HARDWARE-AWARE EXPONENTIAL APPROXIMATION FOR DEEP NEURAL NETWORK

Xue Geng[§], **Jie Lin**[§], **Bin Zhao**[♮], **Zhe Wang**[§], **Mohamed M. Sabry Aly**[♯], **Vijay Chandrasekhar**[§]
[§]I^{2}R, A*STAR, Singapore {geng_xue,lin-j,wang_zhe,vijay}@i2r.a-star.edu.sg
[♮]IME, A*STAR, Singapore zhaobin@ime.a-star.edu.sg
[♯]School of CSE, NTU, Singapore msabry@ntu.edu.sg

ABSTRACT

In this paper, we address the problem of cost-efficient inference for non-linear operations in deep neural networks (DNNs), in particular, the exponential function e^x in softmax layer of DNNs for object detection. The goal is to minimize the hardware cost in terms of energy and area, while maintaining the application accuracy. To this end, we introduce Piecewise Linear Function (PLF) for approximating e^x. First, we derive a theoretical upper bound of the number of pieces required for retaining the detection accuracy. Moreover, we constrain PLF to bounded domain in order to minimize bitwidths of the lookup table of pieces, resulting in lower energy and area cost. The non-differentiable bounded PLF layer can be optimized via the straight-through estimator. ASIC synthesis demonstrates that the hardware-oriented softmax costs 4x less energy and area than the direct lookup table of e^x, while with comparable performance on benchmark datasets.

1 INTRODUCTION

Deep neural networks (DNNs) has achieved massive success in a broad of applications in computer vision Krizhevsky et al. (2012), NLP, speech, games Chouard (2016); Silver et al. (2016), etc. This has also triggered rapid development of customized hardware for accelerating the inference phase of DNNs Han et al. (2016); Chen et al. (2017); Jouppi et al. (2017), by hardware-algorithm co-optimization. The seminal work is the custom Application-Specific Integrated Circuit (ASIC) called Tensor Processing Unit (TPU) Jouppi et al. (2017), which is 15x to 30x faster at inference than the general purpose NVIDIA GPU K80. To optimize DNNs algorithm for inference speed up in hardware, recent work mainly focused on reducing either the size of multiply-and-accumulate (MAC) operands (e.g. reduce bitwidth of weights and activations) or the number of MAC operations (e.g. weight or channel pruning), as MAC operations in linear CONV and FC layers account for over 99% of total operations in modern DNNs such as CNN and LSTM.

It's worthing noting that non-linear operations (e.g. softmax and sigmoid) could also be identified as the computational bottleneck of DNNs. An example is the softmax function $\sigma(x_c) = e^{x_c}/\sum_{j=1}^{C} e^{x_j}$, which is widely used in modern DNNs for different applications, e.g. Faster R-CNN for object detection. In the classification stage of Faster R-CNN, the cost of computing softmax is proportional to $\{\#$ bounding boxes to be evaluated$\} \times \{\#$ detection classes$\}$, which easily becomes the key challenge for inference phase of Faster R-CNN (e.g. $300 \times 10,000$ = 3million per image). However, there is very few publicly available research work optimizing non-linear operations for DNNs-specific hardware. Google TPU developed the *activation* unit to support non-linear operations Jouppi et al. (2017), but without revealing technical details.

In this paper, we aim to develop cost-efficient inference mechanism for non-linear operations in DNNs. In particular, we are interested in approximating e^x in softmax layer. The approximation of e^x is expected to minimize the energy and area cost, without incurring considerable performance loss in accuracy. To fulfill all these constraints, we propose a bounded Piecewise Linear Function (PLF) for approximating e^x. The bounded PLF is hardware-friendly in the sense that it is multiplication-free and requires small lookup table size, which are favorable properties for DNNs specific hardware. For instance, 32-bit fixed-point addition can be 30x less energy and 25x less area than 32-bit fixed-

point multiply Dally (2016). To the best of our knowledge, this is the first publicly available research work towards hardware-aware softmax.

One may note that we did not consider the classical polynomial approximation approaches for e^x, such as Taylor series Nilsson et al. (2014), which would introduce far more multiplication operations than lookup table based approximation schemes, in order to preserve application accuracy.

2 APPROXIMATE e^x WITH BOUNDED PIECEWISE LINEAR FUNCTION

Preliminary. Towards hardware-aware softmax, the key is to develop cost-effective approximation for e^x. A straightforward approach is Piecewise Linear Function (PLF), which typically approximates non-linear curve with a set of line pieces Amin et al. (1997); Namin et al. (2009). In geometry, PLF approximates e^x with S continuous pieces uniformly defined over a finite domain of $x \in [x_l^1, x_r^S]$, each of those pieces is an affine function with slope α^s

$$\begin{gathered} f^s(x) = \alpha^s * (x - x_l^s) + y_l^s = \alpha^s * x + (y_l^s - \alpha^s * x_l^s) \\ x \in [x_l^s, x_r^s], \quad y_l^s = e^{x_l^s}, \quad s \in [1, S] \end{gathered} \tag{1}$$

At inference phase, a lookup table of pieces is built to decide which piece is chosen for computing the approximate value $f^s(x)$ in Eq 1, given x. The cost of computing $f^s(x)$ contains 1 multiply ($\alpha^s * x$) and 1 addition ($\alpha^s * x_l^s + y_l^s$ is pre-computed and stored in lookup table, together with α^s).

In Table 1, we evaluate PLF in terms of detection accuracy on 2 benchmark datasets with 2 state-of-the-art object detection networks. Specifically, we train the DNNs with softmax, and replace e^x in softmax layer with PLF at inference phase. We also include another lookup table baseline (*Lookup-e^x*) that directly stores the exact value of e^x over uniformly sampled x in $[x_l^1, x_r^S]$. We observe that PLF requires much smaller table size (4x) than Lookup-e^x.

Next, we address 3 remaining issues for PLF. First, we analyze how does lookup table size (the number of pieces) affect application accuracy, in particular, in the context of object detection with DNNs. Second, we propose to constrain the domain of x for PLF (termed Bounded PLF) to save bitwidths of lookup table values, resulting in lower energy and area cost. Finally, we further reduce the energy and area cost of Bounded PLF by removing the multiply operation $\alpha^s * x$.

PLF: Lookup table size? We provide a theoretical analysis to derive the upper bound of lookup table size S (i.e. the number of pieces in PLF), without reducing application accuracy of object detection. In the classification stage of object detection networks (e.g. Faster R-CNN or R-FCN), the output of softmax is C-d confidence scores for each of the P region proposals (a.k.a. ROI bounding boxes). For each class, the region proposals are sorted by their corresponding confidence scores, followed by the subsequent non-maximum suppression (NMS) for bounding box filtering. To maintain application accuracy with PLF, the sufficient condition is that PLF does not change the order of region proposals generated by softmax.

We define the adversarial noise Zhou et al. (2017) between a consecutive pair of region proposals sorted by softmax, $\mathbf{r}^*(x_{p,c}) = ||\sigma(x_{p,c}) - \sigma(x_{p+1,c})||^2/2$, which is the minimum noise that would change the order of the pair. Let $\mathbf{r}$ denotes the error between softmax and the approximate value of PLF for the ith bounding box with the cth class,

$$\mathbf{r}(x_{p,c}) = ||\sigma(x_{p,c}) - \frac{f^s(x_{p,c})}{\sum_j^C f^s(x_{p,j})}||, \qquad p \in (0, P-1) \tag{2}$$

To maintain the order of predicted bounding boxes for class c, PLF has P pairwise constraints,

$$||\mathbf{r}(x_{p,c})||^2 + ||\mathbf{r}(x_{p+1,c})||^2 \leq \mathbf{r}^*(x_{p,c}), \qquad i \in (0, P-1) \tag{3}$$

Given that the domain of x is fixed for the uniform PLF, Eq 3 is determined by the lookup table size S. We use Bellman-Ford algorithm Goldberg & Radzik (1993) to solve these inequalities, and derive a theoretical upper bound for the table size S, which enables efficient binary search of the practical minimal table size that retains application accuracy.

Bounded PLF. The entries in the lookup table are stored as fixed-point number representations, arbitrary domain of x probably leads to higher bitwidths of entries as well as higher energy and area cost. One can see from Eq 1 that the range of y_l^s depends on the domain of x_l^s, e.g. to maximize

Table 1: Comparisons of PLF with softmax and lookup table of e^x on 2 object detection benchmark datasets PASCAL VOC2007 Everingham et al. (2010) and MS-COCO 2014 Lin et al. (2014). Following the standard protocol, we report results on VOC2007 test set and MS-COCO minival set in terms of mean Average Precision (mAP). Both Faster-RCNN and R-FCN with various backbone networks (VGG and ResNet) are pre-trained with softmax. At inference phase, we replace e^x in softmax layer with either PLF or Lookup-e^x. For both approximation methods, we report the minimum number of table size that maintains comparable detection accuracy with softmax.

Methods	min. table size	VOC2007 mAP		COCO mAP@[0.05-0.95]	
		Faster R-CNN (VGG16)	R-FCN (ResNet50)	Faster R-CNN (VGG16)	R-FCN (ResNet101)
PLF	32	72.06	71.69	24.10	28.10
Lookup-e^x	128	72.55	71.64	24.10	27.40
Softmax	–	72.52	71.76	24.20	28.20

Table 2: ASIC synthesis experiments to evaluate the hardware cost of approximating e^x in terms of bitwidth of lookup table values as well as area and energy cost. We compare the Bounded PLF (BPLF) with Lookup-e^x, using the clipped domain of x with or without training. Results are reported on PASCAL VOC2007 test set with Faster R-CNN + VGG16. Both approaches are synthesised using UMC CMOS 65nm Standard cell library. It's worth noting that we adopted hard-coded lookup table, as using memory to store lookup table is not efficient for the silicon area if table size is less than 1k entries. Again, we report the minimum number of table size that maintains comparable detection accuracy with softmax. Numbers in bracket represent the number of sign bits, integer bits and fractional bits for entries in the lookup table, respectively.

Methods		VOC2007 mAP	min. table size	bitwidths			*area* (μm^2)	*energy* (normalized)
				α^s	x	e^x		
Clip	BPLF	71.93	8	(0,4,0)	(1,4,2)	(0,20,1)	277	1.8x
($\gamma = 12$)	Lookup-e^x	72.59	128	–	(1,4,2)	(0,20,1)	921	6.4x
Trained Clip	BPLF	70.58	2	(1,3,0)	(1,3,2)	(0,10,3)	20	1.0x
($\gamma = 6$)	Lookup-e^x	70.70	64	–	(1,3,2)	(0,10,3)	38	4.5x

the precision and avoid overflow, the minimum number of integer bits for e^{12} is 20-bit. To minimize bitwidth, a clipped layer is applied at the bottom of PLF to bound the domain of $x \in [-\gamma, \gamma]$

$$h(x) = min(max(x, -\gamma), \gamma) \tag{4}$$

where γ is a pre-defined positive threshold. Smaller γ results in lower hardware cost, but may incur performance loss in accuracy at inference phase, due to the precision loss of e^x. To alleviate this issue, we propose to train the DNNs with the clipped layer combined with the PLF layer, as the replacement of e^x in softmax layer. As Eq 1 and Eq 4 are non-differentiable functions, we use Straight Through Estimator (STE) Bengio et al. (2013) to enable the back-propagation.

Bounded PLF without Multiply. Finally, the multiply term $\alpha^s * x$ in Eq 1 can be also transformed to bit shifting operation, by approximating α^s with 2^b where b is integer constant. As a result, the energy and area cost of Bounded PLF are further reduced.

Table 2 compares the Bounded PLF (BPLF) with Lookup-e^x in terms of hardware cost. ASIC synthesis experiments using UMC CMOS 65nm Standard cell library are performed to generate the hardware metrics, including area and energy cost. First, we observe that the trained clipped layer (γ reduced from 12 to 6) helps reducing the number of minimum table size as well as the bitwidths of lookup table values, with sacrificing slight mAP. Second, BPLF dramatically outperforms Lookup-e^x with much less area and energy cost (e.g. 2x-4x). These results demonstrate the effectiveness of the proposed BPLF in both hardware and software aspects.

3 Conclusion

Future work includes (1) extending the approximation of e^x to softmax function by taking the division into account; (2) beyond optimizing algorithm for hardware, exploring hardware-algorithm co-optimization for softmax approximation; (3) evaluating the softmax approximation approach on other applications like language modeling.

REFERENCES

Hesham Amin, K Memy Curtis, and Barrie R Hayes-Gill. Piecewise linear approximation applied to nonlinear function of a neural network. *IEE Proceedings-Circuits, Devices and Systems*, 144 (6):313–317, 1997.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL `http://arxiv.org/abs/1308.3432`.

Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2017.

Tanguy Chouard. The go files: Ai computer wraps up 4–1 victory against human champion. *Nature, doi*, 10, 2016.

William Dally. High-performance hardware for machine learning. In *Cadence ENN Summit*, 2016.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

Andrew Goldberg and Tomasz Radzik. A heuristic improvement of the bellman-ford algorithm. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1993.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *Proceedings of the 43rd International Symposium on Computer Architecture*, pp. 243–254. IEEE Press, 2016.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12. ACM, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Ashkan Hosseinzadeh Namin, Karl Leboeuf, Roberto Muscedere, Huapeng Wu, and Majid Ahmadi. Efficient hardware implementation of the hyperbolic tangent sigmoid function. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pp. 2117–2120. IEEE, 2009.

Peter Nilsson, Ateeq Ur Rahman Shaik, Rakesh Gangarajaiah, and Erik Hertz. Hardware implementation of the exponential function using taylor series. In *NORCHIP, 2014*, pp. 1–4. IEEE, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. *arXiv preprint arXiv:1712.01048*, 2017.