PredictiveNet: An Energy-efficient Convolutional Neural Network via Zero Prediction

Yingyan Lin, Charbel Sakr, Yongjune Kim, and Naresh Shanbhag
Dept. of Electrical and Computer Engineering, University of Illinois at Urbana Champaign
Email: {yingyan, sakr2, yongjune, shanbhag}@illinois.edu

Abstract—Convolutional neural networks (CNNs) have gained considerable interest due to their record-breaking performance in many recognition tasks. However, the computational complexity of CNNs precludes their deployments on power-constrained embedded platforms. In this paper, we propose predictive CNN (PredictiveNet), which predicts the sparse outputs of the non-linear layers thereby bypassing a majority of computations. PredictiveNet skips a large fraction of convolutions in CNNs at runtime without modifying the CNN structure or requiring additional branch networks. Analysis supported by simulations is provided to justify the proposed technique in terms of its capability to preserve the mean square error (MSE) of the non-linear layer outputs. When applied to a CNN for handwritten digit recognition, simulation results show that PredictiveNet can reduce the computational cost by a factor of $2.9 \times$ compared to a state-of-the-art CNN, while incurring marginal accuracy degradation.

I. INTRODUCTION

Many emerging applications in pattern recognition and data mining require the use of machine learning (ML) algorithms to process massive data volumes on energy-constrained platforms [1]. Convolutional neural network (CNN) is a powerful machine learning algorithm that achieves state-of-the-art performance in various recognition tasks [2]. For example, the convolutional network, Microsoft ResNet, was able to achieve a better-than-human accuracy of 3.6% at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015 [3], which is a benchmark in object category classification and detection on hundreds of object categories and millions of images. Although CNNs have provided an efficient method to constrain its complexity by weight sharing and restricting to local connections, the required amount of computation is still large, with convolutional operations usually taking up over 90% of the total computational power [4]. For example, a state-of-the-art CNN, AlexNet, requires 606 million MACs per $227 \times 227$ image ($13 \times$ MACs/pixel) [4]. This high computational complexity of CNNs hinders their implementations on power-constrained embedded platforms.

Substantial research efforts have been invested in reducing the computational cost of CNNs. One line of research attempts to reduce the precision of weights and activations, and has shown that 8-bit or even binary fixed-point representation is sufficient for evaluating a CNN [5, 6]. Another approach focuses on optimizing the structure of CNN itself. The work in [7] employs a three-step method, where the network is trained to learn important connections, prune redundant connections in pre-trained CNNs, and then retrain the pruned networks to restore the performance. Zhang et al. [8] proposed to replace convolutional layers by several convolutional layers applied sequentially, which have a lower total computational cost. Other research thrust exploits sparsity in well-trained CNNs or enhances sparsity in CNNs via regularization, and skips operations with zero entries (zero-skipping) [4]. Recent work [9, 10] showed that it is possible to avoid evaluation of certain computations with marginal performance loss. In [9], a linear regression model was trained for each convolutional layer to predict the importance of each convolutional filter, and prune low-impact filters at runtime. Panda et al. [10] proposed Conditional Deep Learning (CDL) by adding a linear network to each convolutional layer and monitoring the output to decide whether classification can be terminated at the current stage. In this paper, we propose a new technique referred to as predictive CNN (PredictiveNet) that predicts the sparse outputs (zero prediction) and thereby avoids computing those. In this way, a significant reduction in the number of convolutional operations is achieved without altering the structure or introducing additional side networks. Thus PredictiveNet has negligible overhead and can easily be applied in addition to existing techniques to obtain an even greater reduction in computational cost. PredictiveNet first evaluates the most significant bit (MSB) part of the convolution to predict whether the nonlinear layer output corresponding to the current convolution is zero, and then decides if the remaining least significant (LSB) part’s computation can be skipped or not. PredictiveNet takes advantage of the fact that the MSB part has exponentially larger contribution to the final output. When evaluated on a CNN for handwritten digit recognition using the MNIST dataset, simulation results show that PredictiveNet can achieve about $5.8 \times$ reduction in the required computation compared to a conventional CNN and about $2.9 \times$ reduction compared to a zero-skipping CNN [4], while incurring negligible loss in accuracy.

The rest of this paper is organized as follows. Section II provides background on CNNs and sparsity. Section III presents the PredictiveNet technique and analysis to justify its effectiveness. Simulation results are shown in Section IV. Finally, Section V concludes the paper.
II. BACKGROUND

A. Convolutional Neural Networks (CNNs)

CNNs are a class of multi-layer neural networks [11]. A CNN consists of a cascade of multiple convolutional layers (C-layers), subsampling layers (S-layers) (feature extractor), and fully-connected layers (F-layers) (classifier). In a C-layer, dot products (DPs) between receptive fields and weight vectors are computed, to which a bias term is added, and passed through a nonlinear function to generate the output feature maps (FMs). The computation of one output pixel for the C-layer is described as follows:

\[ z_m[j] = f(y_m[j]), (m = 1, \ldots, M) \] (1)

\[ y_m[j] = \sum_{l=1}^{L} w_m^T x_{jl} + b_m, (m = 1, \ldots, M) \] (2)

where \( L \) and \( M \) are the number of input and output FMs, respectively. \( w_m \) is the \( N \)-tuple weight vector connecting the \( j^{th} \) input FM \( X_l = [x_{l1}, \ldots, x_{lj}] \) (where \( x_{jl} \) is the \( j^{th} \) receptive field in \( X_l \)) to the \( m^{th} \) convolutional output \( y_m \), \( b_m \) is the bias term, and \( z_m \) denotes the \( m^{th} \) output FM in the C-layer. Equation (2) shows that the \( j^{th} \) pixel \( y_m[j] \) of the \( m^{th} \) convolutional output \( y_m \) is obtained by first performing DPs between the \( L \) input vectors \( x_{jl} \) and the weight vectors \( w_m \), and summing up the result. The nonlinear function \( f(\cdot) \) typically takes a sigmoid or hyperbolic form. However, a rectified linear unit (ReLU) has emerged recently as increased evidence shows that it improves performance of CNNs [12]. The S-layer reduces the dimension of its input FMs via either an average or a max pooling.

B. Sparsity in CNNs

Enforcing sparsity has been shown to improve classification accuracy in machine learning models, especially for deep neural networks [12, 13]. For example, the use of the ReLU creates sparse representations thereby enabling CNNs to reach their best performance. Specifically, assuming uniform distribution for the weights and the inputs, at least 50% of the distribution for the weights and the inputs, at least 50% of the hidden units are set to zero after passing through ReLU, which can easily be increased with sparsity-inducing regularization. In practice, it is commonly observed that the sparsity is more than 50% in state-of-the-art CNNs specifically in deeper stages of the networks [14].

III. THE PROPOSED PREDICTIVENET TECHNIQUE

This section describes the PredictiveNet technique and its analytical justification.

A. Principle and Architecture

Without loss of generality, we drop the indices for \( j \) and \( m \) in (1) and (2) and assume \( f(\cdot) \) is a ReLU, i.e.,

\[ z = \max \left( \sum_{l=1}^{L} w_l^T x_l + b, 0 \right) \] (3)

where \( w_l^T x_l = \sum_{i=1}^{N} w_l[i] x_l[i] \).

We first decompose \( x_l[i], w_l[i] \), and \( b \) into MSB and LSB parts. If we assume that \( B_{lsb} \) is the precision of the MSB part of \( w_l, x_l, \) and \( b \), then:

\[ z = \max (y, 0) = \max \left( \sum_{l=1}^{L} w_l^T x_l + b, 0 \right) \]

\[ = \begin{cases} y_{msb} + y_{lsb}2^{-(B_{msb}-1)} & \text{if } \sum_{l=1}^{L} w_l^T x_l + b > 0 \\ 0 & \text{otherwise} \end{cases} \] (4)

where \( y_{msb} \) and \( y_{lsb} \) can be expressed as follows:

\[ y_{msb} = \sum_{l=1}^{L} w_{l,msb} x_{l,msb} + b_{msb} \] (5)

\[ y_{lsb} = \sum_{l=1}^{L} (x_{l,msb}w_{l,msb} + x_{l,lsb}w_{l,lsb}) + b_{lsb} \] (6)

where \( x_{l,msb}, w_{l,msb}, \) and \( b_{msb} \) denote the MSB parts of \( x_l, w_l, \) and \( b \), respectively. Also, \( x_{l,lsb}, w_{l,lsb}, \) and \( b_{lsb} \) denote the LSB parts of \( x_l, w_l, \) and \( b \), respectively.

The PredictiveNet architecture (Fig. 1) includes C-MSB and C-LSB blocks that predicts the sign of \( y \) by computing only \( y_{msb} \) in (5). If \( y_{msb} < 0 \) (i.e., \( \text{sign}(y_{msb}) = 1 \)), then we set \( z = 0 \) without computing \( y_{lsb} \) in (6) and \( y_{msb} + y_{lsb}2^{-(B_{msb}-1)} \) in (4). If \( y_{msb} \geq 0 \) (i.e., \( \text{sign}(y_{msb}) = 0 \)), then C-LSB computes (6) and sets \( z = y_{msb} + y_{lsb}2^{-(B_{msb}-1)} \) in (4). By doing so, PredictiveNet avoids evaluating a significant number of convolutions while incurring only marginal accuracy loss.

The reasons for the accuracy loss to be marginal are as follows: 1) the contribution of \( y_{lsb} \) for calculating \( z = 2^{-(B_{msb}-1)} \) smaller than \( y_{msb} \) as shown in (4); 2) the specific value of \( y_{msb} + y_{lsb}2^{-(B_{msb}-1)} \) is not important if it is negative due to the rectification effect of ReLU; 3) the high sparsity in CNNs as mentioned in Section II-B implies that the term \( y_{msb} + y_{lsb}2^{-(B_{msb}-1)} \) is very likely to be negative, which will result in zero C-layer outputs after being passed through the ReLU function.

B. Analysis

In this subsection, analysis and empirical simulation results are presented to justify why PredictiveNet incurs marginal accuracy loss while greatly decreasing the computational cost. Our analysis is based on the trade-offs between accuracy and
TABLE I

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>MSB-CNN MSE</th>
<th>PredictiveNet MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_0$</td>
<td>$y_{msb} \leq 0, y \leq 0$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$H_1$</td>
<td>$y_{msb} \leq 0, y &gt; 0$</td>
<td>$y$</td>
<td>$y$</td>
</tr>
<tr>
<td>$H_2$</td>
<td>$y_{msb} &gt; 0, y \leq 0$</td>
<td>$-y_{msb}$</td>
<td>$0$</td>
</tr>
<tr>
<td>$H_3$</td>
<td>$y_{msb} &gt; 0, y &gt; 0$</td>
<td>$y - y_{msb}$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

precision. Recently, such a trade-off has been analytically characterized for simple machine learning algorithms such as SVM [15]. Such insights have not yet been leveraged for complex algorithms such as CNNs.

Assume that $B_{w}$, $B_{x}$ and $B_{b}$ denote the required precisions of $w_t$, $x_t$, and $b_t$ respectively. Also, let $B_{w,msb}$, $B_{x,msb}$ and $B_{b,msb}$ denote the precisions of $w_{l,msb}$, $x_{l,msb}$, and $b_{msb}$ in (5), respectively. For convenience, we term the CNN comprising only C-MSB as MSB-CNN, and the CNN implemented using $B_{w}$, $B_{x}$ and $B_{b}$ as the full precision CNN (FP-CNN), respectively.

In Table I, we compare the ReLU output errors of MSB-CNN and PredictiveNet with respect to the outputs of FP-CNN (i.e., $y$) in (3) for four disjoint events from $H_0$ to $H_3$. Note that each possible outcome is included in exactly one of these events. The mean squared errors (MSE) at the outputs of the ReLU with respect to FP-CNN are:

$$MSE_{\text{MSB-CNN}} = E[Y^2[H_1]P(H_1)] + E[Y_{msb}^2[H_2]P(H_2)] + E[|Y - Y_{msb}|^2[H_3]P(H_3)]$$

(7)

$$MSE_{\text{PredictiveNet}} = E[Y^2[H_1]P(H_1)]$$

(8)

where upper case letter denotes random variables. By comparing (7) and (8), we see that:

$$MSE_{\text{PredictiveNet}} < MSE_{\text{MSB-CNN}}.$$  (9)

Furthermore, $P(H_1)$ has been found to be small in practice and can be upper bounded as follows:

$$P(H_1) \leq \Delta_w^2E_1 + \Delta_x^2E_2 + \Delta_b^2E_3$$  (10)

where $\Delta_w = 2^{-(B_{w,msb}-1)}$, $\Delta_x = 2^{-(B_{x,msb}-1)}$ and $\Delta_b = 2^{-(B_{b,msb}-1)}$ are the quantization noise step sizes of $w_{l,msb}$, $x_{l,msb}$ and $b_{msb}$, respectively, and $E_1$, $E_2$, and $E_3$ are given in the Appendix along with the proof of (10).

Similarly, $E[Y^2[H_1]]$ can be upper bounded as follows:

$$E[Y^2[H_1]] \leq \Delta_w^2E_4 + \Delta_x^2E_5 + \Delta_b^2E_6$$  (11)

where $E_1 = E[\sum_{l=1}^L ||X_l||^2[H_1]]$ and $E_5 = \sum_{l=1}^L ||w_l||^2$. The proof of (11) can also be found in the Appendix.

Combining (10) and (11), we can obtain an upper bound on $MSE_{\text{PredictiveNet}}$:

$$MSE_{\text{PredictiveNet}} = \Delta_w^4E_6 + \Delta_x^4E_7 + \Delta_b^4E_8 + (\Delta_w\Delta_x)^2E_9 + (\Delta_w\Delta_b)^2E_{10} + (\Delta_x\Delta_b)^2E_{11}$$  (12)

where $E_{01}, \ldots, E_{11}$ are the cross product terms associated with the product of (10) and (11).

We see that every term in (12) is a fourth order multiplicative combination of quantization steps. Each quantizations step is of the order of $2^{-B_{msb}}$. Hence, the upper bound in (12) is of the order of $2^{-4B_{msb}}$.

Figure 2 shows empirical values of $MSE_{\text{MSB-CNN}}$ and $MSE_{\text{PredictiveNet}}$ for the two C-layers in a CNN designed for handwritten digit recognition [16]. Figure 2 supports (9) and shows that the MSE of PredictiveNet is much smaller than the MSE of MSB-CNN. This results from the exponentially larger weighting factor of $y_{msb}$ contributed to $y$ over that of $y_{lsb}$ and the high sparsity of the C-layer outputs in well trained CNNs.

IV. SIMULATION RESULTS

In this section, we evaluate the performance of PredictiveNet based on the MNIST database for handwritten digit recognition.

A. System Set-up

The parameters of the CNNs are from [16]. The term $b_m$ and kernel $w_{ml}$ in (2) are trained using the back propagation algorithm [11]. The following four architectures are considered: 1) a conventional FP-CNN or FP-CONV; 2) a FP-CONV using zero-skipping [4] or FP-ZS; 3) a CNN with PredictiveNet applied to FP-ZS; 4) a MSB-CNN. For FP-CNN, $B_w = B_b$ and $B_x$ are set to 7 bits and 8 bits, respectively, ensuring the fixed-point classification error rates to be within 0.1% of the floating-point error rates of 0.023. The classification error rate is defined as: $pe = P(T \neq T)$ ($T$ and $t$ are the classifier decision and the true label, respectively).

B. System Performance and Computational Cost

Figure 3 compares the performance of FP-CNN (FP-CONV, FP-ZS). PredictiveNet and MSB-CNN in terms of classification error rates and computational cost normalized over that of FP-CONV. It is shown in Fig. 3 that PredictiveNet is able to achieve a classification error rate that is only 0.3% larger than that of FP-CNN while reducing the computational cost by a factor of 2.9×. On the other hand, when compared to MSB-CNN, PredictiveNet reduces the classification error rate by about 10.8× at the cost of 2.4× greater computational cost. Table II summarizes the required number of FAs to implement the four CNNs. Figure 3 and Table II show that PredictiveNet's...
accuracy is slightly worse than FP-CNN but with significantly lower complexity.

V. CONCLUSION

In this paper, we proposed the PredictiveNet technique, which predicts sparse nonlinear outputs and skips corresponding convolution operations for reduced complexity CNN design. Analysis was performed to justify the effectiveness of PredictiveNet and predict the behavior of CNNs with respect to its precision. This work opens up another possibility to greatly reduce CNNs’ computational cost without degrading their detection accuracy.

ACKNOWLEDGMENT

This work was supported in part by Systems on Nanoscale Information fabric(s) (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

REFERENCES


APPENDIX

We provide a detailed derivation of (10) and (11).

\( P(H_1) = P(Y_{\text{msb}} \leq 0, Y > 0) = P(Y > 0) P(Y_{\text{msb}} \leq 0 | Y > 0) \)
\[
= \frac{1}{2} P(Y > 0) \int f_X|Y > 0(x) P(|q_y| > Y | Y > 0, X = x) dx
\]
\[
\leq \frac{P(Y > 0)}{24} \int f_X|Y > 0(x) \sum_{l=1}^{L} \left[ \Delta_{w}^2 ||x_l||^2 + \Delta_{\beta}^2 ||w_l||^2 + \Delta_{b}^2 \right] dx
\]
\[
= \frac{1}{24} \left( \frac{\Delta_{w}^2 \sum_{l=1}^{L} ||x_l||^2 + \Delta_{\beta}^2 ||w_l||^2 + \Delta_{b}^2 \sum_{l=1}^{L} w_l^T x_l + b_l^2}{||x_l||^2 + \Delta_{\beta}^2 ||w_l||^2 + \Delta_{b}^2} \right) Y > 0
\]
\[
= \frac{1}{24} \left[ \frac{\sum_{l=1}^{L} ||x_l||^2 + \Delta_{\beta}^2 ||w_l||^2 + \Delta_{b}^2}{||x_l||^2 + \Delta_{\beta}^2 ||w_l||^2 + \Delta_{b}^2} \right] \sum_{l=1}^{L} w_l^T x_l + b_l^2
\]
\[
+ \frac{1}{24} \left( \Delta_{w}^2 \sum_{l=1}^{L} ||w_l||^2 + \Delta_{\beta}^2 \right) E \left[ \sum_{l=1}^{L} w_l^T x_l + b_l^2 \right]
\]
\[
= \Delta_{w}^2 E_1 + \Delta_{\beta}^2 E_2 + \Delta_{b}^2 E_3
\]

where \( f_X|Y > 0(x) \) is the conditional distribution of \( X \) given \( Y > 0 \) and \( q_y = \sum_{l=1}^{L} (q_{w_l} x_l + q_{\beta} w_l) + q_b \). Note that \( q_{w_l}, q_{\beta}, \text{ and } q_b \) are the quantization noise terms of \( w_{l,\text{msb}}, x_{l,\text{msb}}, \text{ and } b_{\text{msb}}, \text{ respectively. } \) \( I_A \) denotes the indicator function of the event \( A \). The \( \frac{1}{2} \) in the second step is due to the symmetric distribution of \( q_y \). The fourth step comes from Chebyshev’s inequality. Note that

\[
E_1 = \frac{1}{24} \sum_{l=1}^{L} \left[ \frac{||x_l||^2 \cdot I_{Y > 0}}{\sum_{l=1}^{L} w_l^T x_l + b_l^2} \right]
\]
\[
E_2 = \frac{1}{24} \sum_{l=1}^{L} \left[ \frac{||w_l||^2 \cdot I_{Y > 0}}{\sum_{l=1}^{L} w_l^T x_l + b_l^2} \right]
\]
\[
E_3 = \frac{1}{24} \left[ \frac{I_{Y > 0}}{\sum_{l=1}^{L} w_l^T x_l + b_l^2} \right].
\]

Furthermore, under \( H_1 \) we have \( y_{\text{msb}} = y + q_y \leq 0 \) and \( y > 0 \) which means that \( 0 < y \leq -q_y \) (i.e., \( |y|^2 \leq |q_y|^2 \)). Hence,

\[
E[Y^2|H_1] \leq E[q_y^2|H_1] = \Delta_{w}^2 E \left[ \sum_{l=1}^{L} ||x_l||^2 |H_1 \right] + \Delta_{\beta}^2 \sum_{l=1}^{L} ||w_l||^2 + \Delta_{b}^2.
\]