CEST: Computation-Efficient *N:M* Sparse Training for Deep Neural Networks

Chao Fang[1], Wei Sun[2], Aojun Zhou[3], Zhongfeng Wang[1]
[1]ICAIS Lab, Nanjing University [2]Eindhoven University of Technology [3]CUHK-Sensetime Joint Lab, CUHK

***Abstract*—*N:M* fine-grained structured sparsity has attracted attention due to its practical sparsity ratio and hardware-friendly pattern. However, the potential to accelerate *N:M* sparse deep neural network (DNN) training has not been fully exploited, and there is a lack of efficient hardware supporting *N:M* sparse training. To tackle these challenges, this paper presents a computation-efficient scheme for *N:M* sparse DNN training, called CEST. A bidirectional weight pruning method, dubbed BDWP, is firstly proposed to significantly reduce the computational cost while maintaining model accuracy. A sparse accelerator, namely SAT, is further developed to neatly support both the regular dense operations and *N:M* sparse operations. Experimental results show CEST significantly improves the training throughput by $1.89-12.49\times$ and the energy efficiency by $1.86-2.76\times$.**

I. Introduction

N:M fine-grained structured sparsity [1]–[5], where only *N* out of consecutive *M* elements in a group can be nonzero, has attracted increasing attention due to its practical sparsity ratio as well as its hardware-friendly pattern. However, the potential to accelerate *N:M* sparse deep neural network (DNN) training has not been fully exploited, and there is a lack of efficient hardware supporting *N:M* sparse training. From the perspective of algorithm, previous works solely accelerate DNN training by introducing *N:M* sparsity in either forward [1] or backward pass [5], lacking a unified method to integrate *N:M* sparsity into both forward and backward passes for further DNN training speedup. From the perspective of architecture, the existing Ampere GPUs only supports static 2:4 sparse acceleration, which limits the speedup potential for DNNs leveraging more flexible *N:M* sparsity. Unlocking these missing opportunities, we propose an algorithm-hardware co-design training scheme for DNNs, called Computation-Efficient *N:M* Sparse Training (CEST), with innovations from the training algorithm and hardware architecture. The contributions can be summarized as follows:

- **Algorithm for *N:M* sparse DNN training**: We propose a bidirectional weight pruning method, namely BDWP, leveraging *N:M* sparsity during both forward and backward passes of DNN training. Compared to SDGP [5], BWDP reduces $2\times$ number of operations while the trained models achieve 5.77% higher accuracy.
- **Hardware architecture for DNN training**: We design a sparse accelerator for DNN training, namely SAT, to support computation-efficient *N:M* sparse operations besides the regular dense operations. It achieves $1.89-12.49\times$ higher throughput and $1.86-2.76\times$ greater energy efficiency than prior training accelerators [6]–[9].
- **Scheme for efficient DNN training**: We present CEST, an efficient scheme for DNN training that incorporates the BDWP algorithm and the SAT architecture. It improves the training speed by $1.93\times$ on average compared to the conventional dense training scheme deployed on SAT.

II. BDWP Algorithm

BDWP preserves values with the N most significant magnitude in each group of M elements. For a convolutional layer, BDWP removes pruned elements in each group across the input channels in the forward pass, and across the output channels in the backward pass, respectively. For a linear layer, BDWP is applied to each group across input features in the forward pass and across output features in the backward pass, respectively. The training process of BDWP is shown in Algorithm 1. Every iteration to train an L layer network consists of three steps. BDWP is first applied to generate *N:M* sparse weights, $\widetilde{w}_t$ and $\widetilde{w}_t^T$, for further computation in FF and BP, respectively. In FF, as shown in Line 3, the activations perform operations with the *N:M* sparse $\widetilde{w}_t$, which have been slimmed for $\frac{M}{N}$ times. In BP, as shown in Line 7, the activation gradients are obtained by performing operations with output gradients and $\widetilde{w}_t^T$. The other steps of the training process stay the same as the standard training flow.

Algorithm 1 Training an L layer network using BDWP

Input: A mini-batch of input activations and labels (x_t^0, y_t), current weights w_t, sparse ratio N and M.
Output: Updated weights w_{t+1}.
Step 1: Generate *N:M* sparse weights
1: $\widetilde{w}_t, \widetilde{w}_t^T \leftarrow$ BDWP(w_t, N, M).
Step 2: Forward Pass
2: **for** $l = 1$ to L **do**
3: $x_t^l \leftarrow$ **FF**(x_t^{l-1}, $\widetilde{w}_t^l$).
4: **end for**
5: Compute the gradient of the output layer $g_{a_t^L}$.
Step 3: Backward Pass
6: **for** $l = L$ downto 1 **do**
7: $g_{a_t^{l-1}} \leftarrow$ **BP**($g_{a_t^l}$, $\widetilde{w}_t^T$).
8: $g_{w_t^{l-1}} \leftarrow$ **WU**($(x_t^{l-1})^T$, $g_{a_t^l}$).
9: **end for**
10: Optimize w_{t+1} with momentum SGD.

III. SAT Hardware Architecture

SAT, as shown in Fig. 1, consists of three major computing engines: i) an *N:M* sparse tensor computing engine (STCE), ii) a weight update vector engine (WUVE), and iii) an online sparse reduction engine (OSRE).

978-3-9819263-7-8/DATE23/© 2023 EDAA

TABLE I
COMPARISON BETWEEN SAT AND PREVIOUS FPGA-BASED TRAINING ACCELERATORS

Accelerator	Platform	Network	Precision	Logic Util.	DSP Util.	Freq. (MHz)	Power (W)	Throughput (GOPS)	Comp. Effi. (GOPS/DSP)	Energy Effi. (GOPS/W)
SAT (this work)	**XCVU9P**	**ResNet-18**	**FP16+FP32**	**432K (37%)**	**1216 (18%)**	**180**	**17.94**	**299.86**	**0.25**	**16.71**
TODAES'22 [7]	ZCU102	VGG-16	FP32	N/A	1508 (60%)	100	7.71	46.99	0.03	6.09
ICCAD'20 [6]	Stratix 10 MX	VGG-like	FP16	221K (31%)	1046 (26%)	185	∼20.00	∼158.54	0.15	∼9.00
FPGA'20 [8]	Stratix 10	AlexNet	FP32	616K (66%)	1796 (31%)	253	N/A	∼24.00	0.01	N/A
FPT'17 [9]	ZU19EG	LeNet-10	FP32	329K (63%)	1500 (76%)	200	14.24	86.12	0.06	6.05

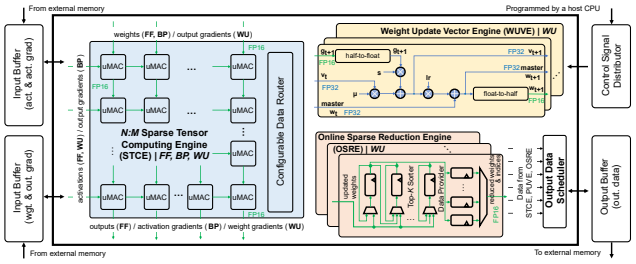

Fig. 1. The overall microarchitecture of SAT is composed of three computing engines, namely STCE, WUVE, and OSRE, respectively.

STCE unifies all the matrix multiplications of FF, BP, and WU steps, and supports both sparse and dense operations generated by BDWP (Line 3, 7, 8 in Algorithm 1), significantly boosting computational efficiency. There are 32x32 unified multiply-and-accumulate (uMAC) units orchestrated as a systolic array in STCE. A data router module is tightly coupled with the uMAC array and distributes the data to the uMAC array according to configurations of DNN layers.

WUVE is a dedicated optimizer for momentum stochastic gradient descent, capable of updating weights in an mixed-precision scheme (Line 10 in Algorithm 1). To minimize quantization errors in the WU step, WUVE specifically elevates the numerical precision of gradients from FP16 to FP32. Moreover, WUVE provides 32-parallel lanes to improve computational efficiency, and each lane consists of three FP32 multipliers, two FP32 adders, one FP16-to-FP32 switcher, and one FP32-to-FP16 switcher.

OSRE performs online BDWP sparse reduction operations by taking as input the dense weights in a group from WUVE, and generating as output the *N:M* sparse weights along with the corresponding indices (Line 1 in Algorithm 1). There are 32-parallel lanes with deep pipelines in OSRE, and each lane consists of a top-*K* sorter and a data provider. The top-K sorter sequentially receives dense data in a group with a size of *M*, and after *M* cycles, the *K* data sorted in the top-*K* sorter with their indices in the group are passed to the data provider.

IV. EXPERIMENTAL RESULTS

Accuracy Robustness of BDWP. We evaluate BDWP on the robustness of model accuracy in comparison with SDGP [5] and SRSTE [1] using ResNet9 on CIFAR10, ViT on CIFAR100, and ResNet18 on Tiny ImageNet, respectively. All networks are trained by an NVIDIA RTX 2080 Ti card with a batch size of 512. BDWP reduces 2× number of operations in comparison with SDGP and SRSTE under the same *N:M* sparsity ratio, and achieves comparable accuracy (merely 0.45% loss on average) to the dense baseline in 75% sparsity ratio. Compared to SDGP, BDWP is much more robust in training accuracy. Note that ViT and ResNet are distinct on model architectures, which indicates BDWP can be easily migrated to other DNNs with robust training accuracy.

Hardware Implementation of SAT. We have synthesized SAT on a Xilinx Virtex UltraScale+ VCU1525 card containing an XCVU9P FPGA. In our implementation, SAT works at 180 MHz and supports BDWP with 2:8 sparsity during training. Table I compares SAT with prior FPGA-based training accelerators for DNNs. SAT exhibits the highest throughput, computational efficiency, and energy efficiency. Compared to [6]–[9], SAT can improve the training throughput by $1.89-12.49\times$ and the energy efficiency by $1.86-2.76\times$.

Sparse Training Efficiency of CEST. For training ViT and ResNet18 models, we evaluate the training time deployed on SAT to achieve a validation accuracy of 59.5% and 62.0%, respectively. CEST (BDWP on SAT) is compared to training methods varying the conventional dense training, SDGP, and SRSTE. The required training time of CEST can be significantly reduced by 48.4% for ViT and 50.8% for ResNet18, respectively, when compared to that of the dense baseline. Moreover, CEST significantly outperforms SDGP and SRSTE by 35.7% on average in terms of training speed.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 62174084, 62104097, in part by the High-Level Personnel Project of Jiangsu Province under Grant JSSCBS20210034, and in part by Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant No. 149.

REFERENCES

[1] A. Zhou *et al.*, "Learning N:M Fine-grained Structured Sparse Neural Networks from Scratch," in *ICLR*, 2021.
[2] W. Sun *et al.*, "DominoSearch: Find Layer-wise Fine-grained N:M Sparse Schemes from Dense Neural Networks," *NeurIPS*, 2021.
[3] I. Hubara *et al.*, "Accelerated Sparse Neural Training: A Provable and Efficient Method to Find N:M Transposable Masks," *NeurIPS*, 2021.
[4] C. Fang *et al.*, "An Algorithm–Hardware Co-optimized Framework for Accelerating N:M Sparse Transformers," *TVLSI*, 2022.
[5] B. McDanel *et al.*, "Accelerating DNN Training with Structured Data Gradient Pruning," in *ICPR*, 2022.
[6] S. K. Venkataramanaiah *et al.*, "FPGA-based Low-batch Training Accelerator for Modern CNNs Featuring High Bandwidth Memory," in *ICCAD*, 2020.
[7] Y. Tang *et al.*, "EF-Train: Enable Efficient On-device CNN Training on FPGA Through Data Reshaping for Online Adaptation or Personalization," *TODAES*, 2022.
[8] K. He *et al.*, "FeCaffe: FPGA-enabled Caffe with OpenCL for Deep Learning Training and Inference on Intel Stratix 10," in *FPGA*, 2020.
[9] Z. Liu *et al.*, "An FPGA-based Processor for Training Convolutional Neural Networks," in *FPT*, 2017.