Fractional Fourier Transform Meets Transformer Encoder

Furkan Şahinuç and Aykut Koç, *Senior Member, IEEE*

***Abstract*—Utilizing signal processing tools in deep learning models has been drawing increasing attention. Fourier transform (FT), one of the most popular signal processing tools, is employed in many deep learning models. Transformer-based sequential input processing models have also started to make use of FT. In the existing FNet model, it is shown that replacing the attention layer, which is computationally expensive, with FT accelerates model training without sacrificing task performances significantly. We further improve this idea by introducing the fractional Fourier transform (FrFT) into the transformer architecture. As a parameterized transform with a fraction order, FrFT provides an opportunity to access any intermediate domain between time and frequency and find better-performing transformation domains. According to the needs of downstream tasks, a suitable fractional order can be used in our proposed model FrFNet. Our experiments on downstream tasks show that FrFNet leads to performance improvements over the ordinary FNet.[1]**

***Index Terms*—Encoder, FNet, Fourier Transform, Fractional Fourier Transform, Transformer.**

I. Introduction

THERE is a recent growing trend of integrating signal processing tools into machine learning models. Transforms from classical signal processing commonly appear in various deep neural network models. Especially, Fourier and wavelet transforms (FT and WT) are suitable tools for either processing the input data or being utilized as a part of model blocks [1]. It is well known that the relationship between convolution and FT enables models to implement calculations in the Fourier domain, particularly for convolutional neural networks (CNNs). WT is also utilized in imaging tasks [2], [3], [4], [5]. For instance, in [2], a fractional wavelet scattering network (FrScatNet) is introduced for the medical imaging classification task. Feature vectors are extracted by fractional wavelet transform (FrWT), and conventional machine learning algorithms train these features. In [3], authors develop a mathematical theory of FrScatNet and offer a rigorous mathematical model for the FrScatNet. Similarly, Dong et al. suggest utilizing a learnable scattering transform filter bank instead of traditional convolution filters to avoid a large number of parameters and the risk of overfitting [5]. In [4], the WT is integrated with CNNs by replacing the standard down-sampling part with the discrete wavelet transform (DWT) to obtain a more robust network against noise.

As signal processing tools contribute to deep learning enhancements, the opposite can also be frequently observed. For instance, in [6], transformer-based methods are used in light field image super-resolution tasks. For automatic speech recognition, the non-autoregressive transformer model is proposed to reduce the inference time of autoregressive transformers [7]. Tian et al. approach the automatic speech recognition problem from a different perspective. They propose a hybrid model by integrating autoregressive and non-autoregressive models [8]. Transformer-based models can also be adapted to skeleton-based action recognition tasks [9], [10].

One of the principal goals of the studies done at the intersection of signal processing and deep neural network models is the search for more sustainable and efficient models in terms of the computational costs of model training. Although deep learning dominates several fields with state-of-the-art performances, many question marks have accompanied this success regarding sustainability. The effects of the notion "the more model parameters, the more performance" can be observed especially in the popular state-of-the-art deep language models [11], [12], [13], [14], [15]. On the other hand, consciousness about the environmental effects of training deep learning models is gaining momentum [16], [17]. Increasing the number of model parameters also brings a large carbon footprint and high computational cost, which most communities in machine learning cannot easily manage [18]. This concern can limit the novel contributions due to difficulties in implementing experiments. Therefore, researchers work on alternative language modeling methods to obtain similar performances without extreme computational burden. In [19], all but one attention head is fixed for the neural machine translation task. It is shown that fixing attention heads does not change model performance. Similarly, there are studies utilizing fixed Gaussian distributions instead of self-attention heads. It is also observed that changing cross-attention more critically affects machine translation performance compared to self-attention [20]. There are similar studies in computer vision and image processing. Tolstikhin et al. propose to change attention layers in vision transformer [21] with multi-layer perceptron layers without significant performance losses in image classification [22].

FT is a frequently utilized tool deployed in deep learning because of its close relationship with convolution. Mathieu et al. propose a pointwise product in the Fourier domain instead of calculating convolutions to accelerate the training process [1].

Manuscript received 19 July 2022; revised 11 October 2022; accepted 25 October 2022. Date of publication 28 October 2022; date of current version 9 November 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ran Tao. *(Corresponding author: Aykut Koç.)*

Furkan Şahinuç is with the ASELSAN Research Center, 06200 Ankara, Turkey, and also with the Department of Electrical and Electronics Engineering, Bilkent University, 06800 Ankara, Turkey.

Aykut Koç is with the Department of Electrical and Electronics Engineering, Bilkent University, 06800 Ankara, Turkey, and also with the National Magnetic Resonance Research Center, Bilkent University, 06800 Ankara, Turkey (e-mail: aykut.koc@bilkent.edu.tr).

Digital Object Identifier 10.1109/LSP.2022.3217975

[1]Source codes are available at https://github.com/koc-lab/FrFNet

Similarly, in [23], training of the images is completely implemented in the Fourier domain resulting in efficient training. In [24], the Fourier domain is utilized differently by GFNet for the image classification task. Self-attention layer is replaced with 2D-FT, applying learnable filters and inverse 2D-FT. Filtering operations can also be applied in natural language processing (NLP) models. In [25], spectral filters are applied to neuron activations to capture better information in different language levels such as word, utterance, and document. Different levels of filtering are inspired by the traditional high/band/low-pass filters. Several other studies that benefit from Fourier features also exist [26], [27], [28], [29].

The breakthrough work in [30] integrates FT and transformer architecture to develop the FNet model, where the whole attention layer in the encoder transformer is replaced with FT layers. This modification leads to a significant acceleration in training, and performance degradation is relatively small. One of the main mechanisms of neural networks lies in the flow of information across layers and in their ability to make vast amounts of interconnections during that information flow. To this end, FT offers a straightforward and efficient alternative way of establishing connections across layers. Although the FNet model initiates a new direction; much space exists to enhance it.

In this letter, we introduce the FrFNet, a model where the tokens of the transformer are mixed with the fractional Fourier transform (FrFT) [31], [32], [33]. While FT stands for a single transformation from the time domain to the spectral domain, FrFT enables us to scan a family of transforms at a continuum of intermediate domains between time and frequency, controlled by a fractional order parameter. Being the direct generalization of FT, FrFT both inherits FT's power in the simplest possible way and provides an additional degree of freedom such that a continuum of alternative transformations (thus alternative information flows in the neural models) are accessible. The motivation of the FNet is to obtain a model with a parameter-free layer, a single linear transformation in this case, instead of an attention layer. We take this motivation one step further by introducing a layer capable of implementing infinitely many transformations for token-mixing, increasing model accuracy. Equally importantly, FrFT also has a well-established fast digital computation algorithm in $\sim N \log N$ time [33], which makes it possible to enjoy its advantages without paying for additional computational costs. The only cost is the addition of a single fraction order parameter as a hyperparameter, which is negligible compared to the number of parameters within transformers. Our main contributions are summarized in two folds:

- We introduce the FrFT to the transformer encoders and propose the FrFNet model.
- We obtain performance improvements on important NLP benchmarks by FrFNet without suffering additional computational costs compared to the FNet.

The organization of the manuscript is as follows. In Section II, preliminary information on FrFT and its related properties are given. Details of our proposed model are presented in Section III. In Section IV, experimental details, results, and discussions are provided. Finally, we conclude in Section V.

II. Fractional Fourier Transform

The FrFT is a generalized version of the ordinary FT [31], [32], [33] that can transform an input function to any intermediate domain between time and frequency domains, which allows us to make a more powerful spectral analysis. The fractional transformation order a ($a \in \mathbb{R}$) is the parameter of FrFT. ath order FrFT is the ath power of the ordinary FT and denoted by $\mathcal{F}^a$. $\mathcal{F}^1$ corresponds to the ordinary FT. $\mathcal{F}^0$ and $\mathcal{F}^2$ are equal to the identity and parity operators ($\mathcal{F}^2 = \mathcal{P} : \mathcal{P}\{f(u)\} = f(-u)$), respectively. In general, FrFT is index additive, that is the ath order FrFT of a' order FrFT equals to $\mathcal{F}^{a+a'}$.

The formal definition of the ath order FrFT $\mathcal{F}^a\{f(u)\}$ of the input function $f(u)$ is given as follows:

$$\mathcal{F}^a\{f(u)\} = \int_{-\infty}^{\infty} K_a(u,u')f(u')du,'$$
$$K_a(u,u') = A_\phi e^{i\pi(u^2 cot\phi - 2uu' csc\phi + u'^2 cot\phi)},$$
$$A_\phi = \sqrt{1 - icot\phi},\ \ \phi = a\pi/2, \tag{1}$$

when $a \neq 2k$ where k is any integer. The kernel $K_a(u,u')$ is defined as $\delta(u-u')$ for $a = 4\,k$ and $\delta(u+u')$ for $a = 4k+2$. Here, it can be inferred that FrFT is periodic with period 4. In other words, $\mathcal{F}^a = \mathcal{F}^{4k+a}$. Hence, the range of transform order can be taken as $(-2, 2]$ or [0,4). Along with index additivity property, $\mathcal{F}^3 = \mathcal{F}^{-1+4} = \mathcal{F}^{-1}$ corresponds to the inverse FT and $\mathcal{F}^2 = \mathcal{F}^{-2}$ equals to parity operator. The kernel $K_a(u,u')$ can be expressed with its spectral expansion:

$$K_a(u,u') = \sum_{k=1}^{\infty} \psi_k(u) e^{-i\frac{\pi}{2}ka} \psi_k(u'), \tag{2}$$

where $\psi_k(u)$ is the kth Hermite-Gaussian function.

Equation (1) is the underlying continuous domain definition. However, sequence processing is done in the discrete domain. To implement the fractional Fourier layer for the transformer, one needs to deploy the discrete version of FrFT. For a given sequence $\{x_0, x_1, \ldots..., x_{N-1}\}$ of length N, the ordinary discrete Fourier transform (DFT) is: $X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi i}{N}nk}$, where $k \in \{0, 1, \ldots., N-1\}$. DFT computation can also be implemented via multiplication of DFT matrix with input sequence and the DFT matrix W is $W[m,n] = e^{\frac{-2\pi i}{N}mn}/\sqrt{N}$, where $m, n \in \{0, 1, \ldots., N-1\}$. The discrete fractional Fourier transform (DFrFT) matrix can also be thought as discrete analogy of (2) [32]. The formal definition of DFrFT is:

$$F^a[m,n] = \sum_{k=0, k\neq(N-1+(N)_2)}^{N} u_k[m] e^{-i\frac{\pi}{2}ka} u_k[n], \tag{3}$$

where u_k is the kth discrete Hermite-Gaussian function. The difference in the summation range comes from eigenvector calculations in the process.

III. FrFNet: Fractional Fourier Transform Based Transformer Encoder

Conventional transformer encoder architecture consists of consecutive encoder blocks and input/output layers. Each encoder block includes an attention layer and a feed-forward layer with normalization stages after both. FNet removes the attention layer from every encoder block and deploys FT to mix the input tokens [30]. FT is efficiently implemented with the well-known fast Fourier transform (FFT) algorithm.

In the proposed FrFNet, we replace the attention blocks with FrFT blocks, where the discrete implementation is done

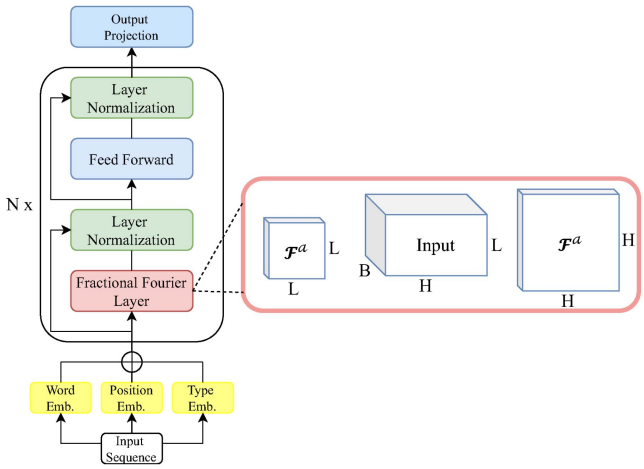

Fig. 1. The FrFNet architecture consists of N consecutive encoders. The FrFT layer is used instead of the attention layer. FrFT of input layers is calculated via DFrFT matrices. L and H stand for maximum sequence length and hidden dimension size, respectively. B indicates the batch size.

by DFrFT. The overall illustration of the proposed encoder architecture and FrFT layer is given in Fig. 1. Here, the input is represented as a batch of $L \times H$ matrices, where L represents the maximum sequence length (the number of allowed tokens) and H represents the hidden state dimension for each token. To compute the DFrFT in this 2D structure, we use two DFrFT matrices that correspond to calculating FrFTs along each dimension of the input (when the batch size is 1). Since 2D-DFrFT is separable, it can be calculated by multiplying a given input matrix from left and right (i.e, DFrFTs across rows and columns) with two DFrFT matrices given in (3) of size $L \times L$ and $H \times H$, respectively. The details of constructing DFrFT matrices and their efficient computation on the order of FFT can be found in [32] and [33], respectively.

In the attention mechanism of the transformer, input values are transformed into query (Q), key (K), and value (V) matrices [34]. Attention output is calculated as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4}$$

where d_k is the query and key dimension used for scaling purposes. To capture information from different positions of the input sequence, multi-head attention, which is the parallel computation of attention using different projections of the queries, keys, and values, is usually utilized. A single attention head is computed by projections as follows:

$$head_i = Attention(QW_i{}^Q, KW_i{}^K, VW_i{}^V), \tag{5}$$

where $W_i{}^Q, W_i{}^K, W_i{}^V$ are the projection matrices for queries, keys, and values, respectively. After all attention heads are computed in parallel, they are concatenated and subjected to a final linear transformation as follows [34]:

$$MultiHead(Q, K, V) = Conc(head_1, .., head_h)W^O, \tag{6}$$

where W^O is the matrix of the linear output projection. On the other hand, the FrFNet model does not require such calculations. First, DFrFT matrices are calculated for the given fractional order and dimension. Then, the 2D-FrFT of the input is calculated through matrix multiplications. Since DFrFT matrices are calculated once, there is no additional cost. In addition, there is no back-propagation cost for DFrFT, unlike the linear transformation of the attention mechanism, since it is a deterministic function. Mathematical representations of the FrFNet through an encoder layer are given as follows:

TABLE I
MODEL AND TRAINING PARAMETERS USED IN THE HYPERPARAMETER (HP) SEARCH AND PRE-TRAINING STAGES

Parameter Name	HP Search	Pre-training
Embedding dimension	256	768
Hidden dimension	256	768
Maximum sequence length	512	512
Model blocks	4	12
Learning rate	10^{-4}	10^{-4}
Training steps	5×10^4	10^6

$$X = \mathcal{F}^a\{x\}, \tag{7}$$

$$\hat{X} = \frac{(X + x) - \mathbb{E}[X + x]}{\sqrt{Var(X + x) + \epsilon}}\gamma_1 + \beta_1, \tag{8}$$

$$\hat{X}_{int} = \hat{X}\Phi(\hat{X}W_1 + b), \tag{9}$$

$$y = D(\hat{X}_{int}W_2), \tag{10}$$

$$\hat{y} = \frac{(y + \hat{X}) - \mathbb{E}[y + \hat{X}]}{\sqrt{Var(y + \hat{X}) + \epsilon}}\gamma_2 + \beta_2, \tag{11}$$

where x is either the input embeddings or the output of the previous encoder. γ_1, β_1, γ_2, and β_2 are learnable parameters of the layer normalizations. ϵ is used for numerical stability. W_1, W_2, and b are linear transformation parameters and bias in the feed-forward network. Φ is the cumulative distribution function for the Gaussian distribution. Finally, D represents the drop-out function.

The encoder's input is fed to the FrFT layer in (7). Then, layer normalization is applied to the sum of the FrFT output and the original input in (8). Two linear transformations are applied in the feed-forward layer given in (9) and (10). GELU activation ($x\Phi(x)$) is used between linear transformations [35]. The drop-out function is applied at the output of the feed-forward layer [36]. At the end of the encoder, second layer normalization is applied as in (11) to the sum of input and output of the feed-forward layer. At the end of the last encoder block, output projection is applied for output logits:

$$y_{final} = \hat{y}W_o + b_o, \tag{12}$$

where W_o and b_o are output projection and bias parameters.

It should be noted that the output of the DFT includes complex values. Since training is done over real values, the FNet model considers only the real parts of the output sequence. In FrFNet, we maintain this convention. In addition, DFT has the conjugate symmetry property. In 2D matrices, where M and N are dimensions, $X[M - u, N - v] = X^*[u, v]$. The FNet model ignores this property and makes redundant calculations due to repeated real values. However, conjugate symmetry does not exist in fractional transform unless $a = 1$. Therefore, the FrFNet model does not implement any redundant calculations.

TABLE II
GLUE VALIDATION RESULTS. FOR EACH TASK, EXPERIMENTS ARE REPEATED THREE TIMES

Model	SST-2	QQP	STS-B	MNLI-M	MNLI-MM	QNLI	RTE	MRPC	Avg.
FNet	0.80	0.83	0.72	0.69	0.70	0.77	**0.56**	0.73	0.725
FrFNet- (-1)	0.82	**0.84**	0.73	**0.70**	**0.71**	**0.81**	**0.56**	**0.76**	0.741
FrFNet- 0.750	0.81	0.81	0.51	0.65	0.66	0.74	0.48	0.68	0.668
FrFNet- 0.994	**0.83**	0.82	**0.74**	0.68	0.70	**0.81**	**0.56**	0.74	0.735
FrFNet- 1.208	0.81	0.81	0.32	0.66	0.68	0.75	0.51	0.67	0.651
FrFNet- 1.998	0.74	0.75	0.12	0.57	0.57	0.62	0.46	0.67	0.562

Average accuracies are reported except for STS-B (Spearman correlation). The best scores for each task are emboldened. MNLI-M and MNLI-MM indicate matched and mismatched versions of the MNLI dataset.

IV. EXPERIMENTS & RESULTS

Training a transformer encoder model consists of two main stages: pre-training and fine-tuning [12]. The model is trained on the unlabeled corpus in the pre-training to learn general features of the input sequence (i.e., natural language text). It consists of two objectives: masking language modeling (MLM) and next sentence prediction (NSP). On the other hand, fine-tuning is a task-dependent training such as natural language inference, sentence similarity, or textual entailment. While training the proposed model, we follow the same pre-training and fine-tuning procedures of BERT [12].

We deploy FrFNet in the GLUE benchmark [37]. The fractional order of FrFNet is to be determined before pre-training and fine-tuning experiments. We implement a hyperparameter (HP) search to calculate the optimal order. Then, the model is pre-trained and fine-tuned with the optimal fractional order. Model and training parameters used in HP search and pre-training stages are given in Table I.

A. Determining the Fraction Order of FrFT

An objective function should be maximized or minimized in the HP search for the fraction order. We search for the best fraction order in our setup by maximizing MLM accuracy. Although NSP is very beneficial for natural language inference tasks, it is easier than MLM due to its binary nature. Therefore MLM accuracy is more distinctive than NSP in evaluating fractional order performance. Furthermore, MLM provides more stabilized results than the NSP objective. In other words, NSP accuracy fluctuates more than MLM between training steps. There are also studies showing that MLM is an essential objective in pre-training [38].

The HP search is done in the pre-training stage, and the pre-training is quite a computationally costly process. To alleviate this cost, we conduct our search in a smaller and established model whose size is similar to BERT-mini [12]. Optuna library is utilized in HP [39]. The models are trained on the English Wikipedia dump provided by Tensorflow Datasets [40]. 50 experiments are implemented with fraction values between 0-2. Experiments are implemented on four GTX1080 Ti GPUs. Fraction orders around 0.75, 0.99, 1.20, -1, and 1.99 give the best MLM scores on the validation set.

B. Pre-Training and Fine-Tuning Experiments

During pre-training, the models are trained with the selected fraction values on the same corpus. Since those models will be our main models and be evaluated on downstream tasks, the BERT-base model size is used in experiments.

After pre-training, model performance is evaluated by fine-tuning the pre-trained model for downstream tasks. To this end, GLUE benchmark is used [37]. This benchmark includes sentiment analysis (SST-2), textual similarity (STS-B), paraphrasing (MRPC), semantic equivalence (QQP), natural language inference (MNLI, QNLI), and textual entailment (RTE) datasets. In Table II, model performances on validation splits of GLUE tasks are given, where we compare FrFNet models that are trained by using fractions $a \in \{-1, 0.750, 0.994, 1.208, 1.998\}$ with the FNet that uses the ordinary FT ($a = 1$). The models trained with fractions -1 and 0.994 obtain the best performance in almost every dataset. Furthermore, Table II shows that using suitable fractions for the training can perform better than FNet. Therefore, task-specific configurations can be implemented without high computational costs to acquire competitive performances compared to models that use attention mechanisms. Another advantage of FrFNet is that model construction can be more flexible. For example, the best fractional value for each encoder can individually be determined according to the target task.

V. CONCLUSION

We integrate the FrFT into transformer encoder models for sequence processing in the context of language processing. Fractional orders as a degree of freedom in FT means that one can deploy a family of transformations with infinitely many members. Combining such a flexible and powerful tool with state-of-the-art deep learning models leads to performance improvements compared to the ordinary FT. Our experimental results show that using FrFT in pre-training and fine-tuning improves performance in most NLP downstream tasks. We believe that introducing FrFT to transformer encoders stimulates further developments where signal processing meets deep neural networks. There are also many powerful time-frequency analysis tools other than FrFT, and integrating them into deep learning models is a possible future research direction. Furthermore, instead of hyperparameter searching for the FrFT order, learning the optimal fractional order during the training or examining the usage of the different fractional values in consecutive encoders are also immediate future works.

REFERENCES

[1] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014.

[2] L. Liu, J. Wu, D. Li, L. Senhadji, and H. Shu, "Fractional wavelet scattering network and applications," *IEEE Trans. Biomed. Eng.*, vol. 66, no. 2, pp. 553–563, Feb. 2019.

[3] J. Shi, Y. Zhao, W. Xiang, V. Monga, X. Liu, and R. Tao, "Deep scattering network with fractional wavelet transform," *IEEE Trans. Signal Process.*, vol. 69, pp. 4740–4757, 2021.

[4] Q. Li, L. Shen, S. Guo, and Z. Lai, "WaveCNet: Wavelet integrated CNNs to suppress aliasing effect for noise-robust image classification," *IEEE Trans. Image Process.*, vol. 30, pp. 7074–7089, 2021.

[5] Z. Dong, R. Zhang, X. Shao, and Z. Kuang, "Learning sparse features with lightweight ScatterNet for small sample training," *Knowl.-Based Syst.*, vol. 205, 2020, Art. no. 106315.

[6] Z. Liang, Y. Wang, L. Wang, J. Yang, and S. Zhou, "Light field image super-resolution with transformers," *IEEE Signal Process. Lett.*, vol. 29, pp. 563–567, 2022.

[7] N. Chen, S. Watanabe, J. Villalba, P. Żelasko, and N. Dehak, "Non-autoregressive transformer for speech recognition," *IEEE Signal Process. Lett.*, vol. 28, pp. 121–125, 2021.

[8] Z. Tian, J. Yi, J. Tao, S. Zhang, and Z. Wen, "Hybrid autoregressive and non-autoregressive transformer models for speech recognition," *IEEE Signal Process. Lett.*, vol. 29, pp. 762–766, 2022.

[9] J. Gao, T. He, X. Zhou, and S. Ge, "Skeleton-based action recognition with focusing-diffusion graph convolutional networks," *IEEE Signal Process. Lett.*, vol. 28, pp. 2058–2062, 2021.

[10] J. Kong, Y. Bian, and M. Jiang, "MTT: Multi-scale temporal transformer for skeleton-based action recognition," *IEEE Signal Process. Lett.*, vol. 29, pp. 528–532, 2022.

[11] M. E. Peters et al., "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2018, pp. 2227–2237.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 4171–4186.

[13] A. Radford et al., "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, pp. 1–24, 2019.

[14] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[15] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 1877–1901.

[16] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *J. Mach. Learn. Res.*, vol. 21, no. 248, pp. 1–43, 2020.

[17] D. Patterson et al., "The carbon footprint of machine learning training will plateau, then shrink," *Computer*, vol. 55, no. 7, pp. 18–28, 2022.

[18] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 3645–3650.

[19] A. Raganato, Y. Scherrer, and J. Tiedemann, "Fixed encoder self-attention patterns in transformer-based machine translation," in *Proc. Findings Assoc. Comput. Linguistics: Empirical Methods Natural Lang. Process.*, 2020, pp. 556–568.

[20] W. You, S. Sun, and M. Iyyer, "Hard-coded Gaussian attention for neural machine translation," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 7689–7700.

[21] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2021.

[22] I. O. Tolstikhin et al., "MLP-Mixer: An all-MLP architecture for vision," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 24261–24272.

[23] H. Pratt, B. Williams, F. Coenen, and Y. Zheng, "FCNN: Fourier convolutional neural networks," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2017, pp. 786–798.

[24] Y. Rao, W. Zhao, Z. Zhu, J. Lu, and J. Zhou, "Global filter networks for image classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 980–993.

[25] A. Tamkin, D. Jurafsky, and N. Goodman, "Language through a prism: A spectral approach for multiscale language representations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 5492–5504.

[26] H. M. El-Bakry and Q. Zhao, "Fast object/face detection using neural networks and fast Fourier transform," *Int. J. Signal Process.*, vol. 1, no. 3, pp. 182–187, 2004.

[27] T. Highlander and A. Rodriguez, "Very efficient training of convolutional neural networks using fast Fourier transform and overlap-and-add," in *Proc. Brit. Mach. Vis. Conf.*, 2015, pp. 1–9.

[28] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, "Fast convolutional nets with fbfft: A. GPU performance evaluation," in *Proc. Int. Conf. Learn. Representations*, 2015.

[29] J. Ryu, M.-H. Yang, and J. Lim, "DFT-based transformation invariant pooling layer for visual classification," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 84–99.

[30] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "FNet: Mixing tokens with fourier transforms," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2022, pp. 4296–4313.

[31] H. M. Ozaktas, Z. Zalevsky, and M. A. Kutay, *The Fractional Fourier Transform With Applications in Optics and Signal Processing*. Hoboken, NJ, USA:Wiley, 2001.

[32] C. Candan, M. Kutay, and H. Ozaktas, "The discrete fractional Fourier transform," *IEEE Trans. Signal Process.*, vol. 48, no. 5, pp. 1329–1337, May 2000.

[33] H. M. Ozaktas, O. Arikan, M. A. Kutay, and G. Bozdagi, "Digital computation of the fractional Fourier transform," *IEEE Trans. Signal Process.*, vol. 44, no. 9, pp. 2141–2150, Sep. 1996.

[34] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.

[35] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

[37] A. Wang, A. Singh, J. Michael, F. Hill, and O. Levy, "GLUE: A multi-task benchmark and analysis platform for natural language understanand bowman, samuel," in *Proc. EMNLP Workshop BlackboxNLP: Analyzing Interpreting Neural Netw. NLP*, 2018, pp. 353–355.

[38] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[39] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2623–2631.

[40] M. Abadi et al., T. Developers, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.