Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 9, No. 1, March 2021, pp. 173~184
ISSN: 2089-3272, DOI: 10.11591/ijeei.v9i1.2737

A Low-complexity Complex-valued Activation Function for Fast and Accurate Spectral Domain Convolutional Neural Network

Shahriyar Masud Rizvi, Ab Al-Hadi Ab Rahman, Mohamed Khalil-Hani, Sayed Omid Ayat
VeCAD Research Laboratory, School of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru, Johor, Malaysia

Article Info

Article history:

Received Sep 20, 2020
Revised Jan 15, 2021
Accepted Jan 17, 2021

Keywords:

Convolutional Neural Network
Spectral domain CNN
Activation functions
Embedded systems

ABSTRACT

Conventional Convolutional Neural Networks (CNNs), which are realized in spatial domain, exhibit high computational complexity. This results in high resource utilization and memory usage and makes them unsuitable for implementation in resource and energy-constrained embedded systems. A promising approach for low-complexity and high-speed solution is to apply CNN modeled in the spectral domain. One of the main challenges in this approach is the design of activation functions. Some of the proposed solutions perform activation functions in spatial domain, necessitating multiple and computationally expensive spatial-spectral domain switching. On the other hand, recent work on spectral activation functions resulted in very computationally intensive solutions. This paper proposes a complex-valued activation function for spectral domain CNNs that only transmits input values that have positive-valued real or imaginary component. This activation function is computationally inexpensive in both forward and backward propagation and provides sufficient nonlinearity that ensures high classification accuracy. We apply this complex-valued activation function in a LeNet-5 architecture and achieve an accuracy gain of up to 7% for MNIST and 6% for Fashion MNIST dataset, while providing up to 79% and 85% faster inference times, respectively, over state-of-the-art activation functions for spectral domain.

Corresponding Author:

Shahriyar Masud Rizvi,
VeCAD Research Laboratory, School of Electrical Engineering
Universiti Teknologi Malaysia, Johor Bahru,
81310 Johor, Malaysia.
Email: m.rizvi@graduate.utm.my

1. INTRODUCTION

An artificial neural network (ANN) is based on a collection of connected nodes called neurons. ANNs must have an input layer, one or more hidden layers and an output layer. The output of each neuron is computed (activated) by some non-linear function of the sum of its inputs. It is highly desirable that the activation functions are non-linear and differentiable. The hidden layers should employ nonlinear activation functions so as to enable the network to learn complex relationships from the input data [1]. In fact, through nonlinear activation functions an ANN can learn any nonlinear behavior, provided the network has enough neurons and layers [2].

A convolutional neural network (CNN) is a specific variant of ANNs, built as a series of convolutional layers that are interleaved with pooling (or subsampling) layers, a fully-connected multilayer perceptron, and ends with a *softmax* layer [3]. CNN is a fundamental example of deep learning that is commonly applied to analyzing visual imagery. CNNs have shown tremendous success in computer vision applications such as

image and video recognition [4, 5] and object detection and segmentation [6, 7]. However, CNNs are both computationally intensive and memory intensive, making them too slow and difficult to be adopted in resource-constrained, low-power mobile and embedded systems. This is because the CNN architecture is dominated by convolutional layers, which have very high computational complexity and, as a result, consumes 90% of the computation and require large memory usage [3].

In a conventional (spatial domain) CNN, convolution requires multiple multiplications and additions (i.e. multiply-accumulate operations) to compute a single pixel of an output feature map. Consequently, computations in the convolution layers are slow because there are huge numbers of dot products and accumulations that must be performed for each kernel. It has been shown that, for an input array of size NxN and a kernel array of size kxk, the computational complexity is $O(N^2k^2)$ [8]. Hence, it is imperative that the CNN architecture is optimized to reduce the computation cost of convolutions and minimize the memory footprint.

An alternative way to compute CNN is to purely perform the computations in the spectral (or frequency) domain. To perform convolution equivalent in spectral domain, real-valued input data are first transformed to complex-valued Fourier space. In the spectral domain, the convolution equivalent can be performed as a single point-wise product [8, 9, 10, 11]. In other words, multiple real-valued multiplications in spatial domain are realized with one complex-valued multiplication (from here on we will refer to it as complex multiplication) in spectral domain. The computational complexity in the spectral domain reduces to $O(N^2)$, which is significantly less than $O(N^2k^2)$ in the spatial domain [8, 9]. Researchers have shown that this attractive feature has allowed spectral domain CNNs to offer 55% less computations in LeNet-5 and 67% less computations in AlexNet [12]. Consequently, convolution can be computed with significantly lower complexity if it is performed in spectral domain, rather than spatial domain.

One of the key issues in spectral domain CNNs is the lack of activation functions that are effective in the spectral domain [8, 9, 13]. Non-linear nature of activation functions is incompatible with the Linear Time Invariant (LTI) property of spectral domain. Thus, conventional spectral domain CNN approaches are forced to perform the activation function in spatial domain [8, 9, 10]. In such techniques, before performing activation function, spectral domain data is transformed to spatial domain using an IFFT block, and then spatial activation function is performed, and finally, the data is transformed back to the spectral domain using an FFT block. Thus, this approach requires multiple spatial-spectral domain switching. These domain transformations are computationally intensive, and hence negate some of the gain in computational complexity achieved with spectral domain CNN [9, 12, 14]. In some of these approaches spectral pooling is performed to downsample spectral domain feature maps, which is considered equivalent to max pooling in spatial domain [11]. Figure 1 gives the functional block diagram that illustrates the realization of LeNet-5 CNN modeled using the above-mentioned conventional approach in spectral domain CNN. Since full-connection and *softmax* layers are not very computationally intensive, one can keep these layers in spatial domain [14]. That is why after the last layer of the feature-learning segment, the feature maps are transformed to spatial domain.

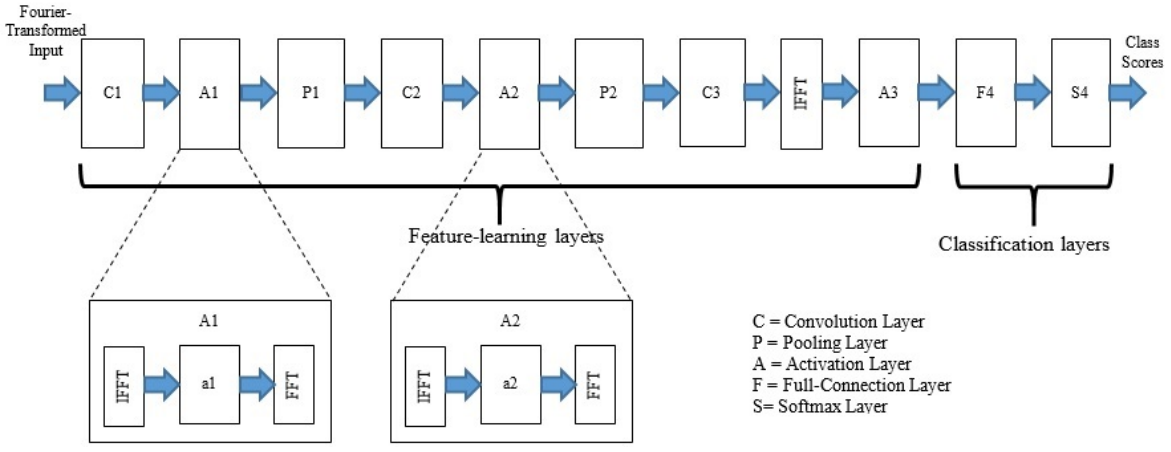

Figure 1. Functional block diagram of conventional spectral domain LeNet-5 CNN model

Francesca in [15] was one of the first to propose a spectral activation function. It was based on the modified Heaveside function and utilized Laplace Transforms. This activation function was only demonstrated in concept; it was not applied in a CNN model. Ko et al. in [12] proposed a linear approximation of

sigmoid/*tanh* type activation function for the spectral domain CNN. However, the authors did not explain how nonlinearity is achieved with a linear function nor provide any details on the implementation of the function.

Recently, Uijens in [13] proposed that a spectral activation function can be developed in the spectral domain by computing Fourier equivalent of a point-wise spatial activation function, approximated with the Taylor series. The square function is proposed for the spatial function, which is equivalent to auto-convolution in Fourier space (Fourier-space input convolved with itself) and hence very computationally intensive. Ayat et al. in [14] proposed a spectral version of ReLU, the most commonly used activation function in CNNs. This activation function, called SReLU, approximates ReLU with a quadratic function using the Taylor series and computes its Fourier equivalent. This activation function also has to compute auto-convolution. This work achieved 99% classification accuracy on MNIST dataset with the LeNet-5 CNN. However, just like [13], its main drawback is that it is computationally complex.

There is a class of activation functions, called complex-valued activation functions, which are applied in complex-valued neural networks (CVNNs) including complex-valued CNNs (CVCNNs) [16]. CVCNNs are applied to classify complex-valued data, that is, inputs that are already in complex form such as complex-valued images generated by Synthetic Aperture Radar (SAR) and functional Magnetic Resonance Imaging (fMRI). This is in contrast to spectral domain CNNs which classify real-valued data that are transformed into complex-valued Fourier space. In a CVCNN, instead of realizing convolution as point-wise products (as is done in spectral domain CNNs), complex-valued convolutions are performed [16].

Some complex-valued activation functions are split-type activation functions, where a spatial activation function such as *tanh* (or *sigmoid*) is applied to real and imaginary parts of the complex inputs separately. Recently, a split-type activation function, called ℂReLU, has been proposed for CVCNN, where ReLU is applied separately to the real and imaginary parts [17]. This work has attained excellent accuracy for CIFAR-10 and MusicNet datasets using ResNet and VGG CNNs respectively. A split-type *tanh* function is used in a spectral domain CNN as shown in the work of [18]. However, *tanh* functions possess singular points, which requires scaling of inputs and initial weights [16].

Another approach in complex-valued activation function involves compressing the complex input space for activation by passing input values in certain quadrants, while blocking input values in other regions. One such example is zReLU, which is proposed in [19]. However, this activation function has been tested only on specialized datasets but not on standardized datasets [19]. A similar activation function is modReLU that preserves pre-activation phase but modifies magnitude by the amount of a trainable parameter [20].

One of the earliest spectral domain CNN models was proposed by Mathieu et al. in [9]. In this work, only the convolution operation (realized as point-wise product) was performed in the spectral domain, while pooling and activation functions were performed in spatial domain. Spectral pooling was first proposed by Rippel et al. in [11], In this work, convolution and pooling operations were realized in spectral domain. Recently, Guan et al. in [18] proposed a spectral domain CNN model with computations of convolution and activation function (which is complex-valued version of *tanh*) performed in the spectral domain. In these models either pooling or activation function or both were performed in spatial domain and hence, required multiple spatial-spectral domain switching (using FFTs and IFFTs).

The first spectral domain CNN model without multiple domain switching was developed by Ko et al. in [12], who implemented convolution, pooling and activation function (realized as linear approximation of *sigmoid*/*tanh*) in spectral domain. Pratt et al. in [21] has also developed another spectral domain CNN model without multiple domain transformations, but do not specify the type of activation function they use or whether they have used any activation function at all. More recently, Ayat et al. in [14] proposed another spectral model for CNN without multiple domain transformations. They apply their SReLU spectral activation function and implement a fused version of convolution in spectral domain where pooling operation is merged with convolution.

This work applies a complex-valued activation function for spectral domain CNNs, where pre-activation values that have positive-valued real or imaginary part are transmitted through the activation function. Here, all feature extraction layers-convolution, pooling and activation-are realized in spectral domain. This activation function is beneficial in two ways. First, proposed activation function does not require computationally expensive domain switchings, as the activation function works in complex domain. Second, it is computationally much lighter than state-of-the-art spectral activation functions such as SReLU [14].

The main contributions of this research work are the following: (a) we propose a new complex-valued activation function for spectral domain CNN. The function has low computational complexity in both forward and backward propagation and exhibits sufficient nonlinearity to ensure high classification accuracy can be achieved, and (b) we show that a spectral domain CNN model applying this activation function offers more accurate and faster inference over state-of-the-art spectral and complex-valued activation functions. In addition to having a computationally light activation function, this CNN model removes the need for multiple spatial-spectral domain switching, which are computationally expensive. This makes such spectral domain CNN

solutions ideal for implementing CNNs in embedded hardware systems that have tight constraints in computing resources and power consumption.

The remainder of this paper is organized as follows: Section 2 presents proposed activation function and CNN model for this work. Experimental work, analysis and results are described in Section 3. Finally, Section 4 presents conclusions.

2. RESEARCH METHOD

2.1. Proposed activation function

The general nature of a complex-valued activation function is that they pass input values in certain segment of input space and blocks, from propagating through, input values in other areas of the input space. The area of input that a complex-valued activation function propagates to its output is called activation area [16, 17, 22]. This activation area is a complex valued space.

We propose to apply the following complex-valued activation function in our spectral domain CNN model. This activation function passes pre-activation values that have either positive-valued real part or positive-valued imaginary part. This idea is partly inspired by the operation of Rectified Linear Unit (ReLU) that passes only positive valued inputs (and rectifies the negative ones) and are widely employed for real-valued CNNs. The proposed activation function g(x), where x is the input to the activation function, is defined as

$$Z = g(x) = \begin{cases} x, \mathcal{R}(x) > 0 \ \vee \ \Im(x) > 0 \\ 0, \quad \text{otherwise} \end{cases} \tag{1}$$

where, $\Re(x)$ and $\Im(x)$ represent real and imaginary part of the complex variable x. Figure 2 (shaded region) illustrates the activation area of the proposed activation function.

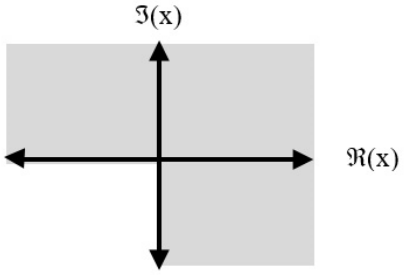

Figure 2. Activation area (shaded region) of proposed activation function

When compared to similar activation functions such as zReLU in [19] and ℂReLU in [17], the proposed activation function preserves more of the input magnitude and phase but still provides nonlinearity by rectifying inputs that have negative-valued real or imaginary segments. This is supported by the results of training and test accuracies of our spectral domain CNN model that utilizes this activation function (which is presented in Section 3 of this paper).

Differentiability is considered an important property of activation functions [23]. The derivatives of different layers of neural networks including activation functions are needed to compute error gradients, which allows the network to learn and improve accuracy. In backpropagation, error gradient of a layer is computed from partial derivative of the layer's output with respect to input as well as error gradient of the following layer with respect to current layer's output [1]. So, if input and output of layer i are x_i and z_i and that of the following layer (layer i+1) are x_{i+1} and z_{i+1}, the error gradient of layer i with respect to its input ($\partial E/\partial x_i$) is

$$\frac{\partial E}{\partial x_i} = \frac{\partial z_i}{\partial x_i} \cdot \frac{\partial E}{\partial x_{i+1}} = \frac{\partial z_i}{\partial x_i} \cdot \frac{\partial E}{\partial z_i} \tag{2}$$

where $\partial z_i/\partial x_i$ is the derivative of the current layer's output, while $\partial E/\partial z_i$ is the error derivative of the next layer with respect to that layer input (which happens to be current layer's output).

For this work, we have kept the backpropagation architecture largely in spatial domain. So, the error gradient with respect to input of the activation function $\partial E/\partial x_i$ needs to be in spatial domain. Here, the error gradient coming from the following layer ($\partial E/\partial z_i$) is in spatial domain. However, the output derivative of the activation function block ($\partial z_i/\partial x_i$) cannot be computed in spatial domain directly as there is no spatial equivalent of the proposed complex-valued activation function. Therefore, the output derivative of the activation function block ($\partial z_i/\partial x_i$) must be computed in Fourier space first. So, if x and z are input and output

of a layer in spatial domain and their equivalent in Fourier space are X and Z, one needs to compute $\partial Z_i/\partial X_i$, the output derivative in Fourier space, where $Z = \mathcal{F}(z)$ and $X = \mathcal{F}(x)$. Computing output derivatives of the activation function is very simple since it either passes input values intact if input values lie within the activation area ($\Re(Z)>0 \vee \Im(Z)>0$) and derivative becomes 1, or passes 0 when the input values lie beyond this area, where derivative would be 0. This is expressed by the following Equations 3, 4 and 5. Finally, after the error gradient is computed in Fourier space, it is transformed to spatial domain to be consistent with the spatial domain nature of backpropagation architecture.

$$Z_i = \mathcal{F}(z_i) \quad (3)$$

$$X_i = \mathcal{F}(x_i) \quad (4)$$

$$\frac{\partial Z_i}{\partial X_i} = \begin{cases} 1, & \mathcal{R}(Z_i) > 0 \ \vee \ \Im(Z_i) > 0 \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

As shown above, the proposed activation function and its derivative are computationally inexpensive. This is an important requirement for activation functions in CNNs. Part of the reason why ReLU has become the dominant activation function for spatial domain CNNs is that ReLU and its derivative are inexpensive to compute. The proposed activation function is also differentiable everywhere in the activation area except at the boundary of the activation area. ReLU is similarly not differentiable at origin, which serves as the boundary between values that are passed and values that are zeroed out. One should note that even though previously activation functions were expected to be non-constant, continuous and differentiable everywhere, many modern activation functions including ReLU do not fulfill all these criteria [4, 24].

2.2. A spectral domain CNN model for evaluating different activation functions

As described in Section 1, there have been a couple of models for realizing CNNs in spectral domain. Some realize only certain feature-extraction layers in spectral domain such as convolution [9], convolution and pooling [11] and convolution and activation function [18]. These models require multiple spatial-spectral domain switching as some feature-extraction layers are realized in spatial domain. On the other hand, some works realize all feature-extraction layers in spectral domain such as convolution, pooling and activation function [12] and convolution and activation function [14] (the latter work does not use a separate pooling layer as pooling operation is merged with convolution). We have developed a spectral CNN model for evaluating different activation functions, where all feature-extraction layers including convolution, pooling and activation function are realized in spectral domain.

In this spectral domain CNN model, we realize convolution with point-wise products in spectral domain. To perform convolution equivalent in spectral domain, first, real-valued input data are transformed to complex-valued Fourier space. The well-known convolution theorem establishes the duality between convolution in spatial domain and point-wise product in spectral (Fourier) domain. It states that convolution of two spatial inputs, say x and w, is equivalent to point-wise product of Fourier-transformed inputs, X and W, as defined in Equation 6, where $X = \mathcal{F}(x)$ and $W = \mathcal{F}(w)$. To compute an output pixel, a point-wise product requires just one complex multiplication between an input pixel and a kernel element in Fourier space. If convolution were performed on the equivalent spatial domain inputs, computing one output pixel would have required k^2 real-valued multiplications and k-1 real-valued additions. However, as equation 7 shows, each complex multiplication requires four real-valued multiplications and two real-valued additions. Here, A and B are real and imaginary components of X, while C and D are real and imaginary components of W. For typical-sized inputs and kernels in image processing, the number of multiplications and additions needed for point-wise products is much less than spatial convolution [8, 9, 10, 11].

$$X = \mathcal{F}(x) \quad (6a)$$

$$W = \mathcal{F}(w) \quad (6b)$$

$$x * w = \mathcal{F}^{-1}(X.W) \quad (6c)$$

$$\begin{aligned} X.W &= (A + iB).(C + iD) \\ &= A.C - B.D + i(B.C + A.D) \end{aligned} \quad (7)$$

We apply spectral pooling to perform dimensionality reduction in our model. Spectral pooling does this by eliminating higher frequencies in the spectral domain feature maps, while retaining lower frequencies. Spectral pooling causes minimal loss of information to the input feature maps as information in natural images is largely concentrated on lower frequencies. With spectral pooling one can crop any complex-valued input feature map to any arbitrary size determined by the designer [11]. This is illustrated in Equation 8, where X and Y represent input and output of the spectral pooling layer with dimensions of PxQ and RxS, respectively.

$$Y = \text{Crop}\,(X, R, S),\quad X \in \mathbb{C}^{PxQ},\quad Y \in \mathbb{C}^{RxS},\quad R < P, S < Q \qquad (8)$$

We have moved pooling layer before activation layer as was done in [14]. This allows the activation function to operate on a smaller feature map, which can enhance the speed of the activation layer. Following pooling operation, we apply our proposed complex-valued activation function to provide nonlinearity. This topology serves as the initial model of a feature-learning block of our spectral domain CNN model, as illustrated in Figure 3. Transformation between spatial to spectral domain is only needed in the first and last feature-extraction blocks (before the first and after the last convolution layers).

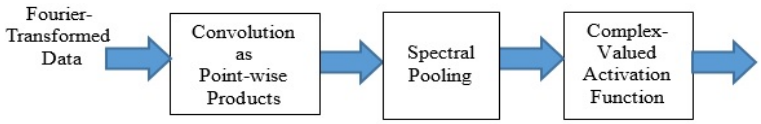

Figure 3. A feature-learning block of our spectral domain CNN model

The proposed activation function removes the need for multiple domain switching between spatial and spectral domains in the feature-learning segment of the network. Since input data and feature maps in spectral domain are complex-valued, a complex-valued activation function is compatible with spectral domain CNN networks. One should note that this CNN model performs feature-extraction in Fourier space, but unlike CVCNNs, it is meant to classify real-valued data.

We have developed a spectral domain CNN model for LeNet-5, where spatial convolution and pooling are replaced with point-wise product and spectral pooling and proposed complex-valued activation function is employed in place of *sigmoid* or ReLU activation functions (for first two feature extraction blocks) in the original model. The classification layers are kept spatial domain, as full connection or *softmax* layers are not that computationally intensive, when compared to convolution. In the last feature extraction block, an IFFT layer is utilized to transform spectral data to real-valued data for full-connection block. A spatial ReLU activation function follows this IFFT layer. A dropout layer has been added at the end of this last feature learning block to enhance generalization capability of the model. Figure 4 illustrates the general architecture of this spectral domain CNN model. For our model, we have kept the number of output feature maps same as the number used in MatConvNet's default LeNet-5 implementation (information about our CNN implementation environment including MatConvNet is provided in Section 3.1). Table 1 shows output feature map dimensions and their numbers as well as kernel dimensions for different layers in this spectral domain LeNet-5 model for digit and object recognition with MNIST and Fashion MNIST datasets.

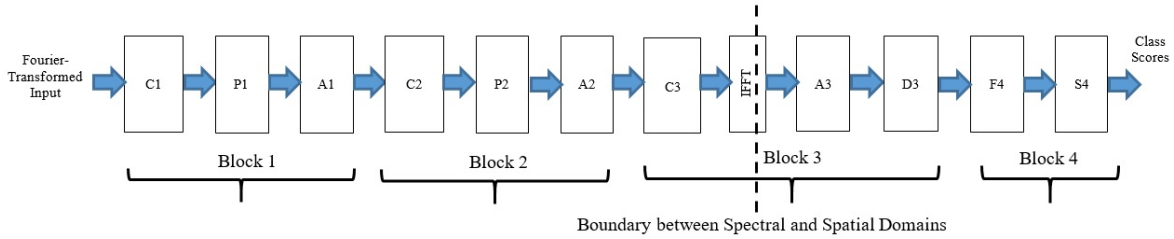

Figure 4. Functional block diagram of our spectral domain LeNet-5 CNN model

Table 1. No. of output feature maps and their dimensions in our spectral domain LeNet-5 CNN model

No of output feature maps		Dimensions of output feature maps and kernels		
		Layer	Dimension of output feature maps	Dimension of kernels
Block	Value	Input	28x28	-
1	20	C1	28x28	5x5
		P1	12x12	-
		A1	12x12	-
2	50	C2	12x12	5x5
		P2	4x4	-
		A2	4x4	-
3	500	C3	1x1	4x4
		A3	1x1	-
		D3	1x1	-
4	10	F4	1x1	1x1
		S4	1x1	-

As described above, the feedforward model is mostly in spectral domain, with only last activation layer and classification layers computed in spatial domain. For ease of computing error gradients, the backpropagation model is kept largely in spatial domain. The convolution in backpropagation architecture is computed in spatial domain as cyclic convolution, as in [14], to make it compatible with point-wise products in feedforward architecture. Error gradients for pooling and activation functions are initially computed in spectral domain and then converted to spatial domain.

2.3. Computational complexity analysis of proposed activation function

Computational complexity refers to the amount of resources (i.e. time, memory) required to execute a computational problem or algorithm. The type of computational complexity that would be useful to analyze running time of an algorithm such as inference time of a CNN model is time complexity. Time complexity describes the amount of time required to run an algorithm. Time complexity of an algorithm can be computed based on the number of elementary operations that need to be executed by the algorithm, assuming each elementary operation takes constant time to be executed by a given hardware platform. Sometimes time complexity can be difficult to compute exactly in this manner. In such cases, asymptotic behavior of the complexity is measured, that is running time of the algorithm for large inputs. Here, coefficients and lower-order terms of the complexity expression are ignored [25]. From here on when we refer to computational complexity, it would mean time complexity.

Computational complexity of some activation functions can be estimated based on the number of elementary operations. For example, ReLU is realized by a simple max operation, as shown in Equation 9, where X and Y represent input and output of the ReLU layer with both feature maps having a dimension of PxQ. For a ReLU layer, each pixel of the output feature map requires one max operation to be performed on the input, which checks whether input is larger than 0. For a 1-dimensional input of size N, each element (or pixel) of the output of this layer would require just 1 max operation on an input element and therefore, N output elements (or pixel) would require N number of operations. Thus the computational complexity of a ReLU activation layer with such 1-dimensional input (of size N) is N. In computer vision, inputs are generally 2-dimensional and with equal height and width. So, when a 2-dimensional image data with height P = N and width Q = N is provided to a ReLU activation layer, total number of elementary operations (in this case, max operations) that it needs to execute is N^2. So, naturally the computational complexity of ReLU activation function can be estimated to be $O(N^2)$.

$$Y = \max(0, X),\ X \in \mathbb{R}^{PxQ},\ Y \in \mathbb{R}^{PxQ} \quad (9)$$

Complex-valued activation functions such as ℂReLU [17], zReLU [19] and proposed activation function perform two max operations on an input element and hence perform $2N^2$ operations on a NxN input. These activation functions perform an additional N^2 operations, as compared to ReLU since they operate on both real and imaginary components of the input. However, their complexity is still $O(N^2)$ since constants are ignored in complexity estimation. Complexity of the spectral activation function proposed by [12] cannot be reliably estimated as they do not provide adequate implementation details for it. Another spectral activation function is *tanh* as proposed by [18]. *Tanh* implementations can vary in different implementation platforms. One of the typical real-valued *tanh* implementations involve multiple exponentiations with a complexity of $\log^3 N^2$ and some simpler operations (multiplication, division, addition and subtraction) of N^2 complexity [16, 26]. When *tanh* is applied on a spectral network, it is applied separately on both real and imaginary components of the input. Since computational complexity describes asymptotic behavior and ignores constants [25], the computational complexity would be $O(\log^3 N^2)$.

Spectral activation functions developed by [13] and [14] have to employ highly computationally intensive spectral convolution or auto-convolution. Spectral convolution for each output pixel would entail a complexity of N^2k^2, where N is the input feature map dimension and k is the kernel dimension. One should note that unlike spatial convolution, where k would be smaller in size than N, in case of auto-convolution, k is larger than N as it is a padded version of the input feature map [14]. Here, every output pixel is computed with k^2 complex multiplications, that entails $4k^2$ real-valued multiplications and $2k^2$ real-valued additions, that is $6k^2$ real-valued operations in total. This can be clearly seen in Equation 7. For an N^2 dimensional feature map, the number of real-valued operations become $6N^2k^2$ and hence complexity here is $O(N^2k^2)$. Since [14] utilizes a quadratic equation, it requires an extra $2N^2$ real-valued additions, but complexity remains $O(N^2k^2)$. Among all the related spectral or complex-valued activation functions, the proposed activation function as well as zReLU [19] and ℂReLU [17] require least number of operations ($2N^2$), possess the smallest complexity ($O(N^2)$) and do not require any computationally intensive auto-convolution. Table 2 lists computational complexity of these spectral and complex-valued activation functions that were developed for CNNs.

Table 2. Computational complexity of spectral and complex-valued activation functions

Activation function (AF)	Computational complexity
AF in [13]	$O(N^2k^2)$
tanh [18]	$O(\log^3 N^2)$
SReLU [14]	$O(N^2k^2)$
zReLU [19]	$O(N^2)$
ℂReLU [17]	$O(N^2)$
Proposed AF	$O(N^2)$

3. RESULTS AND DISCUSSION

3.1. Experimental Environment

To compare our work with previous research in this area we have tested our CNN model on the classic recognition dataset MNIST and its modern variant Fashion MNIST. MNIST is a widely used dataset in deep learning community with well-established benchmark accuracy. Introduced in 1998, MNIST is a dataset for digit classification. It consists of 28x28 greyscale images of handwritten digits of 10 classes (0 to 9). There are 60,000 training images and 10,000 test images [27]. Despite being developed 3 decades ago, MNIST continues to serve as the go to dataset for deep learning researchers for establishing proof-of-concept when exploring new algorithms. Fashion MNIST has the same structure (number of training and testing images) and images of same size as MNIST but consists of images of fashion articles such as shirts, bags and sandals. It is considered a more challenging dataset than MNIST [28].

For the experimental work for this research, we have described our CNN architectures in Matlab (v. 2018b) simulation environment utilizing MatConvNet (v.2015) toolbox that is tailor made for training CNNs. MATLAB is a development platform for scientists and engineers that has usage in diverse areas ranging from machine learning and computer vision to communications and computational finance. MatConvNet provides lot of built-in functions for CNNs, supports computation with GPUs and training large models (i.e. AlexNet, VGG) and complex datasets (i.e. ImageNet) [29]. The CNN model for this work was trained and tested on a desktop PC powered by 4-core Intel Core i7 4790K CPU with 8 GB DDR3 RAM operating at 4.00GHz and taking advantage of a single NVIDIA GeForce GTX 1070 GPU with 4GB RAM.

3.2. Experimental Work, Results and Discussion

We have analyzed test accuracy of our LeNet-5 CNN model employing proposed activation function for recognizing handwritten digits in the MNIST dataset and fashion articles in the Fashion MNIST dataset. Previous works have applied their activation functions in different CNN models and some were tested on different datasets. For instance, the complex-valued activation function ℂReLU [17] and auto-convolution-based spectral activation function [13] have been tested on CIFAR-10 dataset. Another complex-valued activation function zReLU [19] was tested on a specialized dataset. A feature-extraction block in [14] consists of a convolution layer (fused type) followed by an activation function, while the same block in the SpecNet model by [18] has convolution layer followed by activation function followed by a spatial pooling layer. We have applied all these activation functions in our CNN model and tested them on the same MNIST and Fashion MNIST datasets so that a fair comparison can be made between this work and related previous works.

The original work that introduced the world to LeNet-5 and modern CNN networks, LeCun et al. in [27], attained 99.05% accuracy on MNIST dataset. This was a spatial domain CNN network. Highlander et al. in [8] developed a spectral domain CNN architecture for LeNet-5 that uses spatial activation function and attains 92.46% test accuracy. So, this network has multiple transformations between spatial and spectral domains to perform activation function in spatial domain. The work by Ko et al. in [12] presents a spectral domain CNN solution without domain transformations that employs a linear spectral activation function but provides only training accuracy. This work does not describe basic properties of their activation function and how it was implemented. Recently Pratt et al. in [21] developed another spectral domain CNN approach without domain transformations, attains 97% test accuracy but does not mention the type of activation functions, or whether any such has been used at all. Due to lack of implementation details for these two activation functions, we have not applied them in our model to compare our work with theirs. Very recently, Ayat et al. in [14] has achieved an accuracy similar to spatial domain CNN solutions, attaining 99.20% accuracy. Spectral domain CNN model with split type *tanh* proposed by Guan et al. in [18] attains 95% accuracy. This is the average accuracy of six models they propose, where the models differ based on compression rates for input feature maps. These spectral domain CNN solutions as well as the spatial domain CNN of LeCun et al. in [27] evaluated their models on MNIST dataset using LeNet-5 CNN. For this same MNIST dataset, our spectral domain LeNet-5 CNN model with proposed activation function attains a test accuracy of 97.65%, an accuracy very close to all these previous works—in some cases outperforming them—as can be seen in Table 3. However, to have a fair comparison of test accuracies (since all these works use different architectures for LeNet-5 model (having different types of feature-extraction blocks) and different number of parameters (such

as number of output feature maps)), all these activation functions have been evaluated in the same model discussed in Section 2.2. Accuracy and inference times attained by this model with many of these activation functions (as well as proposed activation function) are listed in Tables 4 and 5 for MNIST and Fashion MNIST datasets, respectively.

Table 3. Test accuracy of LeNet-5 CNN model attained by some of the previous works on spatial and spectral CNNs evaluated on MNIST dataset

Work	Year	Domain	Activation Function (AF)	Test Accuracy
LeCun et al. [27]	1998	Spatial	*Sigmoid*	99.05%
Highlander et al [8]	2016	Spectral	-[*1]	92.46%
Ko et al. [12]	2017		Linear approximation of *tanh/sigmoid*	96%[*2]
Pratt et al. [21]	2017		-[*1]	97%
O. Ayat et al. [14]	2019		SReLU	99.20%
B. Guan et al. [18]	2019		*tanh*	95%
This work	2020		Proposed AF	97.65%

[*1] Information regarding activation function applied in these models are not provided in these works.
[*2] Only training accuracy is provided by Ko et al. in [12]

Our spectral domain LeNet-5 model was trained using Adagrad optimizer. For MNIST and Fashion MNIST datasets, the model was trained for 60 epochs with a logarithmically spaced learning rate varying from 0.005 to 0.0001. The network was trained using all the 60,000 training samples of MNIST and Fashion MNIST datasets. At the end of the aforementioned epochs, the model attained a training accuracy of 98.52% (error of 1.48%) and 88.95% (error of 11.05%) for MNIST and Fashion MNIST datasets, respectively. Figure 5 shows the training and test errors for these datasets.

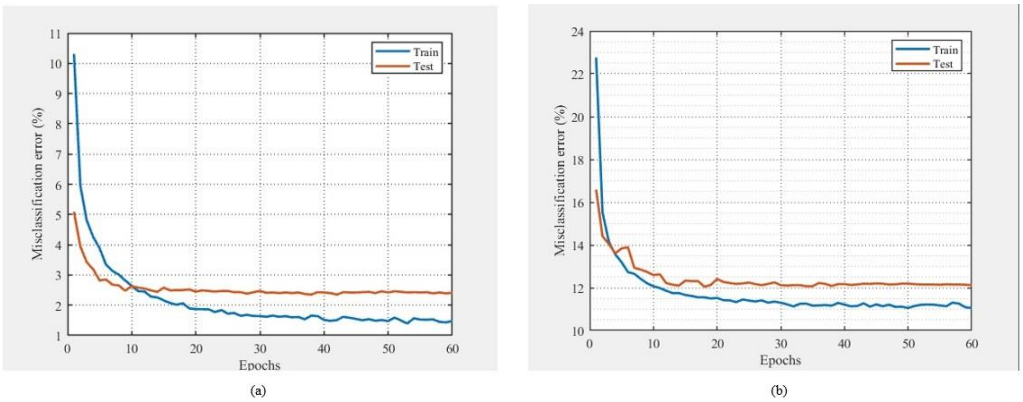

Figure 5. Training and test errors for our spectral domain CNN model for (a) MNIST dataset and (b) Fashion MNIST dataset

For inference, all test samples-10,000 in total-were tested, for MNIST and Fashion MNIST datasets. During each epoch of training, MatConvNet applies inference on test data and records inference times. We took average of inference times (for all 10,000 test samples) recorded for first 30 epochs and in multiple runs to compare inference times of our model with different activation functions. We recorded peak accuracy attained on test data after training was completed. Tables 4 and 5 documents the peak test accuracy and inference time attained with different activation functions for MNIST and Fashion MNIST datasets, respectively. The difference in accuracy, denoted as $Diff_{Acc}$ in tables 4 and 5, is the change in accuracy for an activation function (say A) as compared to the accuracy attained with proposed activation function (say B). So, a negative $Diff_{Acc}$ (when B-A is negative) for an activation function would mean its accuracy is lower than accuracy attained with proposed activation function, while a positive $Diff_{Acc}$ (when B-A is positive) would mean the opposite (proposed activation function attaining a lower test accuracy). The percentage difference in inference time is denoted as $Diff_{Inf}$ in tables 4 and 5. This reflects the percentage change in inference time when compared to the inference time attained with the proposed activation function. So, if the inference time of our model with proposed activation function is denoted as Q and inference time of the same model with another activation function is denoted as P, $Diff_{Inf}$ is calculated as 100*(P-Q)/Q. Here also a negative $Diff_{Inf}$ would mean inference time with an activation function is longer than inference time with proposed activation function, while a positive $Diff_{Inf}$ would mean inference time attained with proposed activation function is longer.

For MNIST dataset, our spectral model with proposed activation function attained 97.65% peak accuracy (error of 2.35%) in inference, outperforming four state-of-the-art spectral or complex-valued

activation functions when employed in the same model. As table 4 illustrates, our activation function outperforms *tanh* [18] by 7% and SReLU [14] and zReLU [19] by about 1%. When inference time is compared, this work offers about 79% and 8.5% shorter inference times than SReLU and zReLU, respectively. The inference times with *tanh* and proposed activation function are virtually same differing by a mere 0.15%. Our test accuracy is almost same as ℂReLU [17] (difference of 0.24%) but we offer about 9.6% shorter inference time with our activation function. Even though zReLU, ℂReLU and the proposed activation function have the same computational complexity, the run time of their elementary functions may not be the same. For example, proposed activation function has to modify only a small number of input values (values in the third quadrant), when compared to ℂReLU. That is why our model provides faster inference with proposed activation function than with ℂReLU despite both activation functions having the same computational complexity. Overall, our approach outperforms three state-of-the-art spectral or complex-valued spatial activation functions in terms of both test accuracy and inference time, in varying degrees. In case of *tanh*, proposed activation function outperforms it significantly in terms of accuracy while offering almost same inference time.

Table 4. Test accuracy and average inference times for our spectral domain LeNet-5 CNN model with different activation functions evaluated on MNIST dataset

Work	Year	Activation Function (AF)	Peak Test Accuracy	$Diff_{Acc}$	Mean Inference Time (s)	$Diff_{Inf}$
O. Ayat et al. [14]	2019	SReLU	96.31%	-1.34%	10.09	-79.04%
B. Guan et al. [18]	2019	*tanh*	90.46%	-7.19%	5.63	+0.15%
N. Guberman [19]	2016	zReLU	96.68%	-0.97%	6.12	-8.49%
C. Trabelsi et al. [17]	2017	ℂReLU	97.41%	-0.24%	6.18	-9.58%
This work	2020	Proposed AF	97.65%	0.00%	5.64	0.00%

The work by [14] proposing SReLU is the only work that applies a spectral CNN model for Fashion MNIST dataset, achieving a test accuracy of 80.54%. We evaluate our model for Fashion MNIST using the same activation functions that were employed to test our model for MNIST dataset, including SReLU. As table 5 shows, our model employing proposed activation function achieves 87.95% test accuracy (error of 12.05%) for Fashion MNIST dataset, outperforming state-of-the-art spectral or complex-valued activation functions. Our activation function outperforms SReLU [14] and *tanh* [18] in test accuracy by about 6% and 2.8%, respectively. It also offers about 85% and 7.6% faster inference time over the aforementioned activation functions. Our activation function achieves almost same test accuracy as ℂReLU [17] (a difference of about 0.5%) but offers about 14% faster inference time. In case of comparison with zReLU [19], proposed activation function achieves about 4% higher test accuracy but inference time with our activation function is slightly longer, by about 5%. In the end, we offer better accuracy and inference time over three state-of-the-art spectral/complex-valued activation functions. In case of one activation function (zReLU), we offer better accuracy.

Table 5. Test accuracy and average inference times for our spectral domain LeNet-5 CNN model with different activation functions evaluated on Fashion MNIST dataset

Work	Year	Activation Function (AF)	Peak Test Accuracy	$Diff_{Acc}$	Mean Inference Time (s)	$Diff_{Inf}$
O. Ayat et al. [14]	2019	SReLU	81.89%	-6.06%	10.95	-85.08%
B. Guan et al. [18]	2019	*tanh*	85.16%	-2.79%	6.37	-7.62%
N. Guberman [19]	2016	zReLU	84.03%	-3.92%	5.63	+4.85%
C. Trabelsi et al. [17]	2017	ℂReLU	87.46%	-0.49%	6.73	-13.71%
This work	2020	Proposed AF	87.95%	0.00%	5.92	0.00%

4. CONCLUSION

In this work, it was demonstrated that a spectral domain CNN model employing proposed complex-valued activation function can provide a low-complexity CNN model that offers very accurate and fast inference and can do so without multiple and expensive spectral-spatial domain switchings. This work has demonstrated the effectiveness of this model by applying it for two types of recognition tasks—recognotion of hand-written digits and objects. Its performance was evaluated in terms of accuracy and inference time on standard recognition benchmark datasets (MNIST and Fashion MNIST). The significant acceleration of inference speed achieved in our spectral domain CNN model while providing high classification accuracy can potentially lead to effective CNN solutions for implementation in resource and energy-constrained embedded systems. As future work, one can explore effectiveness of our activation function in terms of classification accuracy, reduction in computational complexity and speedup of inference time when applied in larger

networks and tested for more complex datasets. Another avenue for future work would be to develop a methodology to train our CNN model entirely in spectral domain.

REFERENCES

[1] N. Buduma and N. Locascio, *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. Sebastopol, CA: O'Reilly Media, Inc., 2017.

[2] H. Aghdam and E. Heravi, *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. Cham, Switzerland: Springer International Publishing, 2017.

[3] W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review", *Neural Computation*, vol. 29, pp. 2352-2449, 2017. Available: https://www.researchgate.net/publication/317496930_Deep_Convolutional_Neural_Networks_for_Image_Classification_A_Comprehensive_Review

[4] A. Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks", *2012 Neural Information Processing Systems conference*, Vol. 1, pp. 1097-1105, 2012. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[5] O. Russakovsky, J. Deng, H. Su, J. Krause et al, "ImageNet Large Scale Visual Recognition", *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, 2015. Available: https://link.springer.com/article/10.1007/s11263-015-0816-y?sa_campaign=email/event/articleAuthor/onlineFirst#

[6] R. Girshick et al., "Fast R-CNN", *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1440-1448, 2015. Available: https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf

[7] J. Long et al., "Fully Convolutional Networks for Semantic Segmentation", *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431-3440, 2015. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Long_Fully_Convolutional_Networks_2015_CVPR_paper.pdf

[8] T. Highlander et al., "Very Efficient Training of CNNs using Fast Fourier Transform and Overlap-and-Add", *arXiv: 1601.06815 [cs]*, Jan, 2016. Available: https://arxiv.org/pdf/1601.06815.pdf

[9] M. Mathieu *et al.*, "Fast Training of Convolutional Networks through FFTs", *arXiv: 1312.5851 [cs]*, Mar 2014. Available: https://arxiv.org/pdf/1312.5851.pdf

[10] N. Vasilache *et al.*, "Fast Convolutional Nets with Fbfft: A GPU Performance Evaluation". *arXiv: 1412.7580 [cs]*, Apr 2015. Available: https://arxiv.org/pdf/1412.7580.pdf

[11] O. Rippel *et al.*, "Spectral Representations for Convolutional Neural Networks". *arXiv: 1506.03767 [stat]*, Jun 2015. Available: https://arxiv.org/pdf/1506.03767.pdf

[12] J. Ko *et al.*, "Design of an Energy-Efficient Accelerator for Training of Convolutional Neural Networks using Frequency-Domain Computation", *2017 ACM/IEEE Design Automation Conference*, pp. 1-6, 2017. Available: https://dl.acm.org/doi/pdf/10.1145/3061639.3062228

[13] W. Uijens, *Activating Frequencies-Exploring Non-Linearities in the Fourier Domain*. Master thesis, Delft University of Technology, Netherlands, 2018. Available: https://repository.tudelft.nl/islandora/object/uuid%3Ab6dfdac6-691d-44a4-bacb-645df5cfdeaf

[14] S. Ayat, M. Khalil-Hani, A. Ab Rahman and H. Abdellatef, "Spectral-based convolutional neural network without multiple spatial-frequency domain switchings", *Neurocomputing*, vol. 364, no. 2019, pp. 152-167, 2019. Available: https://www.sciencedirect.com/science/article/pii/S0925231219310148

[15] M. Francesca *et al.*, "Spectral Convolution Networks", *arXiv: 1611.05378 [cs]*, Nov 2016. Available: https://arxiv.org/pdf/1611.05378.pdf

[16] S Scardapane *et al.*, "Complex-valued Neural Networks with Non-parametric Activation Functions", *arXiv: 1802.08026 [cs]*, Feb 2018. Available: https://arxiv.org/pdf/1802.08026.pdf

[17] C. Trabelsi *et al.*, "Deep complex networks", *arXiv: 1705.09792 [cs]*, Feb 2018. Available: https://arxiv.org/pdf/1705.09792.pdf

[18] B. Guan *et al.*, "SpecNet: Spectral Domain Convolutional Neural Network", *arXiv: 1905.10915 [cs]*, Nov 2019. Available: https://arxiv.org/pdf/1905.10915.pdf

[19] N. Guberman, "On Complex Valued Convolutional Neural Networks", *arXiv: 1602.09046 [cs]*, Feb 2016. Available: https://arxiv.org/pdf/1602.09046.pdf

[20] M. Arjovsky *et al.*, "Unitary Evolution Recurrent Neural Networks", *arXiv: 1511.06464 [cs]*, May 2016. Available: https://arxiv.org/pdf/1511.06464.pdf

[21] H Pratt *et al.*, "FCNN: Fourier Convolutional Neural Networks", *2017 European Conference On Machine Learning & Principles and Practice Of Knowledge Discovery In Databases*, pp. 786-798, 2017.

[22] C-A. Popa, "Complex-Valued Convolutional Neural Networks for Real-Valued Image Classification", *2017 International Joint Conference on Neural Network*, pp. 816-822, 2017. Available: https://ieeexplore.ieee.org/document/7965936

[23] B. Karlik and A. Olgac, "Performance analysis of various activation functions in generalized MLP architectures of neural networks", *International Journal of Artificial Intelligence And Expert Systems*, vol. 1, no. 4, pp. 111-122, 2011. Available: https://www.cscjournals.org/manuscript/Journals/IJAE/Volume1/Issue4/IJAE-26.pdf

[24] X. Glorot *et al.*, "Deep Sparse Rectifier Neural Networks", *2011 International Conference on Artificial Intelligence and Statistics*, pp. 315-323, 2011. Available: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf

[25] M. Sipser, *Introduction To The Theory of Computation*, 3rd Edition. Canada: Cengage Learning, 2013.

[26] T. Ahrendt, "Fast Computations of the Exponential Function", 1999 Annual Symposium on Theoretical Aspects of Computer Science, pp. 302-312, 1999. Available: https://link.springer.com/content/pdf/10.1007/3-540-49116-3_28.pdf
[27] Y. LeCun, L. Botou, Y. Bengio and P. Haner, "Gradient-Based Learning Applied To Document Recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
[28] H. Xiao *et al.*, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms", *arXiv:1708.07747 [cs]*, Sep 2017. Available: https://arxiv.org/pdf/1708.07747.pdf
[29] A. Vedaldi *et al.*, "MatConvNet: Convolutional Neural Networks for MATLAB", *arXiv: 1412.4564 [cs]*, May 2016. Available: https://arxiv.org/pdf/1412.4564.pdf

BIOGRAPHY OF AUTHORS

Shahriyar Masud Rizvi received his BS and MS degrees in Electrical Engineering in 2000 and 2002, respectively, from University of Wyoming, Laramie, WY, USA. He is currently pursuing a PhD degree in Electrical Engineering at Universiti Teknologi Malaysia (UTM). His Ph.D. research focuses on low-complexity deep neural networks and spectral image processing. He also serves as an Associate Professor in Faculty of Engineering at American International University-Bangladesh (AIUB).

Ab Al-Hadi Ab Rahman received his Ph.D. degree from École Polytechnique Fédérale de Lausanne, Switzerland in 2013, M.Eng. degree from Universiti Teknologi Malaysia in 2008, and B.S. degree from University of Wisconsin-Madison, USA in 2004. His current research interests are on optimization methods and design automation for applications in video coding and deep learning. He has authored and co-authored more than 30 journals and conference papers, mainly with contributions in developing new design methodologies and techniques for high performance and low power systems. He is currently a senior lecturer at Universiti Teknologi Malaysia.

Mohamed Khalil-Hani received his Bachelor's degree in Communications Engineering from the University of Tasmania, Hobart, Australia, in 1978 and his M.Eng. in Computer Architecture from Florida Atlantic University, Boca Raton, USA, in 1985. He obtained his Ph.D. degree from Washington State University, Pullman, USA, in1992, specializing in digital systems and computer engineering. He is a senior member of IEEE. Currently, he is a professor in digital systems, computer architecture and FPGA system-on-chip (SoC); serving in the University of Technology, Malaysia (UTM) since 1981. His current research interests include real-time digital computing, hardware/software codesign and FPGA SoC architectures for applications in pattern recognition, image processing, neuro-intelligent hardware systems, biometrics, and quantum computing emulation. He has worked and published extensively in these research areas for more than 19 years. He has supervised and graduated more than 40 postgraduate research students (Ph.D. and Masters).

Sayed Omid Ayat received his B.Eng. degree in electrical-electronic from Azad University of Najaf Abad, Iran in 2009, and his M.Eng. degree in Electrical-Computer and Microelectronic System from Universiti Teknologi Malaysia in 2014. He obtained his Ph.D. degree in Electrical Engineering from Universiti Teknologi Malaysia in 2020. His research focuses on FPGA system-on-chip (SoC), hardware/software codesign, deep neural networks, and spectral image processing.