Rational SoftMax

Sina Baghal

Abstract

SoftMax operations appear ubiquitously in deep learning models. Traditionally, SoftMax was only used in the last layer of a neural network for generating the probabilities needed in classification tasks. As such, its efficient hardware implementation has been largely neglected. However, with the advent of the Transformer-based models, non-matrix, non-linear operations, such as SoftMax, are becoming one of the main computational burdens when working with deep neural networks and thus, rethinking their implementation is of great importance. In this work, we will present a novel formulation of the SoftMax function called Rational SoftMax which does not require any exponentiation. We will show the complexity of Rational SoftMax is considerably lower in comparison with vanilla SoftMax. Moreover, we will present a fixed-point quantized version of Rational Softmax (called FiSoftMax) which produces ideal accuracy using only the optimal number of bits required for classification *i.e.,* $\lceil \log_2 c \rceil$ where c is the number of classes. Our experiments also demonstrate that both Rational SoftMax and FiSoftMax yield baseline accuracy when used in Resnet18 for Cifar10, Cifar100 and TinyImageNet datasets.

Introduction

SoftMax layer is a key component in different deep learning architectures. However, since it contains exponentiation, its calculation is reasonably high. As such, designing efficient and hardware-aware implementation of SoftMax function has recently attracted researchers' attention (Gao, Liu, and Lombardi 2020; Du et al. 2019; Ham et al. 2020; Zhu et al. 2020; Stevens et al. 2021). The main contribution of this work is to present a novel, exponentiation-free, computationally inexpensive, formulation of SoftMax called RaSoftMax. We will also present FiSoftMax_q which is a fixed-point quantization version of RaSoftMax. We present numerical results in support of the accuracy of RaSoftMax and FiSoftMax_q.

Notation Throughout, c-dimensional Euclidean real space is denoted by $\mathbb{R}^c$. Vectors are denoted using bold letters *e.g.,* $\boldsymbol{x}$. A multiply operation (henceforth denoted by m op) refers to taking the product of two real numbers. Similarly div op and add op are defined. We denote by Round, the round to the nearest integer function, *e.g.,* $\text{Round}(2.3) = 2, \text{Round}(2.7) = 3$.

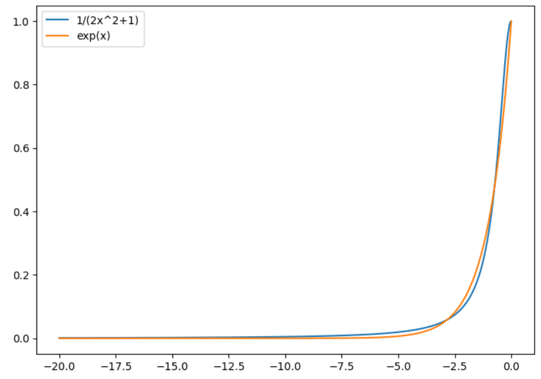

Figure 1: $\exp(t)$ for $t \leq 0$ and its comparison with $\frac{1}{1+2t^2}$

RaSoftMax

SoftMax SoftMax operator is defined as below.

$$\text{SoftMax}(\boldsymbol{x})_i := \frac{\exp(x_i)}{\exp(x_1) + \cdots + \exp(x_c)}, \quad \forall \boldsymbol{x} \in \mathbb{R}^c.$$

Notice that

$$\exp(\boldsymbol{x} + a) = \exp(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathbb{R}^c \text{ and } a \in \mathbb{R}. \tag{1}$$

Appealing to (1), we will always assume that $\boldsymbol{x} \leq \boldsymbol{0}$. This could be accomplished via a simple linear transformation, namely:

$$x_i \leftarrow x_i - x_{\max}. \tag{2}$$

Figure 1 plots the $\exp(t)$ for $t \leq 0$. Hence, in the computation of the SoftMax layer, after translation (2), we only need to compute $\exp(t)$ where $t \leq 0$. We next make the following important observation.

$$\frac{1}{1+2t^2} \approx \exp(t), \quad \forall t \leq 0. \tag{3}$$

In view of (3), we introduce RaSoftMax as below:

$$\text{RaSoftMax}(\boldsymbol{x})_i := \frac{\frac{1}{1+2x_i^2}}{\frac{1}{1+2x_1^2} + \cdots + \frac{1}{1+2x_c^2}}, \quad \forall \boldsymbol{x} \in \mathbb{R}^c_{\leq 0}.$$

Remark 1 *It is emphasized that we always assume that* $x_{\max} = 0$ *when using RaSoftMax.*

Computational complexity Let $\boldsymbol{x} \in \mathbb{R}^c$ and denote

$$y_i := 1 + 2x_i^2, \quad \forall i = 1, \cdots, c.$$

In order to compute RaSoftMax, we need to compute $\frac{1}{y_i}$ which will be computationally expensive. We will simplify RaSoftMax computation as follows: Notice that

$$\text{RaSoftMax}(\boldsymbol{x})_i = \frac{y_1 \cdots \hat{y_i} \cdots y_c}{\sum_{k=1}^{c} y_1 \cdots \hat{y_k} \cdots y_c}$$

Lemma 1 *Having* $y_1, \cdots, y_c$ *at hand,* c *values* $\{y_1 \cdots \hat{y_k} \cdots y_c\}_{k=1}^{c}$ *are computed using only* $2c - 3$ *m ops.*

Proof (outline): Suppose that Alg_j computes $\{y_1 \cdots \hat{y_k} \cdots y_j\}_{k=1}^{j} \cup \{y_1 \cdots y_j\}$ in a_j m ops. Denote $\tilde{y}_j := y_j y_{j+1}$. Using Alg_j, we may compute $\{y_1 \cdots \hat{y_k} \cdots \tilde{y}_j\}_{k=1}^{j} \cup \{y_1 \cdots \tilde{y}_j\}$ in a_j m-m ops. To obtain $\{y_1 \cdots \hat{y_k} \cdots y_{j+1}\}_{k=1}^{j+1} \cup \{y_1 \cdots y_{j+1}\}$, it suffices to perform two additional m-m op, namely $y_1 \cdots y_{j-1} \cdot y_j$ and $y_1 \cdots y_{j-1} \cdot y_{j+1}$. Hence, $a_{j+1} = a_j + 2$. Since $a_2 = 1$, the claim follows. $\square$

Lemma 2 *For* $\boldsymbol{x} \in \mathbb{R}^c$*, RaSoftMax*$(\boldsymbol{x})$ *can be computed using* $4c, 1$ *and* $3c$*, m, div and add ops respectively.*

Proof : Easily follows from Lemma 1. $\square$

Fixed Point Quantization

Let $z \in [0, 1]$ and fix positive integer q. Fixed point quantization of z using q bits is defined as follows.

$$Q_q(z) := \frac{a}{2^q} \text{ where } a := \text{Round}(2^q \cdot z).$$

We define a fixed point version of RaSoftMax using q bits as follows: Let $z_i := \frac{1}{1+2x_i^2}$. Clearly, $0 \leq z_i \leq 1$. Denote

$$a_i := \text{Round}(2^q \cdot z_i) \text{ thus } Q_q(z_i) = \frac{a_i}{2^q}.$$

After computing $a_1, \cdots, a_c$, we compute $\sum_{i=1}^{c} a_i$. Finally, denote

$$\frac{b_i}{2^q} := Q_q\left(\frac{a_i}{a_1 + \cdots + a_c}\right) \quad \forall i = 1, \cdots, c.$$

Define FiSoftMax_q as follows.

$$\text{FiSoftMax}_q(\boldsymbol{x}) := \left(\frac{b_i}{2^q}\right)_{i=1}^{c}.$$

Notice that $0 \leq b_i \leq 2^q$ for all i.

Experiments

We trained the Resnet18 model where the last SoftMax layer is replaced with RaSoftMax and also FiSoftMax_q where q is to be determined. It should be clear that when training Resnet, in the forward pass, there is no need to compute $\log \text{SoftMax}(\boldsymbol{w})$ as its value is not used anywhere. However, in the backward pass, we have that

$$\nabla \log \text{SoftMax} = \text{SoftMax} - y (= \text{Lables}) \tag{4}$$

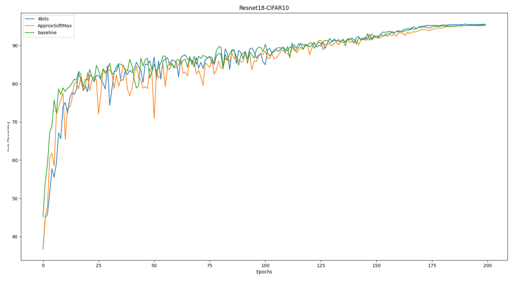

Figure 2: Resnet18 training for Cifar 10 using FiSoftMax_4 (blue). The baseline SoftMax (green) and RaSoftMax (orange) are also plotted here.

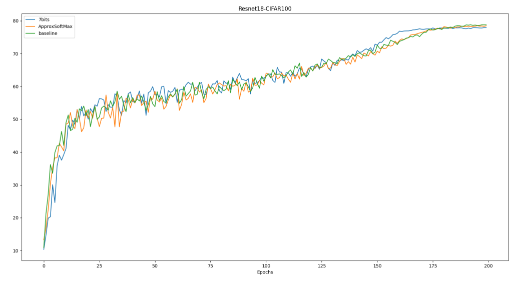

Figure 3: Resnet18 training for Cifar 100 using FiSoftMax_7(blue). The baseline (green) and full precision RaSoftMax (orange) are also plotted.

SoftMax in (4) is replaced with RaSoftMax or FiSoftMax in (4) in our experiments.

Our experiments show that RaSoftMax does not yield any test accuracy drop while FiSoftMax_q yield the same as long as $q \geq \lceil \log_2 c \rceil$ where c is the number of classes. So for example, for the Cifar10 dataset, 4 bits, for Cifar100, 7 bits are required in quantization of SoftMax and for TinyImageNet dataset (200 classes), 8 bits. Less number of bits used in FiSoftMax resulted in the random choice accuracy.

References

Du, G.; Tian, C.; Li, Z.; Zhang, D.; Yin, Y.; and Ouyang, Y. 2019. Efficient softmax hardware architecture for deep

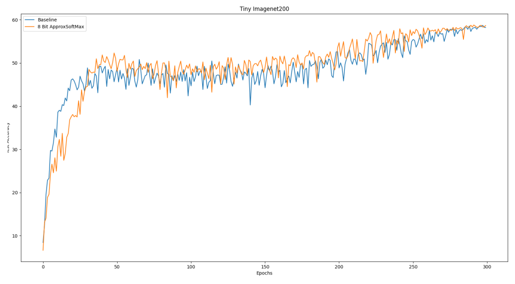

Figure 4: Resnet18 training for Tiny ImageNet containing 100000 images from 200 classes (500 for each class). The plot shows the baseline SoftMax (blue) versus FiSoftMax_8 (orange).

neural networks. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, 75–80.

Gao, Y.; Liu, W.; and Lombardi, F. 2020. Design and implementation of an approximate softmax layer for deep neural networks. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5. IEEE.

Ham, T. J.; Jung, S. J.; Kim, S.; Oh, Y. H.; Park, Y.; Song, Y.; Park, J.-H.; Lee, S.; Park, K.; Lee, J. W.; et al. 2020. Aˆ 3: Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 328–341. IEEE.

Stevens, J. R.; Venkatesan, R.; Dai, S.; Khailany, B.; and Raghunathan, A. 2021. Softermax: Hardware/Software Co-Design of an Efficient Softmax for Transformers. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 469–474. IEEE.

Zhu, D.; Lu, S.; Wang, M.; Lin, J.; and Wang, Z. 2020. Efficient precision-adjustable architecture for softmax function in deep learning. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(12): 3382–3386.