On the Building of Efficient Fourier Convolutional Neural Networks

Daniel Lima-López, and Pilar Gómez-Gil, *Senior Member, IEEE*

***Abstract*—In computer vision tasks, Convolutional Neural Networks (CNN) have been the cornerstone to implement numerous models with outstanding results. Nevertheless, its high computational cost hinders its full potential when working with low-end devices. To overcome this drawback, various models have been proposed to take advantage of the properties of the convolution theorem to reduce calculations in CNN architectures. However, the use of the Fourier domain incurs in an increment of the number of weights of the kernels because they are defined using complex variable. In this work, we present a strategy to reduce such numbers by defining a reduced representation, which considers a limited amount of components to keep the parameter count comparable to that of a standard CNN. Indeed, we introduce a novel layer, called Linear Transform, that learns to identify the most essential components of a frequency representation, building an appropriate reduced representation. Our experiments on image classification using databases MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100 showed that our proposal contains fewer parameters and floating point operations, in comparison with both conventional and Fourier CNN architectures. Moreover, the Linear Transform layer produced an improvement in the accuracy in several cases. Cases where accuracy decreases are also analyzed.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/9912

***Index Terms*—CNN, convolution theorem, Fourier CNN, Linear Transform layer.**

I. Introduction

CONVOLUTIONAL NEURAL NETWORKS (CNN), originally introduced by [1] and specialized by LeCun et al. [2], have been widely used in image classification, where they stand as predominant architectures.

Despite their effectiveness, CNN architectures incur in a high computational operation cost, mainly due to convolution operations. For this reason, several works have proposed strategies to overcome this problem. Among them, some authors have designed models that take advantage of the properties of the convolution theorem [3], which states that the convolution between two signals in spatial representation is equivalent to the Hadamard product of the Fourier transform (FT) of those signals; this last requires fewer computations, due to the availability of efficient FT algorithms, as Fast Fourier Transform (FFT) [4]. Several works have been proposed using this strategy, which we will refer as "Fourier networks" for now on. For example, Mathieu et al. [5] and Lin et al. [6] proposed CNN architectures that calculate convolutions in Fourier domain, while maintaining the rest of the training process in the spatial representation.

Other proposals perform the complete training in Fourier domain incurring in smaller computational cost and producing a faster training [7], but defining all their components using spectral representations, Examples of these architectures include the works of Pratt et al. [8], Ayat et al. [9] and Han et al. [10]. On the other hand, Watanabe et al. [11] proposed a layer that focus its attention on the centre of the frequency representation of an image, and each of its four quadrants. Zack et al. [12] propose the use of a layer that exclusively processes the real part of a frequency representation. Lima-López et al [13], proposed the Butterworth-CNN architecture, which generates complex variable kernels with only 4 parameters, but focused on high resolution images. Fourier networks architectures in [8]–[11], [13] use the pooling technique proposed by Rippel et al. [7], named Spectral Pooling (SP) layer; this layer crops a central region of the frequency representation and eliminates the rest of components.

All Fourier networks face the fact that working in frequency representation requires to define kernels in complex representation, therefore the number of parameters per kernel increase considerably in comparison with CNN, going from $k \times k$ for CNN kernels to $2 \times r \times c$ for Fourier networks kernels, where k represents the CNN kernel size and $r \times c$ is the dimension of the input. This problem is critical in the first layers, where the input dimension is large, unlike standard CNN kernels, where the number of parameters may remain constant regardless of the layer or input dimensions.

To overcome this challenge, in this article we introduce the Reduced (R) and Linear Transform (LT) architectures suitable for low-resolution image classification. The first one operates using a reduced representation of the input, formed by a small amount of its frequency components, whose size remains constant across all convolutional layers, leading to a reduction in the number of parameters. On the other hand, LT-architecture uses a novel layer named Linear Transform (LT), which takes advantage of the frequency representation of the input to build an enhanced and reduced representation. Indeed, LT results in an improvement in accuracy with only a slight increase in the number of parameters and floating point operations. Both architectures exhibit a similar number of parameters per kernel as CNN, but with an important reduction in the number of floating point operations. We assessed our models by embedding them into the Fourier networks archi-

The associate editor coordinating the review of this manuscript and approving it for publication was Gladston Moreira (*Corresponding author: Pilar Gomez-Gil*).

D. Lima-López, and Pilar Gomez-Gil are with the National Institute of Astrophysics, Optics and Electronics, Tonantzintla, Puebla, México (e-mails: daniel_lima_lopez@outlook.com, and pgomez@inaoep.mx).

tectures SB-CNN proposed by Ayat et al. [9], and EF-CNN proposed by Han et al. [10]. These results were compared with a standard CNN akin to LeNet5 [2]; experiments were performed using popular low-resolution databases: MNIST [14], Fashion-MNIST [15], CIFAR-10 [16] and CIFAR-100 [17]. As it is described later, both R- and LT- architectures are designed in a way that makes them best suitable for small inputs, that is, low-resolution images. Fourier networks designed to work with high-resolution images are described elsewhere, as in [13].

The main contributions of this paper are:

- A network architecture, called R-architecture, that reduces the number of parameters and computations on convolutional layers, compared to an equivalent Fourier network architecture, without compromising accuracy and avoiding the use of SP.
- A novel layer called Linear Transform, supporting the novel LT-architecture, that obtains benefits similar to R-architecture while exhibiting an improvement in accuracy, akin to conventional CNN but better to the compared Fourier networks. This improvement comes with a slight increase in the number of parameters and computations, compared to R-architectures.

The rest of this paper is organized as follows: Section II briefly presents the main concepts involved in the design of Fourier networks. Section III presents the currently most popular models in this topic. Section IV describes the proposed architectures. Experimental results and their analysis are shown in Section V, while conclusions and future work are depicted in Section VI.

II. Main Concepts

In this section we introduce some concepts related to Fourier networks: the convolution theorem, the main architecture of a CNN based on Fourier domain and the idea of Spectral Pooling (SP).

A. Convolution Theorem

Given a pair of signals in a spatial representation x, y, and their corresponding FTs $X = \mathcal{F}(x)$, $Y = \mathcal{F}(y)$, where X and Y are frequency domain representations and $\mathcal{F}$ denotes the FT [3], the Convolutional Theorem settles that the convolution of x and y is equivalent to the Hadamard product of X and Y; in a similar way, the Hadamard product of x and y corresponds to the convolution of X and Y, that is:

$$x \circledast y = \mathcal{F}^{-1}(X \odot Y) \tag{1}$$

$$x \odot y = \mathcal{F}^{-1}(X \circledast Y) \tag{2}$$

where $\mathcal{F}^{-1}$ denotes the inverse FT (IFT), $(\circledast)$ stands for convolution and $(\odot)$ stands for Hadamard product.

For two 1D signals of length N each, the computation of FFT has a complexity $O(N \log N)$ [18]; the complexity for the Hadamard product of both signals is $O(N)$ and the calculation of the IFT shows a complexity of $O(N \log N)$. Therefore, the complexity of the whole process is $O(\max\{N \log N, N, N \log N\}) = O(N \log N)$ [10], [11], which is lower than the complexity $O(N^2)$ shown in the convolution operation performed in such signals in their spatial representation.

B. Spectral Pooling (SP)

Pooling layers, proposed by Rippel et al. [7], are a fundamental block of a Fourier CNN architecture aimed to reduce the dimension of the representation in each layer, in order to avoid the training of the network to be exclusively dependent on local spatial characteristics. SP reduces the dimension of its input by cropping the center frequencies of the frequency representation of such input. Later works have acknowledged this layer as the best pooling technique for Fourier Networks [8]–[11]. SP involves a coordinate transformation from the original FT. For a space of size $M \times N$, this transformation is given by $(u, v) \rightarrow (u - M/2, v - N/2)$ [19], where u and v correspond to positions in the input matrix. In this way, the low-frequency components are reorganized towards the center of the new representation, while components distant from the center correspond to high-frequency components.

Given an $M \times N$ input signal in its centered frequency representation, SP keeps the frequencies covered by a centered designer-defined window of dimension $H \times W$, while the rest of the frequencies are eliminated. The $H \times W$ output representation contains only information of the central region of the input representation, similar to a low-pass filter (see Fig. 1).

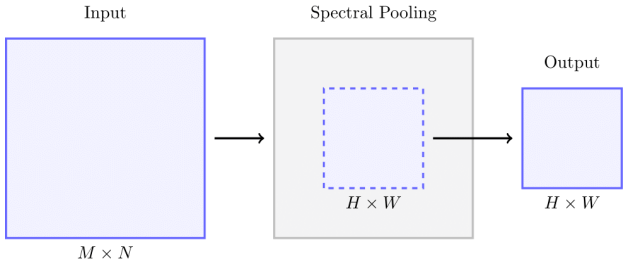

Fig. 1. SP layer mechanism as proposed by [7], illustrated in [20].

III. Fourier Convolutional Neural Networks

Fourier networks as [7]–[12], [21] use the convolution theorem to build architectures lighter than CNN but showing similar performance. These architectures are based on performing a FT on the input image, then convolution operation is replaced by the Hadamard product between the transformed input and kernels, working in complex domain. These kernels may either be defined in complex representation or obtained from the FT of spatial domain kernels. If needed, IFTs are applied in order to facilitate the use of activation functions or pooling layers in the a spatial representation. Otherwise, the rest of components of the CNN architecture, such as pooling layers and activation functions, need to be re-defined in order to operate with frequency representation.

As mentioned earlier, SP [7] is the most suitable pooling technique for frequency representation; however, other activation functions are also available. For example, Ayat et al.

[9] introduced SReLU, which approximates ReLU as a second order polynomial in frequency representation.

Wananabe et al. [11] proposed 2SReLU, which uses the second harmonics of the frequency representation. Based on polar representation, Han et al. [10], Arjovsky et al. [22] and Guberman et al. [23] proposed PhaseReLU, modRELU and zReLU. Trabelsi et al. [21] proposed an extension to ReLU, called CReLU, which handles the real and imaginary parts of the complex representation separately.

With respect to the calculation of the convolution operation, Ayat et al. [9], as well as Han et al. [10], proposed an approach akin to conventional CNN architectures, where the structure includes blocks consisting of Hadamard product computations evaluated with frequency domain activation functions, followed by SP layers, as shown in Fig. 2.

To address the challenge of reducing the number of parameters per kernel, Ayat et al. [9] suggested to avoid the computations of frequency components eliminated by each SP, and consequently, abstaining from specifying the parameters in the kernels that are excluded. However, this approach still presents unnecessary operations and parameters. On the other hand, Han et al. [10] proposed a strategy based on random kernels, which requires only two trainable parameters per kernel. Nevertheless, this incurs in high memory consumption, as it requires two non-trainable parameters per frequency component. Recently, Lima-López et al. [13] proposed an improvement in the use of memory for Fourier Networks, which uses a generation function of kernels based on Butterworth filters. In this model, each kernel is represented by four parameters only, regardless of the size of the input image. However, this approach is geared towards high-resolution images, as the generating functions need images large enough to generate a detailed Butterworth filter. Conversely, in the designs proposed in this paper, we focus on low-resolution images, as in [9], [10].

Notice that a high number of parameters per kernel persists as long as the input size of the layer is large. Moreover, the elimination of components produced by the spectral layers leads to a significant loss of information. Fig. 3 shows an example of an image reconstruction using pooling layers of different sizes. Figure 3 a) is the original image; Fig. 3 b) shows reconstructed images using different sizes, and Fig. 3 c) shows the eliminated information for each case. Notice that, when a big region of components is preserved, such as $\frac{1}{2} \times \frac{1}{2}$ or $\frac{1}{4} \times \frac{1}{4}$ of the original image, there is no noticeable loss of information. Nevertheless, in order to build architectures with a number of parameters similar to a CNN, such representation needs to be compact, but leading to a large loss of information, as shown in Fig. 3 b) when $\frac{1}{16} \times \frac{1}{16}$ and $\frac{1}{32} \times \frac{1}{32}$ are used.

IV. Proposed Architectures

In this section we introduce the new architectures. First it is detailed the implementation of convolution in frequency domain, which is used for both architectures. Next, the R-architecture, the Linear Transform layer and the LT-architectures are detailed; further points about these may be found in [20].

A. Convolution using Frequency Representation

For this, we followed an approach similar to [9]. Given an input feature map $I \in \mathbb{C}^{r \times c \times ch}$, where r, c and ch are the number of rows, columns and channels, respectively, and a target number N of output feature maps denoted each as Y_j, with $i = 1, 2, \ldots, N$. The map Y_j is characterized by a complex weight kernel $W_j \in \mathbb{C}^{r \times c \times ch}$, of the same dimension as the input feature map. The computation of each output feature map is calculated as:

$$Y_j = f\left(\sum_{c=1}^{ch} I_c \odot W_{c,j}\right) \tag{3}$$

where c iterates over the channel dimension, $\odot$ represents the complex Hadamard product and f is a non-linear complex activation function. In this work we used a rectangular complex representation, then Y_j is calculated as:

$$Y_j = f\left(\sum_{c=1}^{ch}\left[\left(I_c^{(re)} \cdot W_{c,j}^{(re)} - I_c^{(im)} \cdot W_{c,j}^{(im)}\right) + i\left(I_c^{(re)} \cdot W_{c,j}^{(im)} + I_c^{(im)} \cdot W_{c,j}^{(re)}\right)\right]\right) \tag{4}$$

where *(im)* and *(re)* are the imaginary and real part of the complex number, respectively. For subsequent layers, the N output feature maps Y_j become the channels of a new input feature map $I' \in \mathbb{C}^{r \times c \times N}$.

B. R-architectures

The R-architectures were designed taking into account that the calculation of every component of the Hadamard product is independent of the rest of components. In the original SP, components eliminated when reducing the dimension of the subsequent layer do not contribute to the learning process. Fig. 4 shows a representation of size $r_0 \times c_0$, where the components in gray color represent the information that is progressively removed through m decrements in the layers dimensions. Consequently, the only region influencing the architecture is the output derived from the final SP layer of dimension $r_m \times c_m$ (marked as blue in Fig. 4). The eliminated components represent a waste of Floating Point Operations (FLOP count) and unnecessary parameters in convolutional layers. We propose using only the set of components that genuinely contributes to the learning process, which we denominate reduced representation.

Any Fourier network may be designed using an R-Architecture as follows: first, a reduced representation is defined as the set of frequency components of the input image covered by a centered window of size $M \times N$. Here $M \times N$ corresponds to the output dimension of the last desired SP layer. The network receives as input such reduced representation, instead of the original frequency representation, as shown in Fig. 5.

The rest of the network keeps the same number of convolutional layers and their configurations, but omits all SP layers. This way, a Fourier Network architecture using SP in the original form, and its counterpart using a R-architecture

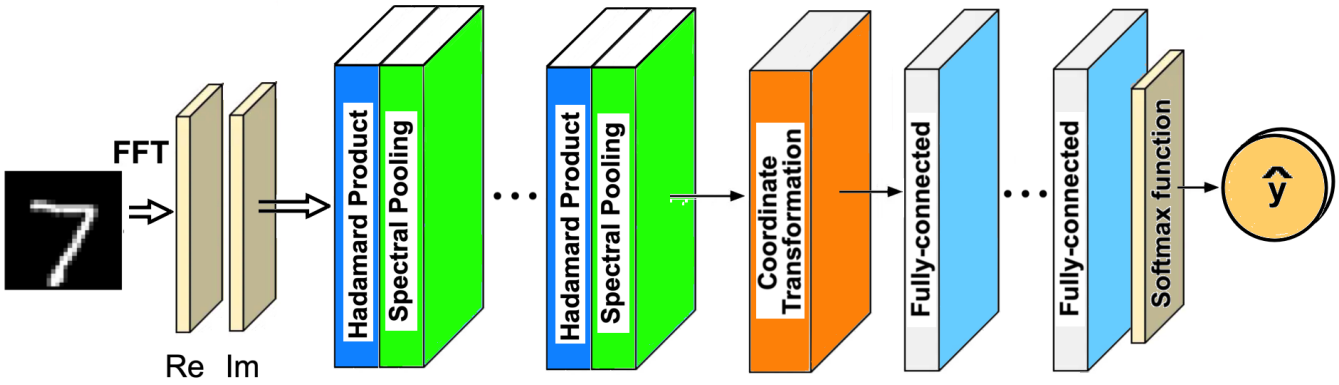

Fig. 2. A general diagram of popular CNN architectures in Fourier domain [20].

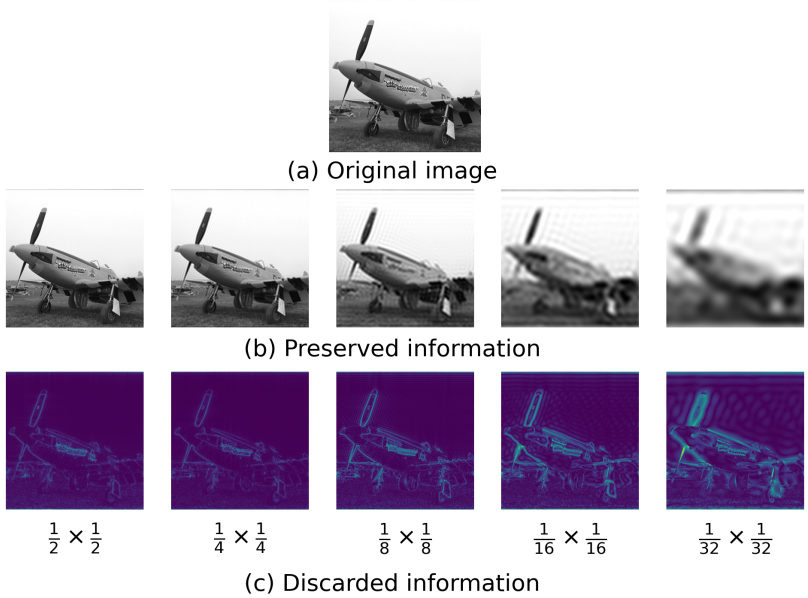

Fig. 3. Reconstruction of the image shown in (a). Part (b) shows the information preserved by SP and part (c) shows the discarded information. Sizes are depicted as a proportion of the original image [20]; image is is part of the Kodak Lossless True Color Image Suite, available at https://r0k.us/graphics/kodak/.

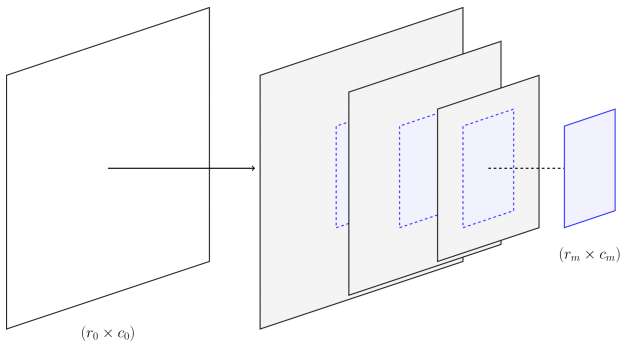

Fig. 4. Reduction of layers dimensions through m SP layers [20].

are equivalent, which guarantees the same accuracy in both designs. Notice that SP layers are not present in R-architectures (Fig. 5), as they are in Fourier networks (Fig. 2).

R-architectures maintain a constant number of parameters per kernel, independently of the layer, similar to conventional CNN filters. On the other hand, unlike to conventional CNNs, R-architectures keep constant the amount of operations per filter, independently of the layer. This advantages allows the use of a large number of filters. As with other applications based on FTs, the size of the frequency initial representation affects the performance of the network. A large set of frequency components could improve the classification accuracy, but with an increment in the number of parameters and FLOPs. Conversely, a smaller reduced representation leads to a greater reduction in the number of parameters and FLOP count, but with the risk of a significant loss of information.

C. LT-architectures

To overcome the issue of loosing information depending upon the size of the input reduced representation, the Linear Transform layer (LT) is proposed, which automatically trans-

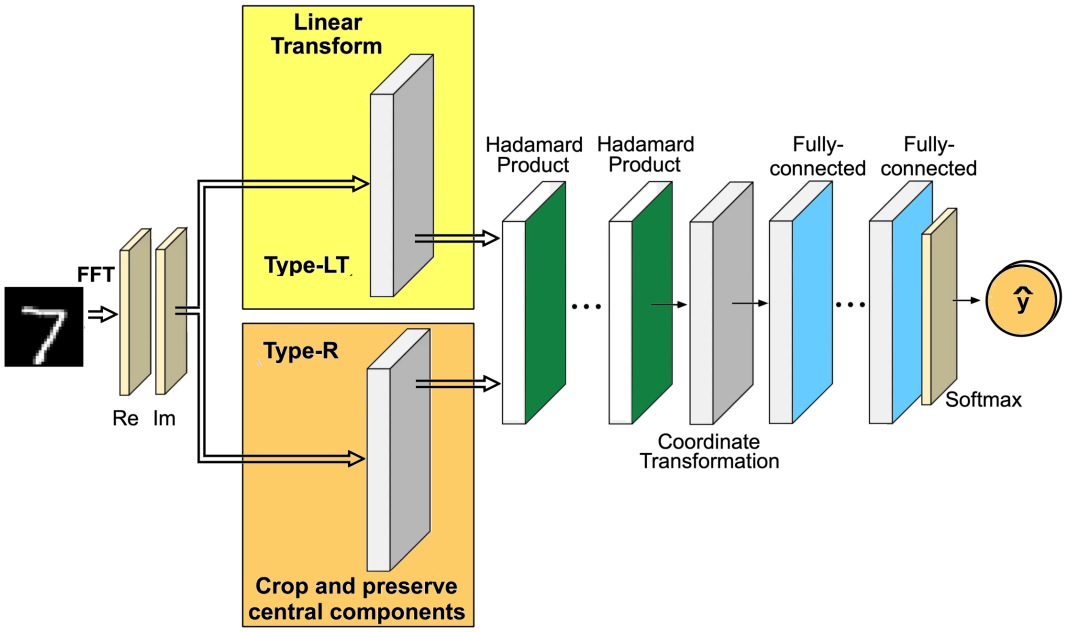

Fig. 5. The proposed Reduced (R) and Linear Transform (LT) architectures [20].

forms an input representation into a smaller one. Fig. 6 depicts the LT process.

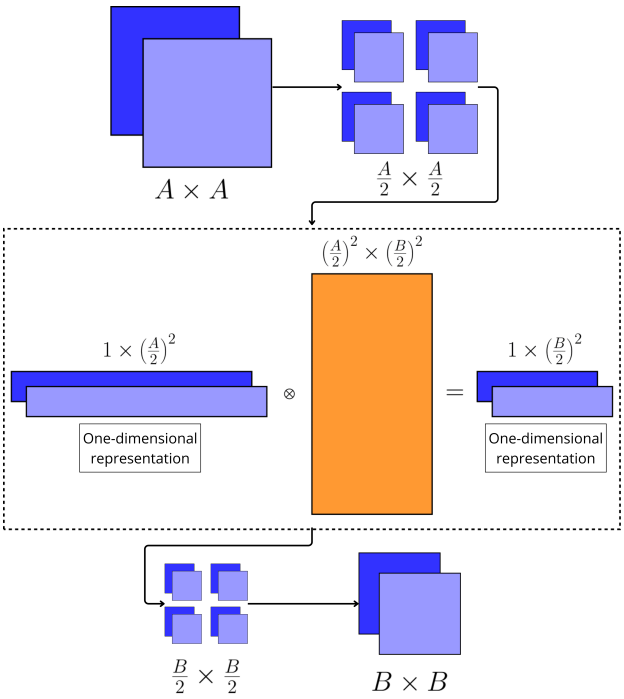

Fig. 6. The Linear Transform layer mechanism per quadrant [20].

Given an initial $A \times A$ representation and a $B \times B$ target representation, $B < A$, LT splits the input representation into quadrants, each $\frac{A}{2} \times \frac{A}{2}$ dimension. Each quadrant is converted to its one-dimensional complex representation of size $1 \times \left(\frac{A}{2}\right)^2$, which is transformed by a $\left(\frac{A}{2}\right)^2 \times \left(\frac{B}{2}\right)^2$ real matrix, initialized with trainable parameters. The result of this process is a one-dimensional complex representation of size $1 \times \left(\frac{B}{2}\right)^2$; next this is transformed into a representation of size $\frac{B}{2} \times \frac{B}{2}$. Finally, the four resulting quadrants are combined to form a representation of size of $B \times B$, which we denominate enhanced reduced representation. This approach applies a separate transformation to each quadrant of the frequency representation, since the coefficients within each quadrant encode similar types of variations in the input signal, and isolating them avoids mixing unrelated spectral information. Moreover, experimental results demonstrates that this design is more effective than performing a single transformation over the entire input, requiring fewer parameters.

The LT-architectures share the same structure as R-architectures, but the first one leverage all frequency components of the input to build an enhanced reduced representation with aid of Linear Transform layer, as shown in Fig. 5. During training, LT layer learns a suitable linear transform able to produce an enhanced reduced representation containing the most relevant information and improving the classification accuracy, with a slight increase in the number of parameters and FLOPs

V. Experimental Results

This section presents the experiments performed to assess R and LT-architectures. First, it is described the experimental setup, followed by a comparative analysis of their performance when compared to other methods. Last, we present an analysis of the reductions in the number of parameters and FLOP count.

A. Experimental Setup

The proposed techniques were evaluated on datasets MNIST [14], Fashion-MNIST [15], CIFAR-10 [16]. An additional accuracy analysis was carried out using CIFAR-100 [17].

Both MNIST and Fashion-MNIST images have a dimension of 28×28 and the dataset include $60,000$ training images and $10,000$ test images in gray scale; CIFAR-10 contains images of size 32×32, with $50,000$ training images and $10,000$ test images, for this case, experiments were performed in gray scale and RGB, denoted as CIFAR-10G and CIFAR-10RGB, respectively. Each of these datasets contains 10 evenly distributed classes. All experiments were implemented using Tensorflow 2.10, and executed in a GeForce RTX 3060 with 12 GB of VRAM, coupled with an AMD Ryzen 5 3600 with 16 GB of RAM.

To assess our proposals, experiments were conducted in both spatial and frequency domains, all following a general structure similar to LeNet-5 [2]. For the network using spatial representation, the architecture is as LeNet-5, denoted as CNN on the following tables. As base Fourier networks, we implemented Efficient Fourier CNN (EF-CNN) proposed by Han et al. [10] and Spectral Based CNN (SB-CNN) proposed by Ayat et al. [9]. Our proposed R and LT-architectures were embedded on that Fourier models, referred as R-EF-CNN and LT-EF-CNN for EF-CNN and R-SB-CNN and LT-SB-CNN for SB-CNN. All architectures were trained using a Stochastic Gradient Descent (SGD) optimizer, with a learning rate 0.0005, a momentum of 0.9 and a batch size of 128. Details of architectures are next:

- **CNN.** This network receives as input the raw image in spatial representation. It contains 3 convolutional layers, with 6, 16 and 120 filters respectively, each using 5×5 kernels and ReLU as activation function, followed by a MaxPooling layer. After these, a flatten layer connecting to a Batch Normalization layer is included, followed by two fully connected layers with 84 and 10 neurons, evaluated with ReLU and Softmax, respectively.
- **EF-CNN and SB-CNN.** Both networks receive as input the frequency representation of images. A 2D centered FT was applied to gray scale images, while for RGB images the same transformation was applied to each channel separately. Convolutional layers are similar to the CNN, with the convolution being performed in frequency representation (see Section IV-A); complex variable kernels are defined as in [10] and [9], for EF-CNN and SB-CNN, respectively; pooling layers are performed using SP, and CReLU [21] is the activation function involved in convolutional layers. Following the third SP layer, both architectures perform a coordinate transformation. EF-CNN calculates the square magnitude of the complex representation, while SB-CNN computes the real part of the Inverse FFT (IFFT). The rest of layers are as in CNN.
- **R-EF-CNN and R-SB-CNN.** Since the base architectures CNN, EF-CNN and SB-CNN present a 4×4 size after the third pooling layer, both architectures receive a 4×4 reduced representation of the image as input (see Section IV-B) in order to keep the most similar consistency in all models. These architectures present the same configuration as their corresponding base architectures EF-CNN and SB-CNN, with the exception that SP layers are excluded.
- **LT-EF-CNN and LT-SB-CNN.** These architectures receive the complete centered FT of the image as input, which is transformed into an enhanced reduced representation of size 4×4 by a LT layer. Subsequently, the rest of the architecture works with this representation, following the same layer configuration as the previously described R-architectures.

B. Experiments with R and LT-architectures

Two experiments are presented. The aim of the first one was to analyze the performance of R and LT-architectures in comparison with the obtained by the base Fourier networks EF-CNN and SB-CNN. The second experiment assessed the performance of the proposed architectures in comparison with CNN, analyzing the impact of increments in the size of the input representation of R and LT-architectures. Each experiment was executed 10 times, with 50 epochs in each running. In the first experiment, the standard partitions of training and test sets of each dataset were used.

Figs. 7 and 8 show the average test accuracy obtained by EF-CNN and SB-CNN respectively; shaded regions indicate standard deviations (SD). Notice that for all cases, the accuracy of both base Fourier networks and R-architectures overlapped throughout the entire training, and their SD are within a comparable range. These observations show empirical evidence that the base and modified R-architectures are equivalent.

Figs. 7 and 8 show that LT-architecture exhibited a better accuracy than base and R-architectures. This enhancement may be due to the fact that LT layer uses all available input information to create an improved reduced representation. Notice that, R-architectures used regions of size 4×4, which represents only a little more than 2% of components of the original region of size 28×28 for MNIST and Fashion-MNIST, and 32×32 for CIFAR-10.

For the second experiment, each dataset was divided as 70% for training, 10% for validation, and and 20% for testing. Table I shows the mean test accuracy obtained by 10 executions using the parameter configuration of the architecture in the epoch that yielded the highest validation accuracy. Results shown in the third column of this table were performed using the configuration described in Section V-A, and are identified as 4×4, reflecting the dimension prior to the flattening layer. Results shown in fourth column were obtained using 8×8 R- and LT-architectures, that is, LT-architectures produce a 8×8 enhanced reduced representation with the first LT layer. The third Max Pooling layer in the CNN was eliminated in order to built a 8×8 representation.

Table I shows that, for 4×4 dimension, the LT-architectures outperformed their equivalent R-architecture, for both EF-CNN and SB-CNN configurations, which may be due to the ability of LT layers to create a more representative reduced representation than R-architectures. It is also noticed that SB-CNN obtained better results than EF-CNN, achieving an accuracy akin to CNN for MNIST and Fashion-MNIST datasets. However, it is noticed a significant difference in accuracy among these architectures when CIFAR-10 is used, mainly when using gray scale images, suggesting that as expected,

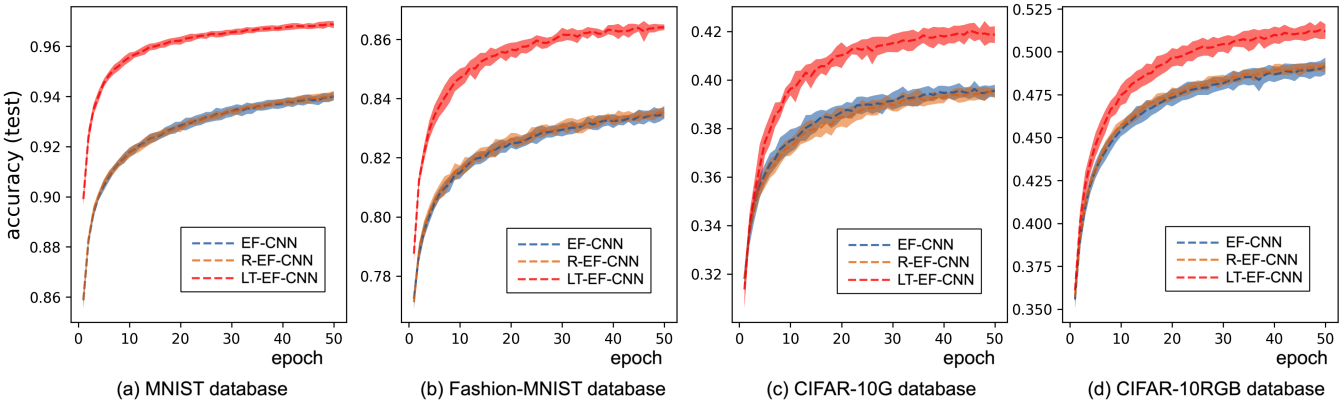

Fig. 7. Mean accuracy on test set using EF-CNN architectures, for the following databases: (a) MNIST, (b) Fashion-MNIST, (c) CIFAR-10G and (d) CIFAR-10RGB.

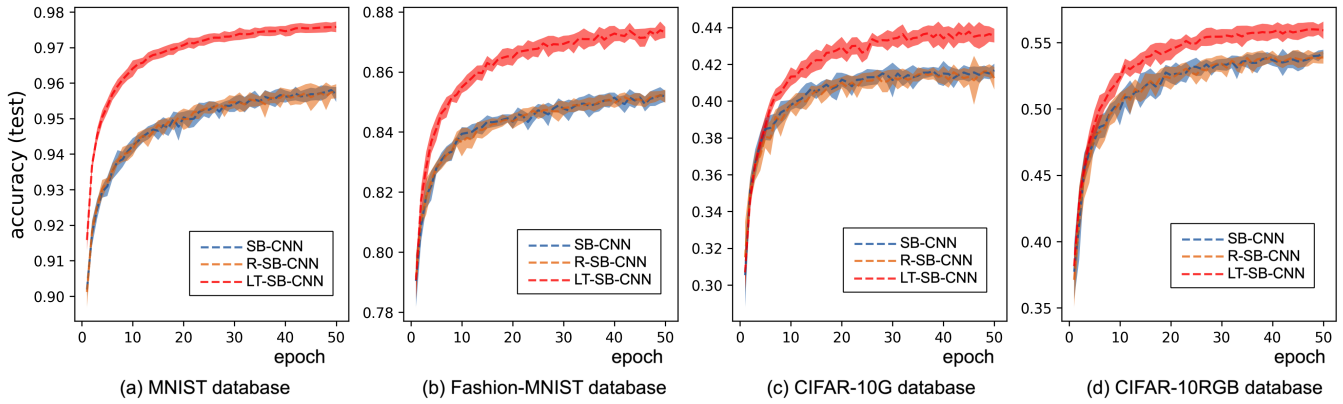

Fig. 8. Mean accuracy on test set using SB-CNN architectures, for the following databases: (a) MNIST, (b) Fashion-MNIST, (c) CIFAR-10G and (d) CIFAR-10RGB.

gray image representation contains less information than color image representation.

For 8×8 R-architectures, there is an increment in accuracy with respect to 4×4, which must probably was caused by the incorporation of more information. However, there is not a significant difference in accuracy between 8×8 R-architectures and their equivalent 4×4 LT-architectures. With respect to 4×4 dimensions, it is evident an increment on the accuracy with LT-architectures, compared to R-architectures. However, this is not observed with 8×8 architectures. This may be because a reduced representation of size 8×8 still contains much of the total information, therefore, Linear Transform layer does not add significant information. Furthermore, since the accuracy of 4×4 LT-architectures is similar to that of 8×8 R-architectures, it may be inferred that LT-layers are capable of generating an enhanced 4×4 reduced representation containing almost the same relevant information as a 8×8 reduced representation.

Additional experiments were conducted using CIFAR-100 [17] to evaluate our models for natural images involving 100 classes; this database contains 32x32 600 images. Results are shown in Table II, noticing an important difference in accuracy when using gray scale with respect to RGB images; indeed CNN presents better results than our proposals. These results are similar to the ones obtained with CIFAR-10, where it is also observed a slight increase in accuracy when working with a higher resolution images, therefore it is inferred that a notable amount of information is lost when working with small frequency representations of natural images; however there is an evident trade-off between accuracy and memory consumption, since a bigger frequency representation would requires a greater amount of parameters to represent their kernels.

Finally, an experiment using CIFAR-10RGB was performed to test the capabilities of LT layer when working with natural images, adding non-linear activation functions in order to capture more complex patterns. LT-SB-CNN architecture with 4×4 and 8×8 output sizes for the LT layer was used, using the same experimental setup as previous experiments. Table III shows the results with the original architecture with no activation function, using CReLU and using modReLU. No improvements are showed in the performance of the model when non-linear activations functions are added.

In summary, R-architectures showed a very similar accuracy as their equivalent base Fourier networks, while LT-architecture showed an improvement. However, a significant difference in accuracy is observed when comparing our proposals with CNN, mainly on natural images. On the other hand, 8×8 R-architectures and 4×4 LT-architectures presented

TABLE I
MEAN TEST ACCURACY ON EACH DATASET WITH THE PROPOSED ARCHITECTURES AND CNN FOR TWO DIFFERENT REPRESENTATION SIZES [20]

Dataset	Architecture	4×4	8×8
MNIST	R-EF-CNN	0.938 ± 0.001	0.972 ± 0.001
	LT-EF-CNN	0.965 ± 0.001	0.974 ± 0.001
	R-SB-CNN	0.956 ± 0.001	0.978 ± 0.001
	LT-SB-CNN	0.973 ± 0.001	0.977 ± 0.001
	CNN	0.988 ± 0.001	0.989 ± 0.000
Fashion-MNIST	R-EF-CNN	0.841 ± 0.001	0.877 ± 0.001
	LT-EF-CNN	0.871 ± 0.002	0.881 ± 0.002
	R-SB-CNN	0.861 ± 0.002	0.888 ± 0.001
	LT-SB-CNN	0.882 ± 0.002	0.889 ± 0.001
	CNN	0.907 ± 0.002	0.910 ± 0.002
CIFAR-10G	R-EF-CNN	0.386 ± 0.002	0.410 ± 0.005
	LT-EF-CNN	0.405 ± 0.003	0.427 ± 0.004
	R-SB-CNN	0.406 ± 0.004	0.441 ± 0.003
	LT-SB-CNN	0.424 ± 0.002	0.447 ± 0.003
	CNN	0.629 ± 0.011	0.629 ± 0.015
CIFAR-10RGB	R-EF-CNN	0.478 ± 0.004	0.488 ± 0.004
	LT-EF-CNN	0.495 ± 0.004	0.519 ± 0.005
	R-SB-CNN	0.530 ± 0.003	0.545 ± 0.003
	LT-SB-CNN	0.549 ± 0.005	0.560 ± 0.005
	CNN	0.638 ± 0.013	0.644 ± 0.015

TABLE II
MEAN TEST ACCURACY ON CIFAR-100 WITH THE PROPOSED ARCHITECTURES AND CNN FOR TWO DIFFERENT REPRESENTATION SIZES

Dataset	Architecture	4×4	8×8
CIFAR-100G	R-EF-CNN	0.123 ± 0.002	0.134 ± 0.003
	LT-EF-CNN	0.132 ± 0.002	0.151 ± 0.003
	R-SB-CNN	0.139 ± 0.002	0.153 ± 0.003
	LT-SB-CNN	0.148 ± 0.003	0.165 ± 0.003
	CNN	0.263 ± 0.010	0.269 ± 0.007
CIFAR-100RGB	R-EF-CNN	0.202 ± 0.004	0.213 ± 0.004
	LT-EF-CNN	0.213 ± 0.003	0.246 ± 0.003
	R-SB-CNN	0.249 ± 0.005	0.263 ± 0.002
	LT-SB-CNN	0.265 ± 0.005	0.283 ± 0.005
	CNN	0.314 ± 0.008	0.324 ± 0.011

similar performance. Consequently, there is no apparent justification for using larger representations, since 4×4 architectures performed similarly while maintaining a lower number of parameters and FLOP count, as explained in the next section.

C. Parameters Count and Floating-point Operations

In this section we present an analysis of the number of parameters and FLOP count produced by CNN, base Fourier networks and the proposed architectures. This analysis is conducted using the setup described in Section V-A. R-architectures used a representation of size 4×4. For CNN and base Fourier networks, we used an input size of 28×28, matching the dimensions of MNIST and Fashion-MNIST. Similar results are observed using CIFAR-10 and CIFAR-100 datasets, despite its slightly larger size of 32×32. The number of parameters was calculated using built-in methods available on Keras; the FLOP count was calculated using the Profiling API provided by TensorFlow.

Tables IV and V detail the number of parameters and FLOP count for each convolutional layer; best results are highlighted. Tables VI and VII show the reduction in the number of parameters and FLOP count per convolutional layer, comparing R-architectures with CNN and base Fourier networks.

TABLE III
MEAN TEST ACCURACY ON CIFAR-100RGB WITH LT-SCB-CNN USING DIFFERENT ACTIVATION FUNCTIONS ON LINEAR TRANSFORM LAYER

Activation	4×4	8×8
No-Activation	0.549 ± 0.005	0.560 ± 0.005
CReLU	0.524 ± 0.006	0.551 ± 0.004
modReLU	0.550 ± 0.003	0.563 ± 0.003

TABLE IV
NUMBER OF PARAMETERS PER LAYER USING A DATASET WITH AN INPUT OF 28×28 FOR CNN, EF-CNN, AND SB-CNN, AND A REDUCED REPRESENTATION SIZE OF 4×4 FOR R-EF-CNN AND R-SB-CNN [20]

Layer	CNN	EF-CNN	SB-CNN	R-EF-CNN	R-SB-CNN
Convolution 1	**156**	9,420	2,364	204	204
Convolution 2	**2,416**	37,824	9,440	3,264	3,104
Convolution 3	**48,120**	192,000	61,680	65,280	61,680
Total	**50,692**	239,244	73,484	68,748	64,988

TABLE V
FLOP COUNT PER LAYER USING A DATASET WITH AN INPUT OF 28×28 FOR CNN, EF-CNN, AND SB-CNN, AND A REDUCED REPRESENTATION OF 4×4 FOR R-EF-CNN AND R-SB-CNN [20]

Layer	CNN	EF-CNN	SB-CNN	R-EF-CNN	R-SB-CNN
FFT	-	15,076	15,076	15,076	15,076
Convolution 1	239,904	37,632	9,408	**768**	**768**
Pooling 1	4,704	0	0	-	-
Convolution 2	943,936	181,888	37,632	14,848	**12,288**
Pooling 2	3,136	0	0	-	-
Convolution 3	4,709,880	929,040	245,760	303,360	**245,760**
Pooling 3	7,680	0	0	-	-
Transformation	-	5,760	41,882	5,760	41,882
Total	5,909,240	1,169,396	349,758	339,812	**315,774**

TABLE VI
REDUCTION IN PARAMETERS OF R-ARCHITECTURES USING A REPRESENTATION OF 4×4, COMPARED TO CNNS AND BASE FOURIER NETWORKS, WITH AN INPUT OF 28×28 [20]

Layer	R-EF-CNN			R-SB-CNN		
	CNN	EF-CNN	SB-CNN	CNN	EF-CNN	SB-CNN
Convolution 1	$\times 0.765$	$\times 46.176$	$\times 11.588$	$\times 0.765$	$\times 46.176$	$\times 11.588$
Convolution 2	$\times 0.740$	$\times 11.588$	$\times 2.892$	$\times 0.778$	$\times 12.186$	$\times 3.041$
Convolution 3	$\times 0.737$	$\times 2.941$	$\times 0.945$	$\times 0.780$	$\times 3.113$	$\times 1.000$
Total	$\times 0.737$	$\times 3.480$	$\times 1.069$	$\times 0.780$	$\times 3.681$	$\times 1.131$

Table IV shows a significant reduction in the number of parameters among base Fourier networks architectures and their corresponding R-architectures. However, the reduction in SB-CNN is not as large as in EF-CNN. Indeed, the parameter count in R-architectures is comparable to that of CNN for all layers; this similarity arises because the representation size remains constant, leading to a consistent number of parameters per kernel, comparable to that of 5×5 CNN kernels. Notice

TABLE VII
REDUCTION IN FLOP COUNT OF R-ARCHITECTURES USING A REDUCED REPRESENTATION SIZE OF 4×4, COMPARED TO CNN AND BASE FOURIER NETWORKS, WITH AN INPUT SIZE OF 28×28 [20]

Layer	R-EF-CNN			R-SB-CNN		
	CNN	EF-CNN	SB-CNN	CNN	EF-CNN	SB-CNN
Convolution 1	$\times 312.375$	$\times 49.000$	$\times 12.250$	$\times 312.375$	$\times 49.000$	$\times 12.250$
Convolution 2	$\times 63.573$	$\times 12.250$	$\times 2.534$	$\times 76.818$	$\times 14.802$	$\times 3.062$
Convolution 3	$\times 15.526$	$\times 3.062$	$\times 0.810$	$\times 19.165$	$\times 3.780$	$\times 1.000$
Total	$\times 17.390$	$\times 3.441$	$\times 1.029$	$\times 18.714$	$\times 3.703$	$\times 1.108$

in Table VI that the reduction factor between R-architectures and CNNs is lower than 1, meaning that the proposed architectures still present a larger number of parameters than CNNs. Nevertheless, a significant reduction factor is observed among R-architectures and their corresponding base Fourier networks. Moreover, the reduction factor between R-architecture and CNN remains uniform across all convolutional layers, since both architectures present a constant number of parameters per kernel. In contrast, the reduction factor among R-architecture and its equivalent base Fourier networks decrease with each convolution layer. This is due to the larger representation size on the first layers, which requires more parameters per kernel.

Additionally, on Table V, it is observed an important reduction in the FLOP count on R-architectures, in comparison with its equivalent Fourier network architecture. This reduction is larger when R-architectures are compared to CNN. Similarly, the reduction factor in FLOP count, observed in Table VII, decreased across each layer, because more operations are needed for bigger representations on first layers. The total reduction factor presents a substantial overall reduction factor for R-architectures when compared to CNN, with $\times 17.390$ and $\times 18.714$ for EF-CNN and SB-CNN, respectively.

Taking into account the total reduction factors of R-architectures, it is reasonable to include additional layers, such as Linear Transform, to build LT-architectures. In this scenario, with an input representation of size 28×28 and an objective reduced representation of size 4×4, LT layer requires only 3,136 parameters and 12,544 FLOPs. The inclusion of this layer does not lead to a significant decrease in the overall reduction factors for parameters and FLOP count, and shows a significant increase in accuracy, as mentioned in Section V-B.

In summary, 4×4 R and LT-architectures present an important reduction in the number of parameters and FLOP count, compared to base Fourier networks and CNN. Furthermore, they achieve comparable accuracy to 8×8 architectures (see Section V-B). Notably, 8×8 architectures require $\times 4$ the number of parameters of 4×4 architectures. Consequently, 4×4 architectures are a better option, since they present a low number of parameters and FLOP count while maintaining comparable accuracy to larger-size architectures.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we present an strategy to reduce both the number of parameters and FLOP per kernel on Fourier networks. This is achieved by the introduction of the R-architecture, which works with a reduced representation of the FT of the input. The implementation of this technique comes at the expense of a significant information loss. To overcome this issue, the LT-architecture is proposed, which includes a novel layer called Linear Transform, that dynamically builds an enhanced reduced representation taking in account all available information.

Experimental results on MNIST, Fashion-MNIST and CIFAR-10, using a reduced representation of size 4×4, showed that R-architectures obtained similar accuracy as base Fourier networks, with no need of pooling layers. Moreover, R-architectures present a significant reduction in the number of parameters and the FLOP count. Notably, the reduction in FLOP count is even more pronounced when comparing R-architectures with CNN. Furthermore, the R-architectures presented a similar number of parameters per kernel as 5×5 CNN kernels.

On the other hand, the LT-architectures presented the same benefits in terms of parameters and FLOP count as R-architectures, but with an increase in classification accuracy, requiring only a slight increase in the number of parameters and FLOP count. The accuracy achieved with these architectures is similar to that obtained by CNN on the MNIST and Fashion-MNIST datasets. However, there is a notable drawback in the accuracy of our proposed architectures when classifying natural images, as observed in the CIFAR-10 and CIFAR-100 datasets. To address this issue, as future work, we propose to design complex variable kernels and pooling techniques than can better represent the complexity involved in these kind of images as well as to study how to leverage different characteristics of the frequency representation of these kind of images. Moreover, it is required to develop new data augmentation techniques. For which, it should be considered that current data augmentation approaches are not suitable for Fourier Networks, as such augmentation models are based on geometric transformations, which are invariant under the Fourier transformation, meaning that no variations in relevant geometric information would be generated with current techniques. Lastly, it could be interesting to test R- and LT-architectures using 1-D signals extending the scope to other domains beyond visual images. However, it is required to consider that a meaningful 1D to 2D transformation should be applied to the input data, in order to leverage the locally-connected spatial relations, obtained by convolutions in CNN-style architectures.

Code metadata. The code required to execute the experiments showed here, as well as other implementation details are available at: https://github.com/daniel-lima-lopez/R-and-LT-architectures

REFERENCES

[1] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989, DOI:10.1162/neco.1989.1.4.541.

[3] J. G. Proakis, *Digital signal processing: principles, algorithms, and applications*, 4th ed. Chennai, India: Pearson Education India, 2007, ISBN:0131873741.

[4] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965, https://doi.org/10.2307/2003354.

[5] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," in *2nd International conference on learning representations, ICLR 2014*, 2014, https://doi.org/10.48550/arXiv.1312.5851.

[6] J. Lin, L. Ma, and Y. Yao, "A Fourier domain acceleration framework for convolutional neural networks," *Neurocomputing*, vol. 364, pp. 254–268, 2019, https://doi.org/10.1016/j.neucom.2019.06.080.

[7] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," *Advances in neural information processing systems*, vol. 28, 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/536a76f94cf7535158f66cfbd4b113b6-Paper.pdf

[8] H. Pratt, B. Williams, F. Coenen, and Y. Zheng, "FCNN: Fourier convolutional neural networks," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I 17*. Springer, 2017, pp. 786–798. [Online]. Available: https://doi.org/10.1007/978-3-319-71249-9_47

[9] S. O. Ayat, M. Khalil-Hani, A. A.-H. Ab Rahman, and H. Abdellatef, "Spectral-based convolutional neural network without multiple spatial-frequency domain switchings," *Neurocomputing*, vol. 364, pp. 152–167, 2019, https://doi.org/10.1016/j.neucom.2019.06.094.

[10] Y. Han and B.-W. Hong, "Deep learning based on Fourier convolutional neural network incorporating random kernels," *Electronics*, vol. 10, no. 16, p. 2004, 2021, https://doi.org/10.3390/electronics10162004.

[11] T. Watanabe and D. F. Wolf, "Image classification in frequency domain with 2SReLU: a second harmonics superposition activation function," *Applied Soft Computing*, vol. 112, p. 107851, 2021, https://doi.org/10.1016/j.asoc.2021.107851.

[12] J. Zak, A. Korzynska, A. Pater, and L. Roszkowiak, "Fourier transform layer: A proof of work in different training scenarios," *Applied Soft Computing*, vol. 145, p. 110607, 2023, https://doi.org/10.1016/j.asoc.2023.110607.

[13] D. Lima-López and P. Gómez-Gil, "Butterworth cnn: an improvement on memory use for Fourier Convolutional Neural Networks," in *2024 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, vol. 8, 2024, pp. 1–6, DOI:10.1109/ROPEC62734.2024.10877077.

[14] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012, DOI:10.1109/MSP.2012.2211477.

[15] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017, https://doi.org/10.48550/arXiv.1708.07747.

[16] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical report*, 2009. [Online]. Available: https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf

[17] ——, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[18] R. C. Singleton, "On computing the Fast Fourier Transform," *Communications of the ACM*, vol. 10, no. 10, pp. 647–654, 1967. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/363717.363771

[19] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011, ISBN:0470844736.

[20] D. Lima-López, "Clasificación de imágenes con arquitecturas ligeras de redes neuronales en el espacio de Fourier," Master's thesis, Computer Science, 2024. [Online]. Available: http://inaoe.repositorioinstitucional.mx/jspui/handle/1009/2612

[21] T. Chiheb, O. Bilaniuk, D. Serdyuk *et al.*, "Deep complex networks," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1705.09792

[22] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *International conference on machine learning*. PMLR, 2016, pp. 1120–1128. [Online]. Available: https://proceedings.mlr.press/v48/arjovsky16.html

[23] N. Guberman, "On complex valued convolutional neural networks," *arXiv preprint arXiv:1602.09046*, 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1602.09046

Daniel Lima-López received a Bachelor degree in Physics from the Benemérita Universidad Autónoma de Puebla and a master degree in Computer Science in the National Institute of Astrophysics, Optics, and Electronics (INAOE), México. His research interests include artificial neural networks focused on deep learning applications.

Pilar Gómez-Gil received a PhD. in computer science from Texas Tech University. She is currently a Titular Researcher in the Computer Science Department at the National Institute of Astrophysics, Optics, and Electronics (INAOE), México. Her research interests include artificial neural networks, time series prediction, signal processing, and pattern recognition with applications in medicine, finances and cosmology.