Low-Energy Convolutional Neural Networks (CNNs) using Hadamard Method

Varun Mannam
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN, USA
vmannam@nd.edu

Abstract—The growing demand for the internet of things (IoT) devices has led to the implementation of computer vision tasks, such as object recognition, in low-power devices. Convolutional neural networks (CNNs) are potential solutions for efficient object recognition and detection. However, the convolutional layer in CNNs consumes a significant amount of energy compared to other layers. To mitigate this problem, a new approach based on the Hadamard transformation is proposed as an alternative to the convolution operation. This approach is demonstrated using two fundamental datasets, MNIST and CIFAR10. The mathematical expression of the Hadamard method shows potential to save energy consumption compared to convolutional layers. The method is implemented on various tasks, such as data encryption, compression, and pattern recognition, to extract principal components and feature extraction. In addition, the Hadamard method uses Hadamard transforms for faster implementation. A. Motivation

In CNNs, convolution takes the majority of the energy. Specifically, convolution with multiple layers consumes more than 90% of the total energy resources [3]. In [4], the energy consumption in 16-bit floating-point (FP) addition is 0.45 pJ, whereas 16-bit FP multiplication is 1.1 pJ. Similarly, for the 32-bit FP, energy consumption for multiplication and addition are 1.0 pJ and 4.5 pJ, respectively. Table I shows the number of multiplications and additions used in convolution for an image of size $N \times N$ and kernel of size $F \times F$. These computations are enormous when the kernel size is large, proving that the current multi-layer CNNs with convolution are not feasible with low-power devices. This requirement motivates us to find an elegant solution for implementing CNNs in energy-constrained devices.

B. Related work

Different methods are proposed in literature to reduce the energy resources in the convolutional layer. Binary weights network (BWN) [7] can compute the convolution with simple...
addition and subtraction. However, this method lacks accuracy. In Hadamard input Network (HIN) [2], the network is modified to use compressed images with the Hadamard transform instead of utilizing the original images as network inputs. The main contribution in reference [2] is to use a combination of BWN and HIN called Binary weight network with Hadamard input (BWNHI) which uses less energy. Our approach extends this idea by using the Hadamard method in each convolutional layer.

The convolution operation can be performed in an orthogonal transform domain. For example, fast Fourier transform (FFT) can be used to reduce resources compared to convolution [5]. In the FFT method [4], in every convolutional layer, input and kernel are transformed to the frequency domain, element-wise multiplied, and finally converted back to the spatial domain using the inverse fast Fourier transform (IFFT). FFT domain multiplication results in the same values as performing convolution in the spatial domain. The non-linearity activation is applied in the spatial domain after convolution operation in each convolutional layer. In the FFT method, the convolution output is complex, whereas it is real in the Hadamard method. This reduces energy consumption, as shown in Table I. Additionally, in the Hadamard method, multiplications during the transformation of input and kernel using WHT are replaced by addition and subtraction, since the Hadamard coefficients are \( \pm 1 \). In all methods, the number of multiplications used in back-propagation is twice that of forwarding propagation. These results show that the Hadamard method is more energy efficient than convolution for large kernel sizes (please see Section IV-C).

### Table I: Number of computations to perform a simple convolution operation of input feature size of \( N \) with a kernel of size \( F \) using orthogonal domains [4]. Here \( N \) indicates input feature size (assuming a single channel) and \( F \) indicates kernel size.

| Method            | Multiplications | Additions |
|------------------|-----------------|-----------|
| Convolution      | \( N^2F^2 \)    | \( N^2F^2 - 1 \) |
| FFT [8]          | \( 3N^2\log N^2 + 4N^2 \) | \( 3N^2(N^2 - 1) + 2N^2 \) |
| Hadamard method  | \( N^2 \)       | \( 3N^2(N^2 - 1) \) |
| (Ours)           |                 |           |

C. Contributions

In this report, the Hadamard method is used instead of convolution to extract the features in the images. The CNNs using the Hadamard method are implemented to show the effect of reduced energy consumption. In single-layer and three-layer CNNs, the performance of the Hadamard method is compared with convolution for MNIST [9] and CIFAR10 [10] datasets. In this paper, Section II explains the Hadamard method with WHT. The CNNs architecture with the Hadamard method is given in Section III. Section IV discusses the simulation results of CNNs with the Hadamard method and convolution on MNIST and CIFAR10 datasets including energy saving for single channel and multi-channel features with various kernel sizes. Conclusions and future work are given in Section V.

II. METHODOLOGY

Hadamard coefficients are arranged either in ascending order or by using the Walsh Hadamard sequence [6]. The first representation of the Hadamard kernel is convenient for representing the sequence order, whereas the second approach is simple for formalization. The Hadamard transform has the recursive property for any order (i.e., any order in which Hadamard transform coefficients are derived from its previous sequence of Hadamard transform coefficients) and is given by

\[
H(2N) = \begin{bmatrix}
H(N) & H(N) \\
H(N) & -H(N)
\end{bmatrix}
\]

(1)

where \( H(2) \) is given by \( H(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \). The simplified version of WHT and IWHT of input image \( X \) and Hadamard transform output \( Y \) is given by

\[
Y = HXH^T
\]

(2)

\[
X = \frac{1}{N^2}YH^TH
\]

(3)

where \( H \) and \( H^T \) are \( N \times N \) Hadamard matrices and \( N \) is the width and height of the image. The IWHT is similar to WHT except for the scaling factor. In the Hadamard method, we know image indices that are multiplied with kernel indices. To simplify further, using WHT, the convolution of image \( x \) and kernel \( h \) using the Hadamard method is given by

\[
x \ast h[k, l] = \sum_{m} \sum_{p} x[m, p]h[k \oplus m, l \oplus p]
\]

(4)

where each index \((k, l, m, p)\) range from 0 to \( 2^n - 1 \), \( n \) number of bits required to represent each indices, \( \ast \) indicates the Hadamard method convolution and \( \oplus \) represents the binary XOR operation [11]. In the same way, the convolution of image \( x \) and kernel \( h \) is given by

\[
x \ast h[k, l] = \sum_{m} \sum_{p} x[m, p]h[k - m, l - p]
\]

(5)

where \( m, p \) are the dimensions of image, \( k, l \) are the dimensions of the convolution output and \( \ast \) indicates convolution. Here the kernel dimensions are much smaller than the image dimension, and the multiplication will happen on the small image section. The sum is over all the tuples of \((m, p)\) that lead to valid subscripts for \( x[m, p] \) and \( h[k - m, l - p] \). Otherwise, those values are considered zeros in the summation. From Eq. 4 and Eq. 5, the two results of the convolution and the Hadamard method are different. Still, their structure is similar in terms of the multiplication coefficients. This similarity suggests that we can extract the features of the images with the Hadamard method, similar to convolution in CNNs.

III. NETWORK ARCHITECTURE

In supervised learning with CNN, kernel coefficients are learned using a training dataset to extract the features from images. In our experiment, we used multi-layer CNNs, which contain convolutional layers followed by a fully connected layer. Fig.1 shows a CNN with a single convolutional layer.
followed by a fully connected layer. Here convolutional layer contains convolution and non-linear activation, followed by the pooling layer. In our approach, we replaced the convolution with the Hadamard method in the convolution operation in the convolutional layer. In each convolutional layer, many kernels (number of features) extract different features from the input image. In the Hadamard method, an input image of size \( C \times H \times W \) and a kernel are transformed using Eq. 2 and then element-wise multiplied. In Eq. 2, we also check for the criteria that input image and kernel size are powers of 2; if not, zeros are appended at the end. The result is transformed to the spatial domain using Eq. 3. The output of the Hadamard method is given to a rectified linear unit (ReLU) non-linear activation to extract all features. The output of non-linear activation is given to the pooling layer to extract the maximum/average value over the cluster of neurons. Lastly, the output from the pooling layer is flattened and then given to the fully connected layer to classify the image among the given classes. The output from the fully connected layer consists of the scores of each class for a given image, and the maximum scored class is considered the predicted class. The Hadamard method is extended to multi-layer CNNs to check the performance compared to convolution. A simple illustration of the Hadamard method in a convolutional layer is shown in Figure 2.

IV. SIMULATION RESULTS

In this work, we consider four configurations:
- MNIST dataset with single-layer CNN
- CIFAR10 dataset with single-layer CNN
- MNIST dataset with three-layer CNN
- CIFAR10 dataset with three-layer CNN

For three-layer CNN architecture, three convolutional layers are cascaded. Simulation results show the performance of the Hadamard method for feature extraction with these datasets. We compare the test accuracy of the Hadamard method and convolution.

The hyper-parameters are tuned to maximize the training accuracy (i.e., minimize the error between the predicted class and the labeled class). In our approach, the following parameters are tuned: batch size (BS), initial learning rate (ILR), weight decay (WD), and the number of output features of each convolutional layer. Additionally, we use “Adam” (adaptive moment estimation) optimizer and “ReLU” activation function. Since our task is image classification, we use cross-entropy as the loss function. We use adaptive learning rate (reduce learning rate by 10 when the training loss is not decreased by a certain threshold over the last few epochs). At each convolutional layer, the input is batch-normalized before passing through the convolution block. In our network, the model parameters (like kernel coefficients and fully connected layer weights and bias) are initialized from a random sample set of a normal distribution, with zero mean and unit variance. Finally, we provide the results with the kernel sizes of \( 3 \times 3 \), \( 5 \times 5 \), and \( 7 \times 7 \). For the energy calculations, we showed our results with larger kernel sizes.

In our simulations, we tune the hyper-parameters for the kernel size \( 3 \times 3 \), and we use these hyper-parameters for all the kernels in each configuration. The tuned hyper-parameters for all configurations are given in Table II for the MNIST and CIFAR-10 datasets.

| Dataset | CNN layers | H-ILR | H-WD | C-ILR | C-WD |
|---------|------------|-------|------|-------|------|
| MNIST   | 1          | 1e-4  | 1e-4 | 1e-4  | 1e-4 |
|         | 3          | 1e-4  | 1e-4 | 1e-4  | 1e-4 |
| CIFAR10 | 1          | 1e-3  | 1e-4 | 1e-3  | 1e-4 |
|         | 3          | 2e-3  | 1e-4 | 1e-3  | 1e-4 |

Table II: Hyper-parameters for all configurations, where CNN networks with 1: single-layer CNN, 3: three-layers CNN, H-ILR: Hadamard method ILR, H-WD: Hadamard method WD, C-ILR: convolution ILR and C-WD: convolution WD.
A. MNIST

MNIST is a dataset of hand-written digits from 0 to 9 grayscale (with a single channel as input) images of size 28 × 28 (W × H) with 60000 training dataset and 10000 test dataset [9].

1) Single-layer CNN: To match the number of parameters in each method, we disabled the bias in the convolution. The single-layer CNN parameters are kernel coefficients of the convolutional layer and weights and biases of the fully connected layer.

| CNN layers | Kernel Size | Hadamard method (Ours) | Convolution method |
|------------|-------------|------------------------|-------------------|
| 1          | 3 x 3       | 97.50                  | 98.34             |
|            | 5 x 5       | 97.95                  | 98.67             |
|            | 7 x 7       | 97.97                  | 98.83             |
| 3          | 3 x 3       | 97.91                  | 99.27             |
|            | 5 x 5       | 98.28                  | 99.37             |
|            | 7 x 7       | 98.43                  | 99.38             |

Table III: MNIST test dataset accuracy (%) for single and three-layer CNNs using convolution and Hadamard methods. CNN layers 1: single-layer CNN and 3: three-layers CNN.

The results show that the test accuracy improved with an increase in kernel size due to more parameters being used to extract the features. The test accuracy in single-layer CNN for all kernels is given in Table III.

![Figure 3: Test accuracy of the MNIST dataset vs. epochs for three-layer CNN using the Hadamard and convolution methods.](image)

2) Three-layer CNNs: To improve network accuracy, we introduce more convolutional layers, which increase the number of parameters and extract more complex features at each layer compared to the previous layer. The test accuracy in three-layer CNN for all kernels is given in Table III, and we observe that when the number of convolutional layers increases, the test accuracy difference between Hadamard transform and convolution also increases. In this way, we show using the Hadamard method, with fewer computations, we got test-dataset accuracy similar to the traditional convolutional layers.

B. CIFAR10

CIFAR10 dataset contains ten different classification images such as plane, cat, dog, etc. with three input channels (C=3 represent R, G, B channels) of size 32×32 (W × H). CIFAR10 dataset has 50000 training images, and 10000 test images [10]. The initial convolutional layer has kernels of size output features × input channels × kernel (i.e., out features × 3 × kernel). At the initial convolutional layer, each input channel is the convolution method (or convolution operation) with the same channel kernel, and all input channels are added together. This dataset uses a batch size (BS) of 20. When the learning rate is increased, the present model will overfit. Incorporating the batch-norm layers in the current architecture before the non-linear activation function improves the test accuracy in the CIFAR10 dataset (both in single-layer and three-layer) [12]. Adding batch norm after convolution method or Hadamard method leverages the training process to use a high learning rate that yields faster convergence. With a high learning rate, there is always an over-fitting issue and batch-norm works as a regularization technique. Adding dropout along with batch-norm at each layer provides better regularization. Dropout is another regularization method that randomly removes a few nodes/input channels with a certain probability over each batch size so that the model will not overfit. For CIFAR10, our network architecture is Hadamard method or convolution method, batch-normalization layers, and ReLU layer followed by a max-pool layer with dropout.

1) Single-layer CNN: The tuned hyper-parameters are given in Table II. The dropout value in our CIFAR10 dataset CNN model is set to 0.2 and 0.3 for Hadamard and convolution methods, respectively. The test accuracy using the CIFAR-10 dataset in single-layer CNN for all kernels is given in Table IV.

| CNN layers | Kernel Size | Hadamard method (Ours) | Convolution method |
|------------|-------------|------------------------|-------------------|
| 1          | 3 x 3       | 60.25                  | 64.06             |
|            | 5 x 5       | 61.43                  | 65.89             |
|            | 7 x 7       | 61.21                  | 66.08             |
| 3          | 3 x 3       | 73.48                  | 82.00             |
|            | 5 x 5       | 72.65                  | 81.97             |
|            | 7 x 7       | 71.08                  | 81.19             |

Table IV: CIFAR-10 test dataset accuracy (%) for single and three-layer CNNs using convolution and Hadamard methods. CNN layers 1: single-layer CNN and 3: three-layers CNN.

![Figure 4: Test accuracy of the CIFAR-10 dataset vs. epochs for three-layer CNN using the Hadamard and convolution methods.](image)

2) Three-layer CNN: To improve the test accuracy, we increase the number of features by two after every convolutional layer with the CIFAR10 dataset. This is because, after each convolutional layer, the image size is reduced by half (because of the max-pool with a kernel size of 2 × 2). To extract more complex features at each successive convolutional layer, the number of output features doubles compared to the previous layer. The number of output features is 32, 64, and 128 in
the first, second, and third convolutional layers, respectively. Further, the test accuracy can be improved by adding a second fully connected layer at the end. The ReLU activation in the first fully connected layer along with dropout is applied. In addition, in the Hadamard method and convolution method in all convolutional and fully connected layers, dropout is set to 0.2.

C. Energy results

Figure 5: Single convolutional layer computational energy saving ratio vs. single channel image size across kernels for 16-bit and 32-bit floating point multiplications.

In this section, energy-saving calculations are shown for the Hadamard method when compared to the convolution method in CNNs. For example, for a single convolutional layer with a single channel image of size $N \times N$ with a kernel size of $F \times F$, then the number of multiplications and additions required to generate convolution output is given in Table. I. Consider, energy consumption for multiplication and addition are $E_m$ and $E_a$, then total energy consumption in convolution method is $N^2 \times F^2 \times E_m + N^2 \times (F^2 - 1) \times E_a$. Similarly, for the Hadamard method, the energy consumption is $N^2 \times E_m + 3N^2 \times (N^2 - 1) \times E_a$. The energy saving ratio is defined as the ratio of the energy consumption of the convolution method to the energy consumption of the Hadamard method. The energy ratio is $\frac{F^2 \times (\alpha + (F^2 - 1))}{\alpha + 3(N^2 - 1)}$ where $\alpha$ is the ratio of $E_m$ and $E_a$. For 16-bit and 32-bit FP, the $\alpha$ values are 2.44 and 4.5, respectively. To show the energy-saving ratio for a single channel image, we perform a simulation across different image and kernel sizes which is shown in Figure. 5. Here considered both 16-bit floating point (FP) and 32-bit floating point multiplication energy values. From the simulation, for the baseline configuration where the threshold is set to 1, the image is less than 128 for the typical kernels of $3 \times 3$ or $5 \times 5$ which is typically the same as most image patches in size.

However, most of the intermediate layers in CNNs are multi-channel features with the number of channels being $C_{in}$. The energy saving for multi-channel inputs is given by $\frac{F^2 \times (\alpha + 1) C_{in}}{\alpha + 2(\log N)^2 + (\log N)^2}$. The simulation result with various image sizes and kernel sizes is given in Figure. 6. In addition, various $C_{in}$ starting from 2 to 5 with floating point multiplications are shown with a baseline energy saving of 1. From Figure. 6, an image of size less than 128 provides energy saving using the Hadamard method across all kernels.

D. Results discussion

For kernel of $5 \times 5$, from the MNIST dataset, there is a 0.71% gap in test accuracy between Hadamard and convolution methods in single-layer CNN, whereas the gap in accuracy increases to 1.09% with three-layer CNN. MNIST is a simple dataset with grayscale images in which most of the features can be extracted using the Hadamard method. In contrast, the CIFAR-10 dataset is complex since 3-channel input images and for the CIFAR10 dataset, in single-layer CNN, there is a gap of 4.46% in test accuracy between the Hadamard and convolution methods. This gap increases to 9.32% for three-layer CNN between the methods. In the CIFAR10 dataset, the test accuracy gap is more due to image content and complex features. With our present architecture, the gap in test accuracy with three-layer CNNs between the Hadamard method and convolution methods can be reduced with proper hyperparameters. The main objective of the proposed method is to minimize computations to achieve the classification task on low-energy devices, such that the entire object classification task can be fit to low-power real-time devices such as mobiles. One exciting application is, proposed simple object detection networks perform inference using the Hadamard method where the multiplication operations are substantially reduced.

V. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In CNNs, convolution is energy expensive, and we identified an alternative method for feature extraction, namely the Hadamard method. We implemented the multi-layer CNNs
with the Hadamard method and verified them with MNIST and CIFAR10 datasets. We observed that the MNIST dataset’s Hadamard transform in CNN achieves performance similar to convolution (test accuracy difference is around 1%). However, in the CIFAR10 dataset, the Hadamard method underperforms convolution (test accuracy difference is 9.32%) since the dataset is complex. Initial results show that using the Hadamard method, we can extract the features in the images using few energy resources with reduced performance.

B. Future work

The test accuracy can be improved in both datasets with operations (adjusting the positions of the coefficients) on the Hadamard method output such that the difference between Hadamard output and convolution is slight and by extending this idea to multi-layer CNNs by comparing the performance of the Hadamard method and convolution methods. Similar to Winograd CNNs [13], performing the Hadamard method without converting back to the spatial domain after each convolutional layer can lead to interesting observations.

REFERENCES

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. The MIT Press.
[2] Deveci, T. Ceren, Serdar Cakir, and A. Enis Cetin. “Energy Efficient Hadamard Neural Networks.” arXiv preprint arXiv:1805.05421 (2018).
[3] T. Abtahi, C. Shea, A. Kulkarni and T. Mohsenin, “Accelerating Convolutional Neural Network With FFT on Embedded Hardware,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 9, pp. 1737-1749, Sept. 2018.
[4] Vasilyev, Artem. “CNN optimizations for embedded systems and FFT.” (2015).
[5] Ušáková, Anna, Jana Kotuliaková, and Michal Zajac. “Using of Discrete Orthogonal Transforms for Convolution.” Journal of Electrical Engineering 53, no. 9-10 (2002): 285-288.
[6] Moiz, Saifuddin. “Fast implementation of Hadamard transform for object recognition and classification using parallel processor.” Ph.D. diss., Ohio University, 1991.
[7] Rastegari, Mohammad, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. “Xnor-net: Imagenet classification using binary convolutional neural networks.” In European Conference on Computer Vision, pp. 525-542. Springer, Cham, 2016.
[8] M. Mathieu, M. Henaff and Y. LeCun. “Fast training of convolutional networks through ffts”. CoRR, 2014.
[9] LeCun, Yann and Cortes, Corinna. “MNIST handwritten digit database.” (2010).
[10] Alex Krizhevsky and Vinod Nair and Geoffrey Hinton. “CIFAR-10 (Canadian Institute for Advanced Research)”. Online: https://math.stackexchange.com/questions/577491/convolution-theorem-for-other-transforms/.
[11] Ioffe, Sergey, and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” arXiv preprint arXiv:1502.03167 (2015).
[12] Liu, Xingyu, Jeff Pool, Song Han, and William J. Dally. “Efficient sparse-Winograd convolutional neural networks.” arXiv preprint arXiv:1802.06367 (2018).
[13] Mannam, Varun, and Arman Kazemi. “Performance analysis of semi-supervised learning in the small-data regime using VAEs.” arXiv preprint arXiv:2002.12164 (2020).