Design of an Energy-Efficient Accelerator for Training of Convolutional Neural Networks using Frequency-Domain Computation

Jong Hwan Ko, Burhan Mudassar, Taesik Na, and Saibal Mukhopadhyay
Georgia Institute of Technology
266 Ferst Drive, Atlanta, GA 30332, USA
{jonghwan.ko, burhan.mudassar, taesik.na}@gatech.edu, saibal@ece.gatech.edu

ABSTRACT

Convolutional neural networks (CNNs) require high computation and memory demand for training. This paper presents the design of a frequency-domain accelerator for energy-efficient CNN training. With Fourier representations of parameters, we replace convolutions with simpler pointwise multiplications. To eliminate the Fourier transforms at every layer, we train the network entirely in the frequency domain using approximate frequency-domain nonlinear operations. We further reduce computation and memory requirements using sinc interpolation and Hermitian symmetry. The accelerator is designed and synthesized in 28nm CMOS, as well as prototyped in an FPGA. The simulation results show that the proposed accelerator significantly reduces training time and energy for a target recognition accuracy.

Keywords

convolutional neural network (CNN); frequency domain; training.

1. INTRODUCTION

Convolutional neural networks (CNNs) have been successfully applied in a wide range of computer vision tasks including image classification [1] and face recognition [2]. Deeper network architectures and larger training data sets have been a key driver for the success of CNNs; however, at the expense of increased training time due to computational complexity. For example, 3-convolution-layer LeNet-5 requires 3 Giga operations (GOPs) to train a 32x32 image, while 13 convolutional layer VGG-16 learns a 224x224 image with 43,142 GOPs [Figure. 1]. The high computational complexity is a major challenge in training CNNs, particularly, in embedded platforms. Therefore, reducing the computation demand of CNNs is critical, specifically, to support in-field and on-chip learning.

As the computation demand of CNNs is largely dominated by convolution layers, a number of studies have explored efficient computation models for convolution layers [3]. Several methods have focused on reducing the number of convolution operations by approximating the network parameters [4]. An interesting alternative approach for fast training is to exploit the duality between spatial- and frequency-domain computation through the Fourier transforms. Recent studies [5][6] have shown the feasibility of replacing convolutions with simpler pointwise multiplications in the frequency domain (FFT-based approach) [Figure. 2(b)]. However, this approach replaced only the operations inside the convolution layer, requiring computationally-intensive Fast Fourier Transforms (FFTs) and Inverse FFTs (IFFTs) at the boundary of every layer. Moreover, frequency-domain mapping of the parameters requires kernels to be prepared as same dimension as feature maps for pointwise multiplications, thereby significantly increasing the total memory required. In summary, although the frequency-domain approach provides fast training of CNNs, the entire network model should be carefully designed to reduce the overhead.

DAC '17, June 18-22, 2017, Austin, TX, USA
© 2017 ACM. ISBN 978-1-4503-4927-7/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3061639.3062228

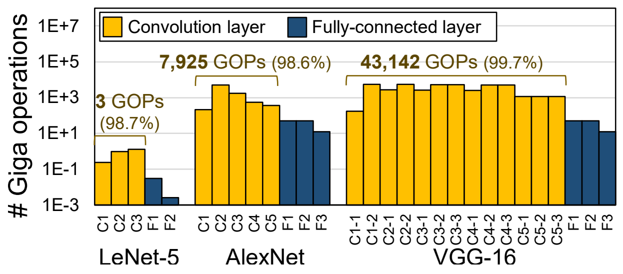


Figure 1. Computational complexity of various CNNs.

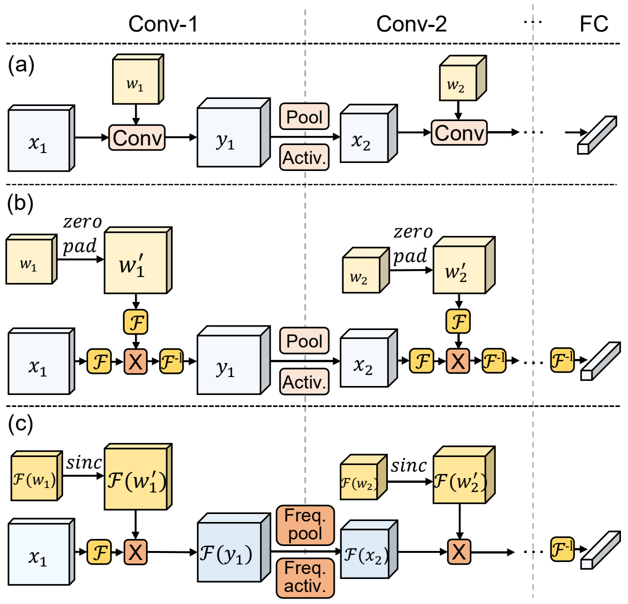


Figure 2. CNN models in forward propagation with (a) conventional spatial-domain approach, (b) FFT-based approach, and (c) the proposed entire frequency-domain approach (w: kernels, x: input features, y: output features, $\mathcal{F}$:FFT, $\mathcal{F}^{-1}$:IFFT, sinc: sinc interpolation).

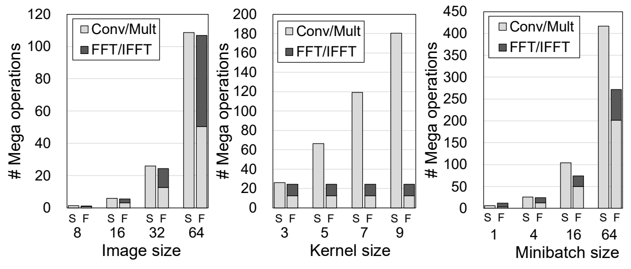


Figure 3. Complexity comparison of the spatial-domain approach (S) and FFT-based approach (F) (Default values: input feature size=32x32 and depth=16, kernel size=3x3 and depth=16, and minibatch=4).

In this paper, we propose an accelerator design that maps the entire convolution layer operations and parameters into the frequency domain for fast and energy-efficient CNN training [Figure. 2(c)]. The key contributions of this paper are:

- We introduce frequency-domain activation functions for the non-linear operations (e.g. pooling, activation function) between convolution layers. Hence, we can train the entire network with frequency-domain computation without the need for the FFTs/IFFTs in-between the layers.
- We propose to store frequency-domain kernels without zero padding to eliminate memory overhead, and expand them before computation through sinc interpolation, which is the frequency-domain duality of zero-padding.
- We store and multiply only half of the parameters using Hermitian symmetry to reduce computation and memory.

We design a digital accelerator for training and inference of CNNs with the proposed approach. The simulations with the MNIST and CIFAR-10 datasets show that the proposed approach with the approximate nonlinear operations achieves the same accuracy as the spatial-domain CNN training model. The design is synthesized in FPGA, as well as in 28nm CMOS technology for performance and energy analysis. The results show that the proposed accelerator significantly reduces training time and energy. The advantage becomes larger for a network with larger input and kernel sizes. For AlexNet, we reduce 4.7X and 6.3X energy and latency than the conventional spatial-domain training.

2. BACKGROUND

With frequency-domain representations of the parameters, convolutions can be performed through simple element-wise multiplications (FFT-based convolution) [5][6]. Let us assume each layer has f' different kernels of depth f, and a set of f input feature maps, which is a part of minibatch S. During forward propagation, each output feature map $y_{s,j}$ is computed as a sum of the input feature maps $x_{s,i}$ cross-correlated with the corresponding kernel $w_{j,i}$, where $j \in f'$, $i \in f$, $s \in S$. Cross-correlation can be obtained as the inverse Fourier transform of the pointwise product between the individual transforms:

$$y_{s,j} = \sum_{i \in f} x_{s,i} \star w_{j,i} = \sum_{i \in f} \mathcal{F}^{-1}\left(\mathcal{F}(x_{s,i}) \cdot \mathcal{F}(w_{j,i}{}')^*\right)$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are FFT and IFFT, and * denotes complex conjugation. Note that the kernels should be zero-padded ($w_{j,i}{}'$) to be the same size as the input feature before applying the FFT, in order to perform desired linear convolution.

During backward propagation, the gradients of loss with respect to the outputs ($\partial L/\partial y_{s,j}$) are convolved with kernels, producing the

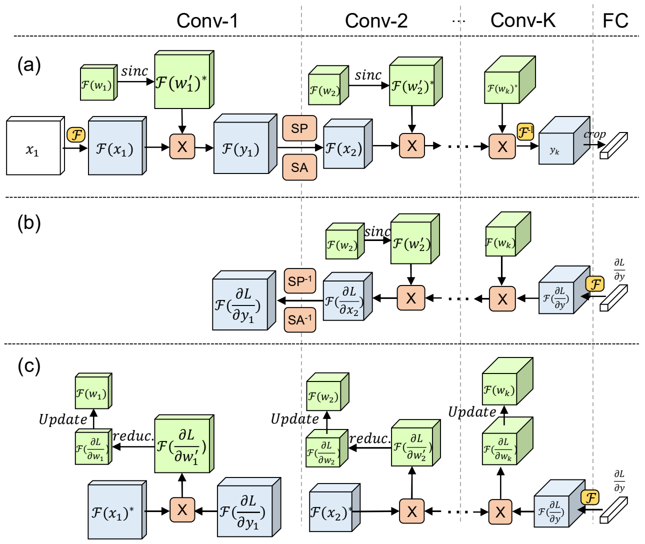


Figure 4. CNN model of the proposed approach in (a) forward propagation, (b) backward propagation, and (c) gradient calculation and update (SP, SA: spectral pooling/activation). **Note that no FFT is required in (b) and (c) if $\partial L/\partial y$ from the fully-connected layer is 1x1 shape.**

gradients with respect to the inputs ($\partial L/\partial x_{s,i}$). As convolutions can be replaced with multiplications, we have:

$$\frac{\partial L}{\partial x_{s,i}} = \sum_{j \in f'} \frac{\partial L}{\partial y_{s,j}} * w_{j,i} = \sum_{j \in f'} \mathcal{F}^{-1}\left(\mathcal{F}\left(\frac{\partial L}{\partial y_{s,j}}\right) \cdot \mathcal{F}(w_{j,i}{}')\right)$$

To calculate the gradients of the loss with respect to the weight ($\partial L/\partial w_{j,i}$), the input feature maps are cross-correlated with the gradients with respect to the outputs. In the frequency domain, it can be calculated by multiplying the two matrices:

$$\frac{\partial L}{\partial w_{j,i}} = \sum_{s \in S} \frac{\partial L}{\partial y_{s,j}} \star x_{s,i} = \sum_{s \in S} \mathcal{F}^{-1}\left(\mathcal{F}\left(\frac{\partial L}{\partial y_{s,j}}\right) \cdot \mathcal{F}(x_{s,i})^*\right)$$

In this approach, every convolution involves three Fourier transforms. As some of the Fourier representations obtained in the forward pass are used in the backward pass, Mathieu et al. [5] proposed storing and reusing them to reduce the transform overhead. However, the complexity comparison with the spatial-domain convolution [Figure 3] shows that the transform overhead still accounts for large portion of the total complexity, making it less efficient than spatial convolution for small kernel and minibatch sizes. Furthermore, this approach requires significant amount of additional memory to store the Fourier representations. In this approach the transforms could not be removed because the output feature maps should be transformed back to the spatial domain for non-linear operations (e.g. pooling, activation function), which do not have exact dual operations in the frequency domain.

In this paper, we use approximate frequency-domain nonlinear operations that enable us to maintain the feature maps in the frequency domain after convolutions. Kernels are also initialized and updated directly in the frequency domain, thereby eliminating the Fourier transforms inside the entire set of convolution layers. To avoid memory increase, we exploit sinc interpolation and subsampling techniques that allow us to store the frequency-domain kernels with the original dimension. Using Hermitian symmetry of the parameters, we further reduce computation and memory demand by storing and computing only part of the parameters.

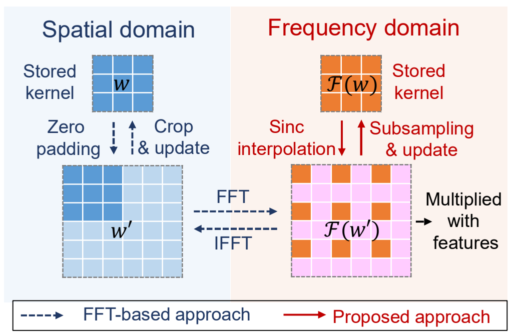


Figure 5. Manipulation of kernel in the FFT-based and proposed approach.

3. PROPOSED APPROACH

3.1 Overall Structure

As Figure 4 illustrates, the proposed training approach uses the same processes as the conventional spatial-domain CNN model, but with the frequency-domain parameters and operations.

At the beginning of the forward pass, a set of input images is transformed into their Fourier representations [Figure 4(a)]. Note that the input images can be prepared as the Fourier representations through the off-line transforms before training in order to reduce the on-line training time. Frequency-domain kernels can be initialized by transforming the spatial-domain initial weights, which can be generated by various initialization techniques. However, this requires the transform overhead for the entire kernel sets, although the transforms are performed only once at the beginning of the training. Alternatively, the weights can be initialized directly in the frequency domain by randomly setting the complex numbers using the initialization techniques.

Note that kernels are initialized and stored without zero-padding ($\mathcal{F}(w)$). They are expanded through sinc interpolation ($\mathcal{F}(w')$) before being multiplied with the input feature maps during forward propagation. The output feature maps are then processed through pooling and activation in the frequency domain. Originally the input feature map dimension ($n\times n$) reduces to $(n-k+1)\times(n-k+1)$ after convolution with $k\times k$ kernel, and then reduces to half ($(n-k+1)/2\times(n-k+1)/2$) by the pooling operation. However, in the proposed approach, the output dimension after convolution is kept same as the input feature maps to avoid the overhead of the transform and cropping. Instead, the $n\times n$ output features are directly reduced to $(n-k+1)/2\times(n-k+1)/2$ dimension by the spectral pooling that supports arbitrary reduction ratio. Feature maps remain in the frequency domain until the last convolution layer, where they are transformed back into the spatial domain and cropped for the desired dimension. Therefore, the proposed approach requires only two Fourier transforms, which are at the boundaries of the entire set of convolution layers.

The entire backward propagation and gradient calculation process is also performed in the frequency domain [Figure 4(b-c)]. As the gradients of loss from the first fully-connected layer ($\partial L/\partial y$) is in the spatial domain, they should be transformed into the frequency domain ($\mathcal{F}(\partial L/\partial y)$) for the backward pass towards convolution layers. However, if the gradients ($\partial L/\partial y$) are a set of scalar (1x1 shape) values, their FFT is not necessary since the FFT of scalar value is an identity function. In backward propagation [Figure 4(b)], the gradients with respect to the output ($\mathcal{F}(\partial L/\partial y)$) are obtained as the Fourier representation by using the frequency-domain backward pooling and activation function. They are then

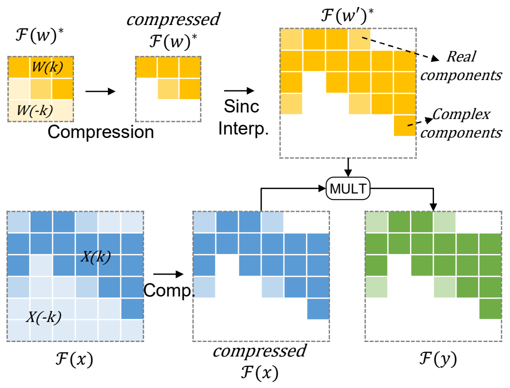


Figure 6. Convolution of the compressed kernel and feature map in forward propagation.

multiplied with the conjugated input feature maps ($\mathcal{F}(x)^*$) to get the gradients with respect to the weights ($\mathcal{F}(\partial L/\partial w')$), which are used to update frequency-domain kernels [Figure 4(c)]. Since the gradients are in the zero-padded dimension, their dimension should be reduced ($\mathcal{F}(\partial L/\partial w)$) in order to update kernels in the original non-zero-padded dimension.

3.2 Application of Sinc Interpolation

In order to implement desired linear convolution through the FFT, $k\times k$ kernels should be zero-padded before the FFT, making their dimension as same as the features ($n\times n$). Therefore, one may consider storing kernels as the Fourier representation of zero-padded weights. However, considering recent networks with large input image size, this approach will lead to significant increase (n^2/k^2) in memory size and bandwidth.

To avoid this memory increase, we propose storing the FFTs of the original kernels ($\mathcal{F}(w)$) and expanding them to be the FFTs of zero-padded kernels ($\mathcal{F}(w')$) before the computation [Figure 5]. For the expansion of kernels, we exploit the zero-padding theorem, stating that the FFT of zero-padded signal can be obtained by convolving the FFT of the original signal with a discrete sinc kernel [7]:

$$G(u,v) = \bar{F}(u,v) * [e^{\frac{-j2\pi u}{n}\left(\frac{k-1}{2}\right)} \frac{sin\left(\frac{\pi uk}{n}\right)}{sin\left(\frac{\pi u}{n}\right)} \cdot e^{\frac{-j2\pi v}{n}\left(\frac{k-1}{2}\right)} \frac{sin\left(\frac{\pi vk}{n}\right)}{sin\left(\frac{\pi v}{n}\right)}]$$

When updating kernels in the backward pass, the gradients $\mathcal{F}(\partial L/\partial w')$ should be reduced to $\mathcal{F}(\partial L/\partial w)$. Generally, the reduction can also be performed through interpolation. However, when the dimension of $\mathcal{F}(w')$ is multiple of $\mathcal{F}(w)$ dimension, we can obtain all the values of $\mathcal{F}(w)$ immediately by subsampling the components in $\mathcal{F}(w')$. Therefore, with a constraint that the dimension ratio of the input features and kernels in a layer is multiples, no sinc interpolation is required during the gradient calculation of the layer.

Sinc interpolation from $k\times k$ to $n\times n$ signal requires k^2n^2 complex multiplications, while memory requirement reduces from $O(n^2)$ to $O(k^2)$. Note that no interpolation is needed at the last convolution layer if the shapes of the input feature maps and kernels are same ($n = k$).

3.3 Application of Hermitian Symmetry

Given an $n\times n$ real-valued signal x, its Fourier transform X has Hermitian (conjugate) symmetry, where negative frequency coefficients are conjugate of positive ones [8]:

$$X(k,l) = X^*(-k \bmod n, -l \bmod n)$$

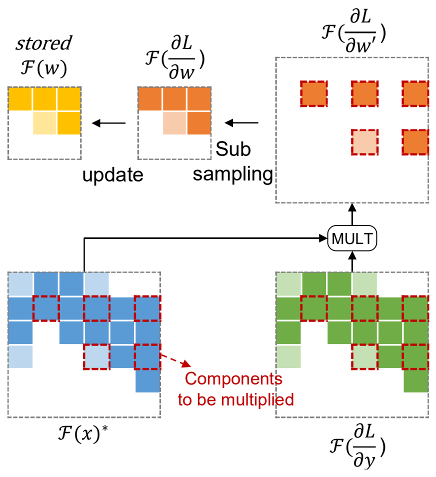

stored
$\mathcal{F}(w)$
$\mathcal{F}(\frac{\partial L}{\partial w})$
$\mathcal{F}(\frac{\partial L}{\partial w'})$
update
Sub sampling
MULT
Components to be multiplied
$\mathcal{F}(x)^*$
$\mathcal{F}(\frac{\partial L}{\partial y})$

Figure 7. Convolution of the output gradients and the input feature maps for obtaining gradients, which are subsampled to update the stored kernels in the backward pass.

This property is commonly used to describe the Fourier representation only with the positive-frequency samples. Since the parameters in general CNNs are real-valued, a prior study [5] has exploited this property to represent the FFT of $n \times n$ parameters using $n/2 \times (n+1)$ components with real and imaginary parts. However, some of these components are real values, which do not need to be stored with the two parts. More precisely, FFT of $n \times n$ real signal has $(n^2 - \alpha)/2$ complex and α real components, where α is 1 for odd n and 4 for even n. Therefore, the memory requirement can be further optimized to n^2 real components, the same space as the original real signal. Note that this approach also allows us to perform half the computation by multiplying only the positive-frequency components.

Figure 6 shows how this property is exploited in the proposed approach. At the beginning of forward propagation, Fourier-transformed kernels and input features are compressed by discarding negative-frequency components. The positive-frequency components in the compressed kernels are interpolated and multiplied with the corresponding components in the compressed feature maps. The original form of feature maps with negative-frequency components is reconstructed simply by taking the complex conjugate of positive-frequency values before being transformed into the spatial domain at the last convolution layer.

Using this symmetry, n^2 complex multiplications reduce to $(n^2 - \alpha)/2$ complex and α real multiplications, where α is 1 for odd n and 4 for even n. For energy-efficient complex

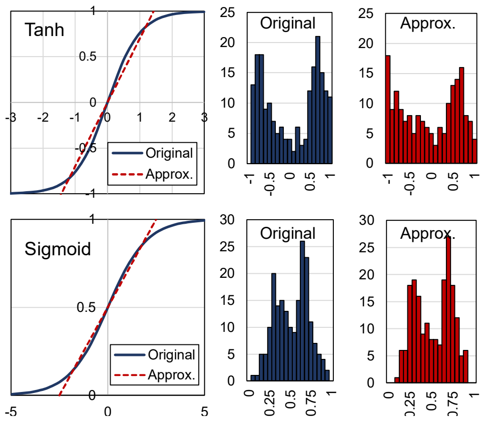


Figure 8. (Left): Approximation of activation functions (tanh, sigmoid) in the frequency domain. (Right): Comparison of output feature distribution.

multiplication, we use a well-known technique that uses three real products per multiplication [9]. Therefore, for convolution of $n \times n$ feature and $k \times k$ kernel, the proposed approach requires $3/2\ (n^2 - \alpha) + \alpha$ real products, while the spatial approach requires $(n - k + 1)^2 \cdot k^2$ real products.

The computational complexity in the backward pass is further reduced in conjunction with the property of zero-padding [Figure 7]. As explained in the Section 3.2, the gradients with respect to zero-padded weights ($\mathcal{F}(\partial L/\partial w')$) are reduced to the gradients without zero padding ($\mathcal{F}(\partial L/\partial w)$) by sampling only certain components. Therefore, only the corresponding components in $\mathcal{F}(x)^*$ and $\mathcal{F}(\partial L/\partial y)$ can be multiplied to obtain the gradients. This approach reduces complexity of gradient calculation from $3/2\ (n^2 - \alpha) + \alpha$ to $3/2\ (k^2 - \alpha) + \alpha$, where n, k are the sizes of the input features and kernels, respectively.

3.4 Frequency-Domain Nonlinear Operations

For a frequency-domain pooling operation, we use the spectral pooling proposed by Rippel et al. [10]. Given an $\alpha \times \alpha$ frequency-domain input feature map with DC component shifted to the center, central submatrix with the desired size $\beta \times \beta$ is cropped. The spectral pooling is claimed to be more beneficial than the spatial-domain max or average pooling, since it maintains more information and also allows any arbitrary output dimensionality.

For a frequency-domain activation function, we introduce approximate sigmoid and tanh functions using linearity of FFT. Generally, the input features to the activation function are normalized to have zero means and unit variances to speedup

Approach	Operation	Computational complexity (# real-number multiplication)			Memory requirement (# real number)		
		Inference	Back propagation	Gradient calculation	Weights	Feature	Additional memory for FFT reuse
Spatial	CONV	$S \cdot f' \cdot f \cdot (n-k+1)^2 \cdot k^2$	$S \cdot f' \cdot f \cdot n^2 \cdot k^2$	$S \cdot f' \cdot f \cdot (n-k+1)^2 \cdot k^2$	$f' \cdot f \cdot k^2$	$S \cdot f \cdot n^2$	-
FFT-based	FFT/IFFT	$2Cn^2 \log n\ (S \cdot f + f' \cdot f + f' \cdot S)$	$2Cn^2 \log n\ (S \cdot f + f' \cdot S)$	$2Cn^2 \log n\ (f' \cdot f)$	$f' \cdot f \cdot k^2$	$S \cdot f \cdot n^2$	$n \cdot (n+1) \cdot (S \cdot f + f' \cdot f + f' \cdot S)$
	MULT	$4n^2 \cdot S \cdot f' \cdot f$	$4n^2 \cdot S \cdot f' \cdot f$	$4n^2 \cdot S \cdot f' \cdot f$			
Proposed	FFT/IFFT	$2Cn^2 \log n\ (S \cdot f)$ *(Only at the 1st and last layer)*	-	-	$f' \cdot f \cdot k^2$	$S \cdot f \cdot n^2$	-
	Interpolation/ reduction	$3/2 \cdot n^2 \cdot k^2 \cdot f' \cdot f$	$3/2 \cdot n^2 \cdot k^2 \cdot f' \cdot f$ *(only when $n \neq k$)*	$3/2 \cdot n^2 \cdot k^2 \cdot f' \cdot f$ *(only when $n \neq a \cdot k$)*			
	MULT	$(3/2 \cdot (n^2-\alpha)+\alpha) \cdot S \cdot f' \cdot f$	$(3/2 \cdot (n^2-\alpha)+\alpha) \cdot S \cdot f' \cdot f$	$(3/2 \cdot (k^2-\alpha)+\alpha) \cdot S \cdot f' \cdot f$			

Table 1. Computation and memory demand for S minibatch, kxk kernel with f 'depth, nxn input feature with f depth.

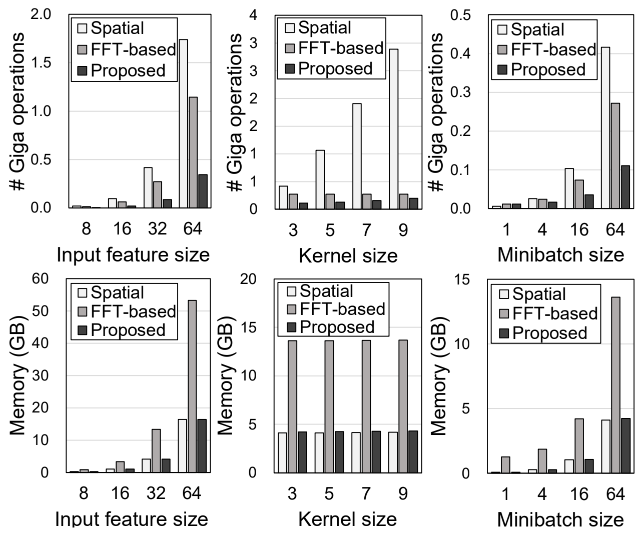


Figure 9. Computation and memory demand for a given convolution layer. Default values: input feature size=32x32 and depth=16, kernel size=3x3 and depth=16, and minibatch=64.

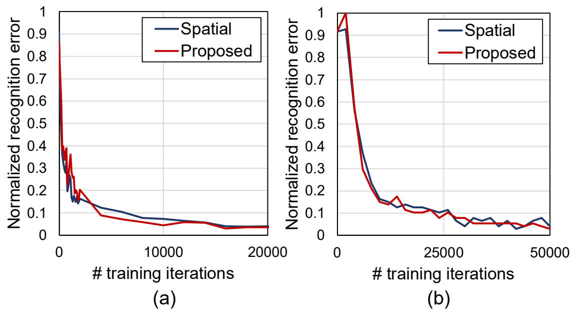


Figure 10. Normalized recognition error vs. number of iteration for (a) MNIST and (b) CIFAR-10 dataset.

training [11]. This motivates us to focus on the central region of the function, which can be approximated by linear slope as shown in Figure 8. In the frequency domain, the linear mapping is easily realized by adding scalar to DC value (shift) or multiplying scalar to all the components (scaling). To validate the approximation, we apply the original and approximate functions to one of the first convolution layer feature maps of a trained LeNet-5 network for CIFAR-10. Figure 8 shows that the distribution after the approximate functions fairly match with the ones with the original functions.

4. Algorithmic Analysis

4.1 Computation and Memory

The complexity of a given convolution layer in terms of the number of multiplication and memory requirement is summarized in Table 1. By reducing both the number of Fourier transforms and the number of multiplications for convolution, the proposed approach requires least computation among the three approaches at any size of input feature, kernel, and minibatch [Figure 9]. The advantage of the proposed method increases as the input image gets larger. Also, its complexity remains almost same regardless of the kernel size since the kernels are zero-padded to be the same size as the feature maps. Therefore, the proposed method motivates the use of large images and kernels in the network. Also note that the proposed approach demands the same memory space as the spatial approach. By eliminating the need for the Fourier

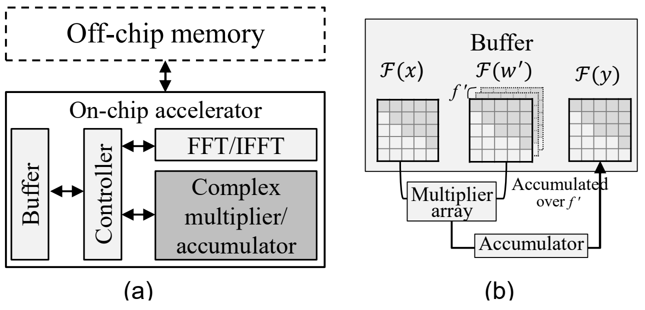
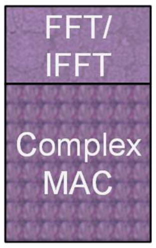


Figure 11. (a) Proposed accelerator design. (b) Multiplier and accumulator process. Bit width: 32-bit real/imaginary. Off-chip memory: DDR3 (70 pJ/bit access E, 12 ns/64 bit latency)



(a)

H/W	Resources	FFT/ IFFT	Complex MAC	Total
ASIC	Area (mm^2)	0.49	1	1.49
	Power (mW)	148	311	459
FPGA	Flip-flops	8,192	23,040	31,232
	Multipliers	176	360	536
	Adders	48	216	264
	Power (mW)	219.2	921.6	1140.8

(b)

Figure 12. (a) ASIC layout and (b) ASIC/FPGA area, power, and utilization of the proposed accelerator. (Operating frequency of ASIC: 500MHz, FPGA: 100 MHz)

transforms, it does not require additional memory for storing the frequency-domain parameters for their reuse.

4.2 Recognition Accuracy

The proposed approach approximates the spatial-domain CNN model in two ways. First, we use the spectral pooling and activation functions, which are approximated using linearity of FFT. Also, the feature map dimension reduces differently from the spatial domain CNN, as discussed in section 3.1. In order to evaluate the effect of these approximations, we compare the recognition accuracy of the LeNet-5 network with MNIST [12] and CIFAR-10 [13] datasets. As Figure 10 shows, the proposed approach achieves similar accuracy trend over training iterations as the original spatial-domain approach. While having the similar number of iterations, we reduce total training time and energy with less demand in computation and memory access per iteration.

5. Hardware Accelerator Design

5.1 Hardware Implementation

For training energy and latency analysis, we implement the proposed accelerator in a hardware [Figure 11(a)]. The 2-D FFT/IFFT module is created based on an array of 2-point 1-D FFT/IFFT implementation presented in [14]. For parallel multiplications of complex numbers, we compose an array of 72 complex-number multipliers. As Figure 11(b) shows, the computation engine is designed to accumulate each output feature plane after element-wise multiplication of an input feature and a kernel. For comparison with the existing approaches, we also design a spatial-domain convolution module using a systolic array of MAC (multiply-and-accumulator) units as presented in [15]. The array includes 120 real-number MAC units, which occupies the same area (1mm^2) as the array of 72 complex multipliers. The design is synthesized with 28nm CMOS technology for energy and latency analysis [Figure 12(a)], as well as in an FPGA to validate the FPGA compatibility [Figure 12(b)].

Network	Approach	Computation (GOPs)		Memory (MB)
		FFT/ IFFT	Conv (Mult)	
LeNet-5	Spatial	-	321.8	1.6
	FFT-based	33.7	113.1	9.1
	Proposed	**4.0**	**45.0**	**1.6**
AlexNet	Spatial	-	1,028,916	534
	FFT-based	28,223	317,316	4,594
	Proposed	**1,621**	**128,030**	**534**

Table 2. Computation/memory demand for one-epoch (128 minibatch) training of LeNet-5 and AlexNet.

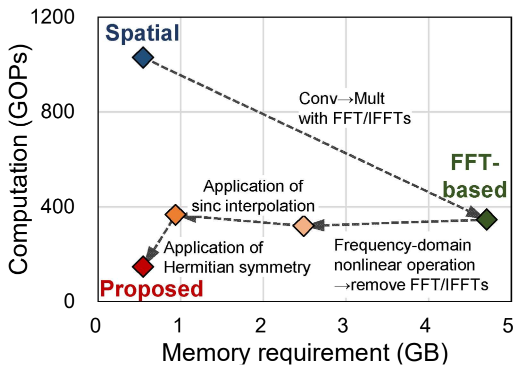


Figure 13. Computation/memory reduction of AlexNet training through the proposed techniques in this work.

5.2 Energy and Latency Analysis

Table 2 shows the hardware cost of the three training approaches when applied to train the LeNet-5 and AlexNet networks for one-epoch with minibatch of 128. Compared to the spatial-domain training, the proposed approach significantly reduces the computational complexity while maintaining the same memory requirement. Figure 13 illustrates how we reduce the hardware demand by applying various techniques.

This reduction translates into 2X lower latency and 2.5X lower energy for LeNet-5 training [Figure 14]. Although the FFT-based convolution approach also reduces computation overhead of convolution, it requires a number of Fourier transforms that demand large amount of latency and energy. Furthermore, its large memory requirement makes it consume more time and energy to access the data from off-chip memory. As Figure 15 shows, the proposed approach achieves more reduction (4.7X latency and 6.3X energy) in case of AlexNet, which has larger input image and kernels. Note that the proposed approach can be used to accelerate inference process as well. For AlexNet, the proposed accelerator requires 4.5X and 6.0X lower latency and energy than the spatial-domain convolution hardware.

6. CONCLUSIONS

Computational complexity is a major challenge in training CNNs using mobile platforms with limited resources. In response, we have designed an accelerator that trains the CNN using efficient frequency-domain computation. By mapping all the parameters and operations into the frequency domain, we perform convolution using simple pointwise multiplications without the Fourier transforms. We maintain the memory requirement same as the conventional approach by exploiting the FFT properties such as sinc interpolation and conjugate symmetry. With the reduced computational and memory overhead, it significantly reduces time

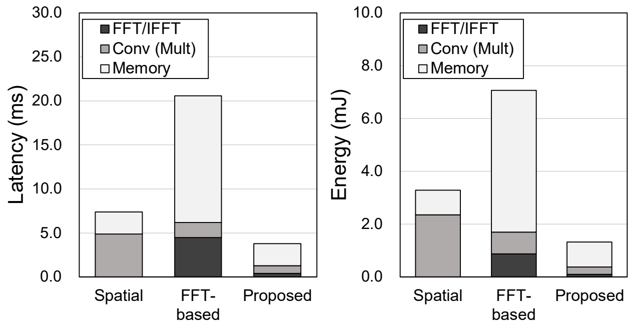


Figure 14. Latency and energy required for one-epoch (128 minibatch) training of LeNet-5.

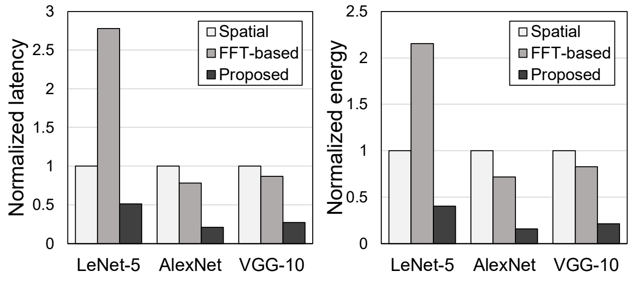


Figure 15. Latency and energy normalized to the spatial-domain approach.

and energy for both training and inference. The future work needs to consider integration of the precision control of the complex-number computation, as well as GPU-based implementation of the approach.

7. ACKNOWLEDGMENTS

This work was supported by Office of Naval Research Young Investigator Program (2012) and National Science Foundation Career Award (CNS# 1054429).

8. REFERENCES

[1] A. Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks," *NIPS '12*, pp. 1–9, 2012.
[2] F. Schroff et al., "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *CVPR '15*.
[3] P. Wang et al., "Accelerating Convolutional Neural Networks for Mobile Applications," *MM '16*.
[4] Jaderberg et al., "Speeding up Convolutional Neural Networks with Low Rank Expansions," *arXiv Prepr. arXiv1405.3866*.
[5] M. Mathieu et al., "Fast Training of Convolutional Networks through FFTs," *ICLR '14*.
[6] N. Vasilache et al., "Fast Convolutional Nets With fbfft: A GPU Performance Evaluation," *ICLR '15*.
[7] R. Bracewell, *The fourier transform and its application,* 1965.
[8] M. Charbit, *Digital Signal and Image Processing Using MATLAB*. John Wiley & Sons, 2010.
[9] R. G. Lyons, *Understanding Digital Signal Processing*, 1997.
[10] O. Rippel et al., "Spectral Representations for Convolutional Neural Networks," *NIPS '15*.
[11] S. Ioffe et al., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167*, 2015.
[12] "MNIST database.", http://yann.lecun.com/exdb/mnist/.
[13] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," *University of Toronto*, 2009.
[14] P. Milder et al., *Computer Generation of Hardware for Linear Digital Signal Processing Transforms*. 2012.
[15] T. Na et al., "Speeding Up Convolutional Neural Network Training with Dynamic Precision Scaling and Flexible Multiplier-Accumulator," *ISLPED 2016*.