Applied Intelligence (2023) 53:4499–4523
https://doi.org/10.1007/s10489-022-03756-1

Computation and memory optimized spectral domain convolutional neural network for throughput and energy-efficient inference

Shahriyar Masud Rizvi[1] · Ab Al-Hadi Ab Rahman[1] · Usman Ullah Sheikh[1] · Kazi Ahmed Asif Fuad[2] · Hafiz Muhammad Faisal Shehzad[3]

Accepted: 10 May 2022 / Published online: 11 June 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Conventional convolutional neural networks (CNNs) present a high computational workload and memory access cost (CMC). Spectral domain CNNs (SpCNNs) offer a computationally efficient approach to compute CNN training and inference. This paper investigates CMC of SpCNNs and its contributing components analytically and then proposes a methodology to optimize CMC, under three strategies, to enhance inference performance. In this methodology, output feature map (OFM) size, OFM depth or both are progressively reduced under an accuracy constraint to compute performance-optimized CNN inference. Before conducting training or testing, it can provide designers guidelines and preliminary insights regarding techniques for optimum performance, least degradation in accuracy and a balanced performance–accuracy trade-off. This methodology was evaluated on MNIST and Fashion MNIST datasets using LeNet-5 and AlexNet architectures. When compared to state-of-the-art SpCNN models, LeNet-5 achieves up to 4.2× (batch inference) and 4.1× (single-image inference) higher throughputs and 10.5× (batch inference) and 4.2× (single-image inference) greater energy efficiency at a maximum loss of 3% in test accuracy. When compared to the baseline model used in this study, AlexNet delivers 11.6× (batch inference) and 5× (single-image inference) increased throughput and 25× (batch inference) and 8.8× (single-image inference) more energy-efficient inference with just 4.4% reduction in accuracy.

Keywords Convolutional neural network · Spectral domain CNN · Energy efficiency · Computational workload · Memory access cost

1 Introduction

Deep neural networks (DNNs) have recently evolved as the prevalent solution for a range of challenging problems in computer vision, such as image recognition [1], image segmentation [2–4], set based image classification [5], and image clustering [6] as well as language processing [7] and autonomous systems [8]. Convolutional neural networks (CNNs) [9, 10], a class of DNNs, have achieved unprecedented success in various fields of computer vision, audio analysis, and text processing including–inter alia–object classification [11, 12], object detection [13, 14], semantic segmentation [15, 16], face verification [17, 18], video understanding [19], audio classification [20, 21], and natural language processing [22]. In the last few years, CNNs have been deployed in diverse applications such as autonomous driving [23], navigation systems [24] and flight safety [25] for drones, skin cancer detection [26], COVID-19 prognosis [27], and VLSI physical design [28].

In CNNs, convolution layers play a central role in feature-extraction [9–11] but demand high computational resources [29]. They account for 90% of CNN operations [30]. Moreover, deeper CNNs (with more convolution layers), which tend to produce higher accuracy, possess a larger number of parameters. This results in increased memory requirements [29]. However, run-time memory during inference is dominated by storage of intermediate output feature maps [31], even with a batch size of 1 [32]. Memory consumed by these output feature maps (OFMs) can exceed parameter memory by 10 to a 100 times [33]. Storing intermediate OFMs requires off-chip dynamic random access memory (DRAM) accesses, which consume more power and energy than computations [29, 34]. In fact, for devices limited by memory bandwidth, the

✉ Shahriyar Masud Rizvi
m.rizvi@graduate.utm.my

Extended author information available on the last page of the article.

memory access cost can be the main bottleneck for power consumption and inference latency including in GPU-based platforms [15, 35, 36]. Therefore, the computational, memory and power budgets of even classical CNNs (with only a few convolutional layers) are such that deploying them in embedded systems or mobile platforms is extremely challenging [29, 30, 32, 35].

A potent approach to reduce computational workload of conventional CNNs is to represent CNNs, especially convolution, in spectral domain using Fourier transformation [37–40]. Spectral domain CNNs (SpCNNs) compute convolution as point-wise multiplication in Fourier space, which significantly reduces the computational workload. Here, each OFM element can be computed from one complex-valued product, instead of many real-valued products accumulated over the receptive field, as is the case in spatial domain [37–40].

A few previous studies have explored methods to reduce memory usage in SpCNNs. Many of these approaches aim to reduce either the size of parameter memory (often by reducing the number of parameters) or the memory access cost of domain transformations (by optimizing Fourier transforms), rather than reducing the memory access cost of intermediate OFMs. For example, Niu et al. [41] compress weights to reduce the number of parameters, while Sun et al. [42] quantize weights to reduce the amount of parameter memory. Studies such as [43, 44] split input images of convolution layers so fast Fourier transform (FFT) is performed on image parts, rather than whole images, to reduce the memory access cost of domain transformations. These latter works compute only convolution in the spectral domain and hence require multiple domain transformations. Most of the above-mentioned works, such as [41–44], require a dedicated hardware accelerator to take advantage of these approaches.

In this work, we analyze computational workload and memory access cost (CMC) of SpCNNs and propose a methodology for SpCNNs to compute CNN inference in a computationally inexpensive and memory-efficient manner. The major contributions of this work include the following.

1. This work reduces the computational workload of CNN by computing the entire feature-extraction segment (consisting of convolution, pooling and activation layers) in the spectral domain. Usage of a computationally light activation function proposed by Rizvi et al. [45] and only one set of domain transformations ensure that the model stays computationally inexpensive.
2. Next, the computational workload and memory access cost of SpCNNs are investigated analytically. This analysis allows designers to estimate the effect of OFM size and depth of different convolution layers on the overall computational and memory costs.
3. Based on the analysis in part 2, a methodology containing three strategies is proposed herein to achieve performance-optimized inference. Here, OFM size (Strategy 1), OFM depth (Strategy 2) or both (Strategy 3) are progressively reduced until an energy-efficient and throughput-optimized inference is achieved under an accuracy constraint. This methodology provides guidelines regarding in which layer and in what quantity OFM size and depth can be optimized to reduce computational workload and memory access cost. It can also provide preliminary insights regarding strategies for faster and more energy-efficient inference, minimal degradation in accuracy, and balanced performance–accuracy trade-off. The proposed methodology is non-intrusive and does not require a specialized accelerator, specialized module, or libraries or any major modification of the CNN model.

The remainder of this paper is structured in the following manner. Section 2 reviews previous studies that are related to this work. Section 3 introduces SpCNNs and the baseline CNN models used in this work for evaluating the proposed methodology. Section 4 discusses the problem formulation and analytically explores the computational workload and memory access cost of SpCNNs. Section 5 describes the proposed methodology for reducing the computational workload and memory access cost to enhance inference performance. It also presents an estimation for gain in inference performance under the proposed methodology. Afterwards, Section 6 discusses experimental results, and finally, Section 7 presents the concluding remarks.

2 Related work

When CNNs are computed in spatial domain, one can address their prodigious computational workload by replacing standard spatial domain convolution with computationally light convolution methods such as depth-wise separable convolution [46, 47] and grouped convolution [11, 48]. However, applying depth-wise convolution to a certain type of convolution layers such as convolution layers with 1×1 kernels results in a significant reduction in accuracy [48]. In the case of grouped convolution, state-of-the-art deep learning frameworks, such as PyTorch [49] or TensorFlow [50], do not provide an expected reduction in inference time [51]. As discussed in Section 1, SpCNNs provide a powerful alternative to these spatial domain approaches for computing CNN in a computationally inexpensive manner.

Among these two approaches discussed above, the spectral domain representation of CNNs was considered to be a more effective option. This is because in addition to offering a significant reduction in the computational workload by computing convolution as a point-wise product, SpCNNs, through spectral pooling, provide another route to machine learning designers to tune computational resources and memory usage. This is because with spectral pooling one can down-sample feature maps to any arbitrary size. Spectral pooling is also known to retain more information after pooling than spatial domain counterparts [39].

Early SpCNNs computed only convolution in spectral domain [37, 38]. In contrast to these works, recent SpCNN models realize the entire feature-extraction segment in spectral domain including non-linear layers of pooling and activation function [40, 45, 52–54]. These solutions require only a single (instead of multiple) set of domain transformations, which does not add any significant computational or memory access cost. This set of transformations include an FFT applied on the input data before the first convolution layer and an inverse FFT (IFFT) after the last convolution layer [40, 45, 52].

Some recent works in SpCNNs have proposed methods to optimize computational workload. Ayat et al. [52] propose a fused convolution layer for spectral domain. In a fused convolution layer, pooling operation is performed before convolution is computed. As a result, fused convolution layers have to process smaller-sized input feature maps (IFMs). Since IFMs and OFMs of convolution layers in SpCNNs have the same size [45, 52], a smaller sized IFM automatically means a smaller-sized OFM. This results in reduced computations, when compared to regular convolution layers. The authors here propose a convolution-based activation function, which approximates ReLU [11] in spectral domain. This activation function is computationally very expensive and thus negates some of the gain in computation reduction. Liu et al. [54] propose methods to obtain optimal coefficients for ReLU approximation first and then modify them with hardware-friendly coefficients for increased computational efficiency. In addition, they propose optimizations for FPGA acceleration such as integer approximation for point-wise products (for convolution layers). Rizvi et al. [45] propose a computationally inexpensive activation function for SpCNNs. Their model exhibits lower computational workload than Ayat et al.'s model [52] when regular convolution layers are employed.

There have been very few works in SpCNNs that investigated the reduction of run-time memory (memory access cost at run-time) by shrinking the size and depth of intermediate OFMs. As discussed before, Ayat et al. [52] reduce feature map size using their fused convolution layer. However, the reduction in memory access cost is not discussed in this work. Guan et al. [55] compress IFMs before convolution by retaining only values that are above a certain threshold, resulting in a sparse input to the convolution layers. The authors here prioritize sparse storage of OFMs, rather than reducing the number of memory accesses. Furthermore, this work realizes pooling in the spatial domain and hence has to contend with multiple sets of domain transformations.

The works discussed above optimize OFMs through size reduction or data-level optimization. However, the reduction of depth of OFMs (also known as the number of output channels), which is a significant contributor to the computational and memory burden, is not considered in these works. Our work demonstrates that reduction of OFM depths (in addition to OFM sizes), under an accuracy constraint, can ensure that inference with SpCNNs is computed in a computationally inexpensive and memory-efficient manner. Furthermore, this can be done without employing any custom compression algorithm or accelerator.

3 Spectral domain CNN models

3.1 Background

CNN architectures are composed of two functional segments, a convolution-based feature-extraction segment and a multilayer perceptron (MLP)-based classification segment. The feature-extraction segment consists of a series of repeating blocks with convolution layer, activation function, and pooling layer [9–11]. Each of these feature-extraction blocks starts with a convolution layer, or CONV layer in short. The activation function either follows the CONV layer or the pooling layer [52]. If the activation function follows the CONV layer, the block can be denoted as a convolution-activation-pooling (CAP) block. Blocks, where CONV layer is followed by a pooling layer, can be denoted as a convolution-pooling-activation (CPA) block. The classification segment is composed of one or more fully connected (FC) layers and an activation layer at the end that performs multi-class, single-label classification [9–11]. Many CNNs, including this work, utilize a *softmax* function for this layer [10–12]. Hereafter, this layer would be referred to as simply the *softmax* layer.

In the case of SpCNNs, at least one additional set of layers is required for domain transformations [40, 45, 52]. One spatial-to-spectral domain transformation through an FFT layer is needed before the first CONV layer to convert spatial input data to spectral domain [40, 45, 52]. For weights, performing FFT during inference is not necessary as trained weights can be readily provided in a spectral format for inference [52]. In SpCNNs, convolution is computed as a point-wise product (also known as Hadamard product) of inputs and kernels in spectral domain [37]. If only convolution is performed in the spectral domain, rather than the full CPA or CAP block, each CONV layer has to be succeeded by an FFT layer [37–39]. If the entire feature extraction is computed in the spectral domain, as in our work, this is not necessary. However, one spectral-to-spatial domain transformation through an IFFT layer would be needed at the end of the feature-extraction segment so that the classification segment can be computed in the spatial domain [40, 45, 52]. Before providing data to the classification segment, the output of the IFFT layer is flattened to a single-dimension (1×1 size) since fully connected layers can only process single-dimensional data [45]. The functional architecture of SpCNN models with one set of domain transformation layers is illustrated in Fig. 1.

3.2 Baseline LeNet-5 and AlexNet SpCNN models

For this work, we employ SpCNN models with CPA type feature-extraction blocks to evaluate our strategies for enhancing inference performance. In each of these blocks, convolution is computed as a point-wise Hadamard product [37] and is followed by a spectral pooling layer, developed by Rippel et al. in [39]. The spectral pooling layer is followed by a complex-valued activation layer suitable for the spectral domain. For activation function, a computationally light and complex-valued activation function proposed by Rizvi et al. [45] is utilized. This activation function propagates input if either the real or imaginary part is positive-valued, otherwise zero is transmitted. We call this activation function PosReLU. The functional architecture of a typical feature-extraction block used in this work is shown in Fig. 2.

In order to evaluate the proposed methodology for gain in inference performance, two classical CNN architectures—LeNet-5 (developed by LeCun et al. [10]) and AlexNet (developed by Krizhevsky et al. [11])—were selected. These two architectures are moderately deep with sufficient number of convolution layers that produce high accuracy in many standard image recognition datasets. This attracts researchers to validate their new models or algorithms (i.e., establish proof-of-concept) on these architectures. In addition, convolution layers in both these architectures are computationally and parameter-wise expensive. Therefore, these architectures are ideal for validating algorithms or optimization strategies for higher computational and memory-efficiency. Furthermore, recent works in SpCNNs such as [45, 52, 54] have validated their models or algorithms on these architectures (e.g., LeNet-5).

In this work, we denote a specific CNN implementation on LeNet-5 and AlexNet architectures, such as an SpCNN implementation or an implementation with specific sets of OFM sizes and depths, as LeNet-5 and AlexNet models. For this work, the number of CONV and fully connected layers for the baseline LeNet-5 model are kept the same as the original model proposed in [10]. In the case of AlexNet, the baseline model shares the same number of CONV layers as the original model [11]. However, the baseline AlexNet model for this work has a single fully connected layer instead of the three that is present in the original model [11]. In this work, we discovered that one fully connected layer for the AlexNet SpCNN model is sufficient to achieve excellent test accuracy for standard datasets utilized in this work. Figures 3 and 4 depict the functional architecture of the baseline SpCNN implementations of LeNet-5 and AlexNet for this work.

It is worth noting that LeNet-5 and AlexNet CNNs can be realized in both spatial and spectral domains. The conventional spatial domain LeNet-5 and AlexNet models and their SpCNN counterparts have largely similar overall functional architecture. However, individual layers that perform similar roles in feature-extraction segments in both spatial domain CNNs and SpCNNs operate differently as the computations are performed in different domains. In addition to convolution being computed differently in spatial and spectral domains, the size of IFMs for the convolution layers also varies between the domains. This is because a convolution operation in spatial domain produces OFMs that are smaller in size than IFMs. For example, a convolution operation of an F×F sized IFM results in an OFM of size (F−k+1)×(F−k+1), where k×k represents the size of the kernel [37]. On the other hand, because of the point-wise nature of the Hadamard product, convolution layers in SpCNNs produce OFMs that have the same size as their IFMs [37]. In addition, the non-linear layers (pooling and activation functions) also operate differently in the two domains. In spatial domain CNNs, pooling methods such as max-pooling [11] are typically employed, while ReLU [11] is widely used as the activation function. SpCNN models typically utilize spectral pooling [39]. Recently, researchers have deployed spectral ReLU (SReLU) [52], complex-valued *tanh* [55] and PosReLU [45] as activation functions for SpCNNs.

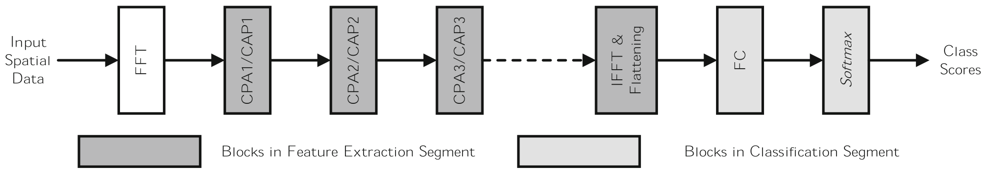


Fig. 1 Generic functional architecture of SpCNN Models with one set of domain transformation layers

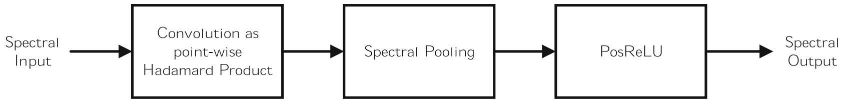


Fig. 2 Functional architecture of feature-extraction blocks for the SpCNN models used in this work

4 Problem formulation

4.1 Computational workload (CW)

CNNs can feature different types of layers. However, five types of layers are essential, namely, convolution, pooling, activation, fully connected, and multi-class classification layer [9–11]. Pooling layers, multi-class classification layers, and most activation functions are executed with simple point-wise operations. As these operations do not require arithmetic operations, they can be ignored when computing the workload of a CNN [45, 52]. The majority of the computational workload (CW) in CNNs is attributable to the multiply-accumulate (MAC) operations [29, 30, 56], which are executed in convolution and fully connected layers [29, 52]. Each multiply-accumulate operation involves one multiplication and one addition. Thus, the CW of a CNN in terms of MACs can be computed by counting the number of these MAC operations.

SpCNNs realize the equivalent of spatial convolution by performing point-wise product of IFMs and kernels in complex-valued Fourier domain [37]. In such solutions, kernels have the same size as IFMs. SpCNNs require a single complex-valued product between an IFM element and a kernel element to produce the corresponding OFM element, instead of the accumulation of many real-valued products over the receptive field, as is the case in the spatial domain [37–39]. When provided with a single IFM of size M×M and a kernel of size M×M, computing an OFM of size M×M would entail M^2 complex-valued multiplications. Because of the multi-dimensional structure of feature maps in CNNs, the complex-valued multiplications would need to be accumulated over the input channels for each OFM element and this needs to be done for each output channel. Thus, the actual number of complex-valued multiply-accumulate operations (CMACs) in an SpCNN would need to take into account the number of input and output channels (hereafter referred to as the input depth and the output depth) in addition to the size of OFMs for each CONV layer. The CW of CONV layers of a spectral domain CNN in terms of CMAC (denoted by CW_{CNV-CM}) is given in (1),

$$CW_{CNV-CM} = \sum_{l=1}^{l_{CV}} D_{i(l)} M_l^2 D_{o(l)} \tag{1}$$

where, l is the index for the layer number, l_{CV} is the number of CONV layers, $D_{i(l)}$ is the input depth for layer l, $D_{o(l)}$ is the output depth for layer l and M_l is the OFM size of layer l.

One CMAC operation constitutes one complex-valued multiplication and one complex-valued accumulation. Each of these complex-valued product is computed from four real-valued products and two real-valued additions, while each complex-valued accumulation involves two real-valued additions. In total, a single CMAC operation constitutes four real-valued products and four real-valued additions, i.e., a total of eight real-valued arithmetic operations in total. Therefore, a single CMAC is equivalent to four real-valued MACs. The CW of CONV layers in terms of MAC (denoted by CW_{CNV-M}) is given in (2).

$$CW_{CNV-M} = 4\sum_{l=1}^{l_{CV}} D_{i(l)} M_l^2 D_{o(l)} \tag{2}$$

In CNNs, images and kernels in CNNs are typically stored in floating-point format [33]. Therefore, each of the eight real-valued arithmetic operations to realize a CMAC is a floating-point operation or FLOP in short. In other words, one CMAC constitutes eight FLOPs. Equation (3) describes this CW of CONV layers in terms of FLOPs (denoted by CW_{CNV-F}).

$$CW_{CNV-F} = 8\sum_{l=1}^{l_{CV}} D_{i(l)} M_l^2 D_{o(l)} \tag{3}$$

The above equations represent the CW of the CNN when performing inference on single images, or equivalently, when the batch size is set to one. When inference is done in batches, the CW needs to be multiplied by the batch sizes.

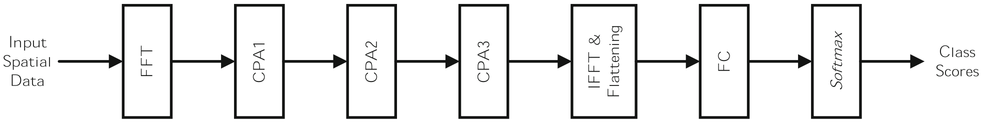


Fig. 3 Functional architecture of the baseline LeNet-5 SpCNN model

The CW of CONV layers in terms of FLOPs can be updated to include the influence of batch size, as shown below in (4),

$$CW_{CNV-F} = 8B \sum_{l=1}^{l_{CV}} D_{i(l)} M_l^2 D_{o(l)} \tag{4}$$

where, B is the number of images in a batch.

SpCNNs require at least one set of layers for domain transformation in the form of one FFT and one IFFT layer. These layers also execute CMAC operations [37, 52]. The FFT and IFFT operations have a complexity of O(Mlog$M$). Each FFT/IFFT operation requires 5Mlog$M$ real-valued FLOPs (0.5Mlog$M$ complex-valued multiplications and $M$log$M$ complex-valued additions), when provided with one-dimensional inputs [57]. However, input images or feature maps in CONV layers are typically provided as two-dimensional data [9, 10]. In that case, each FFT or IFFT operation on an image with height = M and width = M would require $5M^2\log_2 M^2$ FLOPs, which is equivalent to $10M^2\log_2 M$ FLOPs. As feature maps in CNNs also have a third dimension [9–11], the depth of feature maps would need to be multiplied with the above expression to compute the number of FLOPs for an FFT/IFFT layer.

The total CW of a spectral domain CNN in terms of FLOPs (executed by MAC operations), is attributable to the FFT layer at the beginning, the CONV layers, the IFFT layer at the end of the feature-extraction segment and the fully-connected layers in the classification segment. The overall CW of CNN in terms of FLOPs (denoted by CW_{CNN-F}), is given in (5),

$$\begin{aligned} CW_{CNN-F} = {} & 10B \sum_{l=1}^{2} D_{i(l)} M_l^2 log_2 M_l \\ & + 8B \sum_{l=1}^{l_{CV}} D_{i(l)} M_l^2 D_{o(l)} + B \sum_{l=1}^{l_{FC}} D_{i(l)} D_{o(l)} \end{aligned} \tag{5}$$

where, l_{FC} is the number of fully connected layers.

It can be shown analytically that the CW contribution of the CONV layers is the dominant component of the overall CW for a CNN. For example, fully-connected layers perform MAC operations on one-dimensional (1×1) IFMs and kernels in a flattened single-dimensional input stream and hence when compared to the CW of CONV layers, their CW is much less. The MAC operations executed by the single FFT layer and the single IFFT layer are also not significant when compared to the more computationally intensive CONV layers. When analyzing the CW for spectral domain implementations of LeNet-5 and AlexNet by [45] and [58], respectively, one can see that non-CONV layers (fully connected, FFT and IFFT layers) contribute only about 1% to 4% to the overall CW. The remaining 96% to 99% CW is due to the CONV layers. This is shown in Table 1 for single-image inference. Thus, computing the number of FLOPs required by MAC operations in CONV layers is often sufficient to estimate the CW of CNNs, as given in (6).

$$CW_{CNN-F} \approx 8B \sum_{l=1}^{l_{CV}} D_{i(l)} M_l^2 D_{o(l)} \tag{6}$$

In CNNs, OFMs of a particular layer become IFMs of the next layer. In other words, the output depth of the current layer becomes the input depth for the next layer. So, (6) can be rewritten in terms of the output depths of the current layer ($D_{o(l)}$) and the previous layer (($D_{o(l\text{-}1)}$) instead of the input depth ($D_{i(l)}$) and the output depth ($D_{o(l)}$) of the current layer. This is given in (7).

$$CW_{CNN-F} = 8B \sum_{l=1}^{l_{CV}} D_{o(l-1)} M_l^2 D_{o(l)} \tag{7}$$

In CNNs, OFMs of CONV layers go through down-sampling (pooling) in each feature-extraction block. After convolution is performed, pooling layers down-sample feature maps, and thus, each successive CONV layer processes and generates feature maps that are smaller than the previous layer [9–11]. In other words, each CONV layer, other than the first CONV layer, sees a scaled-down version of feature maps originally provided to the CNN. If the first CONV layer produces OFMs of size M_1xM_1, the size of OFMs of a CONV layer l can be expressed as $\alpha_l M_1$x$\alpha_l M_1$, where α_l represents the shrinkage factor for the CONV layer l. Therefore, α_l is essentially the ratio of M_l and M_1. In the case of the first CONV layer, α_l would take the value of 1. Hereafter, CONV layers would be referred to with the word CONV followed by a digit in the subscript representing the index of the layer (e.g., the first and second CONV layers would be referred to as CONV$_1$ and CONV$_2$ layers).

In contrast to the gradual decrease in OFM sizes, CONV layers in CNNs sees a gradual growth in OFM depth. Each CONV layer typically has more output depth than the preceding CONV layer. This happens because earlier CONV layers extract high-level features, and as feature maps go through more and more CONV layers, increased depth allows them to extract more complex features [9–11]. One can express the output depth of a CONV layer in terms

Table 1 CW of different types of layers in selected works in LeNet-5 and AlexNet SpCNNs

Architecture	CW subtotal and percentages of total CW		
	CONV layers	FC layers	FFT/IFFT layers
LeNet-5 in [45]	4,477,440 (99.05%)	5,000 (0.11%)	38,010 (0.84%)
AlexNet in [58]	615,471,360 (95.95%)	21,921,792 (3.42%)	4,033,873 (0.63%)

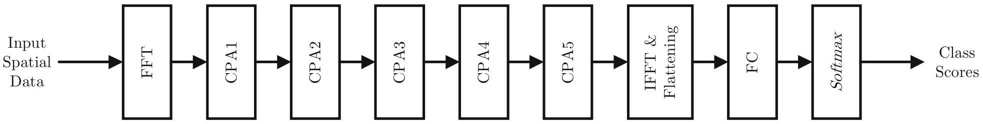


Fig. 4 Functional architecture of the baseline AlexNet SpCNN model

of the output depth of the first CONV layer ($D_{o(1)}$). When computing the CW of a CONV layer (denoted as CW_l) from CONV$_2$ layer and onward, the input depth for a CONV layer can be expressed as $\mu_{l\text{-}1} D_{o(1)}$, while the output depth can be expressed as $\mu_l D_{o(1)}$, where $\mu_{l\text{-}1}$ and μ_l represent the growth factors for the input depth and the output depth for CONV layer l. For computer vision problems, the input depth of the CONV$_1$ layer is a fixed constant. It is one for gray-scale input images and three for full-color (RGB) images. For the CONV$_1$ layer, the output depth is simply $D_{o(1)}$ ($\mu_1 = 1$), as there is no OFM from the previous layer. Thus, the CW expression of CONV$_1$ layer (denoted by CW_1) has a slightly different structure than that of the CW of CONV$_2$ layer and onward. Equation (8) gives the CW expression for the CONV layers (considering gray-scale inputs) in terms of size of OFMs (M_1) and depth of OFMs ($D_{o(1)}$) of the CONV$_1$ layer.

$$\begin{aligned}
CW_{CNN-F} &= CW_1 + \sum_{l=2}^{l_{CV}} CW_l \\
&= 8B\mu_1 D_{\mathrm{o}(1)}(\alpha_1 M_1)^2 \\
&\quad +8B\sum_{l=2}^{l_{CV}} \mu_{l-1} D_{\mathrm{o}(l)} \mu_l D_{\mathrm{o}(l)} (\alpha_l M_1)^2 \\
&= 8BD_{\mathrm{o}(1)} M_1^2 \\
&\quad +8B\sum_{l=2}^{l_{CV}} \mu_{l-1}\mu_l D_{\mathrm{o}(l)}^2 (\alpha_l M_1)^2 \qquad (8)
\end{aligned}$$

As discussed earlier, CONV layers see gradually decreasing OFM sizes, while the opposite takes place in terms of OFM depth. The growth of OFM depths and shrinkage of OFM sizes are illustrated graphically in Fig. 5.

4.2 Memory access cost (MemAC)

The speed of CNN inference, typically measured in terms of latency and throughput, are sensitive to the CW but does not solely depend on it. A major factor at play here is the number of memory access operations, which is known as the memory access cost or MemAC in short. Each CONV layer has to fetch input feature maps and kernel from memory, execute the convolution operation and then store the output feature maps in memory. Thus, total MemAC for a CONV layer, denoted by $MemAC_l$, would need to have $D_{i(l)}M_l^2$ memory accesses to read input feature maps, $D_{i(l)}D_{o(l)}M_l^2$ accesses for reading kernels, and $D_{o(l)}M_l^2$ accesses to store output feature maps. This is shown in (9),

$$\begin{aligned}
MemAC_l &= MemAC_{IFM} + MemAC_{Kern} + MemAC_{OFM} \\
&= D_{i(l)}M_l^2 + D_{i(l)}D_{o(l)}M_l^2 + D_{o(l)}M_l^2 \qquad (9)
\end{aligned}$$

where, $MemAC_{IFM}$ is the number of memory read operations for input feature maps, $MemAC_{Kern}$ is the number of memory read operations for kernels, and $MemAC_{OFM}$ is the number of memory write operations for output feature maps.

The MemAC for IFM, kernel and OFM in (9) can be rewritten in terms of size and depth of OFMs of CONV$_1$ layer as $\mu_{l\text{-}1}D_{o(1)}(\alpha_l M_1)^2$, $\mu_{l\text{-}1}D_{o(1)}\mu_l D_{o(1)}(\alpha_l M_1)^2$, and $\mu_l D_{o(1)}(\alpha_l M_1)^2$, respectively. This is shown in (10).

$$\begin{aligned}
MemAC_l &= \mu_{l-1}D_{o(1)}(\alpha_l M_1)^2 + \mu_{l-1}D_{o(1)}\mu_l D_{o(1)}(\alpha_l M_1)^2 \\
&\quad +\mu_l D_{o(1)}(\alpha_l M_1)^2 \\
&= \mu_{l-1}D_{o(1)}(\alpha_l M_1)^2 + \mu_{l-1}\mu_l D_{o(1)}^2(\alpha_l M_1)^2 \\
&\quad +\mu_l D_{o(1)}(\alpha_l M_1)^2 \qquad (10)
\end{aligned}$$

The total MemAC for the CONV layers, denoted below as $MemAC_{CNN}$, can be computed by summing the memory access operations for individual layers, as shown in (11). The number of kernels for a CNN inference (or training) is the same regardless of whether the inference is run on single images or in batches, and thus, the MemAC for reading kernels are not multiplied by the batch size here. The MemAC of the CONV$_1$ layer, denoted as $MemAC_1$, is taken out of the summation expression as the input depth of the CONV$_1$ layer is a constant unlike other CONV layers, and this input depth is not related to any previous CONV layer.

$$\begin{aligned}
MemAC_{CNN} &= MemAC_1 + \sum_{l=2}^{l_{CV}} MemAC_l \\
&= BM_1^2 + D_{o(1)}M_1^2 + BD_{o(1)}M_1^2 \\
&\quad +B\sum_{l=2}^{l_{CV}} \mu_{l-1}D_{o(1)}(\alpha_l M_1)^2 \\
&\quad +\sum_{l=2}^{l_{CV}} \mu_{l-1}\mu_l D_{o(1)}^2(\alpha_l M_1)^2 \\
&\quad +B\sum_{l=2}^{l_{CV}} \mu_l D_{o(1)}(\alpha_l M_1)^2 \qquad (11)
\end{aligned}$$

If the implementation platform running inference has a large enough cache, the entire set of kernels for all CONV layers can be stored on-chip, necessitating off-chip memory

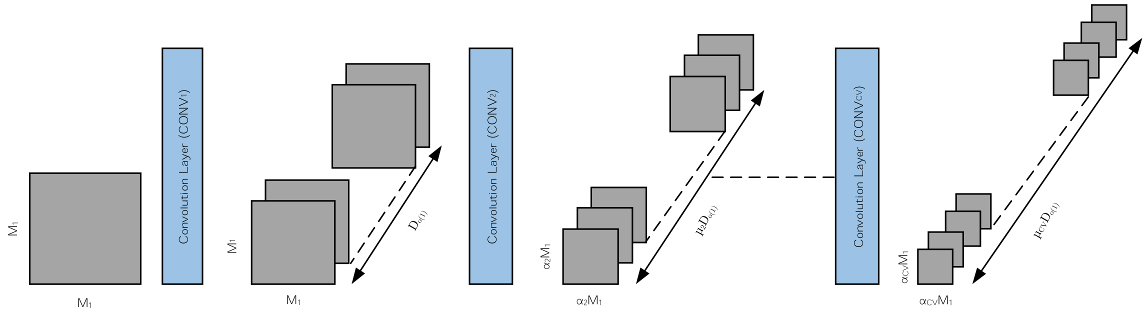

M₁
M₁
Convolution Layer (CONV₁)
M₁
M₁
$D_{o(1)}$
Convolution Layer (CONV₂)
α₂M₁
α₂M₁
$\mu_2 D_{o(1)}$
Convolution Layer (CONV_CV)
α_CV M₁
α_CV M₁
$\mu_{CV} D_{o(1)}$

Fig. 5 Gradual shrinkage of OFM sizes and gradual growth of OFM depths in CNN

access only for input and intermediate feature maps. In such a case, the MemAC would only involve loading and storing input and intermediate feature maps. This MemAC for feature maps, denoted by $MemAC_{CNN-FM}$, is given in (12).

$$\begin{aligned} MemAC_{CNN-FM} = {} & BM_1^2 + D_{o(1)}M_1^2 \\ & + B\sum_{l=2}^{l_{CV}} \mu_{l-1} D_{o(1)} (\alpha_l M_1)^2 \\ & + B\sum_{l=2}^{l_{CV}} \mu_l D_{o(1)} (\alpha_l M_1)^2 \end{aligned} \tag{12}$$

5 Proposed methodology for improving inference performance

As demonstrated in the previous section, the computational workload and memory access cost (CMC) in SpCNNs primarily depend on two parameters, namely, the size of OFMs and the depth of OFMs. This section discusses the methodology proposed in this work that can reduce these parameters to enhance inference performance under an accuracy constraint. The proposed methodology includes three strategies, denoted as Strategy 1, Strategy 2 and Strategy 3, which involve reduction of OFM sizes, OFM depths and both of these parameters, respectively. The three strategies are detailed in the three Sections 5.1, 5.2 and 5.3.

Before outlining the proposed methodology for performance improvement, some of the details of the baseline models, on which the methodology would be applied and evaluated, need to be specified. The basic functional architecture of baseline LeNet-5 and AlexNet SpCNNs were discussed earlier in Section 3.2. For the LeNet-5 baseline SpCNN model, the OFM size of different CONV layers were kept the same as recent SpCNN implementations of LeNet-5 [45, 52]. The AlexNet baseline SpCNN model shares the same OFM sizes of CONV layers as the SpCNN implementation of AlexNet proposed in [58]. Since images of datasets evaluated for this work have a size 28×28, they were resized to 55×55 (after converting input images to spectral domain) before feeding them to the CONV_1 layer. This helped ensure that OFM sizes of all CONV layers (including CONV_1) of the baseline AlexNet model have the same size as in the works of [11] and [58].

For the baseline SpCNN models for this work, the output depths of different CONV layers were chosen to be multiples of two. This allowed for a straightforward and proportionate reduction in the output depths of all CONV layers. For example, the output depths of all CONV layers can be reduced by two, four or eight times across all layers. As the baseline models have output depths that grow with each CONV layer, the reduction of output depths in a proportionate manner still ensures that later CONV layers would still have higher output depths as compared to preceding layers. The sizes and depths of OFMs of each CONV layer for the baseline models are presented in Tables 2 and 3. It also enumerates the values of the growth factor of OFM depths (μ_l) and the shrinkage factor of OFM sizes (α_l) for each CONV layer. As the baseline models are evaluated on datasets that classify objects into 10 categories, the $softmax$ (SM) layers of the models produce 10 classes, as shown in Tables 2 and 3.

5.1 Strategy 1 (S1): OFM size reduction

As discussed previously OFM sizes and depths are the primary contributors to CMC. One approach to reduce CMC is to scale the sizes of OFMs for all CONV layers. In SpCNNs, OFM sizes can be reduced to any arbitrary size using spectral pooling [39].

Table 2 OFM sizes and depths of baseline LeNet-5 SpCNN model

Layer	Size of OFMs	α_l	Depth of OFMs	μ_l
Input	28x28	–	1	–
CONV_1	28x28	1	16	1
CONV_2	12x12	0.42	64	4
CONV_3	4x4	0.14	256	16
FC	1x1	–	10	–
SM	1x1	–	10	–

Table 3 OFM sizes and depths of baseline AlexNet SpCNN model

Layer	Size of OFMs	α_l	Depth of OFMs	μ_l
Input	55x55	–	1	–
CONV$_1$	55x55	1	16	1
CONV$_2$	27x27	0.49	32	2
CONV$_3$	13x13	0.24	64	4
CONV$_4$	13x13	0.24	64	4
CONV$_5$	6x6	0.11	1024	64
FC	1x1	–	10	–
SM	1x1	–	10	–

For most CNN architectures, scaling OFM size of the CONV$_1$ layer would not reduce the CW significantly. The CW of the CONV$_1$ layer is $8D_{o(1)}M_1^2$ for gray-scale inputs and $24D_{o(1)}M_1^2$ for colored inputs (considering B (batch size) is 1). This expression is generic; it has no variable or tunable parameter such as μ_l or α_l. Thus, regardless of whatever OFM size or depth is chosen for any CONV layer, this expression for the CONV$_1$ layer remains true. The expression for a CONV layer other than the CONV$_1$ layer has a form of $\mu_{l\text{-}1}\mu_l D^2_{o(1)}\alpha_l^2 M_1^2$. This would typically be much larger than the expression for the CONV$_1$ layer (C$D_{o(1)}M_1^2$, where C is a constant). Hence, reducing OFM size of the CONV$_1$ layer would have no appreciable effect on reducing the overall CW or MemAC of the CONV layers. For example, for LeNet-5 SpCNN implementation of Rizvi et al. [45], the CW of the CONV$_1$ layer is $8D_{o(1)}M_1^2$. Considering the OFM sizes and depths of this implementation (listed in Table 4), the CW of the other two CONV layers is $13.33D^2_{o(1)}M_1^2$. This is shown in Table 5. It is clear that $13.33D^2_{o(1)}M_1^2 \gg 8D_{o(1)}M_1^2$. If values of M_1 and $D_{o(1)}$ are put down, the CW of the CONV$_1$ layer here turns out to be only 2.8% of the overall CW. For this model, the MemAC of the CONV$_1$ layer is 5.4% of the overall MemAC, as shown in Table 6. Likewise, for the baseline LeNet-5 SpCNN model used in this work, the CW and MemAC of the CONV$_1$ layer are about 2.9% of the overall CW and 5.7% of the overall MemAC.

In the case of the baseline AlexNet SpCNN model used in this work, the CW and MemAC of the CONV$_1$ layer are about 3.7% and 6.3% of the overall CW and MemAC, respectively. When AlexNet SpCNN implementation of Kala et al. [58] is considered, the CW and MemAC of the CONV$_1$ layer become 1.1% and 1.9% of the overall CW and MemAC, respectively. The CW and MemAC of CONV$_1$ and other CONV layers for all the SpCNN models discussed here are presented in Tables 5 and 6. Table 4 provides the OFM size and depth information for the models presented in [45] and [58]. The same information for the baseline models is provided in Tables 2 and 3. From Tables 5 and 6, it is evident that reducing the OFM size of the CONV$_1$ layer does not markedly reduce the CW and MemAC.

Another factor that should be considered when optimizing OFM sizes is that in CNNs, the last CONV layer provides input to the classification segment and typically has the smallest OFM size. For example, in LeNet-5 the OFM size of the last CONV layer is either 5×5 [10, 54] or 4×4 [45, 52]. OFMs generated from this last CONV layer are not downsized by any pooling layer. It was observed in this work that resizing OFMs of the last CONV layer to an even smaller size can adversely affect the recognition capabilities of the CNN. In some CNN architectures such as the original spatial domain AlexNet architecture, OFMs coming out from the last CONV layer are pooled to a smaller size. Here, the output of the last three CONV layers has a size of 13×13. After the last CONV layer, OFMs are pooled to a 6×6 size before feeding them to the fully connected layers. Thus, in the case of AlexNet, the OFM size of the last CONV layer can be kept as small as 6×6, the smallest feasible size. To ensure classification accuracy is not compromised severely, the OFM size of a CONV layer should be reduced only up to the smallest feasible size. Therefore, we suggest that OFM size be reduced for all the middle CONV layers, while keeping the first and the last CONV layers out of the OFM size reduction strategy. The CW and MemAC of CNNs can be expressed in a form that includes the scaling factor β_m for optimizing the size of OFMs, as given in (13) and (14).

$$\begin{aligned} CW_{CNN-F} = {} & 8BD_{o(1)}M_1^2 \\ & +8B\sum_{l=2}^{l_{CV-1}} \mu_{l-1}\mu_l D^2_{o(1)}(\frac{\alpha_l M_1}{\beta_m})^2 \\ & +B\mu_{CV-1}\mu_{CV}D^2_{o(1)}(\alpha_{CV}M_1)^2 \end{aligned} \tag{13}$$

$$\begin{aligned} MemAC_{CNN} = {} & BM_1^2 + D_{o(1)}M_1^2 + BD_{o(1)}M_1^2 \\ & +B\sum_{l=2}^{CV-1} \mu_{l-1}D_{o(1)}(\frac{\alpha_l M_1}{\beta_m})^2 \\ & +\sum_{l=2}^{CV-1} \mu_{l-1}\mu_l D^2_{o(1)}(\frac{\alpha_l M_1}{\beta_m})^2 \\ & +B\sum_{l=2}^{CV-1} \mu_l D_{o(1)}(\frac{\alpha_l M_1}{\beta_m})^2 \\ & +B\mu_{CV-1}D^2_{o(1)}(\alpha_{CV}M_1)^2 \\ & +\mu_{CV-1}\mu_{CV}D^2_{o(1)}(\alpha_{CV}M_1)^2 \\ & +B\mu_{CV}D^2_{o(1)}(\alpha_{CV}M_1)^2 \end{aligned} \tag{14}$$

The scaling factor (β_m) value that reduces OFM size to the smallest feasible value can be called $\beta_{m-\max}$. One can shrink the OFM sizes of all the middle CONV layers proportionally. However, not all layers can have OFM sizes

Table 4 OFM size and depth of CONV$_1$ layer, shrinkage factor (α_l) for OFM size, and growth factor (μ_l) for OFM depth for previous SpCNNs works in [45] and [58]

Architecture	Work	^1M_l	α_l	$D_{o(l)}$	μ_l
LeNet-5	Rizvi et al. [45]	28	{1, 0.42, 0.14}	20	{1, 2.5, 25}
AlexNet	Kala et al. [58]	55	{1, 0.49, 0.24, 0.24, 0.24}	96	{1, 2.66, 4, 4, 2.66}

[1] Represents both length and height of OFM size for CONV$_1$ layer, since they are equal

reduced at the same rate, as OFM size for no layer can shrink beyond the smallest feasible size. For example, the baseline AlexNet model that has five CONV layers has OFM sizes of {55, 27, 13, 13, 6}—values here represent both OFM length and height, since they are equal. As OFM sizes of CONV$_1$ and CONV$_5$ layers are not reduced for reasons discussed above, one can reduce OFM sizes of other layers two-fold resulting in OFM sizes of {55, 13, 6, 6, 6}, where respective β_m values are {1, 2.07, 2.16, 2.16, 1}. If OFM size needs to be reduced further, one can reduce the OFM size of the CONV$_2$ layer to 6 (minimum feasible size), resulting in a model with OFM sizes of {55, 6, 6, 6, 6} with respective β_m values of {1, 4.5, 2.16, 2.16, 1}. Any further reduction in OFM sizes of CONV$_2$ through CONV$_4$ layers would mean one of those layers would have an OFM size smaller than the smallest feasible size. In the case of LeNet-5, the third and the last CONV layer (CONV$_3$) produces OFMs with a size of 4 (i.e., length = 4, height = 4), which is already the smallest feasible size. The CONV$_2$ layer typically produces OFMs with a size of 12 (length = 12, height = 12), which can be reduced up to a size of 4 with a β_m value of 3. In the case of LeNet-5, one can only reduce the OFM size of just one CONV layer, that is the CONV$_2$ layer (as reducing OFM size of the CONV$_1$ layer has a negligible effect on the CW and MemAC). Table 7 shows the typical OFM sizes of middle CONV layers in the baseline LeNet-5 and AlexNet models and how much the sizes can be reduced for these layers.

In this strategy, OFM sizes can be continually reduced until the desired performance is achieved with a tolerable loss of accuracy and without OFM sizes falling below the minimum feasible size. When these goals are achieved, OFM sizes for the middle CONV layers can be finalized. The flow diagram for this strategy of reducing OFM sizes is illustrated in Fig. 6.

One should note that spectral pooling, through which OFM sizes are reduced, does so by discarding the high-frequency segment of the images or feature maps and retaining only their low-frequency content. It is already known that typical inputs to CNNs such as images and time-series data have a spectral bias with most information being retained in low frequencies [39, 59]. As OFM size reduction through spectral pooling preserves most of the information, it is not expected to downgrade accuracy significantly. Works that optimize OFM sizes such as Ayat et al. [52] and Liu et al. [54] have reported this aspect. The current work is different from the above-mentioned works as the former attempts to determine the OFM size of which CONV layer should be optimized and in what amount, rather than optimizing them arbitrarily.

5.2 Strategy 2 (S2): OFM depth reduction

Another method to reduce CMC is to optimize the depths of OFMs. As discussed before CNNs require gradually increasing OFM depths to extract deeper and deeper features with every CONV layer. To preserve this feature after reducing OFM depths of different CONV layers, the relative growth of OFM depths among different CONV layers should be kept the same. This ensures that the CNN model still has gradually growing OFM depths with every CONV layer. For example, if a CNN having three CONV layers with OFM depths of {16, 32, 64} goes through a 2× uniform depth reduction in all CONV layers (resulting in OFM depths of {8, 16, 32}), the same scaling ratio ensures that the output depth of a CONV layer is still twice that of the output depth of the previous CONV layer. We denote this scaling ratio as β_d.

When optimizing the OFM depth, one has to reduce it for every CONV layer, despite the fact that the CONV$_1$

Table 5 CW of CONV$_1$ layer as compared to other CONV layers in selected works in LeNet-5 and AlexNet SpCNNs

Architecture	Work	CW of CONV$_1$ layer	CW of other CONV layers	%CW of CONV$_1$ layer
LeNet-5	Rizvi et al. [45]	$8D_{o(1)}M_1^2$	$13.33D_{o(1)}^2M_1^2$	2.80%
	This work (baseline)	$8D_{o(1)}M_1^2$	$15.67D_{o(1)}^2M_1^2$	2.97%
AlexNet	Kala et al. [58]	$24D_{o(1)}M_1^2$	$22.28D_{o(1)}^2M_1^2$	1.13%
	This work (baseline)	$8D_{o(1)}M_1^2$	$39.68D_{o(1)}^2M_1^2$	3.71%

Table 6 MemAC of CONV$_1$ layer as compared to other CONV layers in selected works in LeNet-5 and AlexNet SpCNNs

Architecture	Work	MemAC of CONV$_1$ layer	MemAC of other CONV layers	%MemAC of CONV$_1$ layer
LeNet-5	Rizvi et al. [45]	$M_1^2 + 2D_{o(1)}M_1^2$	$1.16D_{o(1)}M_1^2 + 1.66D_{o(1)}^2M_1^2$	5.40%
	This work (baseline)	$M_1^2 + 2D_{o(1)}M_1^2$	$1.27D_{o(1)}M_1^2 + 1.95D_{o(1)}^2M_1^2$	5.72%
AlexNet	Kala et al. [58]	$3M_1^2 + 4D_{o(1)}M_1^2$	$2.10D_{o(1)}M_1^2 + 2.78D_{o(1)}^2M_1^2$	1.91%
	This work (baseline)	$M_1^2 + 2D_{o(1)}M_1^2$	$2.34D_{o(1)}M_1^2 + 4.96D_{o(1)}^2M_1^2$	6.28%

layer has a significantly smaller CW and MemAC than other CONV layers. From the CONV$_2$ layer onward the CW and MemAC have a form of $\mu_{l-1} D_{o(1)} \mu_l D_{o(1)} \alpha_l^2 M_1^2$. This shows that the CW and MemAC depend on the output depths of both the current and the previous CONV layer. If both of these terms—$\mu_{l-1} D_{o(1)}$ and $\mu_l D_{o(1)}$—need to be reduced for the CONV$_2$ layer, it automatically necessitates reducing the output depth of the CONV$_1$ layer. Therefore, reduction of OFM depths requires the reduction to be performed on all CONV layers, including the CONV$_1$ layer. The CW and MemAC of CNNs can be expressed in a form that includes the scaling factor β_d for optimizing OFM depths, as given in (15) and (16).

$$\begin{aligned} CW_{CNN-F} &= 8B\frac{D_{o(1)}}{\beta_d}M_1^2 \\ &\quad + 8B\sum_{l=2}^{CV}\frac{\mu_{l-1}D_{o(1)}}{\beta_d}\frac{\mu_l D_{o(1)}}{\beta_d}(\alpha_l M_1)^2 \qquad (15) \end{aligned}$$

$$\begin{aligned} MemAC_{CNN} &= BM_1^2 + \frac{D_{o(1)}}{\beta_d}M_1^2 \\ &\quad + B\frac{D_{o(1)}}{\beta_d}M_1^2 + B\sum_{l=2}^{CV}\frac{\mu_{l-1}D_{o(1)}}{\beta_d}(\alpha_l M_1)^2 \\ &\quad + \sum_{l=2}^{CV}\frac{\mu_{l-1}D_{o(1)}}{\beta_d}\frac{\mu_l D_{o(1)}}{\beta_d}(\alpha_l M_1)^2 \\ &\quad + B\sum_{l=2}^{CV}\frac{\mu_l D_{o(1)}}{\beta_d}(\alpha_l M_1)^2 \qquad (16) \end{aligned}$$

When analyzing the CW and MemAC of different CONV layers using the CW and MemAC expressions developed in Section 4, one can measure the relative contribution of OFM size and depth to these costs. For example, in the case of the LeNet-5 SpCNN model by Rizvi et al. [45], OFM size dominates over OFM depth in the CONV$_1$ layer, in a ratio of 1 to 0.03 (i.e., 33 to 1), as can be seen in Table 8. For other layers, which are computationally and memory-wise more expensive, OFM depth dominates the CW over OFM size. In the case of the CONV$_2$ layer, OFM depth contributes seven times more as compared to OFM size. In the case of the CONV$_3$ layer, the contribution of OFM depth outweighs OFM size by 1562 to 1. This increasing influence of OFM depths is also evident in the MemAC. Here, OFM size dominates the MemAC for the CONV$_1$ layer, while OFM depths dominates it in CONV$_2$ and CONV$_3$ layers. This same pattern is seen in the case of the LeNet-5 baseline model for this work as well. Here also OFM size dominates the CW and MemAC in the CONV$_1$ layer, by a ratio of 1:0.02 (i.e., 50 to 1) and 1:0.04 (i.e., 25 to 1), respectively. In the CONV$_2$ layer, OFM depth contributions in the CW and MemAC are seven and eight times more than OFM size, respectively. In the CONV$_3$ layer, this dominance is more than 1000 times. In Tables 8 and 9, the ratio of OFM size and depth is denoted as RSD in short.

With Table 9, one can again see the dominant contribution of OFM depth in the CW and MemAC for AlexNet SpCNNs. In the case of the AlexNet SpCNN implementation of Kala et al. [58], OFM size contributes more than OFM depth only in the CONV$_1$ layer. For the other CONV

Table 7 OFM sizes of middle CONV layers in the baseline SpCNN models, their smallest feasible sizes, and corresponding β_m values

CNN architecture	OFM size	CONV$_2$	CONV$_3$	CONV$_4$
LeNet-5	OFM size in the baseline model	12	-	-
	The smallest feasible OFM size	4	-	-
	β_m to obtain the smallest feasible OFM size ($\beta_{m-\max}$)	3	-	-
AlexNet	OFM size in the baseline model	27	13	13
	The smallest feasible OFM size	6	6	6
	β_m to obtain the smallest feasible OFM size ($\beta_{m-\max}$)	4.5	2.16	2.16

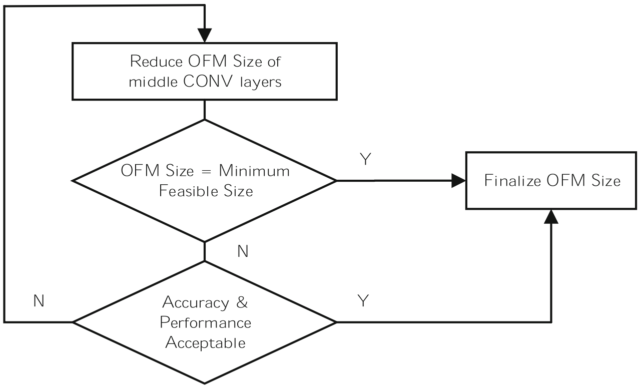

Reduce OFM Size of middle CONV layers
OFM Size = Minimum Feasible Size
Y
Finalize OFM Size
N
N
Accuracy & Performance Acceptable
Y

Fig. 6 Flow diagram for Strategy 1 for reducing OFM sizes of middle CONV layers

layers, OFM depths outweighs the OFM sizes in terms of contributing to the CW and MemAC, by 34 times to over 870 times. In the case of the baseline AlexNet model for this work, it is observed that OFM size dominates the CW and MemAC in two CONV layers, namely, $CONV_1$ and $CONV_2$. For other three CONV layers ($CONV_3$ to $CONV_5$), OFM depths dominates the CW and MemAC in ratios of 1:12 to 1:1851. This analysis, conducted for two separate implementations, each for LeNet-5 and AlexNet, having different set of OFM depths, indicates that OFM depth does dominate the CW and MemAC in CONV layers that are computationally and memory-wise more expensive. Unless the gradual increase of OFM depths and gradual reduction of OFM sizes, which are the norm for CNNs, are unusually small, this observation would remain true.

From the above discussion, it is clear that reducing OFM depths is likely to reduce both the CW and MemAC significantly. As compared to Strategy 1, Strategy 2 is expected to produce larger reductions in the CW and MemAC cost, as OFM depth dominates more than OFM size in computationally and memory-intensive CONV layers. The only possible drawback here is that as CONV layers would have smaller OFM depths, this would mean a lesser number of features would be extracted. This is likely to have some impact on accuracy. This impact is likely to be more in Strategy 2 than with Strategy 1.

Table 8 The ratio of OFM size and OFM depth (RSD) in CW and MemAC of different CONV layers for selected works in LeNet-5 SpCNNs

Work	RSD	$CONV_1$	$CONV_2$	$CONV_3$
Rizvi et al. [45]	RSD in CW	1:0.03	1:7	1:1562
	RSD in MemAC	1:0.05	1:7	1:1597
This work	RSD in CW	1:0.02	1:7	1:1024
(Baseline model)	RSD in MemAC	1:0.04	1:8	1:1044

As discussed earlier, with Strategy 2, one should reduce OFM depths of different CONV layers at the same rate, while ensuring that each CONV layer has a larger OFM depth than the preceding CONV layer Therefore, the $CONV_1$ layer would have the smallest OFM depth. The minimum feasible OFM depth for the $CONV_1$ layer can be made equal to its input depth. Therefore, CNNs that take gray-scale images as inputs would have a minimum feasible OFM depth of 1, while CNNs that take RGB color images as inputs would have a minimum feasible OFM depth of 3. The flow diagram for Strategy 2 is illustrated in Fig. 7. Here, OFM depths can be continually reduced at a uniform rate until performance and accuracy goals are achieved and all the while ensuring OFM depth of the $CONV_1$ layer is above or equal to the minimum feasible depth. After these goals are reached, OFM depths can be finalized for the CNN model.

5.3 Strategy 3 (S3): Reducing both OFM size and depth

As discussed above, optimizing OFM size is limited by which layer can be optimized and by how much. However, because the information in spectral images tend to be heavily concentrated in low-frequency sub-matrix of input image [39], OFM size reduction is unlikely to significantly impact the accuracy. On the other hand, optimizing OFM depth can be done in all CONV layers but might incur a higher cost in accuracy as smaller output depth means a lesser number of features extracted. Considering the characteristics of these two optimization approaches, the optimal approach is to reduce OFM size first. If the performance (throughput or energy efficiency) target is achieved with an acceptable accuracy loss, one can stop the optimization process and finalize the OFM sizes. If this is not reached even after reducing the OFM sizes to the smallest feasible size, one can then start optimizing the OFM depth. OFM depths can be continuously reduced until the performance goal is achieved within the accuracy constraint. This optimization flow of Strategy 3 is depicted in Fig. 8.

It is worth noting that each strategy discussed above has its unique benefits. For instance, Strategy 1 should be the preferred choice to optimize SpCNN models for applications with stringent accuracy requirements. This is because, despite having a lower reduction in CMC than other strategies, Strategy 1 does not reduce OFM depths and therefore has enough features present in IFMs and OFMs to ensure any loss in accuracy is minimal, if at all. On the other hand, for applications, such as those intended for embedded or mobile platforms, that prioritize fast (e.g., lower CW) and energy-efficient (e.g., lower MemAC) inference at the expense of a loss of a few percentages in accuracy, Strategy

Table 9 The ratio of OFM size and OFM depth (RSD) in CW and MemAC of different CONV layers for selected works in AlexNet SpCNNs

Work	RSD	CONV_1	CONV_2	CONV_3	CONV_4	CONV_5
Kala et al. [58]	RSD in CW	1:0.03	1:34	1:582	1:872	1:582
	RSD in MemAC	1:0.04	1:34	1:585	1:877	1:585
This work	RSD in CW	1:0.01	1:0.7	1:12	1:24	1:1820
(Baseline model)	RSD in MemAC	1:0.01	1:0.8	1:13	1:25	1:1851

2 and Strategy 3 would be among the more effective options. Since Strategy 3 optimizes CMC more than Strategy 2, the former typically should be chosen ahead of the latter for recognition tasks. However, specific applications that involve object detection or image segmentation tasks may require precise spatial locations of features present in IFMs. As a result, reducing the size of OFMs (which are IFMs of succeeding CONV layer) may have some impact on accuracy if OFM sizes are reduced significantly. In such scenarios, Strategy 2, which does not reduce OFM sizes, is expected to be more effective than Strategy 3. The impact of OFM sizes and depths on accuracy and performance (throughput, energy efficiency, memory usage) in all three strategies are presented in Section 6.2.

5.4 Performance estimation using the proposed strategies

With the expressions for the CW and MemAC developed earlier, one can compute how much they would be reduced with changes in OFM sizes and depths. Since inference performance (e.g., throughput and energy efficiency) depends on both of these factors, one way to combine their effect is to take a ratio of the CW over the MemAC. This is known as the computation to communication ratio (CTC) [60]. A variation of CTC is known as MACs over CIO (MoC), where MAC and CIO are short for CW in terms of MAC operations and convolutional input/output, respectively [33]. MoC is essentially a ratio of the CW over the MemAC for accessing input and output feature maps—the latter is denoted in this work as $MemAC_{CNN\text{-}FM}$. MoC values are computed for different OFM sizes and depths to estimate how the speed of inference might improve in our methodology. OFM size or depth reduction resulting in a higher value of MoC would indicate that the MemAC is reduced at a higher rate than the CW. A lower value of MoC would indicate the opposite—the CW getting reduced at a higher rate than the MemAC.

One should note that CMC and therefore MoC is only a preliminary estimate to ascertain gain in inference performance. Implementation platform plays a big role in inference performance and hence direct metric (e.g., latency, throughput) is generally more accurate than indirect metrics such as the CW [33, 35].

As discussed earlier in Section 5.1, OFM sizes cannot be arbitrarily reduced across all CONV layers. In the case of LeNet-5, the OFM size of only the CONV_2 layer can be reduced. Baseline LeNet-5 model has OFM sizes of {28, 12, 4}. OFM sizes that are explored here are {28, 9, 4}, {28, 6, 4}, and {28, 4, 4}. The corresponding β_m values are {1, 1.33, 1}, {1, 2, 1}, and {1, 3, 1}. For AlexNet, OFM sizes can be reduced for CONV_2, CONV_3 and CONV_4 layers. Since later CONV layers have smaller OFM sizes, to begin with, the CONV_2 layer can have more variation in OFM sizes as compared to CONV_3 and CONV_4 layers. The baseline AlexNet model has OFM sizes that are {55, 27, 13, 13, 6}.

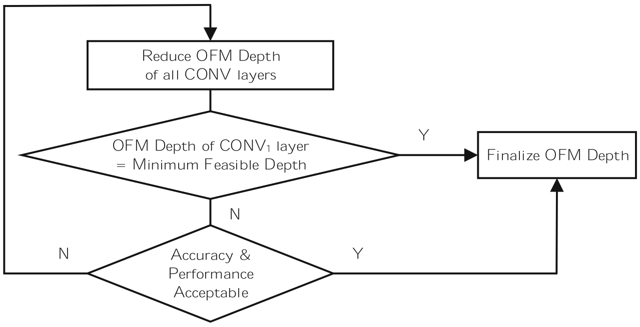

Reduce OFM Depth of all CONV layers
OFM Depth of CONV₁ layer = Minimum Feasible Depth
Y
Finalize OFM Depth
N
N
Accuracy & Performance Acceptable
Y

Fig. 7 Flow diagram for Strategy 2 for reducing OFM depths for all CONV layers

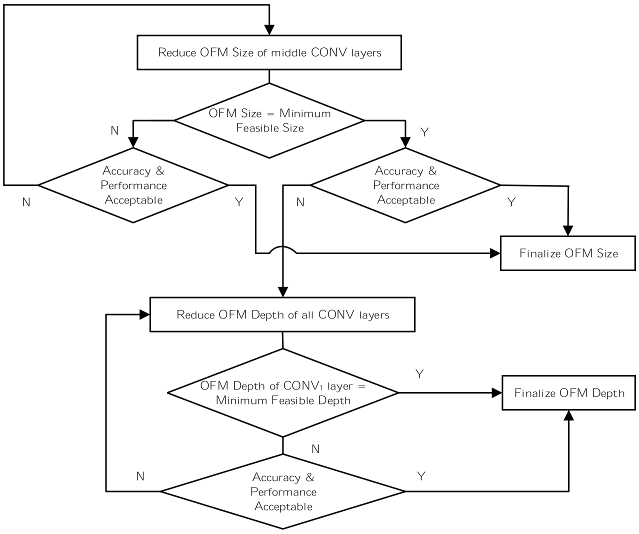

Reduce OFM Size of middle CONV layers
OFM Size = Minimum Feasible Size
N
Y
Accuracy & Performance Acceptable
N
Y
Accuracy & Performance Acceptable
N
Y
Finalize OFM Size
Reduce OFM Depth of all CONV layers
OFM Depth of CONV₁ layer = Minimum Feasible Depth
Y
Finalize OFM Depth
N
N
Accuracy & Performance Acceptable
Y

Fig. 8 Flow diagram for Strategy 3 that combines reduction in OFM sizes and depths

Here, OFM sizes that are explored are {55, 13, 6, 6, 6} and {55, 6, 6, 6, 6}. The corresponding β_m values are {1, 2.07, 2.16, 2.16, 1} and {1, 4.5, 2.16, 2.16, 1}. So, due to reasons discussed here, inference performance for both LeNet-5 and AlexNet can be more effectively elucidated when they are reported against the OFM size of the CONV$_2$ layer, denoted as M_2.

In contrast to OFM sizes, OFM depths can be reduced for all the CONV layers, as discussed in Section 5.2. However, since CONV layers in our methodology go through proportional reduction, the inference performance can be analyzed against the OFM depth of any CONV layer. However, minimum feasible OFM depth is defined for the CONV$_1$ layer. Therefore, we have chosen to report inference performance concerning OFM depth of the CONV$_1$ layer, denoted as $D_{o(1)}$. For both LeNet-5 and AlexNet baseline models, $D_{o(1)}$ is 16. Since our methodology is evaluated on datasets that contain only gray-scale images, the minimum feasible depth is set to 1. Thus, for the CW, MemAC and MoC estimation, $D_{o(1)}$ is reduced up to one. For LeNet-5, the depths that are explored are {16, 64, 256}, {8, 32, 128}, {4, 16, 64}, {2, 8, 32}, and {1, 4, 16}. For AlexNet, the depths that are considered are {16, 32, 64, 64, 1024}, {8, 16, 32, 32, 512}, {4, 8, 16, 16, 256}, and {2, 4, 8, 8, 128}.

When the CW, MemAC and MoC are estimated, the relative performance improvements (relative reduction or gain) for the optimized models are expressed as a multiple of the quantity measured for the baseline model. Thus, if an optimized model has 400,000 parameters as compared to 200,000 parameters for the baseline model, the optimized model has a relative parameter count improvement of two-fold. For representing the relative improvements concisely, the word “times” or “fold” is represented by “×”. Thus, two-fold or two times improvement can be expressed as a 2× improvement.

When Strategy 1 (OFM size reduction) is applied for LeNet-5, it is observed that MoC stays almost the same. The number of parameters can shrink up to 1.45 times. With Strategy 2 (OFM depth reduction) and Strategy 3 (first reducing OFM sizes and then OFM depths), MoC can be reduced up to 15 times as compared to baseline, as shown in Table 10. However, the maximum possible reduction in the number of parameters is higher in Strategy 3 (225×) than Strategy 2 (177×). In the case of AlexNet, Strategy 1 sees MoC stays roughly the same while the number of parameters decreases to a maximum of 1.44 times. With Strategy 2 and Strategy 3, MoC can be reduced up to 17 and 16 times, respectively as compared to the baseline. Similar to LeNet-5, the maximum possible reduction in the number of parameters are higher in Strategy 3 (290×) than Strategy 2 (215×), as can be seen in Table 11. Since the CW involves the product of OFM sizes and OFM depths (depicted in (8)), while the $MemAC_{CNN\text{-}FM}$ contains their contribution in an additive manner (depicted in (12)), the $MemAC_{CNN\text{-}FM}$ would decrease slowly than the CW when OFM depths are progressively decreased. That is why when OFM depths are reduced with or without OFM size reduction, the CW would see a higher rate of reduction than the $MemAC_{CNN\text{-}FM}$ and hence a progressively lower MoC.

6 Results

6.1 Experimental environment, setup, and constraints

Training and inference for CNN models under evaluation were performed in MATLAB computing platform (v. 2020b) from MathWorks using MatConvNet (v. 1.0 beta24), an open-source deep learning library developed by Vedaldi et al. [61] at Visual Geometry Group of the University of Oxford. The host workstation, where training and inference were conducted, is powered by a quad-core 4790K Core-i7 CPU from Intel with 16 GB DDR3 RAM and operating at 4.0 GHz and a GeForce GTX1070 GPU from NVIDIA with 8 GB GDDR5 VRAM, 1920 CUDA Cores, and operating at 1683 MHz.

The proposed methodology was evaluated on LeNet-5 and a modified AlexNet with MNIST [62] and Fashion MNIST [63] datasets. MNIST is a dataset of handwritten digits from 0 to 9 with 70,000 images in total and in 10 classes. A total of 60,000 images are for training and 10,000 are for the test. The fashion MNIST dataset is a dataset of fashion articles such as T-shirts and shoes and with the same number of training and testing images and the number of classes as MNIST. Both datasets take 28×28 gray-scale images as input.

For training, weights were initialized with normally distributed random numbers. A learning rate of 0.0005 was used. All the variants of LeNet-5 and AlexNet models were trained for 50 epochs. During testing, a batch size of 1 and 64 was chosen for single-image and batch inference, respectively.

A constraint on the maximum loss of test accuracy is imposed when the proposed methodology is applied to improve inference performance. The constraint is a 5% loss in test accuracy as compared to the baseline model. Primarily five metrics were used for measuring inference performance, which are test accuracy, throughput, defined as the number of classifications performed per second (cl/s), energy efficiency, defined as the number of classifications per energy consumed (in units of cl/J) [60], memory usage (in MB units), and power consumption (in units of watt (W)). Energy efficiency can be calculated as classification rate over power consumption (in units of cl/J or (cl/s)/W).

Table 10 CW, MemAC, MoC, and parameter estimation for LeNet-5 SpCNN under proposed strategies

				MoC		Number of parameters	
OFM size $M_2(\beta_m)$	OFM depth $D_{o(1)}(\beta_d)$	CW	MemAC	Value	Relative reduction	Value	Relative reduction
S1: Reducing OFM size							
12 (1.0)	16 (1)	3377152	29968	112.69	1.00x	422144	1.00x
9 (1.33)	16 (1)	2861056	24928	114.77	0.98x	357632	1.18x
6 (2.0)	16 (1)	2492416	21328	116.86	0.96x	311552	1.35x
4 (3.0)	16 (1)	2328576	19728	118.03	0.95x	291072	1.45x
S2: Reducing OFM depth							
12 (1.0)	8 (2)	869376	15376	56.54	1.99x	108672	3.88x
12 (1.0)	4 (4)	229888	8080	28.45	3.96x	28736	14.69x
12 (1.0)	2 (8)	63744	4432	14.38	7.84x	7968	52.98x
12 (1.0)	1 (16)	19072	2608	7.31	15.41x	2384	177.07x
S3: Reducing OFM size first and then OFM depth							
4 (3.0)	8 (2)	607232	10256	59.21	1.90x	75904	5.56x
4 (3.0)	4 (4)	164352	5520	29.77	3.78x	20544	20.55x
4 (3.0)	2 (8)	47360	3152	15.03	7.50x	5920	71.31x
4 (3.0)	1 (16)	14976	1968	7.61	14.81x	1872	225.50x

When the performance of the baseline model is compared with optimized models for a specific architecture (LeNet-5 or AlexNet), improvement in inference performance (minimization of memory usage and power consumption and gain in throughput and energy efficiency) averaged over MNIST and Fashion MNIST datasets are documented against loss in test accuracy averaged across these two datasets. The performance metrics of throughput, energy efficiency, power consumption, and memory usage are measured for both single-image and batch inference.

6.2 Experimental results

After training was completed for LeNet-5 and AlexNet models with various OFM sizes and depths with the proposed methodology, the inference was computed for

Table 11 CW, MemAC, MoC, and parameter estimation for AlexNet SpCNN under proposed strategies

				MoC		Number of parameters	
OFM size $M_2(\beta_m)$	OFM depth $D_{o(1)}(\beta_d)$	CW	MemAC	Value	Relative reduction	Value	Relative reduction
S1: Reducing OFM size							
27 (1.0)	16 (1)	30554240	163441	186.94	1.00x	3819280	1.00x
13 (2.07)	16 (1)	21723264	106769	203.46	0.92x	2715408	1.41x
6 (4.5)	16 (1)	21178496	100385	210.97	0.89x	2647312	1.44x
S2: Reducing OFM depth							
27 (1.0)	8 (2)	7735360	83233	92.94	2.01x	966920	3.95x
27 (1.0)	4 (4)	1982240	43129	45.96	4.07x	247780	15.41x
27 (1.0)	2 (8)	519760	23077	22.52	8.30x	64970	58.79x
27 (1.0)	1 (16)	142040	13051	10.88	17.18x	17755	215.11x
S3: Reducing OFM size first and then OFM depth							
6 (4.5)	8 (2)	5391424	51705	104.27	1.79x	673928	5.67x
6 (4.5)	4 (4)	1396256	27365	51.02	3.66x	174532	21.88x
6 (4.5)	2 (8)	373264	15195	24.56	7.61x	46658	81.86x
6 (4.5)	1 (16)	105416	9110	11.57	16.15x	13177	289.84x

these models. The LeNet-5 baseline model attains 97.32% and 88.54% test accuracies for MNIST and Fashion MNIST datasets, respectively. With this baseline model and using batch inference, a 3573.19 cl/s throughput and energy efficiency of 37.73 cl/J are attained with 94.7 W power consumption and 615 MB memory usage. When Strategy 1 (optimizing OFM size) is applied, the performance gain is initially minor but becomes significant when the OFM size of the middle CONV layer gets to the smallest feasible size (3× smaller than the size in the baseline model). As OFM size is reduced, no loss in accuracy is observed. When OFM size of the middle CONV layer is at the smallest feasible size, where M_2 is 4, 1.5× gain in throughput is attained. At this size, there is an only slight reduction in memory usage (1.3×). While there is no noticeable improvement in power consumption, there is a 1.7× gain in energy efficiency. Table 12 shows test accuracies and values of other inference performance metrics as OFM size and/or depth are minimized. From Table 12 onward, accuracy, throughput, power consumption, energy efficiency, and memory usage are abbreviated as acc., thru., power cons., energy eff., and mem. usage, respectively.

For Strategy 2 (optimizing OFM depth), inference performance sees gain at every depth reduction step (every reduction step reduces OFM depths by 2×). The improvements, however, start to saturate after OFM depths are reduced 4×. With a 4× reduction in OFM depths, there is a 3× and 6.7× gain in throughput and energy efficiency, respectively. Power consumption and memory usage were reduced by 2.2× and 1.9×, respectively. All these performance gains are achieved at the cost of a 2.3% loss in accuracy, which is well within our accuracy loss constraint. When OFM depths are reduced by the maximum amount (16×), further performance improvement can be attained, as shown in Table 12. However, these improvements come at a price of accuracy loss of 10%, exceeding our acceptable accuracy loss by 2×.

In Strategy 3, OFM depths are reduced after obtaining the optimal model from Strategy 2. For LeNet-5, the optimal model from Strategy 2 has OFM size reduced to the smallest feasible size. While maintaining this OFM size, OFM depths are progressively reduced. Here, the accuracy constraint is still satisfied when OFM depths are reduced up to 4×. At this set of OFM depths, the improvement in throughput and energy efficiency stands at 3.6× and 8.6×. Furthermore, a 2.4× reduction in power consumption and 1.9× reduction in memory usage are obtained. The loss in accuracy stands at 2.9%, which is within the imposed accuracy constraint. These performance gains with Strategy 3 are higher than what was achieved in Strategy 2. Therefore, the model with $M_2 = 4$ and $D_{o(1)} = 4$ is regarded as the optimal LeNet-5 model. With a 16× reduction in OFM depths, even better inference performance can be attained, but the accuracy loss (13%) is well beyond the accuracy loss constraint.

The behavior of inference performance and accuracy loss with changes in OFM sizes and depths is easy to see when presented graphically. Figure 9 illustrates inference performance improvements in terms of accuracy, energy efficiency, throughput, and memory usage with the reduction in OFM sizes and depths. Subfigures 9a and 9b display loss in test accuracy and gain in energy efficiency, respectively, with changes in OFM sizes and depths. Subfigures 9c and 9d show gain in throughput and reduction in memory usage, respectively, with changes in OFM sizes and depths. Here, the set of entries from 1 to 4 represents Strategy 2, while entries from 5 to 8 and then 9 to 12 represent strategies 1 and 3, respectively. For ease of viewing, negative accuracy loss is represented in the subfigure 9a as zero losses.

In the case of AlexNet, the baseline AlexNet model achieves 95.96% and 86.95% test accuracies for MNIST and Fashion MNIST datasets, respectively. In batch inference, this model has a throughput of 483.50 cl/s and energy efficiency of 3.09 cl/J, as shown in Table 13. The memory and power consumption costs are 2434 MB and 156.4 W, respectively. With Strategy 1, a 1.5× higher throughput and a 1.7× greater energy efficiency are obtained as compared to the baseline model when OFM sizes of the middle CONV layer are at the smallest feasible OFM sizes. When OFM sizes are reduced under this strategy, no loss in accuracy is observed, but the improvements in power consumption and memory usage are minor.

With Strategy 2, significant performance gain is obtained while trading off an acceptable loss in accuracy. When OFM depths are reduced four times from the baseline value, a 10.6× gain in throughput and a 17.7× improvement in energy efficiency are attained at the cost of a 4.15% loss in accuracy. In addition, power consumption and memory usage are reduced by 1.7× and 4.6×, respectively. With an 8× reduction in OFM depths, even better performance can be attained but at the cost of a 14.53% loss in accuracy. A 16× reduction in OFM depths, which would make the OFM depth of the CONV_1 layer equal to the minimum feasible depth, has not been considered for AlexNet. This is because an 8× reduction in OFM depths already exceeded accuracy loss constraint by about three times.

It is observed that the performance improvement is higher in Strategy 3 than Strategy 2 under the same accuracy constraint. After reducing OFM sizes of the middle CONV layers to the smallest feasible size, a 4× reduction in OFM depths produces 11.6× gain in throughput along with a 25.2× gain in energy efficiency at the cost of about 4.4% loss in accuracy. There is also a reduction in power consumption by 2.2× and a 5.6× reduction in memory usage. The inference performance can be improved further

Table 12 Inference performance for LeNet-5 SpCNN

OFM size M_2 (β_m)	OFM depth $D_{o(1)}$ (β_d)	Test acc.$^+$ (MNIST)	Test acc.$^+$ (Fashion MNIST)$^+$	Mean acc. loss^{2*}	Thru.$^+$ (cl/s)	Power cons.* (W)	Energy eff.$^+$ (cl/J)	Mem. usage* (MB)
S1: Reducing OFM size								
12 (1)	16 (1)	97.32%	88.54%	0.00%	3573.19	94.70	37.73	615.10
9 (1.33)	16 (1)	97.07%	88.87%	– 0.04%	3884.03	94.20	41.23	567.80
6 (2)	16 (1)	97.00%	87.73%	0.56%	4220.11	90.60	46.58	523.90
4 (3)	16 (1)	97.06%	89.49%	– 0.34%	5510.48	86.70	63.56	479.50
S2: Reducing OFM depth								
12 (1)	8 (2)	96.57%	86.96%	1.17%	8262.24	74.20	111.35	398.10
12 (1)	4 (4)	94.93%	86.38%	2.28%	10752.69	42.80	251.23	329.10
12 (1)	2 (8)	91.73%	82.82%	5.66%	11201.21	38.80	288.69	314.60
12 (1)	1 (16)	85.83%	79.77%	10.13%	11892.56	38.20	311.32	306.40
S3: Reducing OFM size first and then OFM depth								
4 (3)	8 (2)	96.23%	87.91%	0.86%	9140.00	49.00	186.53	328.70
4 (3)	4 (4)	94.49%	85.59%	2.89%	12855.88	39.70	323.83	319.90
4 (3)	2 (8)	90.62%	81.97%	6.64%	14170.65	37.70	375.88	307.90
4 (3)	1 (16)	81.86%	77.11%	13.45%	15450.95	37.60	410.93	303.90

[2] Negative and positive values in accuracy loss indicate an increase and decrease in accuracy, repectively.

"+" indicates a higher value of this performance metric is better. "*" indicates a lower value of this performance metric is better

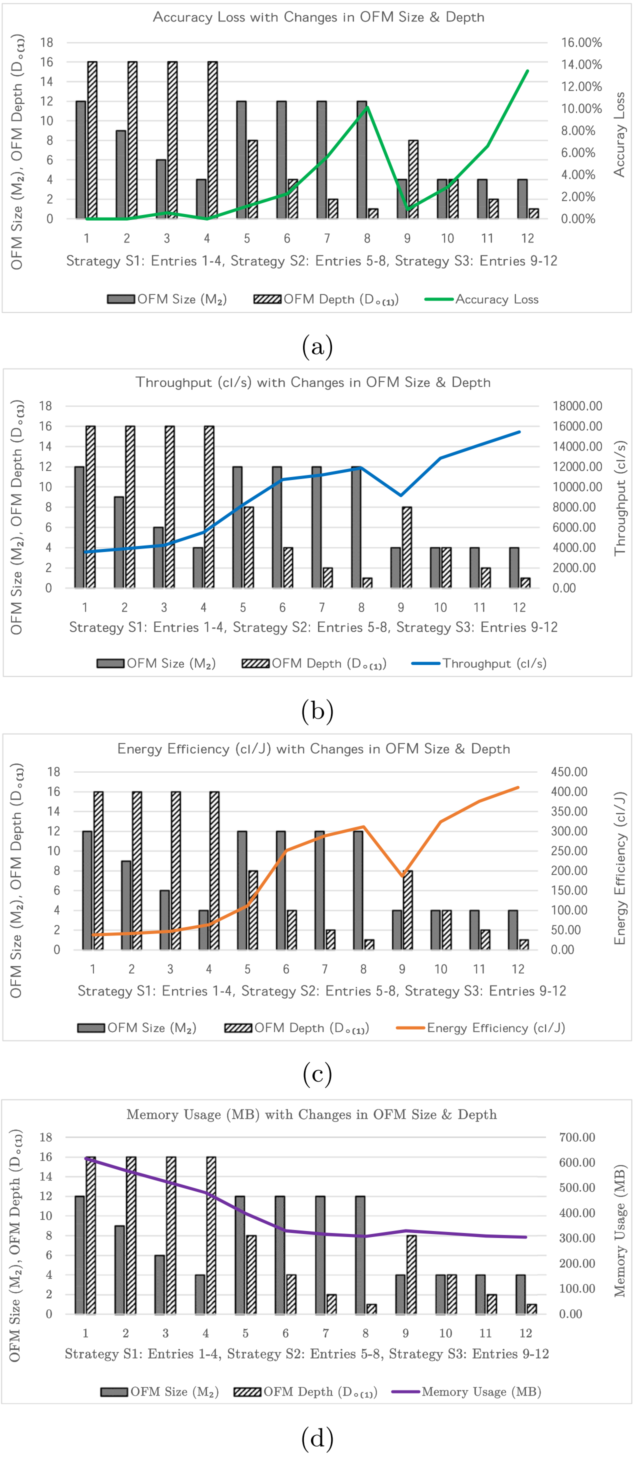


Fig. 9 Inference performance for LeNet-5 SpCNN in terms of (a) accuracy loss vs OFM size and depth, (b) energy efficiency vs OFM size and depth, (c) throughput vs OFM size and depth, (d) memory usage vs OFM size and depth

by reducing OFM depths by 8× but with an unacceptable cost of 11.39% loss in accuracy, as shown in Table 13. Therefore, the model with $M_2 = 6$ and $D_{o(1)} = 4$ is regarded as the optimal AlexNet model. A 16× reduction in OFM depth has not been considered here as the accuracy loss at an 8× reduction already exceeded the tolerance for accuracy loss by more than two times.

Figure 10 illustrates the improvements that are attained in throughput, memory usage and energy efficiency as well as the loss in accuracy with changes in OFM sizes and depths. Subfigures 10a and 10b display the amount of accuracy loss and improvement in energy efficiency, respectively, as OFM sizes and depths are reduced. In subfigure 10a, negative accuracy loss is represented on the figures as zero losses, for ease of viewing. With subfigures 10c and 10d, one can see the improvements in throughput and memory usage with changes in OFM sizes and depths.

The inference performance metrics that were discussed above are for batch inference. However, some embedded and mobile platforms require single-image inference (with a batch size of 1) for real-time tasks. For brevity, inference performance for single-image inference was not reported in Tables 12 and 13. Tables 14 and 15 present the throughput, power consumption, energy efficiency, and memory usage data for the baseline models and the optimal models in single-image inference. With the optimal LeNet-5 model there are 2.4× and 2.6× gains in throughput and energy efficiency, respectively, when compared to the baseline LeNet-5 model. In the case of AlexNet, the optimal model achieves 5.1× gain in throughput and 8.8× gain in energy efficiency, as compared to the baseline AlexNet model. For LeNet-5, there is no noticeable improvement with regard to power consumption and memory usage. However, in the case of AlexNet, a 1.7× reduction in power consumption and 1.4× reduction in memory usage are obtained for the optimal model. When inference is conducted with a single image with a smaller architecture like LeNet-5, the CMC may be too small to allow significant reduction in power or memory consumption. The inference performance for baseline and optimal models in batch inference (taken from Tables 12 and 13) are also added to Tables 14 and 15 such that the performance of the optimal models in both batch inference and single-image inference can be viewed together.

As noted in Section 5.2, reducing OFM depth can have minor impact on accuracy. This is because when CONV layers have smaller OFM depths, they would extract smaller number of features and therefore the optimal models (if chosen through Strategy 2 or Strategy 3) will have slightly lower accuracy than the baseline models. However, as documented in this section, one can achieve significant gains in throughput and energy efficiency with the proposed methodology with a trade-off of minor reduction in accuracy. For example, the AlexNet optimal model achieves 11.6× higher throughput and 25.2× greater energy efficiency with only 4.4% loss in accuracy. In many applications that are deployed on embedded or mobile

Table 13 Inference performance for AlexNet SpCNN

OFM size M_2 (β_m)	OFM depth $D_{o(1)}$ (β_d)	Test acc.$^+$ (MNIST)	Test acc.$^+$ (Fashion MNIST)$^+$	Mean acc. loss[3]*	Thru.$^+$ (cl/s)	Power cons.* (W)	Energy eff.$^+$ (cl/J)	Mem. usage* (MB)
S1: Reducing OFM size								
27 (1.0)	16 (1)	95.96%	86.95%	0.00%	483.50	156.40	3.09	2434.00
13 (2.07)	16 (1)	96.89%	78.08%	3.97%	729.31	154.00	4.74	2194.40
6 (4.5)	16 (1)	96.59%	88.37%	– 1.02%	736.44	143.30	5.14	2106.50
S2: Reducing OFM depth								
27 (1.0)	8 (2)	94.38%	86.73%	0.90%	2131.29	101.10	21.08	1046.40
27 (1.0)	4 (4)	91.29%	83.31%	4.15%	5134.92	93.90	54.68	523.60
27 (1.0)	2 (8)	82.64%	71.20%	14.53%	7923.03	56.50	140.23	360.10
S3: Reducing OFM size first and then OFM depth								
6 (4.5)	8 (2)	95.67%	87.64%	– 0.20%	2414.24	94.30	25.60	786.60
6 (4.5)	4 (4)	93.39%	80.74%	4.39%	5626.88	72.10	78.04	436.50
6 (4.5)	2 (8)	87.68%	72.45%	11.39%	8631.64	43.00	200.74	332.20

[3] Negative and positive values in accuracy loss indicate an increase and decrease in accuracy, respectively.

“+” indicates a higher value of this performance metric is better. “*” indicates a lower value of this performance metric is better

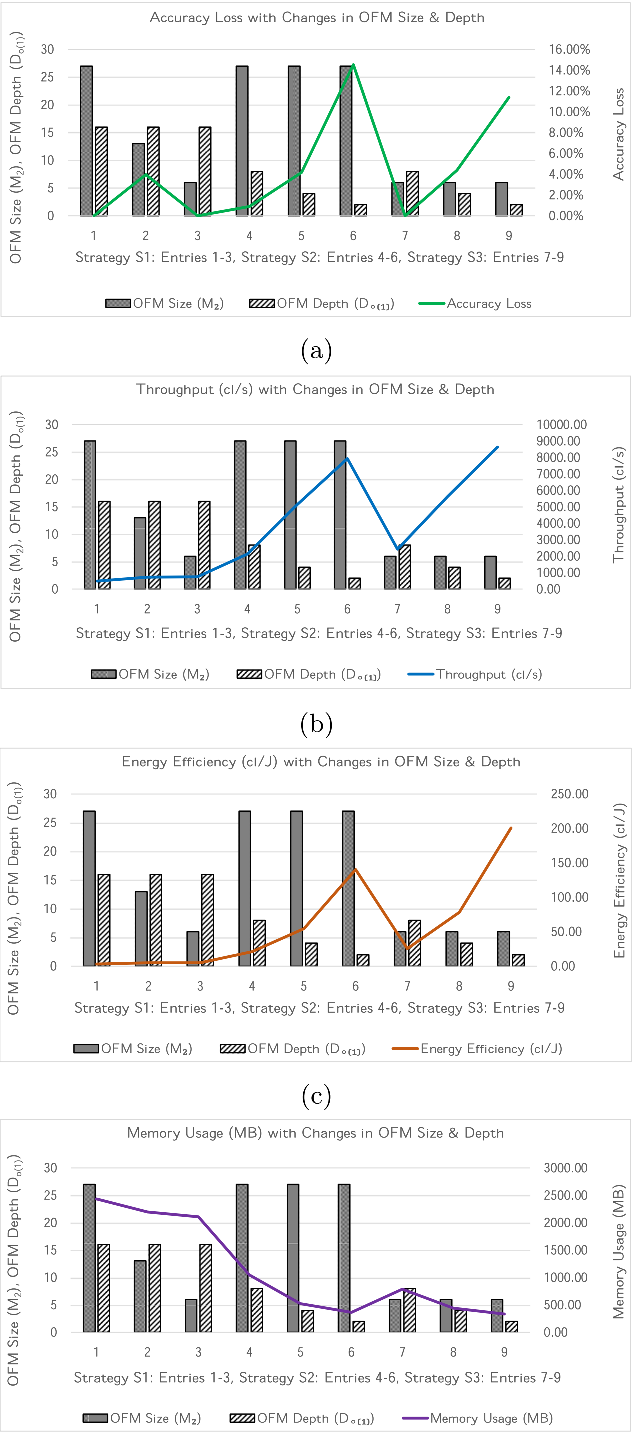


Fig. 10 Inference performance for AlexNet SpCNN in terms of (a) accuracy loss vs OFM size and depth, (b) energy efficiency vs OFM size and depth, (c) throughput vs OFM size and depth, (d) memory usage vs OFM size and depth

platforms, such trade-off where significant performance gains are achieved with a minor loss in accuracy is not only acceptable but desirable [35].

We also compared our optimal model with Ayat et al. [52], Liu et al. [54], and Rizvi et al. [45]—three recent works in SpCNNs. These specific works were chosen for comparison purposes since these works evaluated their SpCNN models with the same datasets such as MNIST and Fashion MNIST just as this work. Additionally, these works used LeNet-5 as their baseline SpCNN model which is one of the two baseline models employed in this work as well. To compare our work with these other works in a fair manner, their works were reproduced in our experimental environment. The OFM sizes and depths of these models were kept intact when they were implemented in our experimental environment.

For the MNIST dataset, our optimal model achieves an accuracy 1.8% higher than Ayat et al. [52] and about 1% higher than Liu et al. [54] but around 3% lower than Rizvi et al. [45]. In the case of Fashion MNIST, our optimal model has an accuracy that is 2.4% and 3% lower than [52] and [45], respectively, but has the same accuracy as [54]. The accuracy values are documented in Table 16. One should note that our baseline model with 97.32% accuracy for MNIST and 88.54% accuracy for Fashion MNIST, has similar or better accuracy than these previous works. As most of the previous works in SpCNN models for AlexNet were on optimizing domain transformations (FFT/IFFT) or inference with hardware accelerators and were evaluated with different datasets, we compared our optimal model with our baseline, rather than these previous works.

When inference performance of our optimal LeNet-5 model is compared with [52], it is observed that during batch inference the optimal model provides 3.1× and 4.2× improvements in throughput and energy efficiency, respectively. Furthermore, there are 1.3× and 1.9× reductions in power consumption and memory usage, respectively. When single-image inference is conducted, the optimal model attains 3.2× higher throughput and 3.4× greater energy efficiency. Even though the reduction in power consumption is minor, memory usage is reduced by 1.7×. These results are shown in Tables 17 and 18.

The optimal LeNet-5 model also surpasses [54] in performance for both batch and single-image inference. In batch inference, the optimal model produces 4.1× higher throughput and 6.9× greater energy efficiency, while reducing power consumption and memory usage by 1.6× and 2.1×, respectively. In single-image inference, 4.1× and 4.2× gains are achieved in throughput and energy efficiency, respectively. The reduction in power consumption is minor but memory usage is reduced significantly, by 1.8×.

With our optimal LeNet-5 model significant performance gain is also attained when compared with [45]. In batch inference, the optimal model achieves 4.2× and 10.5× improvements in throughput and energy efficiency as well as a 2.5× reduction in power consumption and a 2.9×

Table 14 Inference performance of baseline and optimal models for LeNet-5 SpCNN for single-image and batch inference

Work	Thru.$^{+}$ (cl/s)	Power cons.* (W)	Energy eff.$^{+}$ (cl/J)	Mem. usage* (MB)
Single-image inference				
LeNet-5 baseline model	115.21	40.2	2.87	319.6
LeNet-5 optimal model	277.77	37.8	7.35	301.7
Batch inference				
LeNet-5 baseline model	3573.19	94.70	37.73	615.10
LeNet-5 optimal model	12855.88	39.70	323.83	319.90

"+" indicates a higher value of this performance metric is better. "*" indicates a lower value of this performance metric is better

Table 15 Inference performance of baseline and optimal models for AlexNet SpCNN for single-image and batch inference

Work	Thru.$^{+}$ (cl/s)	Power cons.* (W)	Energy eff.$^{+}$ (cl/J)	Mem. usage* (MB)
Single-image inference				
AlexNet baseline model	32.58	67.8	0.48	406.6
AlexNet optimal model	166.09	39.1	4.25	289.4
Batch inference				
AlexNet baseline model	483.50	156.40	3.09	2434.00
AlexNet optimal model	5626.88	72.10	78.04	436.50

"+"indicates a higher value of this performance metric is better. "*" indicates a lower value of this performance metric is better

Table 16 Comparison of the optimal LeNet-5 SpCNN model with some of the recent works in LeNet-5 SpCNN in terms of test accuracy

Work	Test accuracy	
	MNIST	Fashion
Ayat et al. [52]	92.66%	88.00%
Liu et al. [54]	93.52%	85.64%
Rizvi et al. [45]	97.43%	88.62%
This work (optimal model)	94.49%	85.59%

Table 17 Performance comparison of the optimal LeNet-5 SpCNN model with some of the recent works in LeNet-5 SpCNN for batch inference

Work	Batch inference			
	Thru.$^{+}$ (cl/s)	Power cons.* (W)	Energy eff.$^{+}$ (cl/J)	Mem. usage* (MB)
Ayat et al. [52]	4099.07	52.9	77.49	620.0
Liu et al. [54]	3099.86	65.6	47.25	669.0
Rizvi et al. [45]	3077.23	99.7	30.86	929.3
This work (optimal model)	12855.88	39.7	323.83	319.90

"+" indicates a higher value of this performance metric is better. "*" indicates a lower value of this performance metric is better

Table 18 Performance comparison of the optimal LeNet-5 SpCNN model with some of the recent works in LeNet-5 SpCNN for single-image inference

Work	Single-image inference			
	Thru.$^+$ (cl/s)	Power cons.* (W)	Energy eff.$^+$ (cl/J)	Mem. usage* (MB)
Ayat et al. [52]	87.88	41.2	2.13	516.9
Liu et al. [54]	68.10	39.4	1.73	531.5
Rizvi et al. [45]	119.83	42.1	2.85	520.6
This work (optimal model)	277.77	37.8	7.35	301.7

"+" indicates a higher value of this performance metric is better. "*" indicates a lower value of this performance metric is better

reduction in memory usage. In single-image inference, there is a 2.3× gain in throughput along with a 2.6× improvement in energy efficiency. The reduction in power consumption is minor but a 1.7× reduction in memory usage is achieved. Tables 17 and 18 document the performance gain of our optimal model when compared to the above-mentioned previous works.

As discussed in Section 2, the three works that are compared with our optimal model propose different models for LeNet-5 SpCNN and different methods (e.g., fused convolution layers [52], hardware-friendly coefficients for activation function [54], computationally light activation function [45]) to optimize their computational workload. Unlike the above-mentioned works that aim to reduce only the computational cost of SpCNNs, this work proposes a methodology to reduce both computational and memory accesses costs and is able to obtain higher throughput and energy efficiency and smaller memory-footprint with comparable or better accuracy when compared with these works.

7 Conclusion

In this paper, we have demonstrated that the sizes and depths of OFMs are the primary contributors to the computational and memory costs of spectral domain CNNs. These costs can be minimized efficiently by optimizing the depth of OFMs after reducing OFM sizes to the smallest feasible size, without significant loss in test accuracy. Our methodology was evaluated on two well-known CNN architectures—LeNet-5 and AlexNet—with two widely used datasets (MNIST and Fashion MNIST). The performance gains (in terms of throughput, energy efficiency etc.) attained for both single-image and batch inference show that our proposed methodology is highly effective in achieving fast and energy-efficient inference. Another attractive feature of our methodology is that it does not require any specialized compression algorithm or hardware accelerator to achieve its goal. In future, this methodology can be evaluated on more complex CNN architectures such as GoogleNet or ResNet and larger datasets.

Data Availability The datasets employed to evaluate the CNN models in the current study are available in [62] and https://github.com/zalandoresearch/fashion-mnist.

Declarations

Competing interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L (2021) Review of deep learning-concepts, CNN architectures, challenges, applications, future directions. J Big Data 8(1):1–74
2. Ngo L, Cha J, Han J-H (2020) Deep neural network regression for automated retinal layer segmentation in optical coherence tomography images. IEEE Trans Image Process (TIP) 29:303–312
3. Xiao Y, Zijie Z (2020) Infrared image extraction algorithm based on adaptive growth immune field. Neural Process Lett 51(3):2575–2587
4. Yu X, Zhou Z, Gao Q, Li D, Ríha K (2018) Infrared image segmentation using growing immune field and clone threshold. Infrared Phys Technol 88:184–193
5. Zhu W, Peng B, Wu H, Wang B (2020) Query set centered sparse projection learning for set based image classification. Appl Intell 50(10):3400–3411
6. Zhu W, Peng Y (2020) Elastic net regularized kernel non-negative matrix factorization algorithm for clustering guided image representation. Appl Soft Comput 97:106774
7. Otter DW, Medina JR, Kalita JK (2021) A survey of the usages of deep learning for natural language processing. IEEE Trans Neural Netw Learn Syst (TNNLS) 32(2):604–624
8. Grigorescu S, Trasnea B, Cocias T, Macesanu G (2020) A survey of deep learning techniques for autonomous driving. J Field Robot 37(3):362–386
9. LeCun Y, Boser B, Denker J, Henderson D, Howard R, Hubbard W, Jackel L (1989) Handwritten digit recognition with a back-propagation network. In: Proceedings of the 2nd international conference on neural information processing systems (NIPS), pp 396–404
10. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
11. Krizhevsky A, Sutskever I, Hinton G (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
12. Hu J, Shen L, Albanie S, Sun G, Wu E (2020) Squeeze-and-Excitation networks. IEEE Trans Pattern Anal Mach Intell (TPAMI) 42(8):2011–2023
13. Cao C, Wang B, Zhang W, Zeng X, Yan X, Feng Z, Liu Y, Wu Z (2019) An improved faster r-CNN for small object detection, vol 7
14. Aziz L, Haji Salam MSB, Sheikh UU, Ayub S (2020) Exploring deep learning-based architecture, strategies, applications and

current trends in generic object detection: a comprehensive review. IEEE Access 8:170461–170495

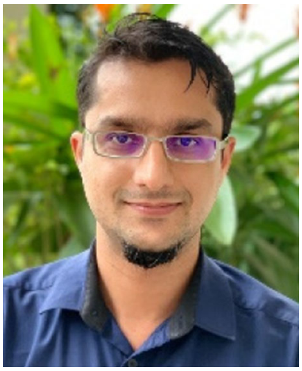
15. Shelhamer E, Long J, Darrell T (2017) Fully convolutional networks for semantic segmentation. IEEE Trans Pattern Anal Mach Intell (TPAMI) 39(4):640–651
16. Li C, Xia W, Yan Y, Luo B, Tang J (2021) Segmenting objects in day and night: edge-conditioned CNN for thermal image semantic segmentation. IEEE Trans Neural Netw Learn Syst (TNNLS) 32(7):3069–3082
17. Kang S, Lee J, Bong K, Kim C, Kim Y, Yoo H-J (2018) Low-power scalable 3-d face frontalization processor for CNN-based face recognition in mobile devices. IEEE J Emerg Sel Top Circuits Syst (JETCAS) 8(4):873–883
18. Jiang L, Zhang J, Deng B (2020) Robust RGB-d face recognition using attribute-aware loss. IEEE Trans Pattern Anal Mach Intell (TPAMI) 42(10):2552–2566
19. Khurana K, Deshpande U (2021) Video question-answering techniques, benchmark datasets and evaluation metrics leveraging video captioning: a comprehensive survey. IEEE Access 9:43799–43823
20. Lin Y, Guo D, Zhang J, Chen Z, Yang B (2021) A unified framework for multilingual speech recognition in air traffic control systems. IEEE Trans Neural Netw Learn Syst (TNNLS) 32(8):3608–3620
21. Kim T, Lee J, Nam J (2019) Comparison and analysis of sample CNN architectures for audio classification. IEEE J Sel Top Signal Process (JSTSP) 13(2):285–297
22. Ramisa A, Moreno-Noguer F, Moreno-Noguer K (2018) Breaking news: article annotation by image and text processing. IEEE Trans Pattern Anal Mach Intell (TPAMI) 40(5):1072–1085
23. Chen L, Lin S, Lu X, Cao D, Wu H, Guo C, Liu C, Wang F.-Y. (2021) Deep neural network based vehicle and pedestrian detection for autonomous driving: a survey. IEEE Trans Intell Transp Syst (TITS) 22(6):3234–3246
24. Miclea V-C, Nedevschi S (2022) Monocular depth estimation with improved long-range accuracy for UAV environment perception. IEEE Trans Geosci Remote Sens (TGRS) 60:1–15
25. Dai Z, Yi J, Zhang Y, Zhou B, He L (2020) Fast and accurate cable detection using CNN. Appl Intell 50(12):4688–4707
26. Esteva A, Kuprel B, Novoa R, Ko J, Swetter S, Blau H, Thrun S (2017) Dermatologist-level classification of skin cancer with deep neural networks. Nature 542(7639):115–118
27. Nayak J, Naik B, Dinesh P, Vakula K, Rao B, Ding W, Pelusi D (2021) Intelligent system for COVID-19 prognosis: a state-of-the-art survey. Appl Intell 51(5):2908–2938
28. Saraogi E, Chouhan G, Panchal D, Patel M, Gajjar R (2021) CNN Based design rule checker for VLSI layouts. In: Proceedings of the 2nd IEEE international conference on applied electromagnetics, signal processing & communication (AESPC), pp 1–6
29. Sze V, Chen Y-H, Yang T-J, Emer J (2017) Efficient processing of deep neural networks: a tutorial and survey. Proc IEEE 105(12):2295–2329
30. Abtahi T, Shea C, Kulkarni A, Mohsenin T (2018) Accelerating convolutional neural network with FFT on embedded hardware. IEEE Trans Very Large Scale Integr (TVLSI) 26(9):1737–1749
31. Jain A, Phanishayee A, Mars J, Tang L, Pekhimenko G (2018) Gist: efficient data encoding for deep neural network training. In: Proceedings of the 45th international symposium on computer architecture (ISCA), pp 776–789
32. Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C (2017) Learning efficient convolutional networks through network slimming. In: Proceedings of the 16th IEEE international conference on computer vision (ICCV), pp 2755–2763
33. Chao P, Kao C-Y, Ruan Y, Huang C-H, Lin Y-L (2019) HarDNet: a low memory traffic network. In: Proceedings of the 17th IEEE/CVF international conference on computer vision (ICCV), pp 3551–3560
34. Chen Y-H, Krishna T, Emer JS, Sze V (2017) Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE Journal of Solid-State Circuits (JSSC) 52(1):127–138
35. Ma N, Zhang X, Zheng H-T, Sun J (2018) Shuffle Net v2: practical guidelines for efficient CNN architecture design. In: Proceedings of the 15th European conference on computer vision (ECCV), pp 116–131
36. Vaze S, Xie W (2020) Namburete, A.I.L.e.: low-memory CNNs enabling real-time ultrasound segmentation towards mobile deployments. IEEE J Biomed Health Inform (JBHI) 24(4):1059–1069
37. Mathieu M, Henaff M, LeCun Y (2014) Fast training of convolutional networks through FFTs. In: Proceedings of the 2nd international conference on learning representations (ICLR)
38. Vasilache N, Johnson J, Mathieu M, Chintala S, Piantino S, LeCun Y (2015) Fast convolutional nets with fbfft: a GPU performance evaluation. In: Proceedings of the 3rd international conference on learning representations (ICLR)
39. Rippel O, Snoek J, Adams R (2015) Spectral representations for convolutional neural networks. In: Proceedings of the 28th international conference on neural information processing systems (NIPS), pp 2449–2457
40. Ko J, Mudassar B, Na T, Mukhopadhyay S (2017) Design of an energy-efficient accelerator for training of convolutional neural networks using frequency-domain computation. In: Proceedings of the 54th ACM/EDAC/IEEE design automation conference (DAC), pp 1–6
41. Niu Y, Zeng H, Srivastava A, Lakhotia K, Kannan R, Wang Y, Prasanna V (2019) SPEC2: SPECtral SParsE CNN accelerator on FPGAs. In: Proceedings of the 26th IEEE international conference on high performance computing, data, and analytics (HiPC), pp 195–204
42. Sun W, Zeng H, Yang Y-h, Prasanna V (2018) Throughput-optimized frequency domain CNN with fixed-point quantization on FPGA. In: Proceedings of the 13th international conference on ReConFigurable computing and FPGAs (ReConFig), pp 1–8
43. Nguyen-Thanh N, Le-Duc H, Ta D-T, Nguyen V-T (2016) Energy efficient techniques using FFT for deep convolutional neural networks. In: Proceedings of the 9th international conference on advanced technologies for communications (ATC), pp 231–236
44. Lin J, Yao Y (2019) A fast algorithm for convolutional neural networks using tile-based fast fourier transforms. Neural Process Lett 50(2):1951–1967
45. Rizvi S, Ab Rahman A, Khalil-Hani M, Ayat S (2021) A low-complexity complex-valued activation function for fast and accurate spectral domain convolutional neural network. Indones J Electr Eng Inform (IJEEI) 9(1):173–184
46. Howard A, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861
47. Zhang X, Zhou X, Lin M, Sun J (2018) Shuffle net: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of the 31st IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6848–6856
48. Huang G, Liu S, Maaten L, Weinberger K (2018) Condensenet: an efficient DenseNet using learned group convolutions. In: Proceedings of the 31st IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2752–2761

49. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: an imperative style, high-performance deep learning library. In: Proceedings of the 32nd international conference on neural information processing systems (NIPS), pp 8024–8035
50. Tensor Flow Lite: ML for mobile and edge devices. https://www.tensorflow.org/lite/. Accessed 1 Nov 2021
51. Gibson J, Pand Cano, Turner J, Crowley E, O'Boyle M, Storkey A (2020) Optimizing grouped convolutions on edge devices. In: Proceedings of the 31st international conference on application-specific systems, architectures and processors (ASAP), pp 189–196
52. Ayat S, Khalil-Hani M, Ab Rahman A, Abdellatef H (2019) Spectral-based convolutional neural network without multiple spatial-frequency domain switchings. Neurocomputing 364:152–167
53. Watanabe T, Wolf D (2021) Image classification in frequency domain with 2SReLU: a second harmonics superposition activation function. Appl Soft Comput 112:107851–107851
54. Liu S, Luk W (2020) Optimizing fully spectral convolutional neural networks on FPGA. In: Proceedings of the 19th IEEE international conference on field-programmable technology (ICFPT), pp 39–47
55. Guan B, Zhang J, Sethares W, Kijowski R, Liu F (2021) Spectral domain convolutional neural network. In: Proceedings of the 46th IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 2795–2799
56. Abdelouahab K, Pelcat M, Berry F (2020) Accelerating the CNN inference on FPGAs. In: Fagerberg J, Mowery DC, Nelson R (eds) Deep learning in computer vision: principles and applications, pp 1–39. Chap 1. CRC Press Taylor & Francis Group, USA
57. Meurant G (1999) Computer solution of large linear systems. Elsevier, Amsterdam.
58. Kala S, Jose B, Paul D, Mathew J (2018) A hardware accelerator for convolutional neural network using fast Fourier transform. In: Proceedings of the 22nd international symposium on vlsi design and test (VDAT), pp 28–36
59. Sadouk L (2019) CNN Approaches for time series classification. In: Ngan C-K (ed) Time series analysis - data, methods, and applications, pp 57–79. Chap 4. IntechOpen, London
60. Wang E, Davis J, Zhao R, Ng H-C, Niu X, Luk W, Cheung P, Constantinides G (2019) Deep neural network approximation for custom hardware: where we've been, where we're going. ACM Comput Surv 52(2):1–39
61. Vedaldi A, Lux M, Bertini M (2018) Matconvnet: CNNs are also for MATLAB users. ACM SIGMultimedia Records 10(1):9–9
62. LeCun Y, Cortes C (2010) MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Accessed 21 Oct 2021
63. Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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), Johor Bahru, Malaysia. He also serves as an Associate Professor at American International University-Bangladesh (AIUB). His research interests include low-complexity and memory-efficient deep learning, synthesis of FSMs from purely behavioral SystemVerilog/VHDL descriptions, computer vision in spectral domain, and hardware/software co-simulation for FPGA-based image processing.

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 50 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.



Usman Ullah Sheikh received his bachelor's degree in mechatronics (2003), master's degree in telecommunications engineering (2005) and PhD degree (2009) in image processing and computer vision from Universiti Teknologi Malaysia. His research work is mainly on computer vision, machine learning and embedded systems design. He is currently a senior lecturer at Universiti Teknologi Malaysia.

Kazi Ahmed Asif Fuad received Erasmus+ Mundus Joint Master Degree in Image Processing and Computer Vision (IPCV) in 2020. Prior to that, he obtained M.Sc. and B.Sc. degrees in Electrical and Electronic Engineering from American International University-Bangladesh (AIUB) in 2013 and 2015, respectively. He is currently pursuing a Ph.D. degree in Computer Science at Oregon State University, Corvallis, USA. He also serves as an Assistant Professor at American International University-Bangladesh (AIUB). His research interests include machine learning accelerators, high performance computing, hardware/software co-simulation for FPGA-based image processing, computer vision, and explainable AI.

Hafiz Muhammad Faisal Shehzad is a lecturer at Depart ment of Computer Science and IT, University of Sargodha, Pakistan. He obtained his PhD degree from Universiti Teknologi Malaysia, Malaysia in 2022. He received his Master of Computer Science degree from University of Sargodha, Pakistan in 2013. His area of research is building information modeling (BIM), system analysis, machine learning and AI. His research interests include technology adoption and implementation, information systems, and database systems.

Affiliations

Shahriyar Masud Rizvi[1] · **Ab Al-Hadi Ab Rahman**[1] · **Usman Ullah Sheikh**[1] · **Kazi Ahmed Asif Fuad**[2] · **Hafiz Muhammad Faisal Shehzad**[3]

Ab Al-Hadi Ab Rahman
hadi@fke.utm.my

Usman Ullah Sheikh
usman@fke.utm.my

Kazi Ahmed Asif Fuad
fuadk@oregonstate.edu

Hafiz Muhammad Faisal Shehzad
muhammad.faisal@uos.edu.pk

[1] VeCAD Research Laboratory, School of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru, 81310, Johor, Malaysia

[2] School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, USA

[3] Department of Computer Science and IT, University of Sargodha, Sargodha, 40100, Punjab, Pakistan