DrowsyNet: Convolutional Neural Networks with Runtime Power-Accuracy Tunability Using Inference-Stage Dropout

Ren-Shuo Liu*, Yun-Chen Lo†, Yuan-Chun Luo‡, Chih-Yu Shen§, and Cheng-Ju Lee¶

Department of Electrical Engineering, National Tsing Hua University, Taiwan
*renshuo@ee.nthu.edu.tw, †yunchen.lo@gapp.nthu.edu.tw, ‡yuanchun@gapp.nthu.edu.tw,
§cyshen@gapp.nthu.edu.tw, ¶lee2832@purdue.edu

***Abstract*—Convolutional neural networks (CNNs) have many future artificial intelligence (AI) applications; CNN hardware is an enabling technology to make this AI future come true. Two main specifications of CNN hardware are power and accuracy, which are determined at the design time by the CNN model, CNN hardware architecture, and the transformation scheme that maps the model onto the hardware.**

This paper points out that the above three design parameters share one common limitation that they are not flexible enough to offer power-accuracy tunability at deployment time and runtime. In response to this demand, we propose DrowsyNet, which randomly drops out a fraction of convolutional neurons to achieve a power-accuracy tradeoff. DrowsyNet is an inference-stage technique, which is different from traditional training-stage dropout. We further present that the dropout rates would better be set non-uniformly among convolutional layers. Experimental results show that up to 50% savings in the number of convolutions are available by trading away less than 11% of top-5 accuracy.

I. Introduction

Convolutional neural networks (CNNs) open up many attractive possibilities of future artificial intelligence (AI) applications. CNN hardware (including chips and silicon intellectual properties (IPs))[1] is an enabling technology to realize this promising AI future. For example, with an system-on-chip (SoC) chip that embeds an image recognition CNN model, a smartphone can react to people and scenes when performing video capturing, and a robot can monitor and understand the foods it cooks.

Two main specifications of a CNN chip are power and accuracy, which are determined at the design time by three categories of parameters. First, the topology and neuron count of the selected CNN model play an important role[2]. Generally speaking, a CNN model with more layers and neurons is more capable of learning things; therefore, a deeper CNN model tends to achieve higher accuracy at the cost of consuming much power to perform convolution operations. Second, the architecture and arithmetic units of a CNN chip affect the power consumption per convolution operation. For example, a CNN chip using integer arithmetic units consumes less power than using floating point arithmetic units, but the former may result in an accuracy loss due to its lower arithmetic precision. Third, a lossy transformation is sometimes utilized to map a CNN model on to CNN hardware, which can affect the power and accuracy, too.

[1]In the following text, "CNN chips", "CNN IPs", and "CNN hardware" are used interchangeably

[2]Datasets and algorithms to train a CNN model can play an even important role though, they are out of the scope of this paper.

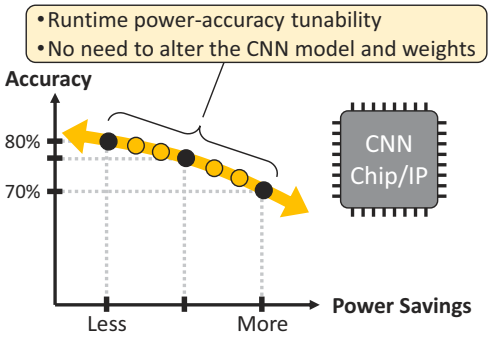

Fig. 1. Objective and advantages of DrowsyNet

We point out that the abovementioned three factors are not flexible at deployment time and runtime, which thus hinder product development, power management, and thermal management. For example, different smartphone models may target different market segments and demand different CNN power-accuracy tradeoff points. It would be desirable for both smartphone vendors and CNN hardware vendors that one (or few) CNN solution covers a wide range of needs. For another example, during video capturing, one may want a smartphone to perform accurate recognition on a portion of frames and less-accurate recognition on some other frames to save battery power and prevent overheating. Thus, it would be efficient if a CNN chip natively supports multiple power-accuracy tradeoff modes to choose from.

In response, we propose DrowsyNet, an extended neural network model that supports runtime power-accuracy tunability as illustrated in Figure 1. DrowsyNet randomly drops out inference-stage convolutional neurons. By doing so, one can gradually trade away a CNN chip's accuracy for the savings in the number of convolutions the CNN chip needs to perform, which means a power consumption decrease.

In addition to naive uniform dropout, we further propose to differentiate dropout rates among different convolutional layers. The rationale behind proposing such non-uniform DrowsyNet is threefold. First, different convolutional layers contain a different number of neurons and contribute a different amount of power consumption. Second, a neuron

978-1-5386-4260-3/18/$31.00 ©2018 IEEE

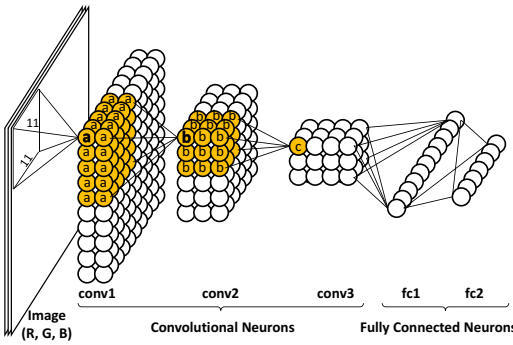

Fig. 2. Convolutional Neural Network

	#Neurons of the Layer		#Mult. per Neuron (aka. Kernel Size)		#Mult. of the Layer
	Ch x H x W = #N		Ch x H x W = #M		#N x #M
conv1	96x55x55	**290 K**	3x11x11	**363**	**105 M**
conv2	256x27x27	**187 K**	96x5x5	**2.4 K**	**448 M**
conv3	384x13x13	**65 K**	256x3x3	**2.3 K**	**150 M**
conv4	384x13x13	**64 K**	384x3x3	**3.5 K**	**224 M**
conv5	256x13x13	**43 K**	384x3x3	**3.5 K**	**150 M**

TABLE I
CONVOLUTIONAL LAYER PARAMETERS OF THE CAFFENET MODEL

with more input synapses consumes more computation energy. Third, different layers of neurons can exhibit different sensitivities to dropout, which needs design space exploration to reveal.

It is noteworthy that the proposed inference-stage dropout is significantly different from the famous Srivastava's dropout [3] from three perspectives. First, Srivastava's dropout is applied during training a neural network instead of during inferencing images. Second, Srivastava's dropout is applied to the fully-connected layers of a CNN instead of the convolutional layers. Third, the objective of Srivastava's dropout is to avoid overfitting instead of reaching a power-accuracy tradeoff point. To our knowledge, this work is the first work focusing on the runtime power-accuracy tunability using inference-stage dropout.

This paper is organized as follows. Section II presents the background of CNNs and related works. Section III presents the proposed Drowsy Neuron model and DrowsyNet. Section IV quantitatively evaluates our observations and designs. Section V concludes this paper.

II. BACKGROUND AND RELATED WORKS

Figure 2 illustrates a CNN model with three layers of convolutional neurons and two layers of fully-connected neurons. A sphere stands for a neuron, and a line stands for an interconnection between two neurons (only a few lines are shown for brevity). Every neuron produces its output by performing *weighted sum* computation. For example, the neuron 'c' computes the weighted sum of the outputs of nine neurons 'b', and the neuron labeled with boldface 'b' computes

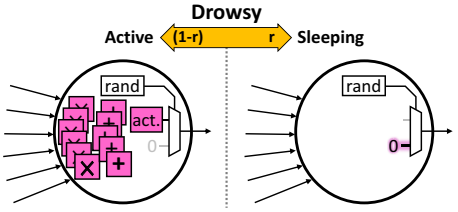

Fig. 3. Proposed Drowsy Neuron Model

the weighted sum of the outputs of 50 neurons 'a'. The first layer of neurons also perform weighted sum except that image pixels are taken as inputs. For example, the neuron labeled with boldface 'a' computes the weighted sum of a patch of image with $11 \times 11 \times 3$ pixels.

Most state-of-the-art CNN models consist of many (e,g, five to up to a hundred) convolutional layers. Table I lists the convolutional layer parameters of CaffeNet [7]. A convolutional layer is composed of around one hundred thousand neurons, and a neuron needs to compute weighted sum over several hundred to several thousand neuron outputs. The required multiplications per convolutional layer thus can reach several hundreds of millions. Therefore, convolutions dominate the computation of a CNN model.

Several prior strategies are available to reduce the amount of convolution computation but are not flexible enough to offer power-accuracy tunability at deployment time and runtime. Deep Compression [5] prunes weights that are close to zero, which reduces the number of multiplications. Eyeriss [6] detects zeros in neuron inputs to skip multiplications. Minerva [2] reduces the precision of multipliers to as low as five-bit. SC-DCNN [1] exploits stochastic computation to turn multiplications into logical XNOR operations.

III. DROWSYNET DESIGN

A. Drowsy Neuron Model

Figure 3 plots the proposed Drowsy Neuron model. Unlike the traditional neuron model, which always performs hundreds (or thousands) of multiplications and additions to produce an output, the proposed Drowsy Neuron model intermittently performs no computation at all and simply outputs a zero. In other words, we intentionally modulate a neuron somewhere between waking and sleeping states (i.e., drowsy).

We define two flavors of Drowsy Neurons, rescaled and non-rescaled. Rescaling can keep the expected value of the Drowsy Neuron independent of its dropout rate. For a rescaled Drowsy Neuron with a dropout rate r, its output is rescaled by $\frac{1}{(1-r)}$. Let $u = \sum_{j=1}^{M}(x_j \cdot w_j)$, where M stands for the number of inputs of a neuron; x_j and w_j stand for the M inputs and M weights of the neuron, respectively. The output of a rescaled Drowsy Neuron is defined as follows:

$$\text{output} = \begin{cases} 0 & \text{if SLEEP} = 1 \\ 0 & \text{if SLEEP} \neq 1 \text{ and } u \leq 0 \\ u \cdot \frac{1}{(1-r)} & \text{if SLEEP} \neq 1 \text{ and } u > 0 \end{cases}$$

Here SLEEP is a random variable with probability $P(\text{SLEEP} = 1) = r$

B. Uniform DrowsyNet

A neural network can adopt one single dropout rate, and we name this type of design Uniform DrowsyNet. In this work, we focus on employing drowsy neurons in the convolutional layers of a neural network because they contribute the majority computation in a neural network. Employing drowsy neurons in the fully connected layers is viable but is left as a future study. When mapping onto a CNN chip, Uniform DrowsyNet incurs insignificant overhead. For example, a CNN chip typically maps convolutions onto processing engines (PEs). A three-bit linear feedback shift register (LSFR) per PE is sufficient to support up to eight power-accuracy tradeoff levels with a dropout rate step of $\frac{1}{8}$.

C. Non-Uniform DrowsyNet

Comparing to Uniform DrowsyNet, which is restricted to adopt one single dropout rate, Non-Uniform DrowsyNet represents larger design space, which can provide better power-accuracy tradeoffs. The challenges of a Non-Uniform DrowsyNet are twofold. First, since a neural network consists of millions of neurons, and each neuron has a dropout rate, searching the optimal settings in this extremely large design space is intractable. Second, Non-Uniform DrowsyNet should not incur significant overhead when mapped onto hardware.

In response to the above challenges, we propose a heuristic that sets non-uniform rates among convolutional layers while keeping a single rate within the same layer. The rationale behind this layer-based heuristic is threefold. First, different layers contain a different number of neurons and contribute a significantly different amount of power consumption. For example, the first convolutional layer of CaffeNet contains around 7× more neurons (290 K) than the fifth convolutional layer (43 K) (Table I). Second, different layers exhibit significantly different per-neuron energy consumptions due to a different amount of computation. For example, every single neuron at the fourth layer of CaffeNet (and the fifth layer, too) needs to perform around $10\times$ more multiplications (3.5 K) than that of the first convolutional layer does (0.36 K). Third, different layers of neurons can exhibit different sensitivities to dropout. Quantitative comparisons are provided in Section IV. Algorithm 1 searches the design space based on the proposed layer-based heuristic

In addition, uniformly adopting rescaling Drowsy Neurons in all convolutional layers is not a good strategy. A heuristic of selecting between rescaled and non-rescaled Drowsy Neuron is described as follow. If the neuron provides its output to another convolutional layer, the expected value of the neuron is relatively important, and thus the rescaled flavor of Drowsy Neuron is used. In comparison, if the neuron feeds its output to a max-pooling layer, the absolute value outweighs the expected value, the non-rescaled flavor of Drowsy Neuron is chosen.

Data:
n : #convolutional layers of the neural network;
N_i : #neurons at the i^{th} layer;
M_i : #multiplies per neuron at the i^{th} layer;
k : #number of different dropout rate values;
Result:
Pareto optimal power-accuracy tradeoff points
for *n-tuple* $(r_1, r_2, ..., r_n)$, *where* $r_i = 0, 1, ..., (k-1)$ **do**
  for i^{th} *layers* **do**
    Dropout rate of the layer = $\frac{r_i}{k}$;
  end
  Power saving $s = \sum_{i=1}^{n}(\frac{r_i}{k} \times N_i \times M_i)$;
  Evaluate accuracy scores, a_{top-1} and a_{top-5};
  Record a power-accuracy tradeoff point (s, a_{top-1});
  Record a power-accuracy tradeoff point (s, a_{top-5});
end
Derive pareto optimal power-accuracy tradeoff points;

Algorithm 1: Search Non-Uniform DrowsyNet

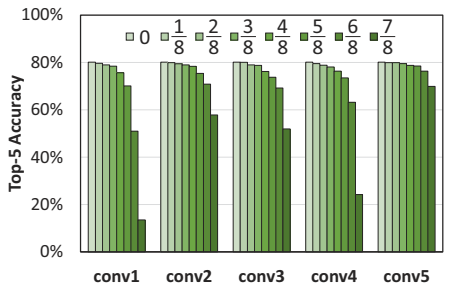

Fig. 4. Sensitivity of Convolutional Layers to Dropout Rates

IV. Evaluations

We perform quantitative evaluation using Caffe [7] framework on a server with an Intel i7 6850K processor and NVIDIA GTX 1080 Ti graphics cards. The Caffe framework is extended to simulate inference-stage dropout. We use the pretrained CaffeNet model [7] to classify the ImageNet dataset [4]. DrowsyNet is applicable to many other models, but related results are not shown due to space limitation. We report the savings in the multiplication as the metric of power saving and the top-1 and top5 scores over 2,500 images as the metric of accuracy.

Figure 4 quantitatively analyzes each layer's accuracy sensitivity to dropout rate by applying DrowsyNet to exactly one convolutional layer at a time with a dropout rate ranging from 0, $\frac{1}{8}$., ..., to $\frac{7}{8}$. Different layers exhibit significantly different sensitivities, and the trend is not intuitive without conducting the experiments. For example, the first layer (conv1) exhibits the highest sensitivity and the fifth layer (conv5) the least. This result seems to results from the fact that the first layer (conv1) contains the most neurons and the fifth layer (conv5) the least among all five layers. However, the second layer (conv2) contains $3\times$ more neurons than the fourth layer (conv4) does, but oppositely, the former behaves less sensitive to dropout than the later.

Figure 5 plots the potential savings in multiplications of different layers. The numbers are normalized to the total

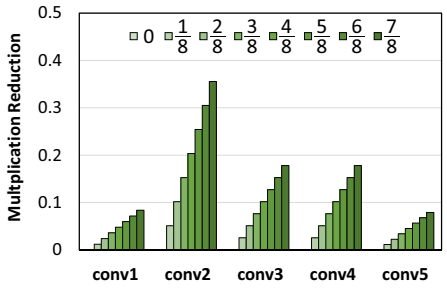

Fig. 5. Multiplication savings of Convolutional Layers to Dropout Rates

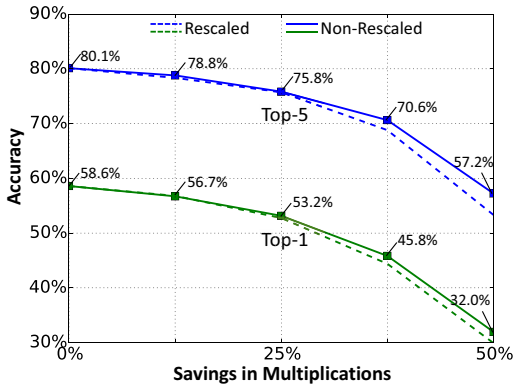

Fig. 6. Power-Accuracy Tradeoffs Achieved by Uniform DrowsyNet

multiplication number of CaffeNet. The findings are interesting. For example, though the second layer (conv2) does not contain the most number of neurons, it can contribute the most multiplication savings because each neuron needs a large number (2.4 K) of multiplications. In contrast, the first layer (conv1) contains the most number of neurons though, it can contribute almost the least savings, only one-third that of the second layer.

Figure 6 demonstrates the power-accuracy tunability achieved by Uniform DrowsyNet. Both rescaled and non-rescaled flavors are evaluated. The results show that non-rescaled one performs better. By setting the dropout rate to $\frac{1}{8}$, $\frac{1}{4}$, $\frac{3}{8}$, and $\frac{1}{2}$, the top-5 accuracy lowers by 1.3%, 4.3%, 9.5%, and 22.9%, respectively, and the top-1 accuracy lowers by 1.9%, 5.4%, 12.8%, and 26.6%, respectively. Note that it is DrowsyNet that makes these power-accuracy tradeoffs available. More importantly, since DrowsyNet involves no changes to the CNN model and weights, the tradeoffs are available not only after a CNN chip is fabricated but also when the chip is deployed in products.

Figure 7 plots the design space of Non-Uniform DrowsyNet. Every dot denotes a setting of drowsy rates to each layers, the X axis denotes the corresponding saving in multiplications, and the Y axis denotes the accuracy. The solid lines show the optimal choices, and the four Uniform DrowsyNet lines in Figure 6 are shown again as the four dashed lines for easy comparisons. Non-Uniform DrowsyNet yields significantly better

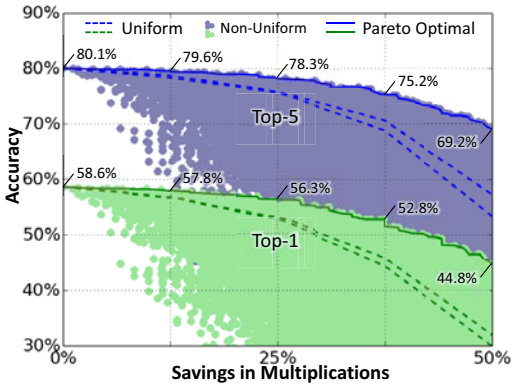

Fig. 7. Power-Accuracy Tradeoffs Achieved by Non-Uniform DrowsyNet

power-accuracy tradeoffs than Uniform DrowsyNet does. For example, the top-1 and top-5 accuracy scores are 12.8% and 12% higher than Uniform DrowsyNet under the condition that 50% of multiplications are saved.

V. Conclusions

This paper proposes DrowsyNet, a runtime power-accuracy tuning technology using inference-stage dropout. We present a Drowsy Neuron model, which intermittently skips all computations and simply outputs a zero. DrowsyNet is different from the existing training-stage dropout [3], and thus DrowsyNet opens up new design space. We propose to adopt non-uniform dropout rates among convolutional layers to take full advantages of DrowsyNet. Experimental results show that DrowsyNet achieves power-accuracy tunability up to 50% savings in multiplications with less than a 11% loss in the top-5 accuracy. We leave measuring power consumption on real platforms as our future work.

Acknowledgment

We thank the anonymous reviewers for their insightful feedback. This work is supported in part by the Ministry of Science and Technology (MOST) of Taiwan under grants 105-2218-E-007-023-, 106-2221-E-007-125-, and 107-2218-E-007-001-.

References

[1] A. Ren et al. SC-DCNN: Highly-scalable deep convolutional neural network using stochastic computing. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2017.

[2] B. Reagen et al. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *International Symposium on Computer Architecture (ISCA)*, 2016.

[3] N. Srivastava et al. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014.

[4] O. Russakovsky et al. Imagenet large scale visual recognition challenge. Technical Report arXiv:1409.0575, 2014.

[5] S. Han et al. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. Technical Report arXiv:1510.00149, 2015.

[6] Y.-H. Chen et al. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. In *International Solid-State Circuits Conference (ISSCC)*, 2016.

[7] Y. Jia et al. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.