PhD Forum: Why TanH can be a Hardware Friendly Activation Function for CNNs

Kamel Abdelouahab, Maxime Pelcat, François Berry

PhD Forum: Why TanH can be a Hardware Friendly Activation Function for CNNs

Kamel Abdelouahab[1], Maxime Pelcat[1,2], and François Berry[1]

[1]*Institut Pascal, Univérsite Clermont Auvergne, France*
[2]*IETR, INSA Rennes, France*

***Abstract*—Convolutional Neural Network (CNN) techniques improved accuracy and robustness of machine vision systems at the price of a very high computational cost. This motivated multiple research efforts to investigate the applicability of approximate computing and more particularly, fixed point-arithmetic for CNNs. In all this approaches, a recurrent problem is that the learned parameters in deep CNN layers have a significantly lower numerical dynamic range when compared to the feature maps. This problem prevents from using of a low bit-width representation in deep layers. In this paper, we demonstrate that using the TanH activation function is way to prevent this issue. To support this demonstration, three benchmark CNN models are trained with the TanH function. These models are then quantized using the same bit-width across all the layers. Efficiency of this method is demonstrated on an FPGA based accelerator, by inferring CNNs with the minimal amount of logic elements**[1]

I. Introduction

Convolutional Neural Networks (CNNs) [2] have become the dominant approach in a variety of computer vision tasks. However, a major drawback of such techniques remains their heavy computational workload. Convolutional stages are the most computationally intensive parts of a CNN and are responsible – in a typical implementation – for more than 90% of the CNN execution time [3]. To accelerate the execution of convolutional parts, multiple approaches advocate the use of approximate computing with fixed point numerical representation of data [4], [5], [6]. However, and as pointed in [4], distinct parts of a CNN have a significantly different Dynamic Range (DR) of data and parameters. More particularly, the network learned parameters tend to be much smaller than the layer outputs especially in the deep layers. As a result, CNNs implementations based on approximate computing resort to dynamic fixed point [4], [6], mini-float [6] ,or long word-length fixed point representation of data [7], [8].

A possible way to address this issue is forcing CNN outputs to be within a limited data-range that matches the DR of parameters. In this paper, the numerical dynamic range of CNN activations is constrained at the output of each conventional layer using the hyperbolic tangent TanH. As a consequence, CNNs can be deployed with the same bit-width across all the layers.

The rest of this paper is organized as follows. Section II recalls the main features of CNNs from a computational point of view, focusing on those related to fixed-point computing and numerical dynamic range. Section III shows how activation functions such *Sigmoid* or *TanH* benefits to fixed-point computing of CNNs. An approximate version of the *TanH* function is described in this section and is used to build Field-Programmable Gate Array (FPGA)-based accelerators for the Cifar10 CNN. Section IV reports resource utilization of these accelerators.

[1]This is a pre-print version. Please refer to final paper in [1]

II. Fixed point computation of CNNs

A typical CNN structure features a deep succession of (L) convolutional layers where a set of 3D convolution kernels carries the feature extraction process. Each output of a given layer involves a multiplication of an input Φ with a kernel Θ, an summation of $C \times K^2 + 1$ terms –where C is the number of inputs and K the size of the convolution kernel–. When using a fixed-point implementation, the bit-width required to cover all the numerical range of the latter processing can be written as follows:

$$b_f = b_\Theta + b_\Phi + \log_2(CK^2 + 1) \tag{1}$$

Where:

- b_f : Bit-width of the layer output
- b_Φ : Bit-width of the learned parameters
- b_Θ : Bit-width of the layer input

This Multiply-ACcumulate process is followed by the application of a non-linear activation function. The ReLU function is wildly used in recent CNN topologies as it grants faster convergence during training and less complex gradient computation [9].

As shown in Fig 1-d, ReLU is defined from $\mathbb{R}$ to $\mathbb{R}^+$. Thus, when applying this activation, the DR of CNNs is halved as all the activations have positive values. As a result, the number of bits required to represent the activations is 1 bit lower ($b_a = b_f - 1$).

When stacking multiple layers, and since the ReLU function has no upper threshold, the inputs of a layer (l) have *the same dynamic range* as the outputs of layer $(l-1)$, as shown in equation 2. Consequently, the numerical range of activations increases over layers. In other words, deeper is the layer, wider is the DR of its activations. This is especially true in the last layers of CNNs where the activations have a significantly larger dynamic range than the parameters DR. As a result,

an expansion of the bit-width is required to keep tolerable precision and prevent overflow.

$$\forall l = 1 : L$$
$$b_a^{(l)} = b_a^{(l-1)} + b_\Theta^{(l)} + \log_2\left[C^{(l)}K^{(l)2} + 1\right] - 1 \quad (2)$$

With the number of bits increasing, the resource utilization of a CNN hardware accelerator grows. For instance, in the case of FPGAs, the hardware cost of fixed point multiplication increases quadratically with the bit-width [10] leading the deep layers of a CNN (the ones with the largest bit-width) to occupy the majority of resources.

A. Controlling DR with activation functions

Training CNNs with ReLU non-linearities is responsible of the expansion of the dynamic range of data, and thus, the bit-width and the resource utilization of CNN accelerator. Using activation functions with thresholds (cf Fig 1-d) such the TanH or Sigmoid cancels this propagation of DR over layers as the outputs of each activation are bounded with two thresholds. As a result, the DR of the inputs of a given layer is constant regardless of the depth of this layer. In this case, and by contrast with equation 2, the DR of CNN outputs using TanH activations can be written as follows:

$$\forall l = 1 : L$$
$$b_a^{(l)} = b_\Phi^{(l)} + b_\Theta^{(l)} + \log_2\left[C^{(l)}K^{(l)2} + 1\right] \quad (3)$$

III. Hardware friendly activation functions

A. Training CNNs with constrained activation functions

Using the Caffe deep learning framework[11], three benchmark CNNs are trained with different activation functions and learning curves are reported in figure 1. For the three networks, ReLU activation function delivers the fastest convergence rates during training. For the SVHN [12], using the TanH function leads to a 2,93% loss in classification accuracy while for the Cifar10 [13], 6% of classification accuracy is lost when compared to the ReLU model. Finally, for LeNet5 [14], both ReLU and TanH models delivers the same accuracy of 99.06%

B. Hardware implementation of TanH

Figure 2-a illustrates a piece-wise approximation of TanH function used to infer CNNs. It is based on a function that either applies a threshold or a linear transformation of the output. The key advantage of such an approximation is that it can be implemented using a small amount of logic elements. As shown in figure 2-b, four comparators are used to check the value of the input which controls a multiplexer that will either output: a value of 1, a value of -1, a right arithmetic shift of the inputs (division by two) or the original value.

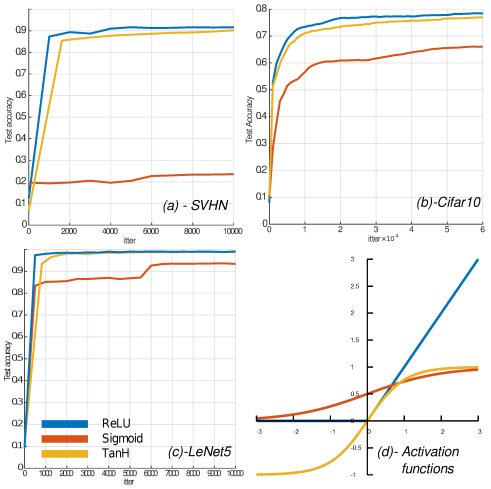

Fig. 1. (a,b,c) Learning curves of 3 benchmark CNNs trained with different activation functions plotted in (d)

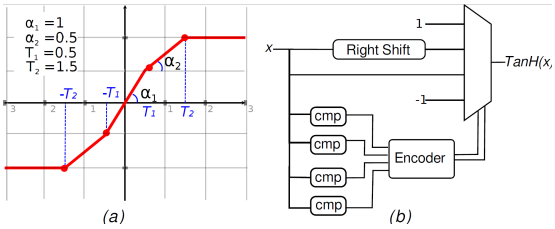

Fig. 2. TanH function: a-Approximation, b-Implementation

IV. Results

In order to evaluate the impact of using TanH function on hardware resources, a small CNN graph with 3 layers is mapped on a Cyclone V FPGA. Each layer respectively involves 96, 1024 and 2048 convolutions of size 5×5. Moreover, Three bit-widths are explored and, at each layer, the mean number of logic elements instantiated to map a single convolution is reported in figure 3. For this experiments, neither dynamic fixed point arithmetic nor floating point have been used.

Fig. 3 shows how logic elements required to map a con-

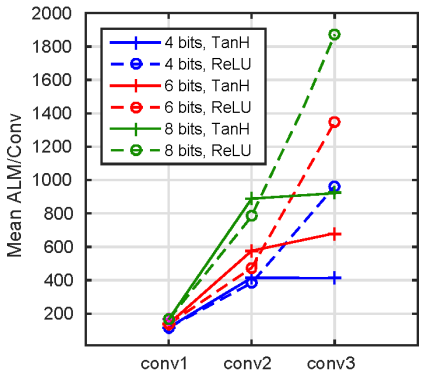

Fig. 3. Logic resources Implemented per convolution across a 3 layer CNN

volution grows when using the ReLU activation. By contrast, applying hyperbolic tangent at the output of each layer constrains the DR of activations and prevents from expending the bit-width in the deeper layers.

REFERENCES

[1] Kamel Abdelouahab, Maxime Pelcat, and Franois Berry. Why TanH is a Hardware Friendly Activation Function for CNNs. In *Proceedings of the 11th International Conference on Distributed Smart Cameras - ICDSC 2017*, pages 199–201, New York, New York, USA, 2017. ACM.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[3] Jason Cong and Bingjun Xiao. Minimizing computation in convolutional neural networks. In *Proceedings of the 24th International Conference on Artificial Neural Networks - ICANN '14*, pages 281–290. Springer, 2014.

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. *arXiv e-print*, 12 2014.

[5] Suyog Gupta, Ankur Agrawal, Pritish Narayanan, Kailash Gopalakrishnan, and Pritish Narayanan. Deep Learning with Limited Numerical Precision. In *Proceedings of the 32nd International Conference on Machine Learning - ICML '15*, pages 1737–1746, 2015.

[6] Philipp Gysel, Mohammad Motamedi, and Soheil Ghiasi. Hardware-oriented Approximation of Convolutional Neural Networks. In *arXiv preprint*, page 8, 2016.

[7] Murugan Sankaradas, Venkata Jakkula, Srihari Cadambi, Srimat Chakradhar, Igor Durdanovic, Eric Cosatto, and Hans Peter Graf. A Massively Parallel Coprocessor for Convolutional Neural Networks. In *Proceedings of the 32nd IEEE International Conference on Application-specific Systems, Architectures and Processors*, pages 53–60. IEEE, 7 2009.

[8] C Farabet, C Poulet, J Y Han, Y LeCun, David R. Tobergte, and Shirley Curtis. CNP: An FPGA-based processor for Convolutional Networks. In *Proceedings of the 15th International Conference on Field Programmable Logic and Applications - FPL '09*, volume 53, pages 1689–1699, 2009.

[9] Alex Krizhevsky, Ilya Sutskever, Hinton Geoffrey E., and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems - NIPS'12*, page 19, 2012.

[10] Jean-Pierre David, Kassem Kalach, and Nicolas Tittley. Hardware Complexity of Modular Multiplication and Exponentiation. *IEEE Transactions on Computers*, 56(10):1308–1319, 10 2007.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the ACM International Conference on Multimedia*, 2014.

[12] Yuval Netzer and Tao Wang. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems - NIPS'11*, page 5, 2011.

[13] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.

[14] Y LeCun, L Bottou, Y Bengio, and P Haffner. Gradient Based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.