Accelerating Convolutional Neural Network Using Discrete Orthogonal Transforms

Eduardo Reis, *Student Member, IEEE*, and Rachid Benlamri, *Member, IEEE*

Abstract—Novel approaches based on spectral domains were proposed over the past ten years in order to accelerate Convolutional Neural Networks (CNNs) by reducing the number of operations required. These works mostly rely on the convolution theorem of the Fourier transform. In this paper, we present the use of spectral domain beyond the Fourier transform for computing convolution operations. First, we review how the domains produced by the Fourier, Hartley and Cosine transforms, which are part of a family of discrete orthogonal transforms, have been used in the literature to design different deep neural network architectures. Then, we investigate how each of these transforms can be used to design CNN models as an alternative to the spatial domain convolution to speedup computation. The proposed orthogonal transforms are benchmarked and their performance is evaluated under the same CNN architectures. A detailed comparative study of these transforms in terms of their learning capabilities and computational complexity is presented. The experimental results show that the Cosine transform yields a superior learning than the Hartley transform for accelerating CNN via a multiplication on the spectral domain. In addition, among the methods that don't exhibit a speed-accuracy trade-off, the Hartley transform presents the best speedup while using the convolution theorem on its spectral domain.

Index Terms—Convolutional neural network, Discrete orthogonal transform, Fourier transform, Hartley transform, Cosine transform, Deep learning.

1 INTRODUCTION

CONVOLUTIONAL Neural Networks (CNNs) became popular over the last decade, enabling models with better than human performance in various fields of pattern recognition such as image classification, image enhancement and object detection. The complexity of the task to be solved drives the number of neurons (i.e. kernels or weights) to be used per layer, as well as the depth of the network. Thus, incrementing these hyperparameters directly increases the number of convolution operations to be computed. This is a common concern to sophisticated architectures used to solve computer vision problems, as these models can take a considerable amount of time to be trained. CNN models [1], [2] that take hours for training using small samples of data, could easily require significantly more time to search for hyperparameters, not to mention the time required to train the final state-of-the-art models [3] which use the entire available data.

Despite having a greater generalization capability for learning models from images, convolutional layers require more operations per neuron when compared with fully connected layers. In comparison, for a given image of size N by N , a neuron of a fully connected layer would compute $\mathcal{O}(N^2)$ multiplications (FLOPs), whereas a neuron of a convolutional layer would compute $\mathcal{O}(M^2K^2)$ FLOPs per output of size M by M and a given kernel of size K by K , where $M = N - K + 1$.

Many approaches have been proposed to accelerate CNNs [4], [5]. In the field of digital signal processing, the convolution theorem property of the Discrete Fourier Trans-

form (DFT) is well known as an alternative for the spatial domain convolution. It yields a faster filtering operation when combined with the Fast Fourier Transform (FFT) algorithm. It was first introduced to speedup the classification time of filtering based models [6], in which a Multi-Layer Perceptron is used as the kernel function of a “convolution-like” computational model for face detection.

The use of the FFT to accelerate the training of CNN models was first demonstrated in [7], [8]. The cost of filtering an image by a kernel using the FFT approach is approximated by $3 \times 2CN^2 \log N + 4N^2$ FLOPs, where $2CN^2 \log N$ is the cost for the 2-D FFT of a given image of size N by N . In this estimate, 3 accounts for two forward transforms to obtain the representations of both the image and the filter kernel in the Fourier domain, and one inverse transform to convert the product of these representations back to the spatial domain. Finally, C is a hidden constant in the FFT complexity and $4N^2$ is the cost for the complex element-wise multiplication on the DFT domain. Figure 1 shows a theoretical comparison between both approaches for filtering a single image. As it can be noticed, the FFT based approach yields more operations than the standard spatial domain convolution up to kernels of size $K = 7$.

The FFT based approach is more efficient when processing multiples images at once, avoiding redundant computation. This is the case for applications that uses mini-batches, such as training neural networks. In this approach, the spectral representation of the filters is shared across all images of a given mini-batch. Having the kernel and image sizes defined, respectively, as K by K and N by N ; let B , f_1 , f_2 , be defined, respectively, as the number of images per mini-batch, channels per image, kernels (neurons) per layer. In Figure 2, we observe the growth in the number of FLOPs for scenarios similar the one of forward pass of first

• E. Reis (edreis@lakeheadu.ca) and R. Benlamri (rbenlamr@lakeheadu.ca) are with the Department of Software Engineering of Lakehead University, Ontario, Canada.

Figure 1: Comparison between the theoretical number of multiplication operations of the FFT based approach and spatial domain convolution for image filtering with different kernel sizes, where $K = \{3, 5, 7, 9, 11\}$.

convolution layer of the LENET-5 [9] model by varying the image size in Figure 2a; the mini-batch size in Figure 2b; and the kernel size in Figure 2c. These comparison shows that the FFT based approach is generally more efficient than the spatial domain convolution for image filtering for kernel as small as size $K = 5$. For scenarios that might actually be quite common, such as the use of small kernels, $K = 3$, combined with small image sizes, $N = 32$, and a small number of input and/or output channels, the FFT approach for filtering images can actually be more expensive even when combined with a mini-batch approach. Also, it can be seen that the FFT based approach is invariant to the kernel size K , given the kernel must be padded to match the image size as shown in Figure 2c. The curve “FC” refers to a fully connected layer used as a lower bound reference. The parameters used in Figure 2 are as follows: Figure 2a has $B = 128$, $f_1 = 3$ and $f_2 = 6$; Figure 2b has $f_1 = 3$, $f_2 = 6$ and $N = 32$; and Figure 2c has $B = 128$, $f_1 = 3$, $f_2 = 6$ and $N = 32$. Though in Figure 2 we observe the spatial convolution with kernel of size $K = 3$ outperforming the FFT based approach, that is not always the case. Figure 3 shows that the performance of the FFT based approach outperforms spatial convolution of size $K = 3$ for larger values of f_1 , f_2 and N . The parameters used in Figure 3 are as follows: Figure 3a has $B = 128$, $f_1 = 96$ and $f_2 = 256$; Figure 3b has $f_1 = 96$, $f_2 = 256$, and $N = 128$; and Figure 3c has $B = 128$, $f_1 = 96$, $f_2 = 256$ and $N = 128$.

Other transforms have also been used to design Deep Neural Network (DNN) models, such as the Hartley Transform [10], [11] and the Cosine Transform [12], [13], [14], [15]. These transforms are part of a family of Discrete Orthogonal Transforms (DOTs), also known as spectral transforms. The spectral domain representation of an image can then be obtained by applying a given 2-D DOT to its spatial domain representation.

Recent review papers on the advances of CNNs, such as [4] and [5], present the use of frequency domain to compute the convolutions of CNNs. These works focus mainly on the domain obtained by the Fourier transform to speedup CNN operations, without further investigating the possibility of using spectral domains produced by other transforms. Furthermore, other works have demonstrated the use of the Hartley [10], [11] and the Cosine [12], [15], [16] transforms to build faster DNN models. In addition, other approaches [13], [14] have also taken advantage of the spectral domain obtained by the Cosine transform to design DNN models despite not focusing on the acceleration of the

model. Both input and output of the models in [10], [11], [14] are represented in their respective spectral domain. The approaches in [10], [11] are based on a Convolutional Neural Network architecture operating entirely on the Hartley domain. The approach in [14] is based on Fully Connected Neural Network operating based on the Cosine domain. These works mainly focus on the Image Super-resolution task, a type of regression problem addressed with DNN models. Hence, their architectures do not include pooling layers and work entirely on the spectral domain. The approaches in [12], [15] combines the Cosine transform with other techniques to compact and speed up pre-trained DNN models.

These studies use DOTs under a different approach from the one described in [7] in multiple aspects. The works in [10], [11] replaces the convolutional layers by element-wise multiplications on the Hartley domain. Also, notice that in [13], [14], [16], the spectral representation obtained from the DCT domain is fed into a FCNN, thus implying element-wise multiplication. However, these element-wise multiplications could not be considered as a close approximation of a convolution operation performed in the spectral domain. This is mainly due to the fact that a small subset of the frequencies are used [14], [16]. As for the work proposed in [13], the Cosine transform used prior to the FC section of the network also does not relate to a conventional convolution operation given that the DCT is applied across the depth of the kernels only. Additionally, though the convolution theorem of the Fourier transform shows that a spatial domain convolution can be replaced by an element-wise product on the spectral domain, this direct equivalence does not hold for the convolution theorems of the Hartley and the Cosine transforms. To the best of our knowledge, no work in the literature investigated these three transforms under the same architecture, such as one that deals with a classification model that contains pooling layers, which represents a typical CNN model.

In this paper, we demonstrate how spectral domains from different Discrete Orthogonal Transforms can be used to accelerate CNNs. The main contributions of this paper are summarized below:

- We provide a critical review of how the Fourier, Hartley and Cosine transforms have been used in the literature to speedup CNNs.
- We extend the framework that applies the Fourier transform to speedup CNN models to work with Hartley and Cosine transforms.
- We use the convolution theorem of the Hartley transform, which is a different approach than a product on the Hartley domain, to design spectral domain CNNs.
- We benchmark the performance of the proposed orthogonal transforms, focusing on their learning capabilities and computation time.

The remainder of the paper is organized as follows. In Section 2, an overview of the recent work involving Fourier, Hartley and Cosine transforms is presented. In Section 3, we extend the framework that applies the Fourier transform to speedup CNN models to work with Hartley and Cosine transforms. We then investigate and discuss the theory related to each of these transforms. In Section 4,

Figure 2: Comparison of the theoretical number of multiplication operations between the filtering using FFT domain and the spatial convolution with different kernel sizes for an input size and a number of kernels similar to the first layer of the LENET-5 model [9], where $B = 128$, $f_1 = 3$, $f_2 = 6$ and $N = 32$ where applicable.

Figure 3: Comparison of the theoretical number of multiplication operations between the filtering using FFT domain and the spatial convolution with different kernel sizes for larger hyperparameters: $B = 128$, $f_1 = 96$, $f_2 = 256$ and $N = 128$ where applicable.

we evaluate the performance of each transform for solving classification problems. In Section 5, we discuss the findings of our investigation. Finally, in Section 6 conclusions are drawn and further research work is suggested.

2 RELATED WORK

2.1 The FFT and Neural Networks

Since the late 90's, spectral convolution has been used to speedup neural networks. An earlier approach [6] presents how the FFT can be used to accelerate a face detection system using of Multi-Layer Perceptron (MLP) to process regions of an image in a sliding window manner. This work demonstrates how the output of the first layer of the network for the overlapping windows can be expressed as a single convolution operation of the entire image by a set of filters. Hence, the inference can be accelerated by performing the convolution in the FFT domain. Techniques like these are categorized as offline approaches, since they aim at the deployment phase, where the model is already known, and thus the FFT of the input images and model weights can be computed only once. Despite such advantage, these offline techniques are not suitable for accelerating CNNs, where the bottleneck of current systems is mainly due to the convolution operation in the training stage.

More recently, the work in [7] introduces how to fast train CNNs through FFT in an online approach by applying the spectral transform into the network architecture. Each layer depends on three different convolution operations. A convolution to compute the forward pass, Equation 1a; and two other convolutions, Equations 1b and 1c, to compute the

backward pass when back-propagating the loss L . In these equations, we have the input feature map x_{f_1} of size N by N , the kernel $w_{f_2 f_1}$ of size K by K , and the output y_{f_2} of a given convolutional layer.

$$y_{f_2} = \sum_{f_1} x_{f_1} * w_{f_2 f_1} \quad (1a)$$

$$\frac{\partial L}{\partial x_{f_1}} = \frac{\partial L}{\partial y_{f_2}} * w_{f_2 f_1}^T \quad (1b)$$

$$\frac{\partial L}{\partial w_{f_2 f_1}} = \frac{\partial L}{\partial y_{f_2}} * x_{f_1} \quad (1c)$$

As highlighted in [7], though the use of FFT for computing a single convolution in the spectral domain requires more computation, the overhead of the FFT is reduced when considering the operation on the batch as a whole, as earlier demonstrated in Figure 3. This work also discusses the reuse of the spectral representations to calculate Equations 1a to 1c, thus enhancing even further the computational gain. However, depending on the size of the batch and the number of convolution layers, caching the spectral representations for x_{f_1} and $w_{f_2 f_1}$ of Equation 1a, to be reused in Equations 1b and 1c for each layer, could be infeasible due to the amount of memory that would be required. Moreover, [7] also suggests taking advantage of the Hermitian conjugate symmetry of FFTs to reduce the stored memory in half, i.e. $N(N+1)/2$. For a given real-valued signal x of size N by N , its Fourier transform has the Hermitian conjugate symmetry, where the negative frequency values are the conjugate of the positive ones as denoted in Equation 2 [17].

This property is more emphasized in [18] to reduce both the FFT representation and the computation complexity of the convolutions.

$$X(k, l) = X^*(-k \bmod N, -l \bmod N) \quad (2)$$

Based on the approach described in [7], many related works [8], [19], [20], [21], [22], [23] were proposed to accelerate CNNs using the spectral domain. These approaches mainly rely on optimizing the computation of the FFT operations, especially for cases where the kernel size is small, and where the spatial convolution would still have an advantage over the spectral convolution if the traditional FFT algorithm were used.

2.2 Other Spectral Domain CNNs

The terms “spectral” and CNNs co-occur throughout recent literature in very different manners. In some works, the term “spectral” is associated with the nature of the problem, i.e. the data domain. For instance, in [24] the input is a hyperspectral image volume wherein each image channel represents the intensity of a specific light wavelength. From a signal processing perspective, the CNN still computes the convolutions on a spatial domain. Nevertheless, most of the related work that uses the terms “spectral” or “frequency domain”, do so to refer to the Fourier domain representations. In contrast, our work refers to Spectral domain Convolutional Neural Networks (SdCNNs) in a more broader sense, considering CNNs that performs the equivalent to the convolution operation on any spectral domain, such as Fourier, Hartley or Cosine transforms. The work described in [25] presents the mathematical formulation of the convolution theorem for these discrete orthogonal transforms.

In this section, we review how recent works have applied convolutional neural networks on domains other than the one produced by Fourier transform.

2.2.1 Hartley Transform

The work in [10] uses the Discrete Hartley Transform (DHT) [26] to elaborate a residual convolutional neural network, building a faster regression model for the image super-resolution (SR) problem. Such a model differs from traditional image SR approaches, such as in [27], in which the high resolution image is obtained directly as the network output. In [10], both input and output are represented in the spectral domain, thus the transformations are applied in an offline manner. Note that such approach implements a CNN entirely on the frequency domain. The DHT also produces a real-valued spectral domain. Thus, the ReLU [28] activation function could be adopted as the non-linear operator. Though the selection of low frequencies could be used as the pooling operator [29], [30], [31], given the nature of the problem, this architecture does not adopt a dimensionality reduction operator. Despite the claim of operating convolution entirely on the frequency domain, this model does not compute the convolution on each layer, but rather generates what here we consider to be an “approximation”, given that the element-wise product on the DHT domain does not exactly match the convolution theorem for the Hartley transform.

The use of the DHT for solving the image SR problem is also presented in [11]. This work is an extension of the approach presented in [10] and it mostly relies on sharing the parameters of four quadrants in the compact weighting layer. Despite reaching an improvement on the network speed, this work still adopts a simplification of the convolution theorem of the DHT. Thus, it does not address the use of the complete convolution theorem for the DHT, neither compares the impact of adopting such simplification on the network learning process.

2.2.2 Cosine Transform

The work in [16] is one of the first to use the Discrete Cosine Transform (DCT) to accelerate DNN models. Unlike the DFT, the DCT produces a real-valued spectral domain [32]. The authors in [16] propose the use of DCT to re-parametrize the input of a MLP network, hence taking advantage of the energy compaction property of the DCT. This process reduces the number of parameters, and thus speedup the network. In particular, a certain number of frequencies are chosen to represent the input image, thus decreasing the size of the model. These frequencies are picked using a zig-zag pattern through the low-frequency diagonals. The training of each layer is done in an unsupervised manner using autoencoders. In the last stage, the network is fine-tuned to build the final classifier. Though this work uses DCT to accelerate DNN models, the network architecture is not based on a CNN architecture. In [14], a solution for the image SR task is based on computing high resolution image patches using a Fully Connected Neural Network with both input and output represented in the DCT domain. This study compares the DCT representation of resized image patches of size 32 by 32 using the bicubic interpolation method to its ground truth. Then, a subset containing the most 512 erroneous frequencies is used to train a FCNN model to reduce the error on these frequencies amplitude. Notice that in both works [14], [16], a subset of frequencies based on the DCT domain is selected and then fed into a Fully Connected network.

A similar work is presented in [13], which uses DCT to build a classifier for scene categorization using spectral features. Here, we highlight some of their similarities: 1) the DCT is computed only one time per batch, and once calculated, the data representation is kept in the spectral domain; and 2) both approaches use DCT to de-correlate the information. In fact, the DCT, like others DOTs, possesses de-correlation properties, concentrating the signal energy into a small number of lower frequency coefficients. Among the DOTs, the Karhunen-Loève Transform (KLT), a.k.a. Eigenvector transform provides optimum de-correlation, thus the most energy compaction. However, given its high computational cost of $\mathcal{O}(N^2)$ for a 1-D signal of size N , it is often replaced by the DCT base II (DCT-II), which is a sub-optimal approximation. Three main differences between the two approaches can also be observed. First, the approaches described in [14], [16] target MLP like models, while the approach in [13] can be applied in either convolutional or fully connected layers. Second, unlike [14], [16], the transform described in [13] is used as one of the layer types present in the architecture. Finally, while the approaches in [14], [16] apply the transform on the model’s

Figure 4: Illustration of our framework, an extension of [7] that operates convolutions on domains beyond the FFT. Figure adapted from [7].

input to de-correlate spatial information, the transformation used in [13] de-correlates the information across multiple features or input channels. Thus, regularizing the network and improving its learning capability by de-correlating the representation across multiple neuron activations, hence avoiding co-adaptation.

Other recent works have adopted the DCT to design CNN models, such as [12] and [15]. These works mostly rely on using the DCT to compress the parameters of the network, thus reducing the amount of memory used by the model. In addition, the approach in [15] applies the DCT to patches of the image, similar to the JPEG compression. Analyzing the effects of convolving image patches is beyond the scope of our work. Thus, we focus on the DCT representation of the entire image.

3 PROPOSED FRAMEWORK

In [7], [8], FFTs have been successfully used to speedup CNN models by accelerating each convolution using a multiplication on the Fourier domain. Such a framework, for fast training of convolutional networks through FFTs, is generalized in this study by using a Transform $\mathcal{T}$, part of the family of DOTs, such as the Hartley and the Cosine transforms. Our approach is illustrated in Figure 4, in which $\mathcal{T}$ and $\mathcal{T}^{-1}$ are generalizing the FFT and the FFT^{-1} operations, respectively. The spectral operation is generalizing the matrix multiplication from the $\mathcal{F}$ domain, which is not always equivalent to the convolution theorem of the $\mathcal{T}$ domain. Therefore, we can combine this framework with either the Fourier, Hartley or Cosine transform, along with their respective spectral domain operation. Recall that the correspondence between time domain convolution and spectral domain operation depends on the adopted transform. Despite using the circular convolution $\otimes_c$ to describe the relationship between spatial and spectral domain, we use the linear convolution operator in the design of our models as described below in this section. To do so, we apply padding to the kernel to match the image size. This is needed to perform the circular convolution on the spectral domain. Then, we crop the output of the circular convolution in order to match the output of the linear convolution $\otimes$ present in CNN models.

The theoretical framework for the use of the Fourier, Hartley and Cosine transforms, along with their respective convolution theorems to speedup CNN models is presented in this section. For simplicity, the notation used here will be

mainly focused on 1-D signals, but it can be extended to 2-D signals without any loss of generalization. Figure 5 shows a comparison between the spectral domain operations using different DOTs and the conventional spatial convolution. An antisymmetric kernel was intentionally chosen to highlight how the filtering technique adopted in [10] for the DHT does not generalize the convolution operation, given that it assumes an even symmetric kernel. In the first row of Figure 5 we present the input image, followed by the filter output using the Fourier transform; the Hartley transform; Hartley transform with simplified convolution theorem based on an even kernel assumption (DHTe); and the product on the Cosine transform. In the second row we present the kernel used for this example and we show the difference between the output of the spatial domain convolution and each of the operations from the first row. The Mean Squared Error (MSE) of the difference is also given.

3.1 Discrete Fourier Transform (DFT or $\mathcal{F}$)

The Discrete Fourier Transform is defined by a complex exponential and maps a real input into the complex domain. This transform allows an input signal $x(n)$ to be represented in terms of a finite number of frequency components $X_{\mathcal{F}}(k)$. The DFT is widely used in signal processing applications for filtering and analysis. The transform and its inverse are defined by Equations 3a and 3b, where $e^{j\theta} = \cos(\theta) + j \sin(\theta)$ and $j = \sqrt{-1}$.

$$X_{\mathcal{F}}(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) \exp(-j2\pi kn/N) \quad (3a)$$

$$x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_{\mathcal{F}}(k) \exp(j2\pi kn/N) \quad (3b)$$

Despite to be known as orthonormal transforms, the DFT and other DOTs are also commonly used and presented as orthogonal transforms. The Equations 4a and 4b present the non-normalized version of the DFT. Note that the normalization factor $\frac{1}{\sqrt{N}}$ is removed from the forward transform and put in the inverse transform.

$$X_{\mathcal{F}}(k) = \sum_{n=0}^{N-1} x(n) \exp(-j2\pi kn/N) \quad (4a)$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_{\mathcal{F}}(k) \exp(j2\pi kn/N) \quad (4b)$$

3.1.1 Convolution Theorem

The convolution theorem for the DFT domain states that the circular convolution $\otimes_c$ of two signals in the spatial domain x and h is equivalent to an element-wise product (i.e. the Hadamard product $\circ$) of their respective counterparts in the Fourier domain. This is given in Equation 5 below.

$$\frac{1}{\sqrt{N}} \mathcal{F}\{x \otimes_c h\}(k) = X_{\mathcal{F}}(k) \circ H_{\mathcal{F}}(k) \quad (5)$$

Observe that by multiplying two DFT domain representations, the implicit normalization factor will appear twice on the right-hand side of the expression, thus requiring

Figure 5: Comparison between spatial convolution and spectral domain operations.

a counter part $\frac{1}{\sqrt{N}}$ in the left-hand side. Such adjustment can be mitigated by using Equation 6. In this equation, the normalization of $H_{\mathcal{F}}$ is dropped, thus yielding $H_{\mathcal{F}}$. This can be explained by the assumption of $h(n)$ being the impulse response representation of a system in which the signal energy was only supplied by one sample [33]. Thus, the normalization should not be applied.

$$\dot{\mathcal{F}}\{x \circledast_c h\}(k) = X_{\dot{\mathcal{F}}}(k) \circ H_{\mathcal{F}}(k) \quad (6)$$

Alternatively, the non-normalized DFT can be adopted to filter an input on the spectral domain, as shown in Equation 7. This notation is very popular for its convenience for using the same functions to process both inputs without changing the equivalence to the circular convolution operation $\circledast_c$ on the spatial domain.

$$\mathcal{F}\{x \circledast_c h\}(k) = X_{\mathcal{F}}(k) \circ H_{\mathcal{F}}(k) \quad (7)$$

3.1.2 Computational Cost

The standard 2D spatial convolution of an image of size N by N by a kernel of size K by K produces an output of size M by M , where $M = N - K + 1$. The computational cost of this operation is $O(M^2 k^2)$. When using the FFT to filter the signal, the filter kernel is padded to match the images size of N by N . Thus having a cost estimate of $3 \times 2CN^2 \log N + 4N^2$, as shown in Section 1. The extra C in the expression is a hidden cost which includes padding the kernel and cropping the output in order to obtain an image of size M by M as in the conventional linear convolution operation.

The complex multiplication on the DFT domain is commonly done using 4 real multiplications. This can be further reduced by using techniques such as the Gauss' complex multiplication [34] or the Karatsuba [35] algorithm. These techniques compute a complex multiplication using 3 real multiplications. Thus, dropping the computation to $3 \times 2CN^2 \log N + 3N^2$. These algorithms replace multiplication by additions, which is a desirable optimization technique for most processors. Given the complex numbers

$x = a + jb$ and $y = c + jd$, their product using the Gauss' complex multiplication is given by Equation 8.

$$\begin{aligned} S_1 &= c(a + b) \\ S_2 &= a(d - c) \\ S_3 &= b(c + d) \\ \Re\{x \cdot y\} &= S_1 - S_3 \\ \Im\{x \cdot y\} &= S_1 + S_2 \end{aligned} \quad (8)$$

3.2 Discrete Hartley Transform (DHT or $\dot{\mathcal{H}}$)

The Discrete Hartley Transform [26], [36], [37] is defined by the *cas* function, thus yielding a real domain spectral representation. The kernels for $\dot{\mathcal{H}}$ and $\dot{\mathcal{H}}^{-1}$ are defined by Equations 9a and 9b, where $\text{cas}(\theta) = \cos(\theta) + \sin(\theta)$. Note that the DHT is involutory, i.e., it is equal to its inverse.

$$X_{\dot{\mathcal{H}}}(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) \text{cas}(2\pi kn/N) \quad (9a)$$

$$x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_{\dot{\mathcal{H}}}(k) \text{cas}(2\pi kn/N) \quad (9b)$$

Furthermore, the even and odd components of the Hartley transform can be written in terms of the Fourier transform, as shown in Equations 10a and 10b.

$$\begin{aligned} X_{\dot{\mathcal{H}}}^e(k) &= \frac{X_{\dot{\mathcal{H}}}(k) + X_{\dot{\mathcal{H}}}(-k)}{2} \\ &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) \cos(2\pi kn/N) \\ &= \Re\{X_{\dot{\mathcal{F}}}\} \end{aligned} \quad (10a)$$

$$\begin{aligned} X_{\dot{\mathcal{H}}}^o(k) &= \frac{X_{\dot{\mathcal{H}}}(k) - X_{\dot{\mathcal{H}}}(-k)}{2} \\ &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) \sin(2\pi kn/N) \\ &= -\Im\{X_{\dot{\mathcal{F}}}\} \end{aligned} \quad (10b)$$

Hence, the Hartley transform itself can be written in terms of the Fourier transform, as shown in Equation 11, where $\Re$ and $\Im$ denote its real and imaginary parts.

$$X_{\mathcal{H}}(k) = \Re\{X_{\mathcal{F}}\} - \Im\{X_{\mathcal{F}}\} \quad (11)$$

Recall the Hermitian symmetry of $X_{\mathcal{F}}$, which comes from the even and odd nature of the cosine and sine, respectively. Thus, it is easy to observe that given $X_{\mathcal{F}}$, $X_{\mathcal{H}}(k)$ can be computed with only an extra addition operation as given in Equation 12.

$$X_{\mathcal{H}}(-k) = \Re\{X_{\mathcal{F}}\} + \Im\{X_{\mathcal{F}}\} \quad (12)$$

Similar to the DFT, the normalization factor of the DHT can also be rearranged to obtain a non-normalized transform, as expressed in Equations 13a to 14b. In such a case, the DHT is not involutory.

$$X_{\mathcal{H}}(k) = \sum_{n=0}^{N-1} x(n) \text{cas}(2\pi kn/N) \quad (13a)$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_{\mathcal{H}}(k) \text{cas}(2\pi kn/N) \quad (13b)$$

$$X_{\mathcal{H}}^e(k) = \Re\{X_{\mathcal{F}}\} \quad (14a)$$

$$X_{\mathcal{H}}^o(k) = -\Im\{X_{\mathcal{F}}\} \quad (14b)$$

It is worth mentioning that there are two natural candidates for the 2D DHT when extending its formulation from the 1D definition as shown in Equation 15 and Equation 16 [37]. Equation 15 is extended from the 1D DHT so that the relation between the DHT and the difference between the even and odd parts of the DFT are preserved. However, this formulation is not separable among the two dimensions, which yields a serious drawback. The separability is an important property for the implementation of the Fast Hartley transform algorithm. Thus, as proposed in [26], the 2D Separable Discrete Hartley Transform (SDHT or $\mathcal{H}_S$) is expressed in Equation 16 as:

$$X_{\mathcal{H}}(k, l) = \frac{1}{\sqrt{NM}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(n) \text{cas} \left[2\pi \left(\frac{kn}{N} \right) + \left(\frac{lm}{M} \right) \right] \quad (15)$$

$$X_{\mathcal{H}_S}(k, l) = \frac{1}{\sqrt{NM}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(n) \text{cas} \left(\frac{2\pi kn}{N} \right) \text{cas} \left(\frac{2\pi lm}{M} \right) \quad (16)$$

3.2.1 Convolution Theorem

The convolution theorem for the DHT domain is given by Equation 17 [25], [26], [37].

$$\frac{1}{\sqrt{N}} \mathcal{H}\{x \otimes_c h\}(k) = X_{\mathcal{H}}(k) \circ H_{\mathcal{H}}^e(k) + X_{\mathcal{H}}(-k) \circ H_{\mathcal{H}}^o(k) \quad (17)$$

Suppose h is an even symmetric kernel. So that,

$$H_{\mathcal{H}}^o = -\Im\{H_{\mathcal{F}}\} = 0 \quad (18)$$

Consequently,

$$\mathcal{H}\{h\} = H_{\mathcal{H}}^e(k) + \underline{H_{\mathcal{H}}^o(k)} \xrightarrow{0} \quad (19)$$

Thus, it can be observed from Equation 20a that for an even kernel, the convolution theorem of the DHT can be simplified into a single product of the representations on the spectral domain. Therefore, having an expression similar to the one done on the Fourier domain, but reducing the number of operations because we have a real multiplication instead of a complex one. This is an advantage commonly explored in signal processing given that most of the designed filters have the even symmetry property. The same principle applies for having an odd symmetry in the kernel as in Equation 20b. For the remaining of this paper, we refer to the combination of the DHT and the simplification on the convolution theorem based on the assumption of an even kernel as the DHTe method for approaching the convolution operation on the DHT domain.

$$\frac{1}{\sqrt{N}} \mathcal{H}\{x \otimes_c h^e\} = X_{\mathcal{H}}(k) \circ \underline{H_{\mathcal{H}}^e(k)} \xrightarrow{H_{\mathcal{H}}(k)} \quad (20a)$$

$$\frac{1}{\sqrt{N}} \mathcal{H}\{x \otimes_c h^o\} = X_{\mathcal{H}}(-k) \circ \underline{H_{\mathcal{H}}^o(k)} \xrightarrow{H_{\mathcal{H}}(k)} \quad (20b)$$

Similar to the DFT, the normalization factor in the DHT expression can be adjusted without changing its equivalence to the convolution operation on the spatial domain, Equations 21 to 22b.

$$\mathcal{H}\{x \otimes_c h\}(k) = X_{\mathcal{H}}(k) \circ H_{\mathcal{H}}^e(k) + X_{\mathcal{H}}(-k) \circ H_{\mathcal{H}}^o(k) \quad (21)$$

$$\mathcal{H}\{x \otimes_c h^e\} = X_{\mathcal{H}}(k) \circ \underline{H_{\mathcal{H}}^e(k)} \xrightarrow{H_{\mathcal{H}}(k)} \quad (22a)$$

$$\mathcal{H}\{x \otimes_c h^o\} = X_{\mathcal{H}}(-k) \circ \underline{H_{\mathcal{H}}^o(k)} \xrightarrow{H_{\mathcal{H}}(k)} \quad (22b)$$

Recall that the properties mentioned above imply holding equivalence between the DHT and the difference between the real and imaginary parts of the DFT as shown in Equation 11. Thus, when extending the DHT along the two dimensions, Equation 15 is the one to be considered. On the other hand, the adoption of the 2D SDHT, described in Equation 16, enables the use of the 2D Fast Hartley Transform (FHT) in a similar fashion to the 2D FFT, but with a more efficient computation. However, the SDHT would yield a different convolution theorem property for 2D as demonstrated in [37].

The general case for the convolution theorem of the 2D SDHT is given by Equation 23, which presents four real multiplications. Like the non-separable 2D DHT extension, this theorem depends of the even and odd parts of the image and kernel signals. However in this case, obtaining these components would require extra operations, whereas for the non-separable version they would be given as the DFT components. In addition, the special case of having a symmetric kernel does not yield a convolution theorem of a single multiplication operation, like its non-separable version would do. Instead, the 2D SDHT convolution theorem

gets reduced from four multiplications to three multiplications on the spectral domain as shown in Equation 24. For the reasons mentioned above, the use of 2D SDHT becomes infeasible, and only the non-separable 2D DHT should be considered when using the Hartley domain to accelerate CNNs.

$$\begin{aligned} \mathcal{H}_S \{x \otimes_c h\} (k, l) = & X_{\mathcal{H}_S}^e(k, l) \circ H_{\mathcal{H}_S}^e(k, l) - \\ & X_{\mathcal{H}_S}^o(k, -l) \circ H_{\mathcal{H}_S}^o(k, -l) + \\ & X_{\mathcal{H}_S}^e(k, -l) \circ H_{\mathcal{H}_S}^o(k, l) + \\ & X_{\mathcal{H}_S}^o(k, l) \circ H_{\mathcal{H}_S}^e(k, -l) \end{aligned} \quad (23)$$

$$\begin{aligned} \mathcal{H}_S \{x \otimes_c h\} (k, l) = & X_{\mathcal{H}_S}^e(k, l) \circ H_{\mathcal{H}_S}^e(k, l) + \\ & X_{\mathcal{H}_S}^o(k, -l) \circ H_{\mathcal{H}_S}^o(k, -l) + \\ & X_{\mathcal{H}_S}^e(k, -l) \circ H_{\mathcal{H}_S}^o(k, l) \end{aligned} \quad (24)$$

3.2.2 Computational Cost

Considering the non-separable 2D DHT, which is obtained from the 2D FFT, the computation of filtering a 2D signal of size N by N using a kernel of size K by K uses only two multiplications on the spectral domain. Hence, the computational cost is estimated by $3 \times 2CN^2 \log N + 2N^2$ FLOPS. The cost can be further reduced if assuming an even kernel, requiring a single multiplication instead, thus yielding an estimate of $3 \times 2CN^2 \log N + N^2$ FLOPS.

3.3 Discrete Cosine Transform (DCT or C)

The Discrete Cosine Transform is largely adopted in image compression algorithms. It is presented under different derivation formulations, also known as “basis”. The term “DCT” is mostly used to refer to the DCT of basis II, or DCT-II transform. This particular basis was first conceptualized and proposed in [32] as an approximation to the eigenvectors of the covariance matrix of a useful class of signals. The true eigenvectors can be obtained by the Karhunen-Loève Transform (KLT) and provide the optimum energy compaction. In addition to having compaction efficiency levels close to the KLT, the DCT is also an alternative to the DFT for mitigating two other aspects, as pointed out in [38]. These two aspects are explained below.

- The DFT yields two components per sample, amplitude and phase, whereas the DCT yields a single component.
- The DFT treats the signal as periodic, which introduces boundary discontinuities that are captured as high-frequency content.

Despite having several other different basis for the DCT, in our work, we focus on the most commonly used basis, the basis II. Equations 25a and 25b provide, respectively, the definition for the DCT-II and its inverse transform [39].

$$X_{\dot{\text{C}}\text{-II}}(k) = \left(\frac{2}{N}\right)^{1/2} \xi_k \sum_{n=0}^{N-1} x(n) \cos \left[\frac{\pi k(n + \frac{1}{2})}{N} \right] \quad (25a)$$

$$x(n) = \left(\frac{2}{N}\right)^{1/2} \sum_{k=0}^{N-1} \xi_k X_{\dot{\text{C}}\text{-II}}(k) \cos \left[\frac{\pi k(n + \frac{1}{2})}{N} \right] \quad (25b)$$

for $k, n = 0, \dots, N - 1$.

The extra normalization factor ξ_k , defined in Equation 26, is required in order to make the basis of the DCT-II transformation orthonormal.

$$\xi_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } k = 0 \text{ or } k = N \\ 1 & \text{otherwise} \end{cases} \quad (26)$$

The DCT can be expressed in terms of the DFT [39], [40] by extending the signal symmetrically in order to cancel the imaginary component.

3.3.1 Convolution Theorem

Unlike the Fourier domain, the convolutional theorem for the Cosine Transform cannot be represented by a simple multiplication in the Cosine domain. According to [25], given $y(n) = x(n) \otimes_c h(n)$, the DCT-II could have its convolution theorem defined by Equation 27.

$$Y_{\dot{\text{C}}\text{-II}}(k) = \frac{1}{\xi_k} X'_{\dot{\text{C}}\text{-II}}(k) H'_{\dot{\text{C}}\text{-I}}(k) - \xi_k X_{\dot{\text{S}}\text{-II}}(k) H_{\dot{\text{S}}\text{-I}}(k) \quad (27)$$

where, $X'_{\mathcal{T}}$ is the equivalent to $\mathcal{T}\{x(n) \cdot 1/\xi_n\}$; and $X_{\text{S-I}}$, $X_{\text{S-II}}$ and $X_{\text{C-I}}$ are respectively, the Discrete Sine Transforms DST-I, DST-II, and the Discrete Cosine Transform DCT-I representations of X .

Given the need for computing two different transforms per signal in this theorem, it is clear that the use of DCT for operating circular convolution is more expensive compared to when using the FFT. To demonstrate it, let's suppose a DCT implementation with similar performance to the FFT algorithm of $\mathcal{O}(N \log N)$ for processing a 1D signal. Then, we have the following comparison:

$$\cancel{2} N \log N + 2\cancel{N} > \cancel{3} N \log N + \cancel{4} N^2 \quad (28a)$$

$$\therefore 2N \log N > 2N \quad (28b)$$

As pointed in [39] and [41], it is very challenging to elaborate a convolution theorem for the DCT-II based on a notation similar to the one used for the DFT.

The product on the normalized DCT-II domain, as demonstrated in [39], is equivalent to having a convolution of the spatial domain of the extensions of the input sequences with a third sequence $\hat{g}(n)$ as expressed in Equation 29.

$$\dot{\text{C}}\text{-II}^{-1} \{X_{\dot{\text{C}}\text{-II}} H_{\dot{\text{C}}\text{-II}}\} = \dot{\mathcal{F}}^{-1} \left\{ \xi_k \hat{X}_{\dot{\mathcal{F}}} \hat{H}_{\dot{\mathcal{F}}} \exp \left(\frac{-j\pi k}{2N} \right) \right\} \quad (29a)$$

$$= \hat{x}(n) \otimes_c \hat{h}(n) \otimes_c \underbrace{\dot{\mathcal{F}}^{-1} \left\{ \xi_k \exp \left(\frac{-j\pi k}{2N} \right) \right\}}_{\hat{g}(n)} \quad (29b)$$

$$= (\hat{x} \otimes_c \hat{h} \otimes_c \hat{g})(n) \quad (29c)$$

When adopting the same approach using one or both unnormalized transforms, as discussed previously for the DFT and DHT, the normalization factor ξ_k is dropped from the expressions as shown in Equations 30. In both cases, it

is important to notice that $\hat{g}(n)$ is a constant, not depending on neither x or h .

$$\mathcal{C}\text{-II}^{-1} \{X_{\mathcal{C}\text{-II}} H_{\mathcal{C}\text{-II}}\} = \mathcal{F}^{-1} \left\{ \hat{X}_{\mathcal{F}} \hat{H}_{\mathcal{F}} \exp \left(\frac{-j\pi k}{2N} \right) \right\} \quad (30a)$$

$$= \hat{x}(n) \otimes_c \hat{h}(n) \otimes_c \underbrace{\mathcal{F}^{-1} \left\{ \exp \left(\frac{-j\pi k}{2N} \right) \right\}}_{\hat{g}(n)} \quad (30b)$$

$$= (\hat{x} \otimes_c \hat{h} \otimes_c \hat{g})(n) \quad (30c)$$

3.3.2 Computational Cost

Unlike the domains of the DFT and DHT, the product in the DCT domain is not equivalent to a spatial domain convolution. Instead, it is related to the convolution of symmetric extension of both input and kernel. The cost for such computation depends on the used DCT algorithm. When computing the Cosine Transform using the FFT [40], the number of multiplications to transform an N by N image is estimated by $2CN^2 \log(N) + 8N^2$. The estimated cost for filtering an image would be $6CN^2 \log(N) + 25N^2$. An alternative approach for Fast Cosine Transform (FCT) [42], that purely computes the DCT without using the FFT, has an estimate of $N/2 \log N$ real multiplications for 1D DCT. Hence, $\mathcal{O}(N^2 \log N)$ for computing the 2D FCT of a single image. Therefore, the cost for filtering an image with a kernel in the DCT domain would be estimated to $3CN^2 \log(N) + N^2$ FLOPS.

3.4 Estimated Computation for Batch Filtering

To illustrate the number of FLOPS that would be required when replacing the spatial convolution by spectral domain operations in the CNN model, Table 1 provides a theoretical comparison of the computation involved in each transform, its spectral domain operation and the total amount of computation for filtering a single image and for filtering the entire batch. The batch filtering in Table 1 is estimated by taking into consideration that the accumulation of the activations produced by the convolution on each channel is done in the spectral domain, prior to the inverse transform, thus further reducing the number of inverse transform computations. This is mainly due to the linear property of DOTs as shown in Equation 31.

$$\sum_i x_i(n) \otimes h_i(n) = \mathcal{T}^{-1} \left\{ \sum_i X_{i\mathcal{T}}(k) \circ H_{i\mathcal{T}}(k) \right\} \quad (31)$$

Table 1 shows the amount of computation for two alternative ways to implement the DCT. Figure 6 shows a theoretical comparison between the DOT based approaches and spatial convolution for filtering a single image. As it can be noticed, the DOT based approach yields more operations than the standard spatial domain convolution up to kernels of size $K = 7$.

The DOT based approaches, like the FFT one, are more efficient when applied to mini-batches. The number of operations in these methods are compared in Figure 7, 8 and 9. As it can be seen in Figure 7–8 and 9, the FCT and the DHTe methods yield the least amount of operations when

Figure 6: Comparison between the theoretical number of multiplication operations of the DOTs based approach and spatial domain convolution for image filtering with different kernel sizes, where $K = \{3, 5, 7, 9, 11\}$.

considering mini-batch filtering, having the order of FLOPS approaching the fully-connected layer. For large values of f_1 and f_2 , the estimated FLOPS for the DCT method via FFT is similar to the DHTe method, and both of these methods have significant reduction in the proportion of the number of operations when compared to the FFT method. On the other hand, when considering small values for f_1 and f_2 , such as those used by the LENET-5 model, the reduction on the proportion of the number of FLOPS is more subtle, as it can be seen in Figure 7. It can be observed that the FFT, FFT+Gauss, DHT and DHTe have a close order of FLOPS for small values of f_1 and f_2 , as shown in Figure 7. These methods have a similar cost for computing the spectral transform, but they differ in the number of multiplications. In addition, the FCT is estimated to have the least amount of FLOPS. Though the FCT method has a similar cost for the spectral operation as the DHTe method, its total estimated FLOPS reflects that if f_1 and f_2 are large numbers, having less computation for the spectral operation has a significant impact on decreasing the overall estimated number of FLOPS. However, if f_1 and f_2 are small numbers, having less computation for the transform operation has a significant impact on decreasing the overall estimated number of FLOPS.

4 EXPERIMENTAL RESULTS

In this section, we present the experiments used to investigate the performance of each of the proposed DOT methods to speedup CNN models. Each method is defined by a *spectral domain* and a *spectral domain operation*. Among the *spectral domain operations*, we have operations that are *equivalent* to convolution theorems, which are a perfect spectral representation of the spatial domain convolution (i.e. DFT and DHT); as well as operations that “*approximates*” the convolution theorem using a simpler spectral domain expression (i.e. DHTe and DCT), as discussed in Section 3. First, we describe the experiment setup used to test, validate, and assess the proposed DOT methods. Then, we analyze their learning capabilities in terms of their loss and accuracy. Finally, we measure their execution time and benchmark their performance using the spatial convolution and the FFT method as baselines.

4.1 Experiment Setup

All experiments are implemented in Python, using the PyTorch [43] and the Torch-DCT [44] libraries under the

Method	Transform	Spectral Operation	Image Filtering	Batch Filtering
FFT	$2N^2 \log N$	$4N^2$	$6N^2 \log N + 4N^2$	$2CN \log N^2 [f_2 B + f_1 B + f_2 f_1] + 4B f_2 f_1 N^2$
FFT+Gauss	$2N^2 \log N$	$3N^2$	$6N^2 \log N + 3N^2$	$2CN \log N^2 [f_2 B + f_1 B + f_2 f_1] + 3B f_2 f_1 N^2$
DHT	$2N^2 \log N$	$2N^2$	$6N^2 \log N + 2N^2$	$2CN \log N^2 [f_2 B + f_1 B + f_2 f_1] + 2B f_2 f_1 N^2$
DHTe	$2N^2 \log N$	N^2	$6N^2 \log N + N^2$	$2CN \log N^2 [f_2 B + f_1 B + f_2 f_1] + B f_2 f_1 N^2$
DCT-II [40]	$2N^2 \log(N) + 8N^2$	N^2	$6N^2 \log(N) + 25N^2$	$[2CN^2 \log N + 8N^2][f_2 B + f_1 B + f_2 f_1] + B f_2 f_1 N^2$
FCT-II [42]	$N^2 \log N$	N^2	$3N^2 \log(N) + N^2$	$CN^2 \log N [f_2 B + f_1 B + f_2 f_1] + B f_2 f_1 N^2$

Table 1: Number of FLOPS per DOT method, in which the number of real-valued multiplications of the DCT approach depends on the algorithm adopted to perform the DCT-II transform.

Figure 7: Comparison of the theoretical number of multiplication operations between the filtering using DOT domains and the spatial convolution with different kernel sizes for an input size and a number of kernels similar to the first layer of the LENET-5 model [9], where $B = 128$, $f_1 = 3$, $f_2 = 6$ and $N = 32$ where applicable.

Figure 8: Comparison of the theoretical number of multiplication operations between the filtering using DOT domains and the spatial convolution with different kernel sizes for an input size and a number of kernels similar to the first layer of the LENET-5 model [9], where $B = 128$, $f_1 = 3$, $f_2 = 64$ and $N = 32$ where applicable.

Figure 9: Comparison of the theoretical number of multiplication operations between the filtering using DOT domains and the spatial convolution with different kernel sizes for larger hyperparameters: $B = 128$, $f_1 = 96$, $f_2 = 256$ and $N = 128$ where applicable.

Figure 10: LUNET-5 architecture, model designed to solve the digit recognition problem [9].

Google Colab [45] environment. The Intel (R) Xeon (R) CPU @ 2.00GHz and a Tesla V100-SXM2-16GB GPU were assignment to the Google Colab runtime when profiling the DOT models. It should be noted that the current stable version of the PyTorch library, version 1.8.1, offers only the implementation of the FFT algorithm. Therefore, the implementations of the Hartley and Cosine transforms, listed in Table 1, are not implemented using the same optimizations (algorithm and code wise) adopted in the FFT. For instance, despite using the FFT, the Hartley transform adopted in our experiments is implemented using PyTorch’s high level API, thus, generating additional overhead.

We benchmark the DOT methods using the LUNET-5 network [9] shown in Figure 10. The ReLU activation function is adopted a non-linear operation across the entire architecture. In this network, the convolutional operations have a kernel of size $K = 5$. The convolution is of type¹ “valid”, i.e., padding is not applied to the input. Hence the output size M of each layer is smaller than its input size N , that is $M = M_{\text{valid}} = N - K + 1$.

The optimizers used in our experiments are Adam, SGD, SGD with Momentum of 0.9, and RMSProp with $\alpha = 0.99$. Notice that the mechanism of the Adam optimizer is based on a combination of the SGD with Momentum and the RMSProp, hence providing a faster convergency than these two optimizers. Thus, most of the experiments will be based on such optimizer. The `StepLR` scheduler is used with a step size of 20 epochs and a $\gamma = 0.5$. We train our model for 40 epochs using a mini-batch of size 128 and a learning rate of 0.001. We consider this to be a reasonable amount of epochs for evaluating each DOT, given that our work focuses on assessing the change in the learning performance within the same model architecture.

Five datasets are used in order to benchmark the proposed DOT methods. Among them, we have the MNIST dataset [9] and some variants of the MNIST dataset such as EMNIST [46], KMNIST [47] and Fashion-MNIST [48]. Additionally, a more complex dataset, CIFAR-10 [49] is also used in our benchmark. Image samples and distribution of target classes are shown for each dataset in Figure 11. A summary of the size of each dataset is presented in Table 2. In the following experiments, seeds for the random processes, such as splitting the dataset, shuffling mini-batches and network weights initialization, are defined in order to have reproducible results when comparing different DOT methods.

1. Recall there are three variants (or modes) of the discrete convolution operation with regard to the size of the output, usually labeled as types “full”, “same” and “valid”. Their output sizes are respectively given by $M_{\text{full}} = N + K - 1$, $M_{\text{same}} = N$ and $M_{\text{valid}} = N - K + 1$.

Dataset	Image Size	Train Samples	Test Samples	Targets
MNIST	(28, 28)	60000	10000	10
EMNIST	(28, 28)	112800	18800	47
KMNIST	(28, 28)	60000	10000	10
Fashion-MNIST	(28, 28)	60000	10000	10
CIFAR-10	(32, 32, 3)	50000	10000	10

Table 2: Datasets adopted for the benchmark of our study

4.2 Learning Capabilities

In this section, we investigate how the use of DOT to design convolution operations of the CNN architecture impacts the performance of the LUNET-5 model. For each method, we compare the loss and accuracy on the training and validation sets. The goal of these experiments is to verify that methods that are equivalent to the spatial domain convolution, i.e. the DFT and DHT methods, also have equivalent learning performance. In addition, these experiments also assess how the model’s performance is affected when the spatial domain convolution is replaced by a spectral operation that is not strictly equivalent to the convolution theorem of the given domain, which is the case for the DHTe and DCT methods. We explore the performance of all the aforementioned methods with the Adam optimizer. Then, we investigate how the methods not equivalent to the spatial domain convolution perform when using the traditional optimizers, SGD, SGD with Momentum, and RMSProp.

4.2.1 FFT and DHT Methods

Both of these methods use their respective convolution theorem to compute the convolution operation. For the DHT method, we adopt the non-separable 2D DHT and its convolution theorem as shown in Equation 21. Hence, by definition, these models would produce the same results to the conventional spatial domain convolution. A comparison between the FFT, DHT and the standard convolution methods is shown in Figure 12, which provides a close view of the results for the training loss on Epochs 7 and 40. As shown in Figure 12, a close overlap of both FFT and DHT methods with the spatial convolution occurs in the early training steps. Despite of having the convolutional theorem of both DFT and DHT domains producing mathematically the exact same results as the spatial domain convolution, Figure 12 shows that both approaches have tiny differences which increase with the training steps. The comparison around Epoch 7 shows that the loss values are very similar even after several thousands of training steps. By comparing these values with the ones at Epoch 40, we observe the same phenomenon but with less similarity. The differences at Epoch 40 can be noticed on the datasets that didn’t fully overfit the training data, i.e. EMNIST and Fashion-MNIST. These differences occurs mainly due to the inherited quantization noise associated with limitations in the numerical representation; which accumulate as it propagates over the training stage. The overall comparison of these models with the accuracy metric over the training phase is provided in Figure 13. The results for the loss and accuracy metrics for both training and validation sets for each dataset are summarized in Table 3, where ΔAcc is the percentage of

Figure 11: Image samples and target classes distribution of the adopted datasets.

change in the accuracy metric on the validation set in terms of the spatial convolution method.

4.2.2 DHTE and DCT Methods

These methods mathematically differ from the convolution theorem of their respective DOT domains. Unlike the previous methods, the DHTE and DCT methods rely on a simplification of their respective DOT convolution theorem, which was shown earlier in Equations 22a and 30, respectively. Figure 14 provides a comparison between the DHTE, DCT and the standard convolution methods, which shows a close view of the results for the training loss on Epochs 7 and 40. As it can be noticed by analyzing the loss on the training and validation sets, the DHTE method causes the CNN to underperform considerably. This comes with no surprise, because the adopted operation assumes that the kernel is even symmetric, but this model, like [10], does not include any constraint or regularization on the network architecture to guarantee such property. Thus, the network would still need to deal with artifacts in the filter output produced by this simplification as demonstrated previously in Figure 5. The overall comparison of these models with the accuracy metric over the training phase is provided in Figure 15. On the other hand, despite not producing the same output as the conventional filtering operation, the DCT method demonstrates a better performance over the DHTE method on the loss metric for the training and validation sets. This is clearly shown in Table 3.

4.2.3 DHTE and DCT Methods using Other Optimizers

In addition to the investigation on the effects of using DOT to learn CNN models with the Adam optimizer, we also investigated how the DHTE and DCT perform when using the SGD, SGD with Momentum, and RMSProp optimizers. Similarly to the Adam optimizer, when combined with RMSProp, the DHTE and DCT methods underperform the standard convolution and equivalent methods as shown in Figure 16 and Table 4. Adam and RMSProp optimizers have the capability of adjusting the learning rate individually per model weight. On the other hand, it is also noticed that when combined with the SGD or SGD with Momentum, the DCT method outperforms the standard convolution and equivalent methods by exhibiting a faster convergence.

Method	Train Loss	Train Acc.	Val. Loss	Val. Acc.	$\Delta\text{Acc} (\%)$
Spatial	9.407258e-06	100.00000	0.000343	99.228766	0.000000
FFT	3.426949e-09	100.00000	0.000346	99.248798	0.020188
DHT	1.612261e-08	100.00000	0.000313	99.348958	0.121126
DHTE	2.124660e-04	98.43750	0.000876	98.046875	-1.191077
DCT	5.112798e-07	100.00000	0.000564	98.838141	-0.393661

(a) MNIST

Method	Train Loss	Train Acc.	Val. Loss	Val. Acc.	$\Delta\text{Acc} (\%)$
Spatial	1.407992e-03	93.75000	0.005730	85.793022	0.000000
FFT	1.021652e-03	95.31250	0.005678	85.862586	0.081084
DHT	9.528674e-04	94.53125	0.005804	85.942851	0.174640
DHTE	1.715477e-03	91.40625	0.005937	81.244649	-5.301565
DCT	1.883421e-03	90.62500	0.004596	85.001070	-0.923096

(b) EMNIST

Method	Train Loss	Train Acc.	Val. Loss	Val. Acc.	$\Delta\text{Acc} (\%)$
Spatial	2.713626e-07	100.00000	0.003218	95.072115	0.000000
FFT	2.769077e-08	100.00000	0.003889	95.162260	0.094817
DHT	4.673161e-08	100.00000	0.003882	95.212340	0.147493
DHTE	1.518097e-05	100.00000	0.006333	89.433093	-5.931310
DCT	2.501922e-04	98.43750	0.004786	92.628205	-2.570585

(c) KMNIST

Method	Train Loss	Train Acc.	Val. Loss	Val. Acc.	$\Delta\text{Acc} (\%)$
Spatial	4.208828e-04	97.65625	0.003672	90.334535	0.000000
FFT	3.199581e-04	98.43750	0.003732	90.024038	-0.343719
DHT	2.808720e-04	100.00000	0.003530	90.134215	-1.866203
DHTE	6.729815e-04	97.65625	0.003857	87.880609	-2.716487
DCT	6.763418e-04	96.09375	0.003783	89.032452	-1.441401

(d) Fashion-MNIST

Method	Train Loss	Train Acc.	Val. Loss	Val. Acc.	$\Delta\text{Acc} (\%)$
Spatial	3.304592e-03	85.15625	0.011912	63.331330	0.000000
FFT	3.632629e-03	82.03125	0.011920	62.149439	-1.866203
DHT	3.752975e-03	84.37500	0.011985	62.239583	-1.723866
DHTE	7.591889e-03	64.84375	0.012793	49.519231	-21.809267
DCT	4.548833e-03	78.12500	0.012163	56.670673	-10.517160

(e) CIFAR-10

Table 3: Comparison of learning capabilities of the LENET-5 model using DOT methods trained by 40 epochs in different datasets, where ΔAcc is the percentage of change in the accuracy metric on the validation set in terms of the spatial convolution method.

Figure 12: Comparison of the loss values for the FFT and DHT methods on the MNIST, EMNIST and Fashion-MNIST datasets over 40 epochs of learning.

(a) MNIST

(b) EMNIST

(c) Fashion-MNIST

Figure 13: Comparison of the accuracy values for the FFT and DHT methods on the MNIST, EMNIST and Fashion-MNIST datasets over 40 epochs of learning.

Figure 14: Comparison of the loss values for the DHTe and DCT methods on the MNIST, EMNIST and Fashion-MNIST datasets over 40 epochs of learning.

Method	SGD				SGD with Momentum				RMSProp			
	MNIST	EMNIST	KMNIST	Fashion-MNIST	MNIST	EMNIST	KMNIST	Fashion-MNIST	MNIST	EMNIST	KMNIST	Fashion-MNIST
Standard	94.9619	77.4187	74.5092	74.9900	98.8081	86.6064	92.7484	88.2712	99.1186	85.7235	94.9720	89.8538
DHTe	87.7404	65.7802	60.4467	73.4976	97.5361	82.9463	86.7288	86.0677	98.2272	81.7102	89.4631	87.8706
DCT	96.7949	81.7530	81.9411	81.1398	98.5877	85.8198	91.6466	88.1611	98.5276	84.6961	92.7384	88.7620

Table 4: Comparison of accuracy on the validation set of the LENET-5 model using DOT methods trained for 40 epochs in different datasets using the SGD, SGD with Momentum and RMSProp optimizers.

Figure 15: Comparison of the accuracy values for the DHTe and DCT methods on the MNIST, EMNIST and Fashion-MNIST datasets over 40 epochs of learning.

Figure 16: Comparison of different optimizers on the loss values for the DHTe and DCT methods on the MNIST, EMNIST and Fashion-MNIST datasets.

4.3 Execution Time

In this section, we compare the execution time of the proposed DOT methods. We use the standard PyTorch profiling tool to produce a trace from which we extract the execution time for each layer of the CNN models. The input data used to profile the network has the following specifications: batch-size, channels per input image, and image size, respectively defined by $B = 128$, $C = 3$ and $N = 32$. Three variants of the LENET-5 network are used to measure the execution time of the proposed methods. The specification of these architectures is given in Table 5. Figure 17 shows the runtime for the forward pass of the three LENET-5 models using spatial convolution, breaking down the execution time of the networks in terms of their layers. It can be observed from this figure that the dominant execution times are those associated with the convolutional layers, Conv-0 and Conv-1, representing most of the total execution time. This is estimated to 59.32%, 63.59% and 79.14% of the total execution time for the spatial convolution of the LENET-5-A, LENET-5-B, and LENET-5-C, respectively. Therefore, in the experimental results presented below we only focus on these two components of the total execution time, ignoring the execution time of the fully connected layers.

It should be noted here that the execution time using the PyTorch library is affected by an internal optimization process that occurs in the first few iterations. Consequently, we do not take into consideration the execution time of these initial steps in our measurements. Furthermore, as highlighted

LENET-5 Configuration		
Input (3 x 32 x 32 Input Size)		
LENET-5-A	LENET-5-B	LENET-5-C
Conv5-6	Conv5-32	Conv5-64
Maxpool 2x2 Stride-2		
Conv5-16	Conv5-64	Conv5-128
Maxpool 2x2 Stride-2		
FC-120		
FC-84		
Softmax-10		

Table 5: Convolutional Neural Network architectures used to profile DOT based methods.

Figure 17: Breakdown of runtime per layer of the forward-pass of each variant of the LENET-5 model.

earlier, the use of the PyTorch high level API in the proposed methods resulted in additional overhead. Therefore, we re-implemented the complex product in Python using this API in order to have a fair comparison with the proposed methods. The FFT method is used here as a baseline for measuring the execution time of the proposed DOT methods, given that the FFT method is already extensively validated in [7], [8]. Table 6 shows the total convolution time of each

Method	Conv-0	Conv-1	Total Conv	S_{FFT}
FFT	1358.875	1288.707	2647.582	1.000
FFT+Gauss	1348.102	1295.859	2643.961	1.001
RFFT	1349.625	1377.289	2726.914	0.971
RFFT+Gauss	1370.641	1411.598	2782.238	0.952
DHT	1330.156	1349.121	2679.277	0.988
DHTe	1181.183	1217.027	2398.210	1.104
DCT-FFT	4673.006	4610.047	9283.053	0.285
DCT-linear	2114.562	2086.913	4201.476	0.630

(a) LENET-5-A

Method	Conv-0	Conv-1	Total Conv.	S_{FFT}
FFT	2624.500	7176.699	9801.199	1.000
FFT+Gauss	2607.625	6514.172	9121.797	1.074
RFFT	2076.148	4410.852	6487.000	1.511
RFFT+Gauss	2081.297	4252.684	6333.980	1.547
DHT	1941.500	3275.258	5216.758	1.879
DHTe	1501.188	1908.742	3409.930	2.874
DCT-FFT	6207.500	5681.719	11889.219	0.824
DCT-linear	2377.766	2611.703	4989.469	1.964

(b) LENET-5-B

Method	Conv-0	Conv-1	Total Conv	S_{FFT}
FFT	4332.000	29193.250	33525.250	1.000
FFT+Gauss	4180.625	24579.562	28760.188	1.166
RFFT	3155.469	17451.031	20606.500	1.627
RFFT+Gauss	3098.688	14787.562	17886.250	1.874
DHT	2741.750	10825.750	13567.500	2.471
DHTe	2050.562	4884.500	6935.062	4.834
DCT-FFT	8415.219	9907.703	18322.922	1.830
DCT-linear	2761.188	5017.062	7778.250	4.310

(c) LENET-5-C

Table 6: Summary of the execution time per layer per DOT methods for both network architecture, where S_{FFT} is the execution speedup of each of proposed DOT methods over the FFT method, used as baseline.

of the methods implemented in this study, as well as S_{FFT} , which is the speedup of the proposed DOT methods over the FFT method.

4.3.1 FFT Based Methods

The FFT method is combined with the Gauss algorithm, resulting in the FFT+Gauss method. As shown in Figure 18, the FFT+Gauss method yields an overall speedup of 1.074 and 1.166 in the total convolution time for the LENET-5-B and LENET-5-C models, respectively. However, for the LENET-5-A model, the FFT+Gauss method does not provide any speedup (1.001), as shown in Table 6.

In addition to the FFT and FFT+Gauss methods, we also use of Real-valued Fast Fourier Transform (RFFT) instead of the FFT as suggested in [7]. In the RFFT method, the entire representation on FFT domain, N^2 samples, can be obtained by computing only approximately half of its samples, i.e. $N(N/2 + 1)$. By taking the conjugate of these samples, the remaining samples can be retrieved using the hermitian symmetry property of the FFT domain for real-values inputs as shown in Equation 2. The amount of computations can be further reduced by not retrieving all the N^2 samples, thus only having to compute the product of the $N(N/2 + 1)$ samples in the FFT domain. The effect of using such an approach to speedup execution time is clearly shown in

Figure 18: Breakdown of runtime per layer of the forward-pass of each variant of the LENET-5 model using the FFT methods.

Figure 19: Breakdown of runtime per layer of the forward-pass of each variant of the LENET-5 model using the DHT methods.

Figure 18 and Table 6, where the RFFT outperform the FFT method, yielding a speedup of 1.511 and 1.627 in the total convolution time for the LENET-5-B and LENET-5-C models, respectively. For the LENET-5-A model, the RFFT method yields a speedup of 0.971, thus performing slower.

The RFFT method can also take advantage of the Gauss multiplication. This combination is here referred to as the RFFT+Gauss method. As clearly shown in Figure 18 and Table 6, this method yields a further speedup of 1.547 and 1.874 in the total convolution time for the LENET-5-B and LENET-5-C models, respectively.

4.3.2 DHT Based Methods

Mathematically, the DHT method uses less operations than the FFT method. In addition to the three sum operations of the DFT-DHT conversion, the spectral operation of the DHT method has only two real multiplications and one sum operation as opposed to the four real multiplications and three sum operations of the complex product in the FFT domain. Figure 19 presents a comparison of the convolution time of the DHT method with that of the FFT. The results in Table 6 show that the total convolution time of the DHT method yields 1.879 and 2.471 speedup in the total convolution time for the LENET-5-B and LENET-5-C models, respectively. For the LENET-5-A model, the RFFT method yields a speedup of 0.988, thus performing slower.

The DHTe, on the other hand, has only one real multiplication in its spectral operation, as opposed to the two real multiplications and four sum operations computed in the DHT method. The difference in the number of operations is reflected in the reduction of their convolution time as observed in Figure 19. As shown in Table 6, the DHTe exhibits a speedup of 1.104, 2.874 and 4.834 in the total convolution time for the LENET-5-A, LENET-5-B and LENET-5-C models, respectively.

4.3.3 DCT Based Methods

Unlike the DHT the DCT method via the FFT requires more computation, as shown earlier in Figure 7 and 9. Hence, an implementation of the DCT approach using the FFT, i.e. DCT-FFT method, occurs to be way more time consuming than the FFT method as it can be seen in Table 6a and 6b, and Figure 20. Therefore, we also adopt the DCT-linear method. Such a method is a variant of the DCT method based on

an approach found in [44], which adopts a fully-connected, a.k.a linear, layer to compute the Cosine transform. This implementation tends to be more expensive if executed on a single processor, but considering the high parallelism of GPUs, it becomes more viable than computing the DCT via FFT as pointed in [44].

Figure 20: Breakdown of runtime per layer of the forward-pass of each variant of the LENET-5 model using the DCT methods.

The results of the DCT-FFT and the DCT-linear methods are presented in Table 6, where the DCT-FFT method exhibits a speedup of 1.830 in the total convolution time for the LENET-5-C. For the LENET-5-A and LENET-5-B models, the DCT-FFT method yields a speedup of 0.285 and 0.824, respectively, thus performing slower. The DCT-linear method however, exhibits a speedup of 1.964 and 4.310 in the total convolution time for both LENET-5-B and LENET-5-C, respectively. For the LENET-5-A, the DCT-linear method also yields a speedup of 0.630, thus performing slower.

5 DISCUSSION

In this paper, we extended the framework that applies the Fourier transform to speedup CNN models to work with Hartley and Cosine transforms. In this section, we provide a general comparison and discussion of the results presented in Section 4. Figure 21 summarizes all computation time results in a single graph for comparison purpose.

5.1 Convolution Equivalent Methods

The FFT, FFT+Gauss, RFFT, RFFT+Gauss and DHT are methods that are equivalent to the spatial domains convolution since their foundation is based on the convolution theorem of their respective DOT. As expected, our experiments, shown in Table 3 and 6, demonstrate that in general CNN can be accelerated by replacing the spatial domain convolution by any of these methods without any significant effect to the model’s learning capability. By comparing the results in Figure 21, we notice that for a CNN that is narrow, the use of DOT methods does not present a significant speedup for models that has a small amount of filters, such as the LENET-5-A. This is because the gain due to DOT based approach was insignificant compared to the overhead generated by our implementation based on the PyTorch high-level API, for such small amount of spectral domain operations generated in the LENET-5-A model. The DHT shows the best performance in terms of speedup among these methods, which is consistent with our initial theoretical estimation in Figure 8. Our results show that it is possible to properly adopt the convolution theorem of the Hartley domain in order to have similar learning capabilities and speedup as the FFT method.

Additionally, in a further extension of this framework which would entirely use spectral domain representation,

i.e. without using DOT to convert between spatial and spectral domain such as in [10], the use of the Hartley transform would be even more advantageous over the Fourier transform for being defined on the real plane. Unlike the Fourier domain, the DHT has the advantage of enabling the traditional non-linear operations (ReLU, Sigmoid, TanH), which are commonly used in the spatial domain, to be also used in the spectral domain representation. The use of these functions in a complex spectral domain, such as the FFT, is not possible in a natural manner. Thus, making the design of a non-linear operation for the FFT domain a challenging task.

5.2 Convolution Approximated Methods

Both DHTe and DCT are methods that are not equivalent to the spatial domains convolution, but instead they are a simplification based on having a single element-wise multiplication in their spectral domain as a replacement of the convolution theorem of their respective DOT. Our experiments, shown in Table 3, indicate that the DCT outperforms the DHTe method, yielding a smaller decrease in the model accuracy (ΔAcc). The inferior performance of the DHTe method is due to the fact that it assumes that kernels are even symmetric, which is usually the case of handmade kernels. As discussed in Section 3, this assumption does not hold for non-symmetric kernels as shown in Figure 5. Hence, it also does not hold for those weights generated by the back-propagation algorithm. This could be addressed by some constraint mechanism or specific type of regularization adopted in the network. Additionally, the DCT method also has the potential to yield a better speedup than the

Figure 21: Breakdown of runtime per layer of the forward-pass of each variant of the LENET-5 model using the DOTs methods.

DHTe method if combined with the Faster Cosine Transform algorithm [42], as shown in Table 1 and Figure 7, 8 and 9.

Despite both of these methods underperform the spatial domain convolution method, they can be still valid approaches to be adopted for problems in which the accuracy can be slightly compromised towards acquiring speedup. Our experiments also show that when using the SGD and SGB with Momentum as optimizers, the DCT was able to show a faster convergence than the spatial domain convolution while maintaining close accuracy values as shown in Table 4.

6 CONCLUSION AND FUTURE WORK

In this paper, we successfully extended the framework that applies the Fourier transform to speedup CNN models to work with Hartley and Cosine transforms. This is achieved by generalizing the spectral transform and the spectral domain operation adopted by the framework. Also, we introduced the use of the complete convolution theorem for the DHT domain, demonstrating that it is superior than the DHTe method based on the approach in [10].

Ideally, the spectral transform and spectral domain operation must be equivalent to a convolution on the spatial domain in order to ensure the derivatives used for the back-propagation are preserved. This combination can naturally be decided by choosing a DOT and its convolution theorem, which is the case for FFT and DHT methods. However, for the DCT method, the convolution theorem requires additional computation, thus making the acceleration infeasible. This issue is addressed in this paper by demonstrating that for the DCT domain the convolution theorem can be replaced by a product on the spectral domain. This yields a convolution of symmetric extensions of the input and the kernel, which is then convolved by a third signal that is invariant to the value of inputs. Our experiments revealed that the DCT method provides a better tradeoff speed-accuracy for accelerating CNNs than the DHTe method. Additionally, the DCT method also has a faster convergence than all the other methods when combined with the SGD and SGD with Momentum.

Though, there are current works showing how to fully execute a CNN entirely in the frequency domain, such as in [10], [11], these models were not compared under a classification task, neither used a pooling operation, given that the nature of the problem considered in these studies is image super-resolution. In future work, we intend to investigate and benchmark the performance of DOTs by considering pooling layers on the spectral domain. Additionally, we intend to explore CNN models for classification and segmentation that fully run on the spectral domain, comparing their performance using multiple DOTs.

ACKNOWLEDGMENTS

The authors would like to thank MITACS for the financial support of part of this research through the Research Training Award (RTA) Summer 2020 program.

REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, pp. 1097–1105. (page 1)
- [2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14, 2015. (page 1)
- [3] H. Zhu, M. Akrou, B. Zheng, A. Pelegris, A. Phanishayee, B. Schroeder, and G. Pekhimenko, "TBd: Benchmarking and analyzing deep neural network training," *arXiv*, pp. 88–100, 2018. (page 1)
- [4] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018. [Online]. Available: <https://doi.org/10.1016/j.patcog.2017.10.013> (page 1, 2)
- [5] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, 2019. [Online]. Available: <https://doi.org/10.1016/j.neucom.2018.09.038> (page 1, 2)
- [6] S. Ben-Yacoub, B. Fasel, and J. Li, "Fast Face Detection using MLP and FFT," in *Conference on Audio and Video-based Biometric Person Authentication*, 1999. [Online]. Available: https://infoscience.epfl.ch/record/82563/files/avbpa_{_}.face.pdf (page 1, 3)
- [7] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pp. 1–9, 2014. (page 1, 2, 3, 4, 5, 14, 15)
- [8] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, "Fast Convolutional Nets With fftfft: A GPU Performance Evaluation," *EWSHM - 7th European Workshop on Structural Health Monitoring*, vol. 127, no. April, pp. 491–499, dec 2014. [Online]. Available: <http://arxiv.org/abs/1412.7580> (page 1, 4, 5, 14)
- [9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. [Online]. Available: <http://ieeexplore.ieee.org/document/726791/> (page 2, 3, 10, 11)
- [10] J. Li, S. You, and A. Robles-Kelly, "A Frequency Domain Neural Network for Fast Image Super-resolution," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2018-July, pp. 1–8, 2018. (page 2, 4, 5, 12, 16, 17)
- [11] S. Xue, W. Qiu, F. Liu, and X. Jin, "Faster image super-resolution by improved frequency-domain neural networks," *Signal, Image and Video Processing*, vol. 14, no. 2, pp. 257–265, 2020. [Online]. Available: <https://doi.org/10.1007/s11760-019-01548-8> (page 2, 4, 17)
- [12] Y. Wang, C. Xu, S. You, D. Tao, and C. Xu, "CNNpack: Packing Convolutional Neural Networks in the Frequency Domain," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, pp. 253–261. (page 2, 5)
- [13] S. H. Khan, M. Hayat, and F. Porikli, "Scene Categorization with Spectral Features," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, 2017, pp. 5639–5649. (page 2, 4, 5)
- [14] O. Aydin, "Image Super-Resolution Using Deep Feedforward Neural Networks in Spectral Domain," Ph.D. dissertation, 2018. [Online]. Available: <http://repository.bilkent.edu.tr/handle/11693/36341> (page 2, 4)
- [15] Y. Wang, C. Xu, C. Xu, and D. Tao, "Packing Convolutional Neural Networks in the Frequency Domain," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2495–2510, 2019. (page 2, 5)
- [16] X. Zou, X. Xu, C. Qing, and X. Xing, "High speed deep networks based on Discrete Cosine Transformation," *2014 IEEE International Conference on Image Processing, ICIP 2014*, pp. 5921–5925, 2014. (page 2, 4)
- [17] G. Blanchet and M. Charbit, *Digital Signal and Image Processing Using Matlab®*. Hoboken, NJ, USA: John Wiley & Sons, Inc., jun 2014. [Online]. Available: <http://doi.wiley.com/10.1002/9781118999554> (page 3)

- [18] J. H. Ko, B. Mudassar, T. Na, and S. Mukhopadhyay, "Design of an Energy-Efficient Accelerator for Training of Convolutional Neural Networks using Frequency-Domain Computation," in *Proceedings - Design Automation Conference*, vol. Part 12828, 2017. (page 4)
- [19] T. Highlander and A. Rodriguez, "Very Efficient Training of Convolutional Neural Networks using Fast Fourier Transform and Overlap-and-Add," pp. 160.1–160.9, 2015. (page 4)
- [20] S. Kala, B. R. Jose, D. Paul, and J. Mathew, "A Hardware Accelerator for Convolutional Neural Network Using Fast Fourier Transform," in *Communications in Computer and Information Science*. Springer Verlag, jun 2019, vol. 892, pp. 28–36. [Online]. Available: http://link.springer.com/10.1007/978-981-13-5950-7_3 (page 4)
- [21] T. Abtahi, C. Shea, A. Kulkarni, and T. Mohsenin, "Accelerating Convolutional Neural Network with FFT in Embedded Hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 9, pp. 1737–1749, 2018. (page 4)
- [22] J. Lin and Y. Yao, "A Fast Algorithm for Convolutional Neural Networks Using Tile-based Fast Fourier Transforms," *Neural Processing Letters*, vol. 50, no. 2, pp. 1951–1967, 2019. [Online]. Available: <https://doi.org/10.1007/s11063-019-09981-z> (page 4)
- [23] K. Chitsaz, M. Hajabdollahi, N. Karimi, S. Samavi, and S. Shirani, "Acceleration of convolutional neural network using FFT-based split convolutions," *arXiv*, 2020. (page 4)
- [24] S. Ozkan and G. B. Akar, "Deep Spectral Convolution Network for HyperSpectral Unmixing," pp. 3313–3317, jun 2018. [Online]. Available: <http://arxiv.org/abs/1806.08562> (page 4)
- [25] A. Ušáková, J. Kotuliaková, and M. Zajac, "Using of Discrete Orthogonal Transforms for Convolution," *Journal of Electrical Engineering*, vol. 53, no. 9, pp. 285–288, 2002. (page 4, 7, 8)
- [26] R. N. Bracewell, "Discrete Hartley transform," *Journal of the Optical Society of America*, vol. 73, no. 12, p. 1832, dec 1983. [Online]. Available: <https://www.osapublishing.org/abstract.cfm?URI=josa-73-12-1832> (page 4, 6, 7)
- [27] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, feb 2016. [Online]. Available: <http://ieeexplore.ieee.org/document/7115171/> (page 4)
- [28] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010. [Online]. Available: <https://icml.cc/Conferences/2010/papers/432.pdf> (page 4)
- [29] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 2015-Janua, pp. 2449–2457, 2015. (page 4)
- [30] J. S. Smith and B. M. Wilamowski, *Discrete cosine transform spectral pooling layers for convolutional neural networks*. Springer International Publishing, 2018, vol. 10841 LNAI, no. 2015. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-91253-0_23 (page 4)
- [31] H. Zhang and M. Jianwei, "Hartley Spectral Pooling for Deep Learning," *CSIAM Transactions on Applied Mathematics*, vol. 1, no. 3, pp. 464–475, jun 2020. [Online]. Available: http://global-sci.org/intro/article_{ }detail/csiam-am/18306.html (page 4)
- [32] N. Ahmed, T. Natarajan, and K. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, jan 1974. [Online]. Available: <http://ieeexplore.ieee.org/document/1672377/> (page 4, 8)
- [33] "Why are magnitudes normalised during synthesis (IDFT), not analysis (DFT)?" [Online]. Available: <https://dsp.stackexchange.com/a/12653/13659> (page 6)
- [34] S. Dasgupta, C. H. Papadimitriou, and U. Vazirani, *Divide-and-conquer algorithms*, 1st ed. USA: McGraw-Hill, Inc., 2006. (page 6)
- [35] D. J. Bernstein, "Multidigit multiplication for mathematicians," pp. 5–6, 2001. (page 6)
- [36] R. Bracewell, "The fast Hartley transform," in *Proceedings of the IEEE*, vol. 72, no. 8, feb 1984, pp. 1010–1018. [Online]. Available: <http://ieeexplore.ieee.org/document/1457236/> (page 6)
- [37] A. B. Watson and A. Poirson, "Separable two-dimensional discrete Hartley transform," *Journal of the Optical Society of America A*, vol. 3, no. 12, p. 2001, 1986. (page 6, 7)
- [38] N. Roma and L. Sousa, "A tutorial overview on the properties of the discrete cosine transform for encoded image and video processing," *Signal Processing*, vol. 91, no. 11, pp. 2443–2464, 2011. [Online]. Available: <http://dx.doi.org/10.1016/j.sigpro.2011.04.015> (page 8)
- [39] K. R. RAO and P. YIP, "CHAPTER 2 - DEFINITIONS AND GENERAL PROPERTIES," in *Discrete Cosine Transform*, K. R. RAO and P. YIP, Eds. San Diego: Academic Press, 1990, pp. 7–26. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780080925349500084> (page 8)
- [40] J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 1, pp. 27–34, 1980. (page 8, 9, 10)
- [41] S. A. Martucci, "Symmetric Convolution and the Discrete Sine and Cosine Transforms," *IEEE Transactions on Signal Processing*, vol. 42, no. 5, pp. 1038–1051, 1994. (page 8)
- [42] B. Lee, "A new algorithm to compute the discrete cosine Transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 6, pp. 1243–1245, dec 1984. [Online]. Available: <http://ieeexplore.ieee.org/document/1164443/> (page 9, 10, 17)
- [43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in Neural Information Processing Systems* 32, no. NeurIPS, pp. 8024–8035, dec 2019. [Online]. Available: <http://arxiv.org/abs/1912.01703> (page 9)
- [44] "DCT (Discrete Cosine Transform) for pytorch." [Online]. Available: <https://github.com/zh217/torch-dct> (page 9, 16)
- [45] E. Bisong, *Google Colaboratory BT - Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, 2019. [Online]. Available: https://doi.org/10.1007/978-1-4842-4470-8_7 (page 11)
- [46] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2017-May, pp. 2921–2926, 2017. (page 11)
- [47] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep Learning for Classical Japanese Literature," *arXiv*, no. NeurIPS 2018, pp. 1–8, 2018. (page 11)
- [48] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv*, pp. 1–6, 2017. (page 11)
- [49] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009. (page 11)

(NCA) of Federal University of Maranhão. His research interests are in the areas of Artificial Intelligence and Data Science.

many referred international journals. His research interests are in the areas of Artificial Intelligence, Semantic Web, Data Science, Ubiquitous Computing and Knowledge Engineering.

Eduardo Reis received the B.Sc. in Computer Science from the Federal University of Maranhão (UFMA) in 2014, the M.Sc. in Informatics from the Pontifical Catholic University of Rio de Janeiro (PUC-RIO) in 2016. Currently, he is a Ph.D. student under the Electrical and Computer Engineering program at Lakehead University. Contract Lecturer at Confederation College. Former member of the Computational Geophysics Group at the TECGRAF Institute of PUC-RIO. Former member of Applied Computing Group

Rachid Benlamri is a Professor of Software Engineering at Lakehead University - Canada. He received his Master's degree and a Ph.D. in Computer Science from the University of Manchester - UK in 1987 and 1990 respectively. He is the head of the Artificial Intelligence and Data Science Lab at Lakehead University. He supervised over 80 students and postdoctoral fellows. He served as keynote speaker and general chair for many international conferences. Professor Benlamri is a member of the editorial board for