Conjugate Adder Net (CAddNet) - a Space-Efficient Approximate CNN

Lulan Shen
McGill University
Montreal, Canada

Maryam Ziaeefard
McGill University
Montreal, Canada

Brett Meyer
McGill University
Montreal, Canada

Warren Gross
McGill University
Montreal, Canada

James J. Clark
McGill University
Montreal, Canada
james.j.clark@mcgill.ca

Abstract

The AdderNet was recently developed as a way to implement deep neural networks without needing multiplication operations to combine weights and inputs. Instead, absolute values of the difference between weights and inputs are used, greatly reducing the gate-level implementation complexity. Training of AdderNets is challenging, however, and the loss curves during training tend to fluctuate significantly. In this paper we propose the Conjugate Adder Network, or CAddNet, which uses the difference between the absolute values of conjugate pairs of inputs and the weights. We show that this can be implemented simply via a single minimum operation, resulting in a roughly 50% reduction in logic gate complexity as compared with AdderNets. The CAddNet method also stabilizes training as compared with AdderNets, yielding training curves similar to standard CNNs.

1. Introduction

Deep learning techniques have revolutionized many information processing tasks. But these methods come with a heavy computational burden, providing a challenge for embedded systems engineers who would like to use these on edge devices with restrictive computational and energy constraints. This has led to the development of compressed networks, which are smaller versions of standard deep neural networks. Even these compressed networks have significant computational demands. In trying to reduce the computational burden even further, designers have looked towards approximation techniques [6]. One example is the recently proposed *AdderNet*, which replaces the multiplications found in convolution operations by addition (actually subtraction) operations followed by absolute values. This results in a reduction in the logic complexity of special purpose implementations. In this paper, we modify the AdderNet approach by using a related, yet simpler, approximation to multiplication that involves addition and absolute value operations. As we will see, this not only reduces circuit complexity, but also provides a more stable training process than the AdderNet approach.

2. Adder Networks

An important component of deep neural networks is the discrete cross-correlation operation:

$$Y(m,n) = \sum_{i=0}^{d}\sum_{j=0}^{d}\sum_{k=0}^{c} X(m+i,n+j,k)*W(i,j,k) \quad (1)$$

In the deep learning literature this is usually referred to as "convolution", although strictly this requires the sign of the indices i and j to be negated in the X term. Here X are the neuron inputs and W are the neuron weights, which are learned.

Chen et al ([2,3,10]) introduced Adder Networks, which replace the convolutions in neural networks by sums of the absolute values of differences:

$$Y(m,n) = -\sum_{i=0}^{d}\sum_{j=0}^{d}\sum_{k=0}^{c} |X(m+i,n+j,k) - W(i,j,k)| \quad (2)$$

The form in equation (2) only requires addition and absolute value operations, whereas the convolution approach requires additions and multiplications. Hence the adder network will require less computing hardware than the convolutional network.

The Adder Network papers demonstrate that the form represented by equation (2) can yield neural network performance comparable to that of convolutional networks. The

Adder Network approach is not without some challenges, however. The first is that applying backpropagation of errors directly using equation (2) does not work well, as the gradients are all either +1 or -1. To counteract this problem, Adder Networks are actually trained using the gradients of the square of the absolute values. The second problem is that, unlike convolution, the outputs Y in equation (2) are always negative due to the absolute value. The Adder Networks papers handle this by applying a batch normalization stage after the summation. As noted by Dong et al [4] the Adder Network exhibits a rather unstable test loss curve during training (see figure 3 later in this paper).

3. Conjugate Adder Multiplier

We propose an approach related to Adder Nets which better approximates the convolution operation, while still avoiding the need for multiplication operations.

The main idea is to replace the multiplication operation in a convolution by differences of even nonlinear functions of conjugate pairs $g(X+W) - g(X-W)$ where $g()$ is an even convex function.

The key point here is that, because of the even order of the nonlinearities, the difference of the nonlinear functions is, to lowest order, proportional to the product of X and W:

$$X * W \approx g(X+W) - g(X-W) = \\ a_{11} X * W + a_{13} X * W^3 + a_{31} X^3 * W + ... \tag{3}$$

If we take the nonlinearity to be the square, $g(x) = x^2/4$, then the higher order terms disappear leaving only the $X * W$ term with $a_{11} = 1$, and equation (3) will be equivalent to the convolution equation (2). This implementation of multiplication as the difference of squares is well known as the "Quarter-Square Multiplier" [11], and has been used to create space-efficient analog neural network circuits [9]. For other types of even nonlinearities there will be some difference from $X * W$, and so equation (3) will not be exactly equivalent to a convolution in these cases. However, equation (3) may be a close approximation to a convolution, and adaptation of the weights W during network optimization may result in good performance of the network as compared with a convolutional network, much as is the case for adder networks.

The most direct implementation is to use the quarter-square function as is, and use this as a direct drop-in replacement for the multiplication operations in a CNN. The advantage of this is that a circuit-level implementation will be more compact than creating a fully parallel two-operand multiplier. A lookup table can be used to implement a quantized version of the square function. For low-bit-width representations these lookup tables can be relatively small compared to that of a two-operand multiplication, since they only have one input dimension. For larger bit-widths, however, lookup tables become impractically large and power hungry.

Piecewise linear functions could be used to allow for good approximations to the squaring operations over a given range, avoiding the need for multiplications, needing only addition/subtraction and/or comparison operations. The simplest piecewise linear implementation comes from using an absolute value operation as the nonlinearity: $g(x) = |x|/2$.

$$X * W \approx \frac{|X+W| - |X-W|}{2} \tag{4}$$

This is seen to be very similar to the Adder Network approach (equation 2) except that there is an additional term involving the absolute value of the sum of the input and weight, whereas the Adder Network has just the absolute value of the difference of the input and weight. We refer to equation (4) as the *Conjugate Adder Multiplier*, or *CAdd*, as it involves the difference of absolute values of the conjugate pair $(X+W), (X-W)$.

It may appear that the Conjugate Adder approach is more computationally expensive than the Adder Networks, as it seems to require three addition operations as compared to the single addition of the Adder Network. However, it can be shown that one can implement the difference of the absolute value of conjugate pairs with *no additions* at all, only a single comparison operation. To see this, we consider the various cases as to the signs of X and W and their relative magnitudes.

$$\begin{aligned} \mathrm{sgn}(X) = \mathrm{sgn}(Y) \Rightarrow & |X+Y| = |X| + |Y| \\ & |X-Y| = |\,|X| - |Y|\,| \\ \mathrm{sgn}(X) \neq \mathrm{sgn}(Y) \Rightarrow & |X+Y| = |\,|X| - |Y|\,| \\ & |X-Y| = |X| + |Y| \end{aligned} \tag{5}$$

Combining these four identities we can see that

$$|X+Y| - |X+Y| = \mathrm{sgn}(X)\,\mathrm{sgn}(Y)\{|X| + |Y| - |\,|X| - |Y|\,|\} \tag{6}$$

Next, we consider the relative magnitudes of X and Y:

$$\begin{aligned} |X| > |Y| \Rightarrow & |\,|X| - |Y|\,| = |X| - |Y| \\ \Rightarrow & |X+Y| - |X-Y| \\ = & 2\,\mathrm{sgn}(X)\,\mathrm{sgn}(Y)|Y| = 2Y\,\mathrm{sgn}(X) \end{aligned} \tag{7}$$

$$\begin{aligned} |X| < |Y| \Rightarrow & |\,|X| - |Y|\,| = |Y| - |X| \\ \Rightarrow & |X+Y| - |X-Y| \\ = & 2\,\mathrm{sgn}(X)\,\mathrm{sgn}(Y)|X| = 2X\,\mathrm{sgn}(Y) \end{aligned} \tag{8}$$

We can merge these two results into a single expression as follows:

$$|X+Y| - |X-Y| = 2\,\mathrm{sgn}(X)\,\mathrm{sgn}(Y)\min(|X|, |Y|) \tag{9}$$

Thus, we see that we can compute the difference of the absolute values of the conjugate pairs merely with a single comparison or minimum operation (assuming the sign operations come for free through the sign bits of the operands).

The Conjugate Adder multiplication is functionally equivalent to the idealized diode ring modulator circuit, which is often used to multiply analog signals [5, 8]. Figure 1 shows the output of the CAdd operation acting on two sinusoidal inputs. It looks quite similar to the action of the standard multiplication operation for these inputs. This is misleading however, as the CAdd operation is very nonlinear and if one of the inputs was scaled quite differently than the other then the output would be saturated or clipped. In using this type of approximation to the multiplication operation it is important that the two operands be scaled properly.

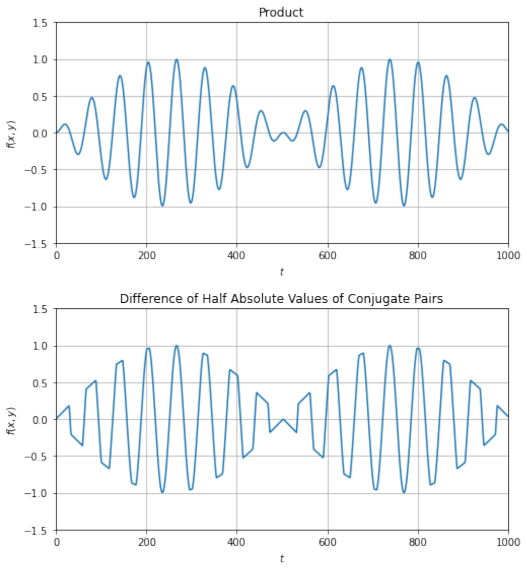

Figure 1. Standard multiplier output (left) and Conjugate Adder multiplier output (right) for 2 sinusoids

To evaluate the accuracy of the CAdd approximation to the multiplication operation, we performed a Monte-Carlo analysis of the Mean Relative Error Distance (MRED), which is given by $\frac{1}{N_s}\sum(\frac{|P-P_{approx}|}{P})$, where P is the true product value, P_{approx} is the approximate value and N_s is the number of sample cases contributing to the average. For 16-bit unsigned integers, the MRED is 0.425. This is higher than other approximate multipliers, such as the efficient logarithmic multiplier of Ansari et al [1], which has an MRED of 0.0289, but has a much lower circuit complexity.

We will refer to a convolutional neural network where the weight-input product is replaced by the Conjugate Adder Multiplier as a *CAddNet*. CAddNets are even more computationally efficient than AdderNets, in that the addition operation is replaced by a single comparison operation. In terms of logic gate complexity and power dissipation, a comparator circuit is less complex than an adder (roughly half as many logic gates are needed), as it can be implemented by computing just the sign bit of a subtraction operation, whereas a full subtraction also requires generation of all the sum bits. The speed of the operation will be similar in both cases, since the main determiner of speed in an adder circuit is its carry chain, which is also present in the comparator circuit.

Another advantage of CAddNets over Adder Networks is that the output of neurons can be both positive and negative. Also, the use of the standard convolution gradient signal during training is better justified in the case of CAddNets as compared with Adder Networks, as there is a closer approximation to the convolution operations. Thus the training process is expected to be closer to that of convnets in terms of learning rate schedules, and again avoids the need for batch normalization to tame the training process (although batch normalization may still be useful to speed up training, as it is for standard convolutional networks).

4. Experiments

We trained AdderNet and CAddNet versions of a standard ResNet20 convolutional neural network on the CIFAR-10 dataset [7]. The relative accuracy after training for 400 epochs with a batch size of 256 for CAddNets and of 128 for standard ResNet20 is shown in Table 1. We also include the accuracy for the binarized network of Zhou et al [12] for comparison. Note that [3] lists a higher accuracy value of 91.84 for the AdderNet version than our value of 91.54.

We follow the same experiment settings as the AdderNet paper [3] for training and testing except that the initial learning rate is set to 0.5 (for CAddNets and standard ResNet) and the hyper-parameter η is set to 0.7 (for CAddNets only). The CAddNet approach has the same drawback as the AdderNet in that the gradients of the output with respect to the weights and inputs are not very informative, yielding only sign information. For this reason we follow the practice in the AdderNet paper [3] and during backpropagation compute the gradients of the square of the output. As in the AdderNet paper, to prevent explosion of the gradients with respect to the inputs, we clamp these to a range of [-1,1]. Also, to have a fair comparison with the AdderNet paper [3] and the Binarized Net of [12], we use full precision standard convolution operations for the first and last layers of the ResNet.

Figure 2 shows that the classification accuracy of the CAddNet method increases more quickly, and with much less variability, than the AdderNet approach. Similarly, in figure 3 we see that the loss on the CIFAR-10 test set for the CAddNet is much less variable during training than for the AdderNet. The reason for this may be due to more informative gradient directions during early training for the CAddNet than for the AdderNet. Figure 4 shows that the histogram of weight values obtained after training is very similar for both the CAddNet and AdderNet. Dong et al [4] point out that the variance of the weights of a standard convolutional network is much lower than that of the AdderNet, and propose that this is what result in the highly variable test loss training curve for AdderNets as compared with convolutional nets. However, we see that the CAddNet has less test loss variability but still has high weight variance. So the explanation proposed by Zhou et al [12] for the test loss variability of the AdderNet cannot be completely correct. Further investigation and analysis is required to see where the stability provided by the CAddNet approach in spite of large weight variance arises.

Table 1. ResNet20 classification accuracy on CIFAR-10.

Method	Accuracy (%)
Standard ResNet	92.59
AdderNet	91.54
CAddNet	91.18
Binary Net	84.87

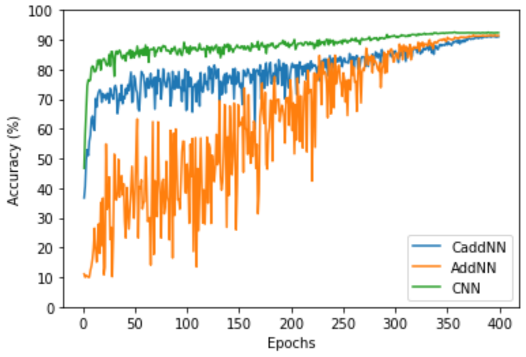

Figure 2. Accuracy of Standard ResNet, CAddNet and AdderNet during training.

5. Conclusions

In this paper we have presented a modified version of the AdderNet approach, where the single absolute value of the difference between the input and the weight is replaced by a conjugate pair of the absolute value of the sum and difference of the input and weight. Although this may seem to result in a more complex implementation, we show that it can be expressed in a simplified form that reduces the conjugate pair operation to a single minimum operation. Thus, the effective implementation complexity is roughly cut in half as compared with the AdderNet. We observe, after training, an accuracy on the CIFAR-10 dataset very close to that obtained by the AdderNet, but with a more rapid increase in accuracy with training, and exhibits less instability during training.

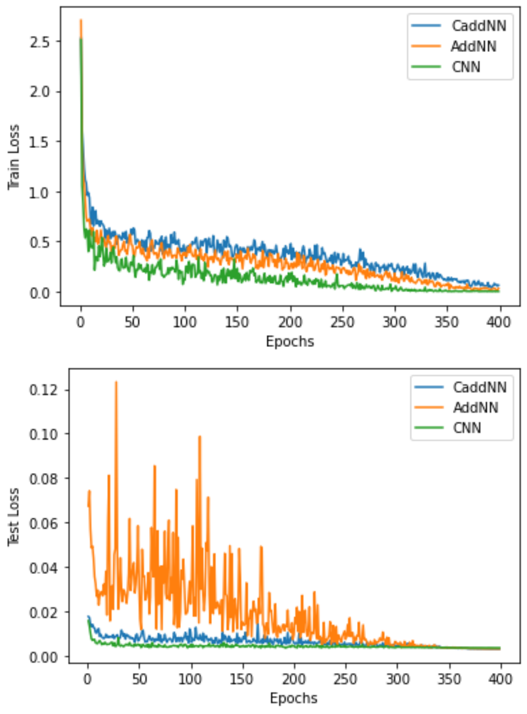

Figure 3. Training (top) and Test (bottom) loss curves for Standard ResNet, AdderNet and CAddNet.

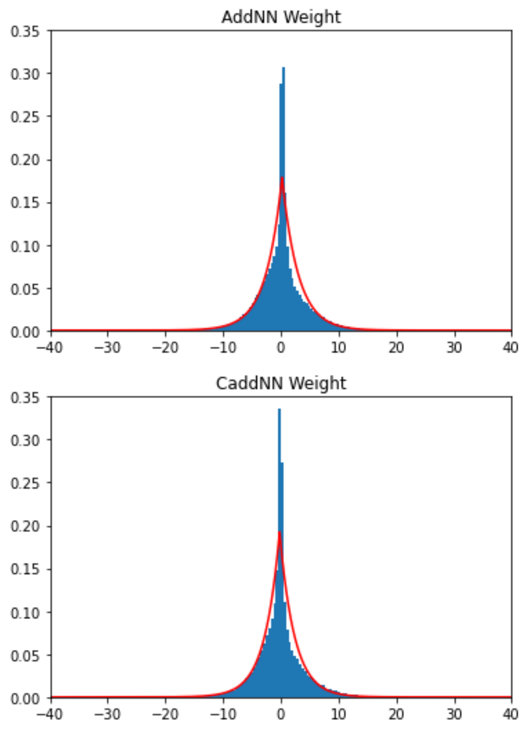

Figure 4. Histogram of learned weight values for AdderNet (top) and CAddNet (bottom).

References

[1] Ansari, M.S., Cockburn, B.F. and Han, J., "A hardware-efficient logarithmic multiplier with improved accuracy". In 2019 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 928-931, March 2019. 3

[2] Chen, H., Wang, Y., Xu, C., Xu, C., Xu, C. and Zhang, T., "Universal Adder Neural Networks," arXiv preprint arXiv:2105.14202, 2021. 1

[3] Chen, H., Wang, Y., Xu, C., Shi, B., Xu, C., Tian, Q. and Xu, C., "AdderNet: Do we really need multiplications in deep learning?" Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1468-1477, 2020. 1, 3

[4] Dong, M., Wang, Y., Chen, X. and Xu, C., "Towards stable and robust addernets." Advances in Neural Information Processing Systems, 2021. 2, 4

[5] Goodman, B., "Diode Modulators," QST Magazine, pp 39-43, April 1953. 3

[6] Han, J. and Orshansky, M., "Approximate Computing: An Emerging Paradigm For Energy-Efficient Design," in IEEE ETS, 2013. 1

[7] Krizhevsky, A. "Learning Multiple Layers of Features from Tiny Images." Technical Report, University of Toronto, 2009. 3

[8] Parker, J., "A Simple Digital Model of the Diode-Based Ring-Modulator," Proc. 14th Int. Conf. Digital Audio Effects, January 2011. 3

[9] Saxena, N. and Clark, J.J., "A four-quadrant CMOS analog multiplier for analog neural networks," IEEE Journal of Solid State Circuits, pp 746-749, June 1994. 2

[10] Wang, Y., Huang, M., Han, K., Chen, H., Zhang, W., Xu, C. and Tao, D., "AdderNet and its minimalist hardware design for energy-efficient artificial intelligence," arXiv preprint arXiv:2101.10015, 2021. 1

[11] Whitbread, J. and Glaisher, L., "The method of quarter squares," Nature, 40(1041), pp 573–576, 1889. 2

[12] Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H. and Zou, Y. "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients." arXiv preprint arXiv:1606.06160, 2016. 3, 4