Flexconv: Continuous kernel convolutions with differentiable

kernel sizes. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=3jooF27-0Wy

11. Romero, D.W., Kuzina, A., Bekkers, E.J., Tomczak, J.M., Hoogendoorn, M.: CKConv: Continuous kernel convolution for sequential data. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=8FhxBtXSl0
12. Romero, D.W., Lohit, S.: Learning partial equivariances from data. Advances in Neural Information Processing Systems **35**, 36466–36478 (2022)
13. Romero, D.W., Zeghidour, N.: DNArch: Learning convolutional neural architectures by backpropagation. In: ICML 2023 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators (2023), https://openreview.net/forum?id=rlQPdYh9JD
14. Scardapane, S., Van Vaerenbergh, S., Hussain, A., Uncini, A.: Complex-valued neural networks with nonparametric activation functions. IEEE Transactions on Emerging Topics in Computational Intelligence **4**(2), 140–150 (2020). https://doi.org/10.1109/TETCI.2018.2872600
15. Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., Catanzaro, B.: Megatron-lm: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv:1909.08053 (2019)
16. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Advances in neural information processing systems **33**, 7462–7473 (2020)
17. Tygert, M., Bruna, J., Chintala, S., LeCun, Y., Piantino, S., Szlam, A.: A mathematical motivation for complex-valued convolutional networks. Neural Computation **28**(5), 815–825 (2016). https://doi.org/10.1162/NECO_a_00824
18. Wood, L., Larson, E.C.: Parametric spectral filters for fast converging, scalable convolutional neural networks. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2800–2804. IEEE (2021)
19. Wood, L., Tan, Z., Stenbit, I., Bischof, J., Zhu, S., Chollet, F., et al.: Kerascv. https://github.com/keras-team/keras-cv (2022)