DEEP FRACTIONAL FOURIER NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Emirhan Koç
August 2024

Deep Fractional Fourier Networks
By Emirhan Koç
August 2024

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Aykut Koç(Advisor)

Haldun M. Özaktaş

S. Figen Öktem Seven

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

DEEP FRACTIONAL FOURIER NETWORKS

Emirhan Koç
M.S. in Electrical and Electronics Engineering
Advisor: Aykut Koç
August 2024

This thesis introduces the integration of the fractional Fourier Transform (FrFT) into the deep learning domain, with the aim of opening new avenues for incorporating signal processing into deep neural networks (DNNs). This work starts by introducing FrFT into recurrent neural networks (RNNs) for time series prediction, leveraging its ability and flexibility to perform infinitely many continuous transformations and offering an alternative to the traditional Fourier Transform (FT). Despite the initial success, a significant challenge identified is the manual tuning of the fraction order parameter a, which can be cumbersome and limits broader applicability. To overcome this limitation, we introduce a novel approach where the fraction order a is treated as a learnable parameter within deep learning models. First, a theoretical foundation is established to support the learnability of this parameter, followed by extensive experimentation in image classification and time series prediction tasks. The results demonstrate that incorporating a learnable fraction order significantly improves model performance, particularly when integrated with well-known architectures such as ResNet and VGG models. Furthermore, the thesis proposes fractional Fourier Pooling (FrFP), a pooling technique that replaces traditional Global Average Pooling (GAP) layers in Convolutional Neural Networks (CNNs). FrFP enhances feature representation by processing intermediate signal regions, leading to better model performance and offering a new perspective on integrating signal transformations within deep learning frameworks. Overall, this thesis contributes to the growing body of research exploring advanced signal processing techniques in deep learning, highlighting the potential of FrFT as a powerful tool for improving model accuracy and efficiency across various applications.

Keywords: Fractional Fourier Transform, signal processing in deep learning, image processing, time series prediction, deep neural networks, convolutional neural networks, recurrent neural networks.

ÖZET

DERİN KESİRLİ FOURIER AĞLARI

Emirhan Koç
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Aykut Koç
Ağustos 2024

Bu tez, kesirli Fourier Dönüşümü'nün (KFD) derin öğrenme alanına entegrasyonunu tanıtarak, sinyal işlemenin derin sinir ağlarlarıyla (DSA'lar) işbirliğine yeni yollar açmayı amaçlamaktadır. Bu çalışma, geleneksel Fourier Dönüşümüne (FD) bir alternatif sunan kesirli Fourier Dönüşümünün sonsuz sayıda sürekli dönüşüm gerçekleştirme esnekliği ve yeteneğinden yararlanarak, KDF'yi zaman serisi tahmini için mükerrer sinir ağlarına (MSA'lar) entegre ederek başlar. İlk başarıya rağmen, tespit edilen önemli bir zorluk, kesir parametresi a'nın elle (manuel) ayarının zahmetli olması ve daha geniş uygulanabilirliği sınırlayabilmesidir. Bu sınırlamayı aşmak için, kesir parametresi a'nın derin öğrenme modellerinde öğrenilebilir bir parametre olarak ele alındığı yeni bir yaklaşım da sunuyoruz. İlk olarak, bu parametrenin öğrenilebilirliğini destekleyen teorik bir temel oluşturulmuş, ardından görüntü sınıflandırma ve zaman serisi tahmini görevlerinde kapsamlı deneyler yapılmıştır. Sonuçlar, öğrenilebilir bir kesir parametresinin dahil edilmesinin, özellikle ResNet ve VGG modelleri gibi iyi bilinen mimarilerle entegre edildiğinde, model performansını önemli ölçüde artırdığını göstermektedir. Ayrıca, tez, geleneksel küresel ortalama havuzlama (KOH) katmanlarını evrişimsel sinir ağlarında (ESA'lar) değiştiren yeni bir havuzlama tekniği olan kesirli Fourier Havuzlaması'nı da (KFH) önermektedir. KFH, ara sinyal bölgelerini işleyerek özellik temsiliyetini geliştirir, bu da model performansını artırır ve sinyal dönüşümlerinin derin öğrenme yapılarına nasıl entegre edilebileceğine dair yeni bir bakış açısı sunar. Genel olarak, bu tez, derin öğrenmede ileri sinyal işleme teknikleri konusunda ilerleyen araştırmalara katkıda bulunarak, KFD'nin çeşitli uygulamalarda model doğruluğunu ve verimliliğini artırmak için güçlü bir araç olma potansiyelini vurgulamaktadır.

Anahtar sözcükler: Kesirli Fourier dönüşümü, derin öğrenmede sinyal işleme, görüntü işleme, zaman serileri tahmini, derin sinir ağları, konvolüsyonel sinir ağları, mükerrer sinir ağları.

Acknowledgement

First of all, I would like to express my sincere gratitude to my advisor, Asst. Prof. Aykut Koç, for his guidance and support throughout my master's studies. His mentorship has significantly contributed to my development as a researcher and a person. The skills and knowledge I have gained under his supervision will be valuable throughout my academic and professional life.

I would also like to thank the rest of my thesis committee: Prof. Haldun Özaktaş and Assoc. Prof. Figen Öktem, for their time and valuable feedbacks.

I thank Enes Erkoç for the countless memories we have shared for the last ten years across different parts of the country will always remain unforgettable. I believe that more of them will be added to them, maybe in different countries. Alp Güven, I miss our deep conversations on science, philosophy, history, and other intellectual topics. I also would like to thank Sami Yavuz, Alp Özalp, and Osman Berke Güney for their help and friendships from the days at Sabancı.

I would like to thank our lab members: Tuna Alikaşifoğlu, Arda Can Aras, Nurullah Sevim, İrem Şanlı, Eralp Kumbasar, Tuna Özateş, Yunus Emre Ekiz and Emre Kulkul.

I also would like to thank to friends at Bilkent. Atakan Topçu, H. Atakan Bedel, M. Hasan Kayapınar, M. Emin Öztürk. It was truly a pleasure for me to meet you. We made so many great memories this past year. I would also like to express my deepest gratitude to Ecem Şimşek for helping me to get to know Bilkent and Ankara better. Tunalı and Bahçeli days, one-day trips to Eskişehir —all of them were amazing, and I cannot wait to do it all again. I enjoy discussing new papers and methods with you, especially in NLP, graphs, and various areas of deep learning. I believe there are many papers ahead where we will be co-authors.

I would like to thank Turkcell Communication Services Inc. for their support under the 5G and Beyond Graduate Scholarship Programme.

Last but not least, I want to thank my family and relatives. A special thanks to my mother, Sadegül, and my sister, Ayşegül, for always believing in me. Their unwavering support has been my backbone, and without them, none of my accomplishments would have been possible. I cannot forget to thank my first-ever friend, Zeki, for his endless brothership since childhood.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning approaches based on neural networks are extensively employed in various domains, ranging from time series and sequence prediction to image processing, computer vision, and natural language processing (NLP) [1–7]. Although deep neural networks (DNNs) offer increased performance through sophisticated algorithms, there remains potential for further advancements by incorporating traditional signal processing techniques and concepts. Recently, there has been a growing tendency toward developing DNN models that are grounded in more concrete theoretical and analytical foundations. To this end, the integration of classical signal processing methods, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), has gained popularity and led to prominent improvements in performance. [8–14].

In [10], wavelet scattering transforms are integrated with CNNs to tackle text-independent speaker identification. The features extracted through wavelet scattering are employed as the initial layer in the CNNs, with the remaining layers trained under supervision. The experiments demonstrated that wavelet scattering transforms provide effective feature extraction for speaker identification tasks. In [9], a fractional wavelet scattering network (FrScatNet) is applied to the gland segmentation task on colon histology images. Features are extracted

via fractional wavelet transform and subsequently used to train CNNs. [11] develops a mathematical model for FrScatNet, achieving superior performance in image classification tasks compared to traditional CNNs. In [12], the authors introduce empirical wavelet transforms based on Fourier-Bessel series expansion (FBSE) for enhanced time-frequency representations (TFR) of non-stationary signals. Inspired by [12], Gupta et al. [13] proposed a novel TFR method leveraging Fourier-Bessel decomposition for sleep stage classification using CNNs. In [14], a new TFR approach is introduced, utilizing improved eigenvalue decomposition of the Hankel matrix and Hilbert transform (IEVDHM–HT) for the classification of epileptic seizures using least-square support vector machines (SVM).

Fourier Transform (FT) is also deployed in machine learning mostly to reach computational accelerations. In [15–17], the convolution property of FT is leveraged such that convolution operations in layers of DNNs are replaced with Hadamard point-wise products in FT domain [17]. Since convolutions are computationally costly, the proposed approach enables significant speed-up without considerable degradation in model performance. [18] proposed a novel convolution operator named fast Fourier convolution that examines the ensemble of local and non-local receptive fields in a convolutional layer. [19] showed that discrete Fourier transform (DFT) based pooling layer enhances translation invariance and shape-preserving properties in CNN architectures whose average/max-pooling layers are replaced with the proposed DFT-pooling operation. In addition to these frequency domain methods that aim to accelerate the model training process, techniques such as average pooling and maximum pooling are also used to reduce the size of the processed image and handle it more effectively. However, although these methods increase efficiency, they can lead to significant information loss due to the reduction in image size, resulting in a decline in model performance. To prevent significant information loss and significantly improve model performance, other methods have been proposed. Xu et al. [20] demonstrated that learning in the frequency domain better preserves image information during preprocessing and achieves higher accuracy compared to traditional pooling techniques. The FcaNet [21] study mathematically showed that traditional global average pooling (GAP) is a special case of feature separation in the

frequency domain and proposed a method incorporating multi-spectral channel attention, which improved model performance in image processing tasks without significantly increasing computational costs.

Similar to the image domain, signal processing tools also enhance the capability of NLP applications. The seminal work FNet [22] shows that substituting self-attention layers in transformer encoder [23] with the ordinary, parameter-free Fourier transform (FT) considerably accelerates the training with limited performance degradation. [24] proposes the introduction of FT into autoencoders to detect anomalies with less noisy features. The proposed model yields competitive performances with the state-of-the-art. FT is also used in RNNs to address vanishing/exploding gradient problems and learn hidden-state information [25]. Several other methods incorporates FT in the DNN models [26–31]

The fractional Fourier transform (FrFT) generalizes the standard FT with a fraction order parameter a that controls the degree of transformation in the time-frequency plane enabling representations of signals in a continuum of infinitely many intermediate domains between time and frequency [32–37]. Thus, FrFT emerges as a powerful tool with diverse application areas ranging from signal processing [38], [39] to optical systems [40–42]. Being the direct generalization of FT, FrFT both inherits FT's power and provides an additional degree of freedom such that a continuum of alternative transformations (and thus information flow or feature extraction alternative in the neural network models) are accessible. 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 single parameter. The parametric nature of FrFT allows us to map a signal into infinitely many signal domains and presents a powerful signal analysis tool. In other words, FrFT enables us to select the most suitable domains that facilitate the analysis and operations on the signal. This degree of freedom and capability of FrFT motivates us to replace FT to select the most representative feature domain that works best in the underlying signal. It is important to note that discrete-time signals can be represented in fractional Fourier domains using a discrete version of FrFT, and there are fast digital computation algorithms [43] with $\mathcal{O}(N \log N)$

time complexity. Analogous to the matrix-vector multiplication in discrete FT (DFT) computation, discrete FrFT (DFrFT) can be calculated by multiplying the sequence vector with the DFrFT matrix, parameterized with fraction order a.

These benefits drive the introduction of FrFT/DFrFT to DNNs, where the underlying models are effectively processed with the most representative signal domain. In [44], authors take the FNet [22] one step further by introducing FrFNet, a model where the FrFT is employed as a token mixer. They improve the FNet on several NLP benchmarks without bearing additional model complexity. In a different application domain, [34] also proposed the FrIT model where self-attention blocks in vision transformer (VIT) [45] are substituted with FrFT to extract both local and global contexts. Similar to [44], FrFIT outperforms the VIT in multimodal remote sensing data classification tasks. In [46], FrFT is employed to extract spectral features in hyperspectral anomaly detection applications.

In previous studies, while efforts were made to find a suitable fractional order parameter, the values were either not fully reported or confined to a narrow range. The main purpose of this thesis is to explore the optimal FrFT parameter a in the context of its application to DNN architectures. To achieve this objective, three subsequent studies are proposed. The first study involves an extensive parameter search, treating the fraction order a as a hyperparameter. The second study defines the fraction order a as a learnable parameter, training the networks to optimize this parameter by maximizing their performance. Motivated by these two studies, FrFT is applied to image classification task as a trainable maximum pooling layer.

This thesis is based on the two published journal articles and one conference proceeding: "Fractional Fourier Transform in Time Series Prediction" [47] and "Trainable Fractional Fourier Transform" [48], "Maximally Selective Fractional Fourier Poooling" [49], which are the throughputs of the works conducted during M.Sc. education.

In this thesis, we first introduce FrFT to the machine learning domain by combining it with RNNs for the time series prediction problem with the motivation of utilizing a generalized transform capable of implementing infinitely many transformations to increase model performance. We present a model where FrFT is applied to each segment of windowed time-series signals in the time domain by replacing STFT-based sequence predictions with fractional orders without changing the number of computation steps. In other words, FrFT is applied as equal as the FT in terms of number of steps. We show that FrFT-based RNN models give better prediction performances compared to those based on either time and frequency domain. Then, we broaden the scope of utilization of FrFT in the context of deep learning and propose the fraction order a as a *learnable* parameter. Thus, we alleviate the need to search for suitable a and mitigate the shortcomings of the prevailing studies, which require an extensive tuning process to reach optimal a. This implies that, through the backpropagation in the network training process, the FrFT layers can adapt and optimize the fraction order parameter, thereby eliminating the need for manual search in determining the proper a value. To solidify our approach theoretically, we establish a rigorous mathematical formulation showing that the fraction order a can be taken into account of parameter update through gradient descent in network training. To empirically validate the efficacy of our proposed theory, we also report experimental results from an extensive and diverse set of tasks, encompassing image classification and time-series prediction. In the last study, we present a technique named fractional Fourier Pooling (FrFP), which is based on the FrFT. In this approach, we substitute the Global Average Pooling (GAP) layers that follow the final convolutional layer in the models with our FrFP layer. This allows for the processing of information within intermediate signal regions defined by the fractional order a.

The remainder of this thesis is structured as follows. Chapter 2 delves into the theoretical aspects of FrFT. Chapters 3, Chapter 4, and Chapter 5 present the proposed methods and their respective performance results. Finally, Chapter 6 discusses the findings and concludes the thesis.

Chapter 2

Fractional Fourier Transform

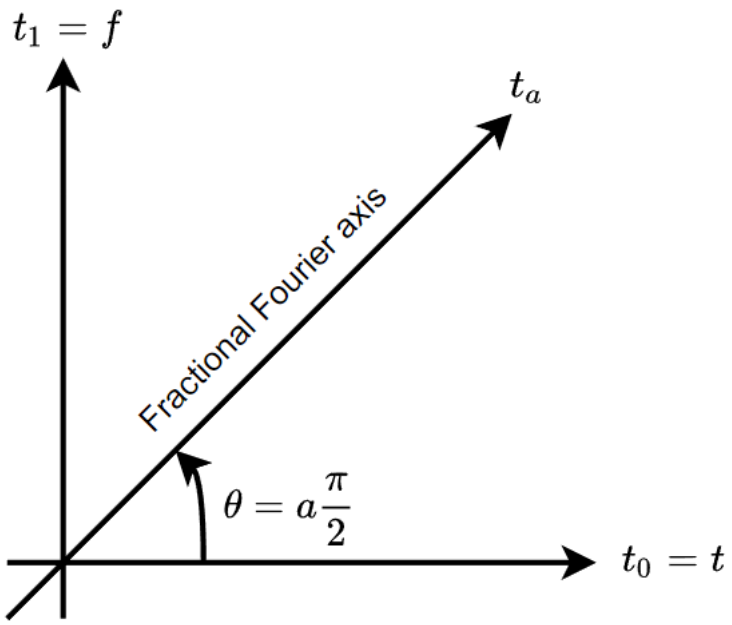

Figure 2.1: FrFT is a generalized version of ordinary FT, providing a unified and intermediate perspective between the time and frequency axis.

FrFT is the generalized version of ordinary Fourier Transform (FT) with a fraction order a that controls the rotation angle θ in the time-frequency plane and enables us to observe a signal in the intermediate domains between time and frequency. The FrFT provides a more versatile analysis tool than the FT,

allowing a better understanding of signals with time-varying or non-stationary characteristics. Its ability to smoothly transition between the time and frequency domains offers valuable insights into the time-frequency behavior of signals. Its flexibility in balancing time and frequency localizations makes it a powerful tool for analyzing and processing signals in various domains. It has applications in a wide range of signal-processing tasks ranging from optical signal processing to communication systems.

As illustrated in the 2.1, the time representation of the signal corresponds to the values on the line passing through the time axis. In contrast, the frequency distribution corresponds to values on the line passing through the frequency axis of the time-frequency plane. The FrFT allows us to observe the representation of these signals in arbitrary intermediate domains between time and frequency. That also corresponds to observing the values in an arbitrary line passing through the origin of the time-frequency plane.

2.1 Formal Definition

The most direct and concrete definition of a–order FrFT for a signal $f(t) \in \mathcal{L}^2(\mathbb{R})$ is provided in *linear integral form* as follows:

$$\begin{aligned} \mathcal{F}_a(u) &= \mathcal{F}^a\{f(t)\}(u) = \int_{-\infty}^{\infty} K_a(u,t) f(t) dt, \\ A_\phi &= \sqrt{1 - j\cot\phi}, \quad \phi = a\pi/2, \\ K_a(u,t) &= \begin{cases} \delta(u-t), & a = 4k \\ \delta(u+t), & a = 4k+2 \\ A_\phi \exp\left(j\pi(u^2\cot\phi - 2ut\csc\phi + t^2\cot\phi)\right), & \text{otherwise,} \end{cases} \end{aligned} \tag{2.1}$$

where $a \in \mathbb{R}$ and $k \in \mathbb{Z}$ [50–54]. Apart from the continuous-time version, DFrFT is also established to transform discrete-time signals into fractional Fourier domains. Similar to the well-defined computation of DFT with matrix-vector multiplication, DFrFT can also be represented as a matrix-vector multiplication [43,55],

where each element of the order-a DFrFT matrix, $\mathbf{F}^a$, is defined as:

$$\mathbf{F}^a[m,n] = \sum_{\substack{k=0 \\ k\neq N-1+(N)_2}}^{N} \boldsymbol{u}_k[m] \exp\left(-j\frac{a\pi}{2}k\right) \boldsymbol{u}_k[n], \tag{2.2}$$

where $(N)_2 \equiv N \mod 2$, $\boldsymbol{u}_k$ is the k^{th} discrete Hermite-Gaussian vector whose derivation thoroughly provided in [2]. Note that the iterated index k takes the values such that $k \in \mathcal{M} \triangleq \{0, 1, \ldots, N-2, N-(N)_2\}$, i.e., the final value of k is $N-(N)_2$ which is $N-1$ if N is odd, and N if it is even. The DFrFT of a discrete-time signal $\mathbf{X} \in \mathbb{R}^N = \{x_1, x_2, \ldots x_N\}$ is formally defined as follows:

$$X_{F^a} = \mathbf{F}^a\mathbf{X}, \tag{2.3}$$

where N is the sequence length and $X_{F^a} \in \mathbb{R}^N$ is the transformed signal in FrFT domain rotated by $\theta{=}a\frac{\pi}{2}$.

2.2 Derivative of the DFrFT Matrix

We provide a derivative manifestation to the discrete case with matrix-scalar differentiation. The derivative of DFrFT matrix Eq. (2.2) with respect to the order a can be computed in element-wise form. Derivation of a matrix with a scalar produces a matrix with the same size (using numerator notation), where the elements of the new matrix are element-wise derivatives of the original matrix with respect to the scalar. The derivative of the DFrFT matrix is denoted as, $\dot{\mathbf{F}}^a[m,n]$, and provided as the following:

$$\begin{aligned}\dot{\mathbf{F}}^a[m,n] \triangleq \frac{\partial}{\partial a}\mathbf{F}^a[m,n] &= \frac{\partial}{\partial a}\sum_{k\in\mathcal{M}} \boldsymbol{u}_k[m] \exp\left(-j\frac{a\pi}{2}k\right) \boldsymbol{u}_k[n] \\ &= -j\frac{\pi}{2}\sum_{k\in\mathcal{M}} k\boldsymbol{u}_k[m] \exp\left(-j\frac{a\pi}{2}k\right) \boldsymbol{u}_k[n]. \end{aligned} \tag{2.4}$$

Element-wise generation of the matrices $\mathbf{F}^a[m,n]$ and $\dot{\mathbf{F}}^a[m,n]$ can be tedious, therefore we can vectorize Eqs. (2.2) and (2.4). First, we define some basic

vectors to help us in the vectorization process.

$$\boldsymbol{\kappa}_N \triangleq \begin{bmatrix} 0 & 1 & \cdots & N-2 & N-(N)_2 \end{bmatrix}^\top, \tag{2.5}$$

$$\boldsymbol{\lambda}_N^{(a)} \triangleq \begin{bmatrix} \lambda_0^a & \lambda_1^a & \cdots & \lambda_{N-2}^a & \lambda_{N-(N)_2}^a \end{bmatrix}^\top, \tag{2.6}$$

where $\lambda_k = \exp\left(-j\frac{\pi}{2}k\right)$. The derivative of a vector with respect to a scalar is also a vector with an element-wise derivative. Then, by taking the derivative of Eq. (2.6) with respect to the order a, we have the following:

$$\begin{aligned} \dot{\boldsymbol{\lambda}}_N^{(a)} \triangleq \frac{\partial \boldsymbol{\lambda}_N^{(a)}}{\partial a} &= -j\frac{\pi}{2}[k\lambda_k^a]_{k\in\mathcal{M}} \\ &= -j\frac{\pi}{2}\begin{bmatrix} 0 & \lambda_1^a & \cdots & (N-(N)_2)\lambda_{N-(N)_2}^a \end{bmatrix}^\top \\ &= -j\frac{\pi}{2}\left(\boldsymbol{\kappa} \odot \boldsymbol{\lambda}_N^{(a)}\right), \end{aligned} \tag{2.7}$$

where $\odot$ operator is the Hadamard (element-wise) product. At this stage, we can directly generate the derivative matrix $\dot{\mathbf{F}}^a$ by an addition to the last step of the algorithm provided in [56] such that the eigenvalues relies in $\boldsymbol{\lambda}_N^{(a)}$ are replaced with the ones in $\dot{\boldsymbol{\lambda}}_N^{(a)}$.

We should indicate that the differentiability of FrFT/DFrFT is the initial step in discussing and establishing its integrability into neural networks. In Chapter Chapter 4, we use these derivations to justify our approach to constructing trainable layers within network architectures.

Chapter 3

Fractional Fourier Transform in Time Series Prediction

The content of this chapter reflects the work described in the following publication:

- E. Koç and A. Koç, "Fractional Fourier Transform in Time Series Prediction," IEEE Signal Processing Letters, vol. 29, pp. 2542–2546, 2022..

In this chapter, we introduce the FrFT into time series prediction with sequential models including RNNs and gated recurrent units (GRU). We compare our results with the baseline time domain and standard FT-based methods. We demonstrated the effectiveness of our approach with several experiments conducted on non-domain specific time series data from a wide range of areas, such as nonlinear differential equations, finance, and electric energy consumption.

3.1 Time Series Prediction Using Fractional Fourier Representations

Our proposed method is structured into two main stages. The first stage, feature extraction, involves a sequence of cascaded operations designed to produce feature vectors from time series data. Likewise the short time Fourier transform (STFT) [57, 58], any univariate data sequence $\mathcal{X}_n = \{x_1, x_2, x_3, ..., x_L\}$ of length L is multiplied with a sliding window w_n of length τ and split into M segments. Resulting segments are stacked to create $\mathbb{X}_W \in \mathbb{R}^{M\times\tau}$. Subsequently, we apply FrFT to each row vector of $\mathbb{X}_W$ by multiplying its transpose with DFrFT matrix of order a, and $\mathbb{X}_{F^a} \in \mathbb{C}^{\tau\times M}$ is obtained. Finally, $\left\{\mathbf{x}_F^m\right\}_{m=1}^M$, which are the columns of $\mathbb{X}_{F^a}$, are extracted as features as the following:

$$X_n[m] = w_n[n - Sm]\mathcal{X}_n[n], \tag{3.1}$$

$$\mathbb{X}_W = \mathbb{W}(\mathcal{X}_n) \in \mathbb{R}^{M\times\tau}, \tag{3.2}$$

$$\mathbb{X}_{F^a} = W^a\mathbb{X}_W^\intercal, \tag{3.3}$$

$$\mathbb{X}_{F^a} \in \mathbb{C}^{\tau\times M} \mapsto \left\{\mathbf{x}_F^m\right\}_{m=1}^M \in \mathbb{C}^\tau, \tag{3.4}$$

where $\mathbb{W}$ is a mapping from a sequence to a matrix where each row is a windowed segment $X_n[m]$ and $W^a \in \mathbb{C}^{\tau\times\tau}$ is a DFrFT matrix of a. The second stage of our approach, referred to as the encoder-decoder phase, employs a many-to-many encoder-decoder architecture built with either GRU or basic RNN cells [59, 60], leading to two proposed model variations. The features derived from the first stage are passed into the encoder component of this stage, while the decoder operates merely on the original time-series data. Our overall model is illustrated in Fig. 3.1.

The model is then trained using the training set of the sequential information that is being studied. The feature extraction procedure is applied to only encoder training sequences and the features $\left\{\mathbf{x}_F^i\right\}_{i=1}^M$ are generated. These features are then fed to the encoder block GRU (or basic RNNs) units sequentially such that

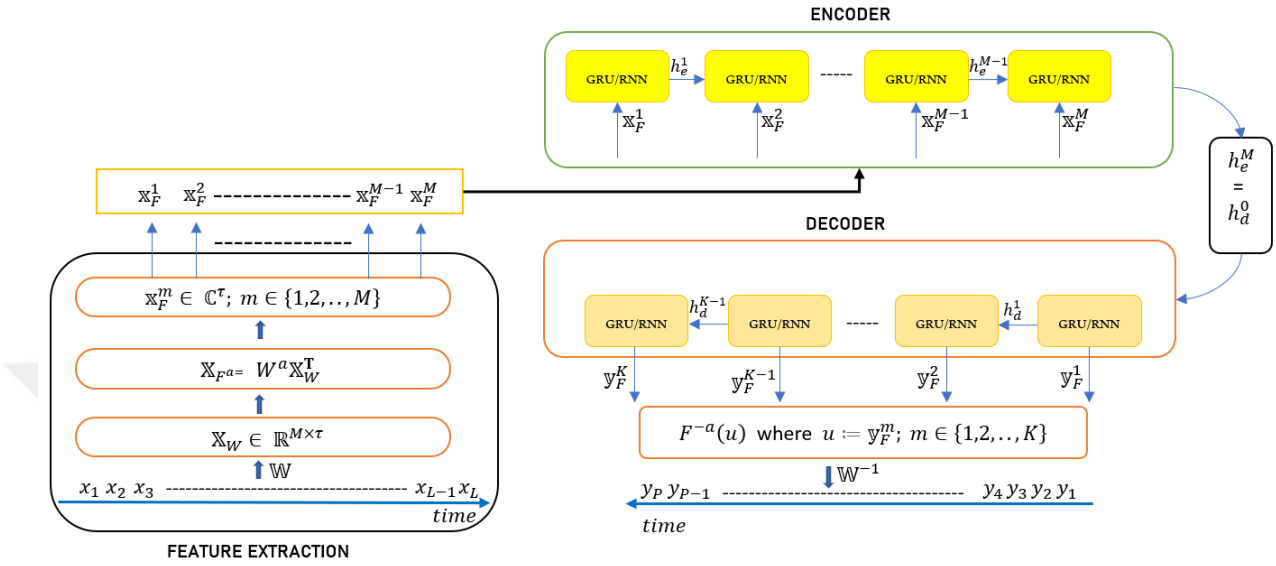

Figure 3.1: Feature vectors are extracted from the univariate time series using 3 cascaded operations. The proposed feature extraction method is applied to a sequence of encoder blocks and fed into GRU/RNN cells. The prediction sequence of length P is generated at the decoder block.

$\mathbf{x}_F^1$ goes to the first encoder unit, and $\mathbf{x}_F^n$ goes to the nth one. In the training procedure's forward pass, the encoder's hidden state is initialized to zero, and weights are initialized with Xavier initializer [61]. In each cell along the encoder, a new hidden state is produced using the current input and previous state and passed to the next cell. The state information of the last cell is transferred to the decoder to initialize the decoder's hidden states. Similarly, hidden state information is produced and propagated as in the encoder. Along the decoder, output vector of each cell is calculated as the multiplication of hidden state vector $\boldsymbol{h}_d$ with output weights W_{hy}, and passed through tanh activation function. The output vectors are multiplied with DFrFT of order $-a$ to calculate the inverse FrFT, then converted to one-dimensional sequences as in the case with the inverse STFT [57, 58]. The resulting sequence is finally used to perform the prediction. It is also necessary to mention that the length of the decoder block can differ from that of the encoder.

In the backward pass, the mean-squared error (MSE) between the predicted sequence and the decoder training sequence is calculated, and weights are updated

via backpropagation using RMSProp optimizer [62]. At the end of the training phase, model parameters, and weights are stored for the inference phase with unseen test data. In Algorithm 1, we present the details of our training process.

Algorithm 1 Learning sequential information using FRFT representations. $\odot$ stands for Hadamard point-wise multiplication.

Input: Encoder feature vectors: $\{\mathbf{x}_F^i\}_{i=1}^E$
Output: Model weights: $\boldsymbol{W}_{final}$
Parameters: E and D (encoder-decoder sequence lengths), T_{iter} (iteration number)
Initialize: $\boldsymbol{W}_0$, $\boldsymbol{h}_e^0 = 0$
for $t = 1$ to T_{iter} **do**
  Forward Pass:
  for $i = 1$ to E **do**
    if GRU **then**
      $z_e^i = \sigma_g(W_{zx}\mathbf{x}_F^i + W_{hz}\boldsymbol{h}_e^{i-1})$
      $r_e^i = \sigma_g(W_{rx}\mathbf{x}_F^i + W_{hr}\boldsymbol{h}_e^{i-1})$
      $\hat{h}_e^i = \tanh(W_{xx}\mathbf{x}_F^i + W_{hh}(r_e^i \odot \boldsymbol{h}_e^{i-1}))$
      $\boldsymbol{h}_e^i = z_e^i \odot \hat{h}_e^i + (1 - z_e^i) \odot \boldsymbol{h}_e^{i-1}$
    else if RNN **then**
      $\boldsymbol{h}_e^i = \sigma_g(W_{xx}\mathbf{x}_F^i + W_{hh}\boldsymbol{h}_e^{i-1})$
    end if
  end for
  $\boldsymbol{h}_d^0 \leftarrow \boldsymbol{h}_e^E$
  for $j = 1$ to D **do**
    if GRU **then**
      $z_d^j = \sigma_g(W_{hz}\boldsymbol{h}_d^{j-1})$
      $r_d^j = \sigma_g(W_{hr}\boldsymbol{h}_d^{j-1})$
      $\hat{h}_d^j = \tanh(W_{hh}(r_d^j \odot \boldsymbol{h}_d^{j-1}))$
      $\boldsymbol{h}_d^j = z_d^j \odot \hat{h}_d^j + (1 - z_e^j) \odot \boldsymbol{h}_d^{j-1}$
    else if RNN **then**
      $\boldsymbol{h}_d^j = \sigma_g(W_{hh}\boldsymbol{h}_d^{j-1})$
    end if
    $\hat{\mathbf{y}}_F^j = \tanh(W_{hy}\boldsymbol{h}_d^j)$
  end for
  $\{\hat{\mathbf{y}}_p\}_{p=1}^P \leftarrow \{\hat{\mathbf{y}}_F^j\}_{j=1}^D$ (inverse fractional Fourier Transform and sequence reconstruction)
  Backward Pass:
  $\mathcal{L}_{MSE}(\hat{y}, y) = \frac{1}{P}\sum_{i=1}^P (\hat{y}_i - y_i)^2$
  Update weights: $\boldsymbol{W}_t \leftarrow \boldsymbol{W}_{t-1}$
end for
return $\boldsymbol{W}_{final}$

Finally, the stored parameters and weights are used to initialize the test model in the inference phase. Features extracted from encoder test sequences are fed to

the encoder network, and the last hidden state is transferred to the decoder. In each cell along the decoder, output vectors are calculated to generate a prediction sequence for decoder test data.

3.2 Experiments & Results

In this section, we first provide the information on the datasets. Then, we delineate the implementation and experimental setup. Finally, we demonstrate our results and compare them with the existing methods.

3.2.1 Datasets

We conducted our experiments on three datasets from different fields. First, we have taken care that the datasets are not domain-specific to conduct a generalizable investigation. Therefore, datasets are selected from different domains, such as differential systems, finance, and energy consumption. Second, time-frequency varying signals are selected since the window-based methods are more suitable for signals with time-frequency variations.

3.2.1.1 Mackey-Glass Chaotic Time Series

It is generated from a nonlinear, time delay differential system that is described by the following differential equation [63]:

$$\frac{dx}{dt} = \frac{\beta x(t-T)}{1+x^{10}(t-T)} - \gamma x(t), \tag{3.5}$$

where $\beta = 0.2$, $\gamma = 0.1$, $dt = 0.1$, and $T = 17$. Starting values up to T-th second are initialized as $1 + u[-0.1, +0.1]$ where u stands for the uniform distribution.

3.2.1.2 USD-TRY Currency Exchange Ratio

Data on the USD to Turkish Lira (TRY) currency exchange ratio [64] are used from January 1, 2007, until January 1, 2020. Given the significant volatility in the USDTRY exchange ratio in this time interval, it is challenging to forecast future values based on historical values. The missing values are filled with the average value of the prior and following days.

3.2.1.3 University of California - Irvine (UCI) Electricity Load Dataset

UCI Electricity Load dataset [65] contains the electricity power consumption of 370 clients between 2011 and 2014, with the kW values recorded in a 15-minute resolution. The data before 2012 is not used due to many missing values. Power consumption (kW) is converted to energy consumption (kWh) by dividing each value by 4. Instead of a 15-minute resolution, daily energy consumption is used. The daily data of each client is also normalized using min-max normalization.

3.2.2 Implementation Details

In our experiments, we initialize the encoder block with a zero state. We set the learning rate to 0.001 during training and apply it over 30,000, 10,000, and 30,000 iterations for the respective datasets. To enhance training efficiency, the learning rate undergoes an exponential decay by a factor of 0.9 every 1,000 steps. For the currency and electric consumption datasets, we employ a Gaussian window of 64 samples, while for the Mackey-Glass dataset, a window length of 128 samples is used. Evaluating computational cost is critical for assessing model performance during both the training and testing phases. For a signal of length N, the Fast Fourier Transform (FFT) and the direct FrFT both have a time complexity of $N \log N$. Alternatively, FrFT can also be computed using matrix-vector multiplication with the DFrFT matrix, which has a time complexity of N^2. Interestingly,

for sequences of 64 and 128 samples, we found that using the DFrFT matrix yields faster computations compared to direct methods. Hence, in our study, we opted to compute the FrFT using the DFrFT matrix approach.

3.2.3 Results

We present the results of six different variants of the proposed FrFT-based time series prediction method in our experiments. Each variant is designed to process features in fractional Fourier domains. GRU64 and RNN64 stand for GRU and RNN architectures with hidden state sizes of 64. Similarly, $\text{GRU64}_{LP_{16}}$ and $\text{RNN64}_{LP_{16}}$ are GRU and RNN architectures with a hidden state size of 64. However, they are fed with feature vectors filtered using a low-pass filter where the cutoff frequency is 1/16 of the original signal bandwidth. cgGRU32 and cgGRU64 architectures stand for complex-gated GRUs with hidden sizes of 32 and 64, respectively. Complex-gated GRU [66] is an advanced architecture capable of being trained with complex-valued weights.

Table 3.1: Mackey-Glass for various architectures and orders a.

	Model					
a	GRU64	RNN64	$\text{GRU64}_{LP_{16}}$	$\text{RNN64}_{LP_{16}}$	cgGRU64	cgGRU32
0.5	0.030	**0.022**	71.84	71.63	**0.005**	**0.022**
0.6	**0.021**	0.034	64.14	64.20	**0.010**	**0.020**
0.7	0.026	**0.026**	47.14	47.55	**0.011**	**0.032**
0.8	**0.020**	**0.017**	2.256	3.614	**0.006**	**0.054**
0.9	0.024	**0.013**	**0.039**	0.027	**0.010**	**0.039**
1.0	0.023	0.034	0.044	0.012	0.015	0.175
1.1	0.027	**0.031**	**0.040**	0.041	**0.004**	**0.078**
1.2	0.033	**0.023**	0.942	2.556	**0.011**	**0.029**
1.3	**0.018**	**0.019**	46.85	46.91	**0.009**	**0.013**
1.4	0.024	**0.016**	64.13	64.19	**0.009**	**0.014**
1.5	**0.016**	**0.017**	71.71	71.81	**0.013**	**0.035**

Table 3.2: Try/Usd for various architectures and orders a.

	Model					
a	GRU64	RNN64	GRU64$_{LP_{16}}$	RNN64$_{LP_{16}}$	cgGRU64	cgGRU32
0.5	**9.33**	9.78	63.22	63.31	9.07	**8.81**
0.6	**9.72**	9.32	49.01	48.95	8.95	**7.65**
0.7	**9.61**	10.17	43.43	43.05	9.03	**8.48**
0.8	9.84	9.28	**10.08**	**10.40**	8.84	**8.49**
0.9	9.99	8.72	**9.91**	**10.39**	9.23	**9.10**
1.0	9.74	8.15	10.74	10.92	8.57	9.73
1.1	**9.54**	9.03	**10.08**	**10.39**	9.33	**8.94**
1.2	**9.43**	9.89	10.83	10.92	8.84	**8.48**
1.3	**9.70**	10.44	44.55	44.58	**8.06**	**8.15**
1.4	10.13	10.05	49.08	48.89	9.38	**8.23**
1.5	**9.05**	10.17	62.70	62.96	9.22	**8.68**

Table 3.3: Electric consumption for various architectures and orders a.

	Model					
a	GRU64	RNN64	GRU64$_{LP_{16}}$	RNN64$_{LP_{16}}$	cgGRU64	cgGRU32
0.5	28.54	11.98	79.52	71.43	17.25	22.30
0.6	21.26	10.92	59.73	55.93	19.39	**20.30**
0.7	20.62	11.58	56.78	47.2	**16.27**	**19.78**
0.8	26.65	**10.34**	12.98	14.33	**15.14**	21.71
0.9	19.95	**10.36**	12.47	**7.51**	18.41	**19.00**
1.0	19.00	10.54	10.12	8.38	16.98	20.63
1.1	**17.24**	11.52	12.94	**8.03**	**16.86**	**20.62**
1.2	**18.58**	**10.40**	11.89	8.65	17.18	**19.70**
1.3	25.38	10.64	53.55	47.26	18.68	25.65
1.4	**18.82**	**10.42**	61.86	56.41	19.72	**18.38**
1.5	21.54	12.19	85.77	71.43	18.41	**20.31**

Table 3.4: Best model performances of the proposed method are compared to baselines that use GRU/RNN.

Dataset	**Model**	***NMSPE***(%)
Mackey-Glass	RNN64 (a_l=0.944)	**0.0035**
	cgGRU64 (a=1.1)	0.004
	$\text{GRU64}_{baseline}$	0.23
	$\text{RNN64}_{baseline}$	5.48
TRY-USD	cgGRU32 (a_l=1.277)	**6.59**
	cgGRU64 (a=1.3)	8.06
	$\text{GRU64}_{baseline}$	10.30
	$\text{RNN64}_{baseline}$	12.01
Electric Consumption	GRU64 (a_l=0.858)	**6.20**
	$\text{RNN64}_{LP_{16}}$ (a=0.9)	7.51
	$\text{GRU64}_{baseline}$	10.84
	$\text{RNN64}_{baseline}$	14.76

We utilize normalized mean-squared percentage error (NMSPE) as our assessment metric. It is formally defined as $100\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i)^2}$ and expressed as a percentage, where y_i is the target value and $\hat{y}_i$ is the predicted value. In Tables 3.1-3.3, the performances of various methods are presented with respect to different FrFT orders and architectural differences for all three datasets. In the tables, results for the special case of the ordinary FT (order $a = 1$) are boxed as baselines, and better-performing results with respect to their corresponding baselines are emboldened.

Our experiments are not confined to fixed parameter search such that we also conduct experiments by including the fraction order a in the parameter learning stage. The fraction order a is taken into account a trainable/learnable weight likewise to the remaining trainable network parameters. Instead of manually searching for the optimal value, a is randomly initialized from the uniform distribution $U \sim (0.5, 1.5)$ and learned in the network iteratively to minimize training loss. We denote the learned fraction orders as a_l in order to discriminate them from manually selected fraction orders a that are tabulated in Tables 3.1-3.3. The

performances of the best model with the value of learned fraction order a_l are tabulated in Table 3.4. We also compare our proposed methods with $\text{GRU64}_{baseline}$ and $\text{RNN64}_{baseline}$, which are conventional time series prediction methods that do not use Fourier analysis, and the results can be found in Table 3.4.

In each experiment, the best model performance is achieved by the proposed method, using a manually-adjusted fraction order a, significantly outperforms the baseline methods at fraction orders other than $a = 1.0$. Furthermore, our proposed approach with a learned fraction order a_l not only automates the parameter search process but also significantly enhances model performance. These encouraging results show that the FrFT-based time series prediction improves both conventional baseline methods and the standard FT-based time series prediction.

It should be indicated that the fraction order values are mainly considered as hyperparameter and investigated the suitable values. The notion of trainability requires a stronger theoretical foundation to support experiments with the learnable fractional order a. Therefore, in the following chapter, we present a mathematical formulation illustrating how the FrFT can be integrated into neural network layers as a trainable layer.

Chapter 4

Trainable Fractional Fourier Transform

This chapter introduces the incorporation of the FrFT, a parametric signal transformation, as a trainable layer within neural networks. We showed that the fraction order a can be learned concurrently with remaining network parameters (weights) through backpropagation during the training stage. Our main objective is to establish a theoretical foundation for this concept, which is verified by extensive experiments across various tasks. In our image classification experiments, we replace the traditional average pooling layer, located between the last convolutional layer and the fully connected layers, with a trainable FrFT pooling layer. We use pre-trained models like ResNet18, ResNet34, ResNet101 [67], and VGG16 [68] as our base networks, integrating the FrFT pooling layer and comparing its performance against the original time-domain baselines and the DFT/FFT-based pooling method proposed in [19].

For time series prediction, we devise FrFT as a trainable layer to extract feature vectors by converting the time-domain input data into the fractional Fourier domain. These transformed inputs are then processed through encoder-decoder architectures, including sequence models like RNN, GRU, and LSTM. We compare the performance of these FrFT-based sequence models to those trained on

features in both the time and Fourier domains. Our experimental results support the potential of our utilization of FrFT across different tasks.

The content of this chapter reflects the work described in the following publication:

- E. Koç , T. Alikaşifoğlu, A. C. Aras, and A. Koç, "Trainable Fractional Fourier Transform," IEEE Signal Processing Letters, vol. 31, pp. 751–755, 2024..

4.1 FrFT Layer Update

Representation of a network graph where the ℓ^{th} hidden layer is a FrFT layer is provided in Fig. 4.1, which can be seen as a dimension-preserving fully-connected layer, where all the weights are parameterized with the fractional order a, and the relationship between the activation values of the ℓ^{th} and $(\ell-1)^{\text{th}}$ layers is given by $\boldsymbol{y}^{(\ell)} = \phi\left(\mathbf{F}^a \boldsymbol{y}^{(\ell-1)}\right)$ for some activation function ϕ.

Let C be the defined cost for the network, which can ultimately be expressed as a function of the ℓ^{th}-layer output vector of the network, i.e., $C = f(\boldsymbol{y}^{(\ell)})$. Let $\boldsymbol{\nabla}_{(\ell)} C$ be the row vector representing the gradient of the cost function with respect to the entries of ℓ^{th} layer's output vector. Then, for a learning rate of η, the update policy for the fractional order can be written as:

$$\begin{gathered} a_{\text{new}} = a_{\text{old}} - \eta \left(\boldsymbol{\nabla}_{(\ell)} C\right) \cdot \left(\frac{\partial \boldsymbol{y}^{(\ell)}}{\partial a}\right), \\ \text{with} \\ \frac{\partial \boldsymbol{y}^{(\ell)}}{\partial a} = \dot{\phi}\left(\mathbf{F}^a \boldsymbol{y}^{(\ell-1)}\right) \odot \left(\dot{\mathbf{F}}^a \boldsymbol{y}^{(\ell-1)}\right), \end{gathered} \tag{4.1}$$

where $\dot{\phi}$ is the derivative of the activation function, and $\dot{\mathbf{F}}^a$ is derived in Eq. (2.4). We utilize a basic, fully connected layer to illustrate the concept of

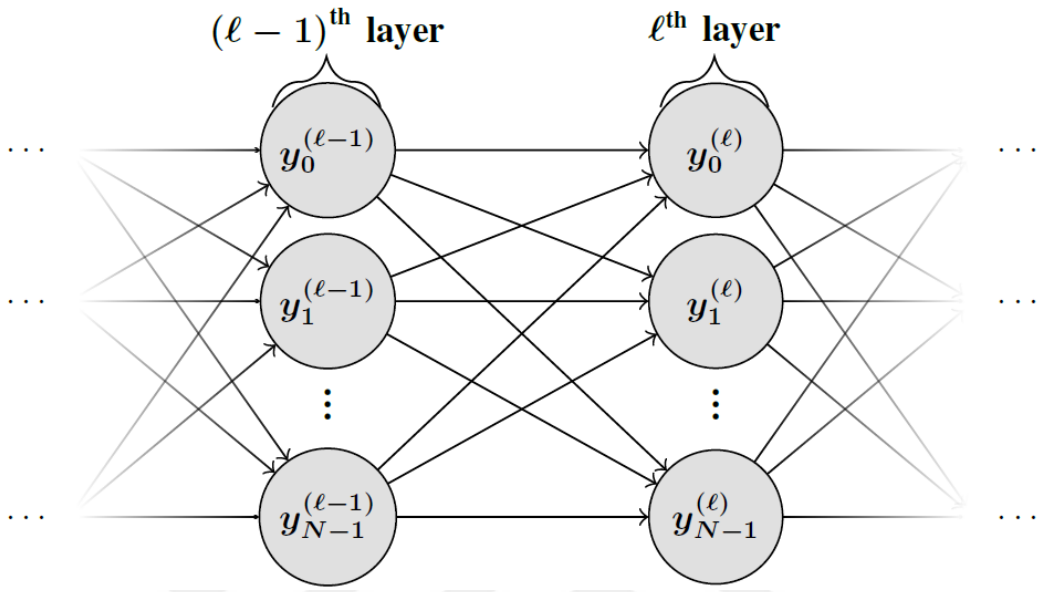

Figure 4.1: Representation of a network graph where the ℓ^{th} hidden layer is the FrFT layer, which can be seen as a dimension-preserving fully connected layer where all the weights are parameterized with the fractional order a, and $y_i^{(k)}$ corresponds to activation value of the i^{th} index of the k^{th} layer's output.

trainability, intending to clarify how the parameter updating mechanism works. The process of parameter update in neural networks, irrespective of the model or architecture, fundamentally depends on backpropagation. Crucially, the principle of learning the optimum a remains consistent across different applications that utilize those abovementioned deep neural network architectures; fully connected layers, CNNs, and RNNs, in their tasks. Hence, the derivations for learning a constitute a comprehensive formulation applicable to different neural network training processes, leveraging the backpropagation mechanism and relying on the gradient descent algorithm. This consistency originates from a common principle of weight updates across these models, which is primarily driven by the gradient descent algorithm. As a result, we opted to employ a fully connected layer as an abstraction to effectively illustrate this concept.

4.2 Trainable FrFT Layers

We introduce the FrFT as a trainable layer in different DNNs, including CNNs, and sequence models such as RNN, GRU, and LSTM. In CNN architectures, the FrFT pooling layer is placed between the last convolutional and fully connected layer by substituting the conventional average pooling layer. In sequence models, FrFT is employed as a learnable feature extraction method, where the time domain input features are transformed into fractional Fourier domain and processed in the time series prediction models.

4.2.1 Trainable FrFT Magnitude Pooling Layer

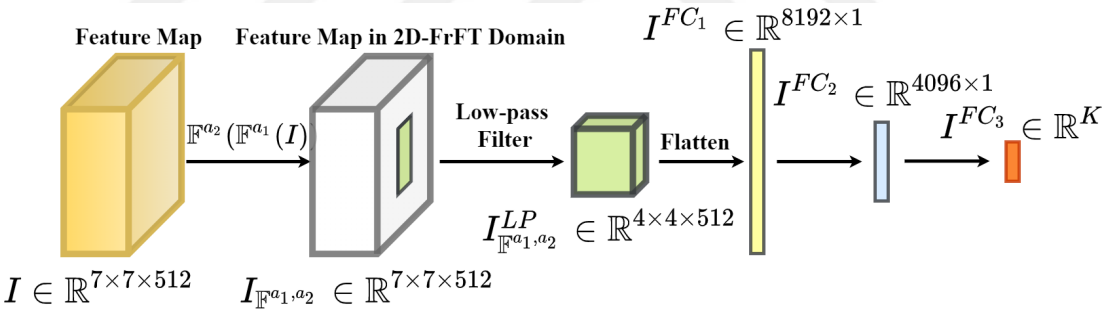

Figure 4.2: The feature map of size 7×7 in the last convolutional layer is transformed to the FrRT domains using 2D-FrFT with fraction orders a_1 and a_2. The high-frequency components are removed in the transformed domain, and the fully connected layer takes primarily the low-frequency information contained in the 4×4 area at the center of the FrFT coefficients. K stands for the number of image categories.

The last convolution layer produces feature maps $\mathbf{I} \in \mathbb{R}^{B\times C\times M\times M}$, where B is the batch size, C is the channel size, and M is the spatial resolution. Explicitly, the $M \times M$ feature maps correspond to the activations of neurons encoding the visual information, such as shape and location, which are employed in discriminating different image classes [19]. In the base networks, the last convolutional layer is followed by an average pooling layer such that the feature maps $\mathbf{I} \in \mathbb{R}^{B\times C\times M\times M}$ are transformed into $\mathbf{I}^{AP} \in \mathbb{R}^{B\times C\times1\times1}$, where AP stands for average pooling

operation.

FrFT magnitude pooling applies a 2D-FrFT to each channel of its input feature maps $\mathbf{I} \in \mathbb{R}^{B\times C\times M\times M}$. Note that 2D-FrFT is only two consecutive FrFT operations in an image channel's width and height dimensions [69]. Subsequently, the magnitude of the resulting fractional Fourier coefficients is adjusted by cropping them to a square size of $N \times N$. This cropping effectively removes high-frequency components from the coefficients. The remaining coefficients primarily represent low-frequency information and are then passed on to a subsequent fully-connected layer. In Fig. 4.2, incorporating the FrFT into CNNs as a trainable pooling layer is illustrated.

Formally, given the input feature maps $\mathbf{I} \in \mathbb{R}^{B\times C\times M\times M}$, the 2D-FrFT transformation with fraction orders a_1 and a_2 is defined as:

$$\mathbf{I}_{b,c}^{FrFT} = \mathrm{FrFT}_{a_1}\left(\mathrm{FrFT}_{a_2}\left(\mathbf{I}_{b,c}\right)\right), \tag{4.2}$$

where $\mathbf{I}_{b,c}^{FrFT}$ denotes the transformed feature maps for batch b and channel c. After transformation, the magnitude of the FrFT coefficients is computed and cropped to $N \times N$:

$$\mathbf{I}_{b,c}^{mag} = \left|\mathbf{I}_{b,c}^{FrFT}\right|_{1:N,1:N}. \tag{4.3}$$

These cropped coefficients are then fed into the fully connected layer, which performs classification over K image categories. The final classification output $\mathbf{y} \in \mathbb{R}^{B\times K}$ is obtained, and the cross-entropy loss is minimized for training:

$$\mathcal{L}_{CE} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{k=1}^{K} y_{i,k}\log\hat{y}_{i,k}, \tag{4.4}$$

where $y_{i,k}$ is the ground truth label and $\hat{y}_{i,k}$ is the predicted probability for class k of sample i.

4.2.2 Trainable FrFT Representations in Time Series Prediction

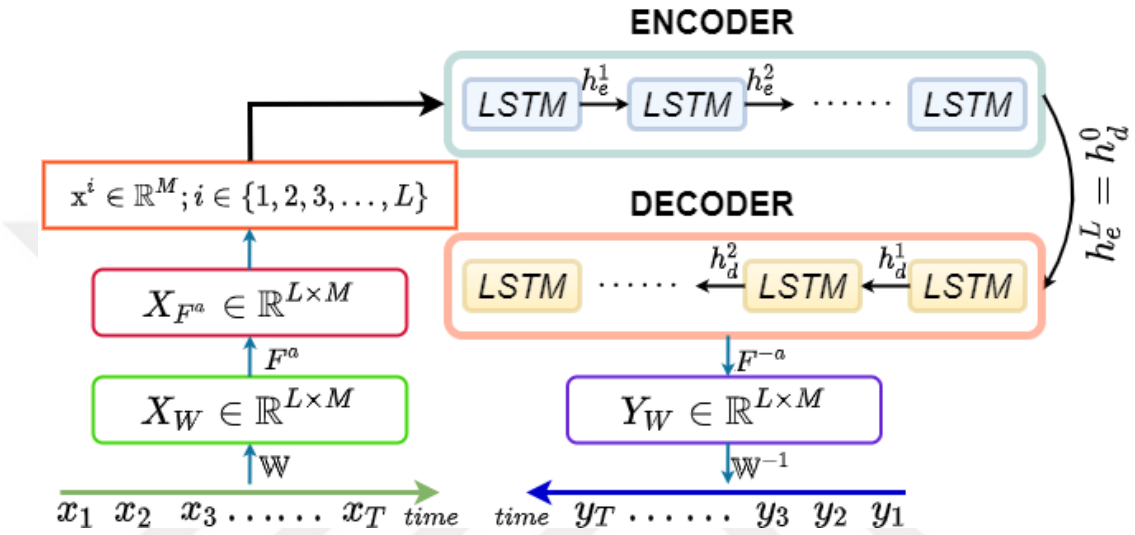

Figure 4.3: FrFT is applied to windowed segments obtained from the univariate time series. The fractional Fourier domain feature vectors are fed into the encoder blocks that consist of LSTM cells. The decoder output is transformed to the time domain using inverse FrFT with fraction order $-a$.

In time series prediction tasks, we use FrFT as a trainable layer to extract features from the input data prior to feeding the features into the encoder-decoder architectures with sequence models. We follow the similar underlying model architectures and training strategy proposed in [47] and [60] to investigate the *learnability* of fraction order a.

Our proposed method consists of two main stages. First, the univariate time series data $\mathbf{X} \in \mathbb{R}^{B \times T}$ is transformed into a multivariate sequence data $\mathbf{X}_W \in \mathbb{R}^{B \times L \times M}$, where B is the batch size, T is the length of univariate time series, L is the multivariate sequence data length and M is the sequence dimension. The transformation is performed such that $\mathbf{X} \in \mathbb{R}^{B \times T}$ is divided into L segments of length M with a sliding window function $\mathbb{W} \in \mathbb{R}^M$. Afterwards, $\mathbf{X}_W \in \mathbb{R}^{B \times L \times M}$ is transformed into FrFT domain with a fraction order a along the M dimension and the resulting feature sets $\mathbf{X}_{\mathbf{F}^a} \in \mathbb{R}^{B \times L \times M}$ are prepared for network training. The second stage comprises the training of encoder-decoder

architecture with basic RNN, GRU, and LSTM cells. The real parts of features $\mathbf{X}_{\mathbf{F}^a} \in \mathbb{R}^{B\times L\times M}$ generated in the initial stage are fed to the encoder of the second stage as a sequence of feature vectors $\mathcal{X}_{\mathbf{F}^a} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^L\}$, where $\mathbf{x}^i \in \mathbb{R}^{B\times M}$ and L is also the number of encoder cells. The state information h_e^L of the last cell is transferred to the decoder to initialize the decoder's hidden states. Similarly, hidden state information is produced and propagated as in the encoder. While the encoder receives input features, the decoder gets zero vectors as input, meaning it receives no input information. The decoder outputs a sequence of vectors $\mathcal{Y} = \{\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^L\}$, where $\mathbf{y}^i \in \mathbb{R}^{B\times M}$. $\mathcal{Y}$ can also be devised as the matrix of feature vectors $\mathbf{Y} \in \mathbb{R}^{B\times L\times M}$. Then, the feature matrix $\mathbf{Y}$ is transformed back to the time domain with DFrFT of order $-a$ constituting the $\mathbf{Y}_{\mathbf{F}^{-a}}$. The resulting time domain feature vectors are then converted to one-dimensional sequence $\mathbf{Y}_{pred} \in \mathbb{R}^{B\times T}$ through inverse window function $\mathbb{W}^{-1}$. Finally, the network is trained to update and optimize network weights, including fraction order a, using backpropagation iteratively and reducing the MSE between $\mathbf{Y}_{pred} \in \mathbb{R}^{B\times T}$ and the ground-truth values $\mathbf{Y}_{target} \in \mathbb{R}^{B\times T}$. The overall architecture using LSTM as a sequence model is depicted in Fig. 4.3.

The formal explanation of feature extraction, network training, and optimization is given as follows:

$$\mathbf{X}_{\mathbf{W}} = \mathbb{W}(\mathbf{X}), \tag{4.5}$$

$$\mathbf{X}_{\mathbf{F}^a} = \mathbf{X}_{\mathbf{W}}\mathbf{F}^a, \tag{4.6}$$

$$\mathbf{X}_{\mathbf{F}^a} \rightarrow \mathcal{X}_{\mathbf{F}^a} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^L\}; \mathbf{x}^i \in \mathbb{R}^{B\times M}, \tag{4.7}$$

$$\mathbf{Encoder}(State = h_e^0, Input = \mathcal{X}_{\mathbf{F}^a}) \rightarrow h_e^L = h_d^0, \tag{4.8}$$

$$\mathbf{Decoder}(State = h_e^L, Input = \mathbf{0}'s) \rightarrow \mathcal{Y} = \{\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^L\}; \mathbf{y}^i \in \mathbb{R}^{B\times M}, \tag{4.9}$$

$$\mathcal{Y} \rightarrow \mathbf{Y} \in \mathbb{R}^{B\times L\times M}, \tag{4.10}$$

$$\mathbf{Y}\mathbf{F}^{-a} = \mathbf{Y}_{\mathbf{F}^{-a}}, \tag{4.11}$$

$$\mathbb{W}^{-1}(\mathbf{Y}_{\mathbf{F}^{-a}}) = \mathbf{Y}_{pred} \in \mathbb{R}^{B\times T}, \tag{4.12}$$

$$\mathcal{L}_{MSE}(\mathbf{Y}_{pred}, \mathbf{Y}_{target}) = \frac{1}{BT}\sum(\mathbf{Y}_{pred} - \mathbf{Y}_{target})^2, \tag{4.13}$$

$$W_t \leftarrow W_{t-1} - \alpha\frac{d\mathcal{L}_{MSE}(\mathbf{Y}_{pred}, \mathbf{Y}_{target})}{dW_{t-1}}, \tag{4.14}$$

where W_t is the updated weights, $\mathcal{L}_{MSE}(\mathbf{Y}_{pred}, \mathbf{Y}_{target})$ is the MSE loss between predicted and ground-truth sequences and α is the learning rate.

4.3 Experiments & Results

We considered a comprehensive range of tasks, including image classification using the CUB-200-2011 [70], CIFAR-100 [71], and CIFAR-10 [72], along with time series prediction using the Electricity Load [73] and Exchange Rate [64] datasets. In image classification, we use the variants of ResNet [67] and VGG16 [68] to conduct our experiments on models with diverse scales. In each dataset, images are resized to 224×224 for compatibility with the CNN models. Encoder-decoder architectures with RNN, GRU, and LSTM are employed in time series prediction tasks.

4.3.1 Experimental Setup

For image classification tasks, each model is trained using the SGD optimizer, with a learning rate of 10^{-3}, a momentum of 0.9, and a regularization constant of 10^{-4}. Due to the complexity of the models, batch sizes of 64 and 32 are chosen for the ResNet variants [67] and VGG16 [68], respectively. The pooling size N for the FrFT pooling layers is set to 4. In time series prediction tasks, each sequence model is trained with the Adam optimizer, also using learning rate of 10^{-3} and an L_2-regularization constant of 10^{-4}. For experiments on the Electricity Load [73] dataset, a batch size of 32 is used. The window size for generating segments and the hidden state dimension of the sequence models are both set to 128.

We also implement baseline models based on the time domain and FFT using

the same configurations to ensure a thorough, rigorous and fair comparison with our FrFT-based models compared to their counterparts. More explicitly, in image classification tasks, we evaluate our approach against models that use traditional average pooling layers [67], [68] and FFT pooling layers [19], both with a pooling size of $N = 4$. For time series prediction tasks, we compare our method to models using input features in the time and Fourier domains.

4.3.2 Datasets

In this work, we employ three datasets for image classification and two datasets for time series prediction experiments. Here, we provide the datasets utilized in image classification tasks. The information for time series datasets is given in Section 3.2.1

4.3.2.1 CUB-200-2011

Consists of $11,798$ bird images of 200 subcategories, of which $5,994$ and $5,794$ are selected for training and testing, respectively. It is commonly used in fine-grained visual categorization tasks. Each image is annotated in detail with one subcategory label, 15 part locations, 312 binary attributes, and one bounding box [70].

4.3.2.2 CIFAR-100

The CIFAR-100 dataset is a subset of the Tiny Images dataset and contains $60,000$ color images of size 32×32. Image categories are distributed evenly such that each class consists of 600 images. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs) [71].

4.3.2.3 CIFAR-10

Likewise, the CIFAR-10 is a subset of Tiny Images datasets and consists of $60,000$ color images in 10 classes of size 32×32, where each image class consists of $6,000$ images [72].

4.3.3 Results

Table 4.1 shows the performance of models with average pooling, FFT-pooling, and FrFT-pooling layers evaluated using CUB-200-2011 [70], CIFAR-100 [71] and CIFAR-10 [72] datasets on ResNet18, ResNet34, ResNet101 [67], and VGG16 [68] architectures. The proposed FrFT-based models consistently outperform the base and FFT-based models over all datasets. In Table 4.2, we compare the performance of LSTM, GRU, and RNN models that extract features in time, FFT, and FrFT domains. Each method is evaluated using root mean square error (RMSE) and mean absolute error (MAE) on Electricity Load and Exchange Rate datasets. In all datasets, the proposed FrFT performs better than time-domain and Fourier-domain-based approaches in time series prediction.

Table 4.1: Accuracy (in %) of FrFT-based method with time and FFT-based methods on three benchmarks and four CNN models.

CNN Model	**Transform**	**Dataset**		
		CUB-200-2011	CIFAR-100	CIFAR-10
ResNet18	Time	70.95	80.78	95.54
	FrFT	**71.56**	**81.20**	**95.70**
	FFT	71.14	81.09	95.63
ResNet34	Time	74.64	82.89	96.34
	FrFT	**74.76**	**83.91**	**96.77**
	FFT	74.30	83.79	96.65
ResNet101	Time	78.81	85.97	97.64
	FrFT	**79.35**	**86.81**	**97.75**
	FFT	79.00	86.71	96.82
VGG16	Time	77.75	77.13	94.43
	FrFT	**78.70**	**77.70**	**94.78**
	FFT	78.13	77.44	94.42

Table 4.2: Comparison of the FrFT-based method with time and FFT-based methods on two benchmarks, and LSTM, GRU, and RNN models.

Model	**Transform**	**Dataset**			
		Exchange Rate		Electricity Load	
		RMSE ↓	MAE ↓	RMSE ↓	MAE ↓
LSTM	Time	1.294	1.193	0.181	0.150
	FrFT	**1.170**	**1.121**	**0.176**	**0.138**
	FFT	1.197	1.136	0.177	0.140
GRU	Time	1.293	1.194	0.197	0.146
	FrFT	**1.201**	**1.124**	**0.182**	**0.144**
	FFT	1.211	1.333	0.184	0.145
RNN	Time	1.185	1.148	0.187	0.151
	FrFT	**1.142**	**1.101**	**0.167**	**0.133**
	FFT	1.226	1.158	0.180	0.142

4.3.3.1 Learned Fraction Orders

The initial value of fraction orders is set as 1.0, ensuring compatibility across diverse models but also steers models to surpass the FT-based approaches. In Table 4.3, the learned fraction orders, a_1 and a_2, are presented for our image classification tasks with ResNet18, ResNet34, ResNet101, and VGG16 on CUB-200-2011, CIFAR-100 and CIFAR-10. Table 4.4 gives the learned fraction order a values for our time series prediction tasks with LSTM, GRU, and RNN models on Exchange Rate and Electric Load datasets.

Table 4.3: The learned fraction order values are presented for image classification tasks. a_1 and a_2 refer to the fraction orders of 2D-FrFT operation in width and height dimensions of the image.

Model	**Datasets**		
	CUB-200-2011	CIFAR-100	CIFAR-10
	a_1 | a_2	a_1 | a_2	a_1 | a_2
ResNet18	1.0600 | 1.0010	0.9951 | 0.9124	1.0140 | 1.1810
ResNet34	0.9992 | 0.9967	0.9978 | 1.0410	0.9638 | 0.8142
ResNet101	1.0910 | 1.2380	0.9978 | 1.0950	1.0030 | 1.2300
VGG16	0.9947 | 0.9760	0.9974 | 1.0040	1.0030 | 0.9550

Table 4.4: The learned fraction order (a) values are presented for time series prediction tasks.

Model	**Datasets**	
	Exchange Rate	Electric Load
LSTM	1.0350	0.9983
GRU	0.9997	0.9973
RNN	0.9998	0.9997

4.3.3.2 Learned Fraction Orders vs Fixed Fraction Orders

We also examine the performance in both tasks using fixed fractional orders such as 0.25, 0.5, and 1.0. Subsequently, we compare the model performance with these fixed orders to the results of our proposed method.

Results indicate that the fixed values of fraction orders still demonstrate promising performance, especially in image classification tasks. However, their results are inferior to those with learned fraction orders denoted as $a_{learned}$, where their exact values for each model, dataset, and task are provided in the Table 4.3 and Table 4.4.

Table 4.5: Accuracy values (in %) for various fraction orders. 0.25, 0.50, 1.00 (FFT) denotes the fixed fraction orders, and $a_{learned}$ represents the trainable fraction orders.

Model	**Order**	**Dataset**		
		CUB-200-2011	CIFAR-100	CIFAR-10
ResNet18	0.25	69.12	80.09	95.64
	0.50	69.87	81.11	95.64
	1.00	71.14	81.09	95.63
	$a_{learned}$	**71.56**	**81.20**	**95.70**
ResNet34	0.25	72.10	83.23	96.42
	0.50	72.11	83.53	96.67
	1.00	74.30	83.79	96.65
	$a_{learned}$	**74.76**	**83.91**	**96.77**
ResNet101	0.25	76.56	86.17	97.36
	0.50	78.32	86.57	97.69
	1.00	79.00	86.71	96.82
	$a_{learned}$	**79.35**	**86.81**	**97.75**
VGG16	0.25	74.66	76.41	93.86
	0.50	77.16	77.01	94.24
	1.00	78.13	77.44	94.42
	$a_{learned}$	**78.70**	**77.70**	**94.78**

Table 4.6: Model performance for fixed and learned fraction orders in time series tasks. 0.25, 0.50, 1.00 (FFT) denotes the fixed fraction orders, and $a_{learned}$ represents the trainable fraction orders.

Model	**Order**	**Dataset**			
		Exchange Rate		Electricity Load	
		RMSE ↓	MAE ↓	RMSE ↓	MAE ↓
LSTM	0.25	1.225	1.479	0.409	0.311
	0.50	1.224	1.151	0.394	0.307
	1.00	1.197	1.136	0.177	0.140
	$a_{learned}$	**1.170**	**1.121**	**0.176**	**0.138**
GRU	0.25	1.201	1.214	0.417	0.320
	0.50	1.203	1.216	0.360	0.274
	1.00	1.211	1.333	0.184	0.145
	$a_{learned}$	**1.201**	**1.124**	**0.182**	**0.144**
RNN	0.25	1.197	1.140	0.343	0.265
	0.50	1.216	1.152	0.370	0.294
	1.00	1.226	1.158	0.180	0.142
	$a_{learned}$	**1.142**	**1.101**	**0.167**	**0.133**

4.3.3.3 Inference Times

In Table 4.7 and Table 4.8, we illustrate the inference times for our image classification and time series prediction tasks, respectively. The results show that our FrFT-based methods outperform both the time and FFT-based approaches without adding a significant computational cost. The differences in inference times between our FrFT-based approach and the other methods are mainly due to our intentional choice of a long sequence length of 128 in the time series prediction tasks. However, it is important to note that this difference does not pose a challenge during the training phase, as the inference times remain within the millisecond range.

Table 4.7: Inference time (in milliseconds) comparison between time, FFT, and FrFT-based methods in classification tasks.

Model	**Transform**	**Dataset**		
		CUB-200-2011	CIFAR-100	CIFAR-10
ResNet18	Time	4.82	0.99	0.99
	FrFT	5.42	1.33	1.37
	FFT	5.18	1.01	1.10
ResNet34	Time	4.67	1.05	1.02
	FrFT	5.51	1.56	1.54
	FFT	4.96	1.01	1.04
ResNet101	Time	4.95	1.13	1.16
	FrFT	6.00	2.68	2.77
	FFT	4.57	1.15	1.20
VGG16	Time	4.34	0.92	0.91
	FrFT	6.00	2.88	2.91
	FFT	4.34	0.98	0.95

Table 4.8: Inference time (in milliseconds) comparison between time, FFT, and FrFT-based methods in time series prediction tasks.

Model	**Transform**	**Dataset**	
		Exchange Rate	Electricity Load
LSTM	Time	1.95	2.00
	FrFT	33.65	20.05
	FFT	2.50	2.80
GRU	Time	1.70	1.75
	FrFT	36.20	22.10
	FFT	2.10	2.40
RNN	Time	1.10	1.40
	FrFT	32.20	28.50
	FFT	2.10	2.35

Chapter 5

Maximally Selective Fractional Fourier Pooling

In this chapter, we present our pooling approach that combines FrFT and GAP techniques which are overwhelmingly applied to CNNs in image classification tasks. More explicitly, to reduce the information loss that may arise from the GAP technique and to improve the model performance of image classification models without significantly increasing computational costs, we introduce a method called FrFP, a FrFT-based pooling technique. Essentially, we replace the GAP layers following the final convolutional layer of the models with our proposed FrFP layer, enabling the processing of information in intermediate signal regions determined by the fractional order a. In this work, we define the fractional order a as a learnable parameter, allowing the model to automatically determine the optimal data representation for the FrFP process.

The content of this chapter reflects the work described in the following publication:

- E. Koç, Y. E. Ekiz, H. Özaktaş and A. Koç, "Maximally Selective Fractional Fourier Pooling," 2024 32nd Signal Processing and Communications Applications Conference (SIU), pp. 1–4, 2024..

5.1 Problem Definition

Mathematically, the average pooling method can be expressed as follows: a feature map $\mathbf{X} \in \mathbb{R}^{B\times C\times H\times W}$, after passing through an average pooling layer that halves the spatial dimensions, is transformed into a new feature map $\mathbf{X}_{avg} \in \mathbb{R}^{B\times C\times H/2\times W/2}$.

A more specialized type of pooling, known as GAP, which is predominantly used between the last convolutional layers and fully connected (FC) layers of deep neural networks (DNNs), does not solely halve the size of each channel but instead averages all pixel values in each channel to convey a single scalar value from each channel to the subsequent layer. More clearly, given an input feature map $\mathbf{X} \in \mathbb{R}^{B\times C\times H\times W}$, the GAP layer generates $\mathbf{X}_{GAP} \in \mathbb{R}^{B\times C\times 1\times 1}$. Here, B denotes the batch size, C denotes the channel dimension, and H and W denote the height and width of an image, respectively.

As observed, the information passed to the next layer following the GAP layer is considerably minimal. Even though this operation aims at increasing efficiency, it can lead to large information loss. More importantly, the proximity of the GAP operation to the classification layer makes this loss irreversible and inevitable [74].

To address these potential tackles with GAP, we propose using the FrFP layer instead. In the FrFP layer, the input feature maps are transformed into the fractional Fourier domain along the H and W dimensions using a two-dimensional FrFT. It is worth noting that the two-dimensional FrFT is essentially an extension of the one-dimensional FrFT process applied separately along both the height and width dimensions of the image [69]. This operation is performed separately for each channel. Subsequently, pixel values in the fractional Fourier domain for each channel are sorted by their absolute magnitudes. From this sorted list, the top k pixel values are selected. This process is applied separately across all channels, and the selected values from all channels are combined into a single feature vector, which is then forwarded to the fully connected (FC) layer, similar to GAP. Specifically, given an input feature map $\mathbf{X} \in \mathbb{R}^{B\times C\times H\times W}$, the

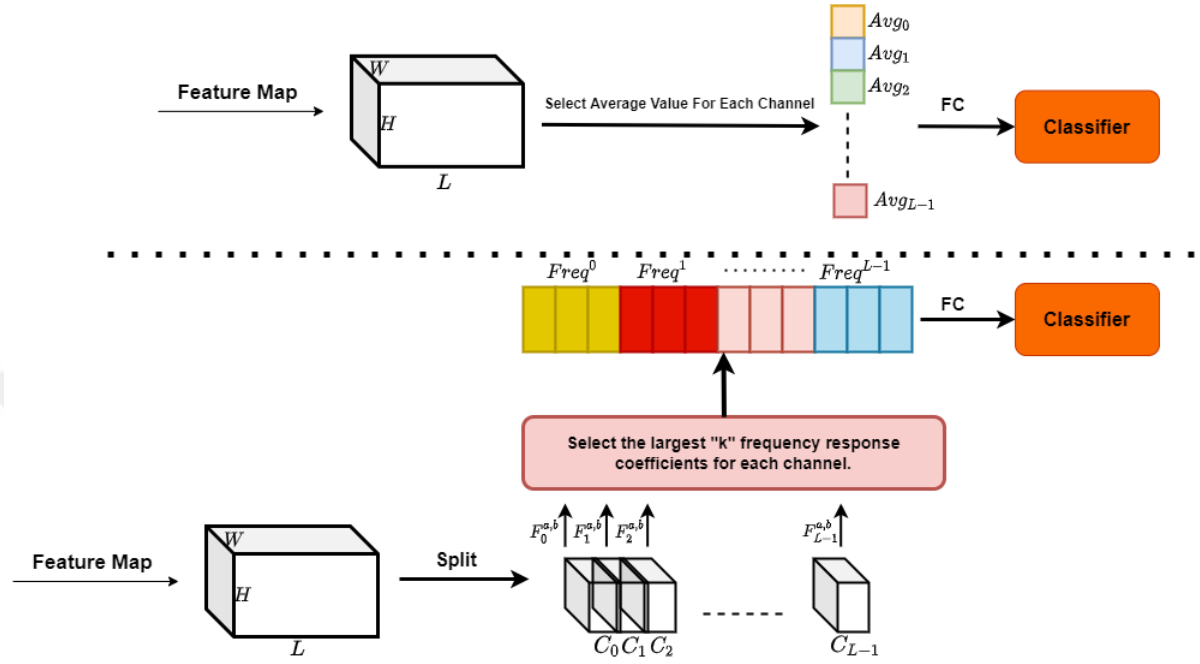

Figure 5.1: **Top:** Illustration of the traditional GAP operation. All pixels in the spatial region of each channel of the feature map are averaged and combined. The resulting new feature vector is forwarded to the fully connected (FC) layer for classification. **Bottom:** Illustration of our FrFP approach. The feature map is first split into channels. Images in each channel are transformed into the fractional Fourier domain using DFrFT, and the pixels/coefficients are sorted by their absolute values. The *k-largest* coefficient values in each channel are selected and combined into a single feature vector. The resulting new feature vector is forwarded to the fully connected (FC) layer for classification. Here, $F^{a,b}$ represents the two-dimensional FrFT.

DFrFT matrices $F_a \in \mathbb{R}^{H\times H}$ and $F_b \in \mathbb{R}^{W\times W}$ are used for multiplication, resulting in $\mathbf{X}_{F_{a,b}} \in \mathbb{R}^{B\times C\times H\times W}$. Here, a and b denote the fractional degrees of the DFrFT operations. As explained in Section 4, we incorporated these fractional degrees into the model's training process, enabling automatic learning. After this stage, the top k pixel (FrFT coefficient) values with the highest magnitude are selected for each channel, and these values are combined into a single feature vector $\mathbf{X}^{max}_{F_{a,b}} \in \mathbb{R}^{B\times kC}$, which is then passed to the fully connected (FC) layer. It is noteworthy that "k" is treated as a hyperparameter set by the user. All these processes are visualized in Figure 5.1.

5.2 Experiments and Results

In this study, we replaced the GAP layers in our image classification models with the proposed FrFP layers. We trained both the original models with GAP and the models with FrFP using the same hyperparameters and training strategy on two different datasets. Specifically, to ensure a fair comparison, we trained and tested the same model with both GAP and FrFP layers using identical training methods and configurations. This section presents details of the experiments and results.

5.2.1 Experiment Details

In this work, we set the batch size to 32 for VGG11 and to 64 for all other models. This choice is due to the larger memory footprint of the VGG11 model compared to the others. Similarly, we set the number of epochs to 100 for VGG11 and to 50 for all other models. We chose the pooling size k to be 16, as we observed that even transferring approximately 1/3 of the information from 7×7 pixel-sized channels was effective in improving model performance. To prevent overfitting, we employed both L_2-regularization and early stopping. We applied the early stopping method with a patience value of 10 on the validation dataset. We used SGD as the optimizer. All hyperparameter values are listed in Table 5.1.

Table 5.1: Hyperparameters to train models

Hyperparameter	**Value**
Batch size	[32, 64]
Initial learning rate	1×10^{-3}
Pool size	16
Epoch number	[50, 100]
Patience	10
Optimizer	SGD
Momentum	0.9
Regularization	Ridge ($\mathcal{L}_2$)
Regularization coefficient(λ)	5×10^{-4}
Scheduler	CosineAnnealingLR

5.2.2 Results

To comprehensively compare our FrFP-based pooling method with the traditional GAP method, we evaluate both FrFP and GAP versions of common CNNs such as DenseNet121, ResNet variants, VGG11, MobileNetV2, EfficientNet, and the encoder-based Swin Transformer V2 model on the CUB-200-2011 [70] and CIFAR-100 [72] datasets. The experimental results show that our FrFP method surpasses the GAP-based models in all model and dataset combinations, except for the performance of the ResNet101 model on the CIFAR-100 dataset. Table 5.2 compares the model accuracy of the two different pooling methods across all models and datasets.

Table 5.2: Accuracy comparison of GAP and FrFP-based methods on two benchmark datasets and nine image classification models.

Model	**Pooling Method**	**Dataset**	
		CUB-200-2011	CIFAR-100
DenseNet121 [75]	GAP	78.65	82.43
	FrFP	**79.59**	**84.36**
ResNet18 [67]	GAP	73.00	79.88
	FrFP	**73.76**	**81.37**
ResNet34 [67]	GAP	75.04	82.50
	FrFP	**75.65**	**83.61**
ResNet50 [67]	GAP	78.58	83.52
	FrFP	**78.81**	**83.75**
ResNet101 [67]	GAP	78.72	**86.87**
	FrFP	**79.48**	86.21
VGG11 [68]	GAP	71.78	75.04
	FrFP	**74.85**	**75.52**
MobileNetV2 [76]	GAP	71.83	79.09
	FrFP	**73.40**	**79.26**
EfficientNet B0 [77]	GAP	74.09	84.88
	FrFP	**76.08**	**85.54**
Swin Transformer [78]	GAP	82.14	87.00
	FrFP	**83.91**	**87.20**

Chapter 6

Discussion and Conclusion

In this thesis, we introduced and explored the integration of the FrFT within the deep learning domain, showcasing its potential to enhance model performance across various tasks. We started our exploration by introducing FrFT into sequence models for time series prediction. Our main motivation behind this work stemmed from FrFT's ability to perform an infinite number of transformations, offering a flexible alternative to FT. While this work shows promising results and underlines the potential of using FrFT in prospective deep learning studies, one limitation remains: similar to current works in literature, the fraction order a has traditionally been considered as a hyper-parameter that requires exhaustive tuning to determine the optimal value. This process can be cumbersome and time-consuming, potentially limiting the broader applicability of the models.

By aiming at addressing this limitation, in the second work of this thesis, we expanded the use of FrFT in the domain of deep learning by introducing the fraction order a as a learnable parameter. First, we established a robust theoretical and analytical foundation, offering a formal explanation of the parameter's learnability. Subsequently, we performed a diverse array of experiments in image classification and time series prediction tasks demonstrating the success of our approach. Moreover, in our image classification models, we selected well-established models such as ResNet18, ResNet34, ResNet101, and VGG16. These models are

commonly used as backbones for large-scale architectures for image segmentation and object recognition tasks, where their intermediate features and classification performance play a pivotal role in the overall success of these tasks. This suggests that our method can also be effectively integrated into large-scale models through their backbones, and improves the performance of the aforementioned tasks.

In the final study, we introduced FrFP, a pooling technique that leverages FrFT to replace GAP layers in CNNs. The FrFP layers processed information within intermediate signal regions defined by the fractional order a, offering a more refined representation of features compared to traditional pooling methods. This approach not only improved model performance but also provided a new perspective on how signal transformations can be integrated into the deep learning pipeline.

Bibliography

[1] C. Sakr and N. R. Shanbhag, "Signal processing methods to enhance the energy efficiency of in-memory computing architectures," *IEEE Transactions on Signal Processing*, vol. 69, pp. 6462–6472, 2021.

[2] L. Cheng, J. Li, and Y. Yan, "FSCNet: Feature-specific convolution neural network for real-time speech enhancement," *IEEE Signal Processing Letters*, vol. 28, pp. 1958–1962, 2021.

[3] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.

[4] A. Y. Yıldız, E. Koç, and A. Koç, "Multivariate time series imputation with transformers," *IEEE Signal Processing Letters*, vol. 29, pp. 2517–2521, 2022.

[5] O. B. Güney, E. Koç, C. Aksoy, Y. Çatak, Ş. S. Arslan, and H. Özkan, "Adaptive boosting of dnn ensembles for brain-computer interface spellers," in *2021 29th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2021.

[6] E. Şimşek, A. Güngör, Ö. Karavelioğlu, M. T. Yerli, and N. G. Kuyumcuoğlu, "Wind power prediction using machine learning and deep learning algorithms," in *2023 31st Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2023.

[7] E. Gundogdu, A. Koç, and A. A. Alatan, "Object classification in infrared images using deep representations," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1066–1070, 2016.

[8] M. Shahbazi and H. Aghajan, "A generalizable model for seizure prediction based on deep learning using CNN-LSTM architecture," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 469–473, IEEE, 2018.

[9] L. Liu, J. Wu, D. Li, L. Senhadji, and H. Shu, "Fractional Wavelet scattering network and applications," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 2, pp. 553–563, 2018.

[10] W. Ghezaiel, B. Luc, and O. Lézoray, "Wavelet scattering transform and CNN for closed set speaker identification," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, IEEE, 2020.

[11] J. Shi, Y. Zhao, W. Xiang, V. Monga, X. Liu, and R. Tao, "Deep scattering network with fractional Wavelet transform," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4740–4757, 2021.

[12] A. Bhattacharyya, L. Singh, and R. B. Pachori, "Fourier–Bessel series expansion based empirical Wavelet transform for analysis of non-stationary signals," *Digital Signal Processing*, vol. 78, pp. 185–196, 2018.

[13] V. Gupta and R. B. Pachori, "FBDM based time-frequency representation for sleep stages classification using eeg signals," *Biomedical Signal Processing and Control*, vol. 64, p. 102265, 2021.

[14] R. R. Sharma and R. B. Pachori, "Time–frequency representation using ievdhm–ht with application to classification of epileptic eeg signals," *IET Science, Measurement & Technology*, vol. 12, no. 1, pp. 72–82, 2018.

[15] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," *arXiv preprint arXiv:1312.5851*, 2013.

[16] H. Pratt, B. Williams, F. Coenen, and Y. Zheng, "FCNN: Fourier convolutional neural networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 786–798, Springer, 2017.

[17] K. Chitsaz, M. Hajabdollahi, P. Khadivi, S. Samavi, N. Karimi, and S. Shirani, "Use of frequency domain for complexity reduction of convolutional neural networks," in *International Conference on Pattern Recognition*, pp. 64–74, Springer, 2021.

[18] L. Chi, B. Jiang, and Y. Mu, "Fast Fourier convolution," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4479–4488, 2020.

[19] J. Ryu, M.-H. Yang, and J. Lim, "DFT-based transformation invariant pooling layer for visual classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[20] K. Xu, M. Qin, F. Sun, Y. Wang, Y.-K. Chen, and F. Ren, "Learning in the frequency domain," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1740–1749, 2020.

[21] Z. Qin, P. Zhang, F. Wu, and X. Li, "FcaNet: Frequency channel attention networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 783–792, 2021.

[22] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "FNet: Mixing tokens with Fourier transforms," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Seattle, United States), pp. 4296–4313, Association for Computational Linguistics, July 2022.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.

[24] D. Lappas, V. Argyriou, and D. Makris, "Fourier transformation autoencoders for anomaly detection," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1475–1479, IEEE, 2021.

[25] J. Zhang, Y. Lin, Z. Song, and I. Dhillon, "Learning long term dependencies via Fourier recurrent units," in *International Conference on Machine Learning*, pp. 5815–5823, PMLR, 2018.

[26] A. Alaa, A. J. Chan, and M. van der Schaar, "Generative time-series modeling with Fourier flows," in *International Conference on Learning Representations*, 2021.

[27] H. Ma, L. Zhang, X. Zhu, and J. Feng, "Accelerating score-based generative models with preconditioned diffusion sampling," in *European Conference on Computer Vision*, pp. 1–16, Springer, 2022.

[28] W. Liu, Q. Liao, F. Qiao, W. Xia, C. Wang, and F. Lombardi, "Approximate designs for fast Fourier transform (FFT) with application to speech recognition," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 12, pp. 4727–4739, 2019.

[29] X. Luo, H. Wang, T. Han, and Y. Zhang, "FFT-Trans: Enhancing robustness in mechanical fault diagnosis with Fourier transform-based transformer under noisy conditions," *IEEE Transactions on Instrumentation and Measurement*, 2024.

[30] K. Dakic, B. Al Homssi, M. Lech, and A. Al-Hourani, "HybNet: A hybrid deep learning-matched filter approach for IoT signal detection," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 1, pp. 18–30, 2023.

[31] H. Veluri, U. Chand, C.-K. Chen, and A. V.-Y. Thean, "A low-latency DNN accelerator enabled by DFT-based convolution execution within crossbar arrays," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[32] Y. Yang and R. Tao, "Single-shot fractional Fourier phase retrieval," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023.

[33] G. Huang, F. Zhang, and R. Tao, "Sliding short-time fractional Fourier transform," *IEEE Signal Processing Letters*, vol. 29, pp. 1823–1827, 2022.

[34] X. Zhao, M. Zhang, R. Tao, W. Li, W. Liao, L. Tian, and W. Philips, "Fractional Fourier image transformer for multimodal remote sensing data classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[35] H. Zhao and B.-Z. Li, "Unlimited sampling theorem based on fractional Fourier transform," *Fractal and Fractional*, vol. 7, no. 4, p. 338, 2023.

[36] X. Zhao, M. Zhang, R. Tao, W. Li, W. Liao, and W. Phlips, "Multisource remote sensing data classification using fractional Fourier transformer," in *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*, pp. 823–826, 2022.

[37] A. Koç and H. M. Ozaktas, "Operator theory-based computation of linear canonical transforms," *Signal Processing*, vol. 189, p. 108291, 2021.

[38] C. Ozturk, H. M. Ozaktas, S. Gezici, and A. Koç, "Optimal fractional Fourier filtering for graph signals," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2902–2912, 2021.

[39] A. Koç, H. M. Ozaktas, C. Candan, and M. A. Kutay, "Digital computation of linear canonical transforms," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2383–2394, 2008.

[40] H. M. Ozaktas and D. Mendlovic, "Fourier transforms of fractional order and their optical interpretation," *Optics Communications*, vol. 101, no. 3-4, pp. 163–169, 1993.

[41] A. Koç and H. M. Ozaktas, "Relationship between the beam propagation method and linear canonical and fractional fourier transforms," *Applied Optics*, vol. 61, no. 34, pp. 10275–10282, 2022.

[42] M. Cheng, L. Deng, X. Wang, H. Li, M. Tang, C. Ke, P. Shum, and D. Liu, "Enhanced secure strategy for OFDM-PON system by using hyperchaotic system and fractional Fourier transformation," *IEEE Photonics Journal*, vol. 6, no. 6, pp. 1–9, 2014.

[43] H. Ozaktas, O. Arikan, M. Kutay, and G. Bozdagı, "Digital computation of the fractional Fourier transform," *IEEE Transactions on Signal Processing*, vol. 44, no. 9, pp. 2141–2150, 1996.

[44] F. Şahinuç and A. Koç, "Fractional Fourier transform meets transformer encoder," *IEEE Signal Processing Letters*, vol. 29, pp. 2258–2262, 2022.

[45] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[46] R. Tao, X. Zhao, W. Li, H.-C. Li, and Q. Du, "Hyperspectral anomaly detection by fractional Fourier entropy," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 12, pp. 4920–4929, 2019.

[47] E. Koç and A. Koç, "Fractional Fourier transform in time series prediction," *IEEE Signal Processing Letters*, vol. 29, pp. 2542–2546, 2022.

[48] E. Koç, T. Alikasifoglu, A. C. Aras, and A. Koç, "Trainable fractional Fourier transform," *IEEE Signal Processing Letters*, 2024.

[49] E. Koç, Y. E. Ekiz, H. Özaktaş, and A. Koç, "Maximally selective fractional fourier pooling," in *2024 32nd Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2024.

[50] A. Koç, H. M. Ozaktas, B. Bartan, E. Gundogdu, and T. Cukur, "Digital computation of fractional Fourier and linear canonical transforms and sparse image representation," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 111–117, IEEE, 2017.

[51] A. Koç, B. Bartan, E. Gundogdu, T. Çukur, and H. M. Özaktaş, "Sparse representation of two-and three-dimensional images with fractional fourier, hartley, linear canonical, and Haar Wavelet transforms," *Expert Systems with Applications*, vol. 77, pp. 247–255, 2017.

[52] A. Koç, "Operator theory-based discrete fractional Fourier transform," *Signal, Image and Video Processing*, vol. 13, pp. 1461–1468, 2019.

[53] T. Alikaşifoğlu, B. Kartal, and A. Koç, "Wiener filtering in joint time-vertex fractional Fourier domains," *IEEE Signal Processing Letters*, vol. 31, pp. 1319–1323, 2024.

[54] T. Alikaşifoğlu, B. Kartal, and A. Koç, "Graph fractional Fourier transform: A unified theory," *IEEE Transactions on Signal Processing*, pp. 1–16, 2024.

[55] H. M. Ozaktas, Z. Zalevsky, and M. A. Kutay, *The Fractional Fourier Transform with Applications in Optics and Signal Processing*. New York: Wiley, 2001.

[56] C. Candan, M. Kutay, and H. Ozaktas, "The discrete fractional Fourier transform," *IEEE Transactions on Signal Processing*, vol. 48, pp. 1329–1337, 5 2000.

[57] Y. G. Jin, J. W. Shin, and N. S. Kim, "Spectro-temporal filtering for multi-channel speech enhancement in short-time Fourier transform domain," *IEEE Signal Processing Letters*, vol. 21, no. 3, pp. 352–355, 2014.

[58] Z. Průša and P. Rajmic, "Toward high-quality real-time signal reconstruction from STFT magnitude," *IEEE Signal Processing Letters*, vol. 24, no. 6, pp. 892–896, 2017.

[59] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1724–1734, Association for Computational Linguistics, Oct. 2014.

[60] M. Wolter, J. Gall, and A. Yao, "Sequence prediction using spectral RNNs," in *International Conference on Artificial Neural Networks*, pp. 825–837, Springer, 2020.

[61] S. K. Kumar, "On weight initialization in deep neural networks," *arXiv preprint arXiv:1704.08863*, 2017.

[62] T. Tieleman, G. Hinton, *et al.*, "Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[63] E. Chng, S. Chen, and B. Mulgrew, "Gradient radial basis function networks for nonlinear and nonstationary time series prediction," *IEEE transactions on neural networks*, vol. 7, no. 1, pp. 190–194, 1996.

[64] "USD/TRY Exchange Rate." `https://finance.yahoo.com/quote/USDTRY=X/`. [Accessed 25-Sep-2023].

[65] "UCI Machine Learning Respitory: Electricity Load Diagram Data Set." `https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014`. [Accessed 21-Jun-2022].

[66] M. Wolter and A. Yao, "Complex gated recurrent neural networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[68] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations (ICLR 2015)*, 2015.

[69] A. Koç, F. S. Oktem, H. M. Ozaktaş, and M. A. Kutay, *Fast algorithms for digital computation of linear canonical transforms*. Springer, 2016.

[70] X. He and Y. Peng, "Fine-grained visual-textual representation learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 2, pp. 520–531, 2019.

[71] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-100 (Canadian Institute for Advanced Research)." 2009.

[72] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (Canadian Institute for Advanced Research)." 2009.

[73] A. Trindade, "ElectricityLoadDiagrams20112014." UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86.

[74] T. Yao, Y. Pan, Y. Li, C.-W. Ngo, and T. Mei, "Wave-vit: Unifying Wavelet and transformers for visual representation learning," in *European Conference on Computer Vision*, pp. 328–345, Springer, 2022.

[75] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

[76] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[77] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, pp. 6105–6114, PMLR, 2019.

[78] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, *et al.*, "Swin transformer v2: Scaling up capacity and resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12009–12019, 2022.