# Exploiting Spatial Correlation in Convolutional Neural Networks for Activation Value Prediction

Gil Shomron Uri Weiser
Electrical Engineering
Technion — Israel Institute of Technology
{gilsho@tx, uri.weiser@ee}.technion.ac.il



## ABSTRACT

Convolutional neural networks (CNNs) compute their output using weighted-sums of adjacent input elements. This method enables CNNs to achieve state-of-the-art results in a wide range of applications such as computer vision and speech recognition. However, it also comes with the cost of high computational intensity.

In this paper we propose to exploit the *spatial correlation* inherent in CNNs, and use it for *value prediction*. We show that spatial correlation may be exploited to predict activation values, thus reducing the needed computations in the network. We demonstrate this method with a heuristic that predicts which activations are zero-valued according to nearby activation values, in a scheme we call *cross-neuron prediction*. Our prediction heuristic reduces the number of multiply-accumulate operations by an average of 40.8%, 36.2%, and 20.8%, with degradation in top-5 accuracy of 2.9%, 5.1%, and 7.6%, for AlexNet, VGG-16, and ResNet-18, respectively.

## 1. INTRODUCTION

Convolutional neural networks (CNNs) are a widely used form of deep neural networks (DNNs). They operate under the premise that the network's input exhibits the property of *spatial correlation*, which is why they are thriving as networks for image classification [1, 2, 3]. CNNs are mainly comprised of convolutional (CONV) layers with a few (normally 1-3) fully connected layers at the end. Each convolution operation within a CONV layer is usually followed by the Rectifying Linear Unit (ReLU) [4] activation function, which is defined as $f(x) = \max(0,x)$. Almost all the multiply-accumulate operations (MACs) within a CNN take place in the CONV layers, e.g., 92% in AlexNet [1] and 99% in VGG-16 [2].

To better understand and improve CNNs, ongoing research attempts to unravel what the hidden layers actually learned during the training process [5][6]. Visualization of the hidden layers unveils strong patterns of spatial correlation. Figure 1 presents a few examples we extracted from AlexNet CONV3 after the ReLU activation; hot colors represent higher values, whereas cool colors represent lower values (e.g., blue pixels are zeros). We observe that similar values are distributed spatially.

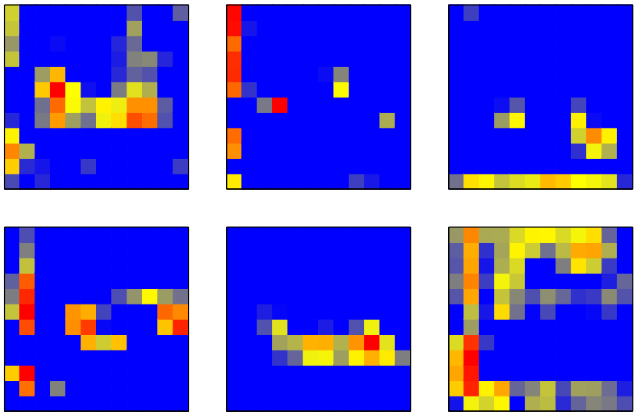

**Figure 1: Examples of AlexNet CONV3 output feature map (13x13) values after ReLU for six arbitrary filter channels. Hot colors represent higher values whereas cool colors represent lower values (e.g., blue pixels are zeros).**

To reduce the computational intensity of CNNs, we propose to exploit this inherent spatial correlation for *value prediction*. We suggest a *cross-neuron prediction* scheme, where activation values are based on already-calculated adjacent activations.

Since the ReLU function cuts off the activation values at zero, we demonstrate our method with a heuristic that predicts zero-valued activations, thereby decreasing the number of MAC operations. Our prediction heuristic achieves a 40.8% reduction in the number of MAC operations, with 4.0% top-1 and 2.9% top-5 accuracy degradation in AlexNet [1]; a 36.2% reduction in the number of MAC operations, with 8.4% top-1 and 5.1% top-5 accuracy degradation in VGG-16 [2]; and a 20.8% reduction in the number of MAC operations, with 11.1% top-1 and 7.6% top-5 accuracy degradation in ResNet-18 [3].

Our contributions in this paper are as follows:

- We quantify the amount of spatial correlation in state-of-the-art CNNs.
- We demonstrate prediction of zero-valued activations using a simple heuristic that exploits the spatial cor-

relation in CNNs, thereby reducing the computational intensity.

The remainder of this paper is organized as follows: In Section 2 we provide motivation for our ideas by presenting spatial correlation measurements of AlexNet, VGG-16, and ResNet-18 and then introduce our prediction method. Section 3 evaluates the prediction performance and its influence on network accuracy. Section 4 presents related work. We conclude in Section 5 and discuss future research trajectories.

## 2. EXPLOITING SPATIAL CORRELATION

To solidify our approach of exploiting spatial correlation for value prediction, we first quantify the amount of spatial correlation in state-of-the-art CNN architectures. Then, we introduce a heuristic for prediction, which we evaluate in Section 3.

### 2.1 Assessing Potential

CNNs operate on adjacent elements to extract unique feature maps. With CNNs for image classification, the network's input comprises correlated pixels that together form an image. A certain degree of spatial correlation has already been observed. However, it must be quantified to understand whether it is worth exploiting it for prediction.

We focus on spatial correlation of zero-valued activations, since it is the most straightforward to measure and later predict. A 2D sliding window (with a stride of 1) is used to measure the degree of spatial correlation. We count the number of times all activations are zero-valued. Since we assume no cross-channel correlation, we measure spatial correlation per channel. Large windows filled with zeros indicate a high degree of spatial correlation, whereas small windows filled with non-zeros indicate a small degree of spatial correlation.

First, sparsity is measured by setting the window size to 1x1 — at this stage, spatial correlation is not a consideration. We evaluate 3 state-of-the-art models — AlexNet [1], VGG-16 [2], and ResNet-18 [3], and we use ILSVRC-2012 [7] as our validation dataset throughout this paper. Figure 2 illustrates the average sparsity results (1x1 bars) for these models. We observe that 73%, 60%, and 41% of activation are zero-valued for AlexNet, VGG-16 and ResNet-18, respectively [8, 9, 10]. The question now is whether those zeros are adjacent.

By increasing the window size we observe that the neighbors of zero-valued activations are also zero-valued. It is especially noticeable in AlexNet and VGG-16, e.g., 40% and 31% of the 3x3 windows are filled with zero-valued activations.

### 2.2 CNNs Explained

CNNs are comprised of CONV layers. Each CONV layer extracts and refines feature maps that originate from the network's input. The basic operation in CONV layers is 2D convolution between the input feature maps (ifmaps) and the corresponding filters, to construct output feature maps (ofmaps). The CONV layer input comprises a set of ifmaps, each called a channel. Therefore, the ifmap dimensions are 3D (height, width, channels). The ifmap-filter convolution is a per channel operation which forms a single 2D ofmap.

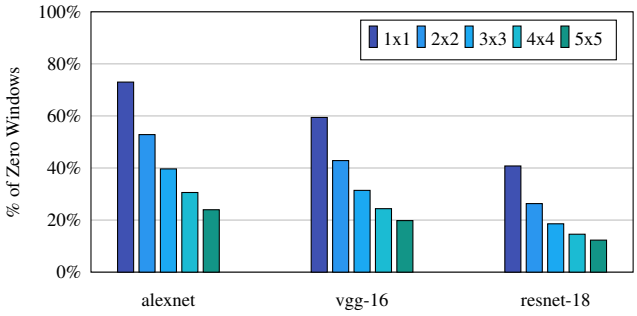


**Figure 2: Spatial correlation measurements of AlexNet, VGG-16, and ResNet-18.**

However, the ofmap itself is multi-channel. Therefore the ifmap-filter convolution is repeated using a different 3D filter. The filters thus have a 4D structure (height, width, ifmap channels, ofmap channels). In addition, a batch of ifmaps may be processed together, which makes the ifmap and ofmap 4D as well (width, height, channels, batch size).

A CONV ofmap is formulated as follows:

$$O(n,c) = \sum_{q=0}^{F_C-1} I(n,q) * F(c,q) + B(c),$$

where $O$, $I$, $F$, $F_C$, and $B$, are the ofmap, ifmap, filter, the number of filter channels, and the bias, respectively. The arguments obey the following conditions: $0 \leq n <$ batch size, and $0 \leq c <$ ofmap channels $=$ filters channels. The convolution ($*$) result is a $O_W \times O_H$ matrix, and the dimensions of the ofmap $O$ are also $O_W \times O_H$. The convolution is defined as:

$$(I_{n,q} * F_{c,q})(x,y) = \sum_{i=0}^{F_W-1} \sum_{j=0}^{F_H-1} I(Sx+i, Sy+j) \times F(i,j),$$

where $F_W \times F_H$ are the filter dimensions, and $S$ is the filter stride. This convolution is calculated per input in the batch, per channel in the ofmap, and for each $x$ and $y$ that obey $0 \leq x < O_W = (I_W - F_W + S)/S$, and $0 \leq y < O_H = (I_H - F_H + S)/S$. $I_W$ is the ifmap width, and $I_H$ is the ifmap height. We illustrate the convolution process in Figure 3.

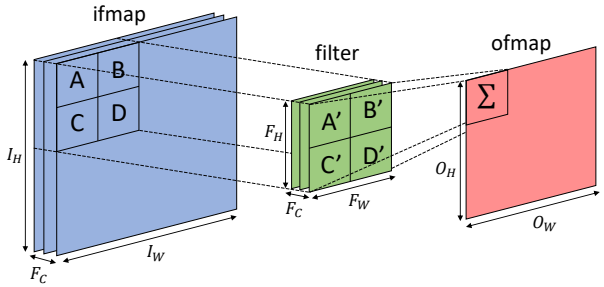


**Figure 3: Illustration of a single convolution with a single filter. This process is repeated for the entire ofmap and for each filter. Eventually, the ofmap depth is equal to the number of filters.**

## 2.3 Cross-Neuron Prediction

Adjacent activations are correlated and, specifically, zero-valued activations, as described in the previous subsection. This characteristic may be exploited for value prediction, thereby decreasing the number of required MAC operations. We propose a cross-neuron prediction method by which activation values are predicted according to nearby activation values.

We demonstrate this method by using a simple heuristic to predict which activations are zero-valued. The ofmaps are divided into square, non-overlapping windows (i.e., with a stride that equals the height and width of the window). The ofmaps are also padded to take care of margins that may form because of the filter and ofmap dimensions. The values of activations positioned diagonally in each window are then calculated. If these activations are zero-valued, the other activations within that window are predicted to be zero-valued as well, thereby saving their MAC operations. Note that predicting these activations to be zero-valued affects the operations that need to take place between the layer's ifmap and the relevant filter. Figure 4 depicts this heuristic.

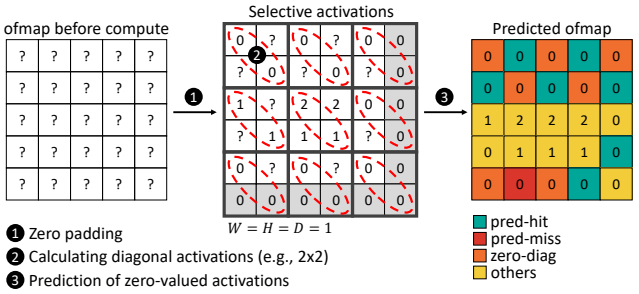


**Figure 4: Illustration of the prediction heuristic. Predicted activations save an entire convolution.**

Increasing the window size also increases the number of activations that may be predicted per window. Potentially, the number of activations that may be predicted equals $W \times H - D$, where $W$, $H$, and $D$ are the width, height, and diagonal dimensions, respectively. Since we deal with square windows, $W = H = D$, and the previous formulation turns into $k^2 - k$, where $k$ is just one of the dimensions. On the flip side, increasing the window size might (1) decrease the number of windows with a zero-valued diagonal; and (2) increase the rate of missed predictions rate, since the area around the diagonal becomes bigger. We evaluate these trade-offs in the next section.

# 3. EVALUATION

In this section we first give a quick overview of our system setup, and then present our evaluation results.

## 3.1 Methodology

We used PyTorch 0.4.0 [11] (Python 3.5.5) pre-trained models of AlexNet [1], VGG-16 [2], and ResNet-18 [3]. The models were evaluated on the entire ILSVRC-2012 [7] validation dataset (50,000 images). We simulate our prediction method by extracting the intermediate values of each hidden layer during feed-forward, altering the data according to the given window size, and pushing it back to the next layer. Statistics such as missed predictions are recorded by comparing the original feed-forward intermediate values with the predicted feed-forward values.

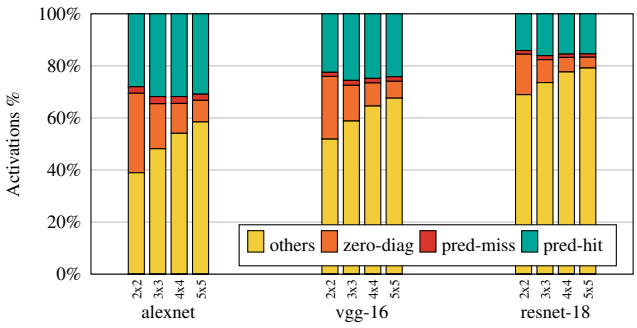


**Figure 5: Prediction simulation results breakdown of AlexNet, VGG-16, and ResNet-18.**

## 3.2 Experimental Results

Figure 5 illustrates the average performance breakdown of each of the models, for different prediction windows, into four categories:

1. `pred-hit` — zero activations that were predicted as zeros, i.e., "hit".
2. `pred-miss` — non-zero activations that were predicted as zeros, i.e., "miss".
3. `zero-diag` — the zero-diagonals on which the prediction was based.
4. `others` — the rest of the activations.

Note that the total number of predicted activations equals `pred-hit+pred-miss`, and it corresponds to `zero-diag` with $k^2 - k$ (as described in Section 2).

**Activation savings.** The trade-offs in increasing the window size are noticeable throughout our measurements. The "sweet spot" of all three models is a window size of 3x3. A larger window of 4x4 or 5x5 decreases the prediction opportunities, since fewer diagonals are equal to zero. On the other hand, a prediction using a smaller window of 2x2 incurs a relatively large prediction overhead (as compared to 3x3, 4x4, and 5x5 windows).

Overall, we are able to save 34.5%, 27.5%, and 17.6% of the CONV layer activation computations in AlexNet, VGG-16 and ResNet-18, respectively.

**MAC savings.** The required number of computations per activation depends on the CONV layer itself. For example, for AlexNet CONV1, 3x11x11 MAC operations are required to compute an activation value, whereas for AlexNet CONV4, 192x13x13 MAC operations are required to compute an activation value — obviously, the latter activation requires more computations than the former.

Figure 6 depicts the average savings results in terms of MAC operations. Our heuristic was able to save an average of 41.9%, 36.2%, and 23.6% MAC operations in AlexNet, VGG-16, and ResNet-18, respectively.

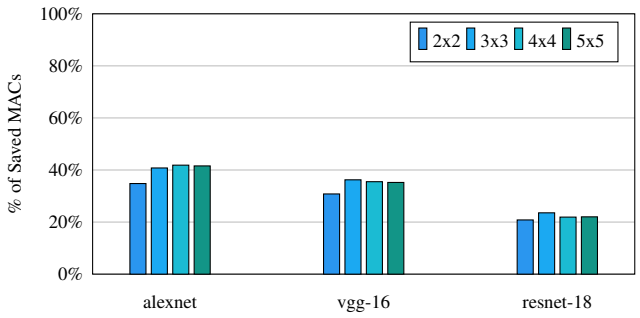


**Figure 6: MAC operations savings. The number of predicted activations (Figure 5) are transformed to MAC operations.**

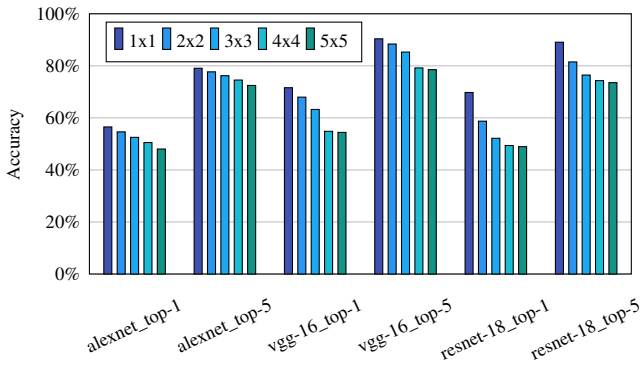


**Figure 7: Top-1 and Top-5 accuracy over ILSVRC2012 validation set for different prediction windows. 1x1 is the original network's accuracy.**

**Impact on accuracy.** Naturally, this type of prediction may accidentally zero-out non-zero activations, which will decrease the network's accuracy.

Figure 7 presents the top-1 and top-5 accuracies of the three models. As discussed earlier, increasing the window size also increases the number of activations that may be predicted, which generally decreases the accuracy. With VGG-16 and ResNet-18, the accuracy reaches a plateau because 4x4 and 5x5 windows are equally effective in these models. Interestingly, we observe that the accuracy decreases as the window sizes increases, whereas the missed prediction rates do not vary dramatically. This is probably due to the aggressiveness of the larger windows, which are more prone to zero activations with large values than are the smaller windows — it is not only *how many* activations are zeroed out, but also *which* activations are zeroed out.

The optimal results with regard to the tradeoff between prediction rate and accuracy can be summed up as follows: Using a 3x3 window, AlexNet achieves a 40.8% savings in MAC operations with 52.5% top-1 accuracy, and 76.2% top-5 accuracy (baseline is 56.5% and 79.1%); using a 3x3 window, VGG-16 achieves a 36.2% savings in MAC operations with 63.2% top-1 accuracy, and 85.3% top-5 accuracy (baseline is 71.6% and 90.4%); and using a 2x2 window, ResNet-18 achieves a 20.8% savings in MAC operations with 58.7% top-1 accuracy, and 81.5% top-5 accuracy (baseline is 69.8% and 89.1%).

## 4. RELATED WORK

**Data prediction.** Value prediction has long been proposed in the context of general purpose processors. Lipasti et al. [12] and Gabbay et al. [13] introduced load value prediction based on previously seen load values. In subsequent work, Lipasti et al. [14] suggested predicting all values produced by instructions that write to a register.

These works have led to different implementations of value predictors. Sazeides et al. [15] explored and defined two classes of value predictors: computational — the predicted value is based on operations on previous values, and context based — the predicted value is based on previous observed values. Wang et al. [16] continued to explore the two classes and presented a hybrid solution that uses both computational and context predictors. Calder et al. [17] also took into consideration the gain and the confidence of the prediction when deciding whether to predict a certain value. Nakra et al. [18] implemented a global value predictor instead of the previous local value predictors; hence, different instructions are used in the prediction process.

Recently, the notion of approximate computing has also emerged in value prediction. Miguel et al. [19] used approximations of load values with applications that can tolerate error or inexactness, thereby eliminating the rollbacks that may occur when dealing with precise predictions. In a sense, applications that use neural networks are tolerant to errors, which is exactly the reason we do not validate and rollback in the case of a prediction miss.

**Prediction and approximation.** The work most closely related to ours is SnaPEA [8]. Akhlaghi et al. propose to predict, per activation, whether the weighted-sum will be below zero, and hence zero because of the ReLU function. Their method is based on static weight reordering and an optimization algorithm that determines the degree of speculation. By doing so, they able to save MAC operations. We categorize their solution as *intra-neuron prediction* (as opposed to our cross-neuron prediction scheme), as no correlation is made between nearby activation to support the prediction decision.

Other relevant works are those related to approximations. Huan et al. [20, 21] extend the zero-valued sparsity to near-zero-valued sparsity, to avoid complex multiplications of near-zero valued data through approximation. When one of the operands (or two of them, obviously) of the multiply operations is close to 0, the multiplication may result in an output close to 0, which can be simply approximated to 0. To understand whether the multiplication result is small, a near-zero approximation unit (NZAU) is proposed. The NZAU is based on a leading-zero count (LZC) algorithm that allows the order of magnitude of the result to be deduced. A threshold is set to determine whether the result should be approximated.

Han et al. [22] introduce "deep compression", which involves three stages: pruning, quantization, and Huffman coding. First, with pruning, all connections with weights below a threshold are removed from the network. The network is then retrained to tune the remaining weights accordingly. Second, network quantization and weight sharing are used to further compress the pruned network by reducing the number of bits

required to represent each weight. Third, Huffman code is used to encode the more common symbols with fewer bits, thereby compressing the network even further.

## 5. CONCLUSIONS AND FUTURE WORK

Value prediction is a well-known technique for speculatively resolving true dependencies [12, 13]. The algorithmic characteristics of CNNs have motivated us to research new approaches that use value prediction in CNNs, with the goal of reducing the computational intensity in these types of networks. Different approaches may be used: prediction based on correlation between nearby activations, as we suggest (i.e., *cross-neuron prediction*), and prediction of independent activations, as proposed in SnaPEA [8] (i.e., *intra-neuron prediction*). We moreover believe that there are other opportunities for prediction heuristics based on correlation between layers — *cross-layer prediction*.

In this paper we exploit an inherent property of CNNs: spatially correlated activations. We choose three state-of-the-art CNN models — AlexNet, VGG-16, and ResNet-18 — and quantify the amount of spatial correlation in each. Then, we introduce a method for predicting that a group of activations will be zero-valued according to their nearby activations. By predicting zero-valued activations we reduce the number of computations. Our method reduces computation by 40.8%, 36.2%, and 20.8% on average, with degradation in top-5 accuracy of 2.9%, 5.1%, and 7.6%, for AlexNet, VGG-16, and ResNet-18, respectively.

We consider two trajectories for future work: (1) algorithmic — prediction of an entire window, in contrast to predicting some of the activations within the window; prediction within the window using an arrangement of activations other than diagonal; mitigation of accuracy losses; and emphasizing spatial correlation during the network training. (2) architecture — the performance, power, and energy implications of exploiting spatial correlation.

## 6. REFERENCES


[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Neural Information Processing Systems (NIPS)*, pp. 1097–1105, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[4] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. of the Intl. Conf. on Machine Learning (ICML)*, pp. 807–814, 2010.

[5] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, "The building blocks of interpretability," *Distill*, 2018. https://distill.pub/2018/building-blocks.

[6] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "ImageNet large scale visual recognition challenge," *Intl. Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[8] V. Aklaghi, A. Yazdanbakhsh, K. Samadi, H. Esmaeilzadeh, and R. Gupta, "SnaPEA: Predictive early activation for reducing computation in deep convolutional neural networks," in *Intl. Symp. on Computer Architecture (ISCA)*, 2018.

[9] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 52, no. 1, pp. 127–138, 2017.

[10] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *Computer Architecture News*, vol. 44, pp. 1–13, IEEE Press, 2016.

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," *Neural Information Processing Systems Workshop (NIPS-W)*, 2017.

[12] M. H. Lipasti, C. B. Wilkerson, and J. P. Shen, "Value locality and load value prediction," in *Intl. Conf. on Arch. Support for Programming Languages & Operating Systems (ASPLOS)*, 1996.

[13] F. Gabbay and A. Mendelson, "Using value prediction to increase the power of speculative execution hardware," *ACM Trans. on Computer Systems*, vol. 16, no. 3, pp. 234–270, 1998.

[14] M. H. Lipasti and J. P. Shen, "Exceeding the dataflow limit via value prediction," *Intl. Symp. on Microarchitecture (MICRO)*, pp. 226–237, 1996.

[15] Y. Sazeides and J. E. Smith, "The predictability of data values," in *Intl. Symp. on Microarchitecture (MICRO)*, pp. 248–258, IEEE Computer Society, 1997.

[16] K. Wang and M. Franklin, "Highly accurate data value prediction using hybrid predictors," in *Intl. Symp. on Microarchitecture (MICRO)*, pp. 281–290, IEEE Computer Society, 1997.

[17] B. Calder, G. Reinman, and D. M. Tullsen, "Selective value prediction," in *Computer Architecture News*, vol. 27, pp. 64–74, IEEE Computer Society, 1999.

[18] T. Nakra, R. Gupta, and M. L. Soffa, "Global context-based value prediction," in *Symp. on High-Performance Computer Architecture (HPCA)*, pp. 4–12, IEEE, 1999.

[19] J. S. Miguel, M. Badr, and N. E. Jerger, "Load value approximation," in *Intl. Symp. on Microarchitecture (MICRO)*, pp. 127–139, 2014.

[20] Y. Huan, Y. Qin, Y. You, L. Zheng, and Z. Zou, "A multiplication reduction technique with near-zero approximation for embedded learning in IoT devices," in *IEEE Intl. System-on-Chip Conf. (SOCC)*, pp. 102–107, IEEE, 2016.

[21] Y. Huan, Y. Qin, Y. You, L. Zheng, and Z. Zou, "A low-power accelerator for deep neural networks with enlarged near-zero sparsity," *arXiv preprint arXiv:1705.08009*, 2017.

[22] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.