A Computational Efficient Architecture for Extremely Sparse Stereo Network

Tan Huang, Sih-Sian Wu, Jan Klopp, Po-Hsiang Yu, Liang-Gee Chen, *Fellow, IEEE*
DSPIC Lab, Department of Electrical Engineering
National Taiwan University, Taiwan
danhuang0313@gmail.com, benwu@video.ee.ntu.edu.tw, kloppjp@gmail.com, r08943024@ntu.edu.tw, lgchen@ntu.edu.tw

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401565

***Abstract*—CNN-based stereo matching methods achieve great performance but come with high computational requirements. Pruning a CNN can reduce the complexity but may in turn lead to memory conflicts, lowering throughput. Our proposed architecture and memory mapping technique aim at reducing conflicts to exploit extremely sparse stereo matching networks. To maintain a high utilization of processing elements, we decompose the de-convolution operation into several convolution operations. The proposed architecture provides a 2.1× speed up over SCNN. Compared to the software implementation, only 0.01% performance drop is observed, so that the proposed architecture obtains state-of-the-art accuracy compared to existing sparsity aware hardware implementations.**

***Index Terms*—sparsity-aware CNN, PE utilization, deconvolution, VLSI architecture, Memory mapping**

I. Introduction

Deep learning based methods have lead to a significant progress in stereo matching and come with much more complexity than hand-crafted stereo methods. Real-time stereo engines are crucial for many computer vision applications such as automatic driving, robotic and augmented reality (AR). However, existing AI engines are not able to deliver enough throughput to achieve real time processing for advanced stereo networks. One way to reduce complexity of CNNs is to prune weights. Although several sparsity aware architectures have been proposed to leverage the complexity reduction, they fail to preserve PE utilization under high sparsity.

Typical stereo matching networks first extract features and downscale them using strided convolutions. Unlike classification networks, whose features contract in the spatial dimension, extracted features are up-scaled by a deconvolution operation to regain the desired spatial extend. Although state-of-the-art stereo matching networks [3] have a remarkable accuracy, the computational complexity is two orders greater than what CNN accelerators targeting classification networks can provide.

To lower the complexity, we use [27] to compact the model and prune the weights to 95% sparsity with only 1.6% accuracy drop. Although the complexity can be reduced by 92%, the current most advanced sparse accelerator [22] can't efficiently exploit the sparsity because of frequent memory accesses. In addition, the deconvolution operation in the later layers yields an irregular, hardware-unfriendly computation pattern.

Since extremely sparse networks introduce memory conflicts that lower PE utilization, we design a memory hierarchy to reduce these conflicts to a level that is practical for a hardware implementation. To further improve the hardware efficiency, the fractional convolution operation (de-convolution) is transformed into multiple regular convolutions.

The contribution of this work is threefold:

1) A memory conflict reduction and speed up architecture for extremely sparse stereo matching networks
2) The sparsity-aware architecture supports strided convolution and de-convolution
3) Compared to state-of-the-art sparse aware architecture, the proposed architecture achieves more than 2 times speed up for various degrees of sparsity.

II. Related Works

Existing CNN-based hardware implementations mostly target on classification networks [14], [16], [19], [24], [26]. The optimization for CNN accelerators can be divided into three categories including optimization for computation [11], memory [5], [8], [9], and a data reuse scheme [2], [6], [7], [15], [23]. All above works focus on dense neural networks and cannot support stereo matching network with high complexity. A pruning technique is widely adopted to provide weight sparsity and further lower computational complexity for neural networks.

The pruning technique for stereo networks [13] is proposed to lower computational complexity with acceptable accuracy loss. Several works [1], [4], [7], [12], [20], [22], [25], [28] target on pruned models are proposed to deal with sparse and irregular computation pattern. Among them, SCNN [22] is the first architecture using Cartesian product to fully exploit both activation and weight sparsity. None of these works can provide enough throughput for stereo matching network because of high complexity and special operations. Although ASV [10] can achieve 30 fps with an interpolation technique for DNN method, it can't support high complexity networks that is more accurate. Our work can provide 10.36 fps by exploiting extremely sparse weight and achieve the best accuracy among those hardware implementations.

III. Proposed Architecture

The proposed architecture is a block-based chess memory mapping to efficiently support unit-stride, stride-2, and de-

978-1-7281-9201-7/21/$31.00 ©2021 IEEE

convolution operations.

A. Memory Conflict of Sparse Neural Network Engines

Reducing memory conflicts is crucial for designing accelerators targeting sparse neural networks because of the irregular dataflow when zero values are skipped. SCNN [22], our baseline architecture, uses the Cartesian product as their inner data flow. In each cycle, **F** filters and **I** inputs are loaded from buffer and fed to **F**×**I** multipliers to compute a Cartesian product. Then, the partial-sums computed from the multipliers are delivered to the accumulation unit according to the coordinates computed from the input's and filter's indices in the coordinate computation unit. That's because the multiplier outputs should be added to the corresponding partial sum in the output activation space. However, coordinates of outputs in the same cycle may be mapped to the same bank but a different address, which would result in a memory conflict. In addition, because a full read and store operation requires two cycles with single port memory, conflicts may also happen in adjacent cycles as shown in Fig. 1. If a memory conflict occurs, the accelerator with no additional buffer stalls until all data has been stored, which lowers efficiency. Although SCNN set number of accumulator bank A to be larger than F×I to reduce bank contention, tile sizes should be limited to 6×6 for best performance, which is so small that induces high percentage of halos. Hence, we propose a new memory mapping approach to reduce conflicts.

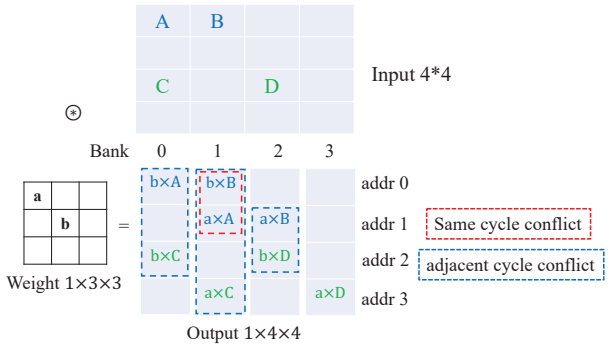
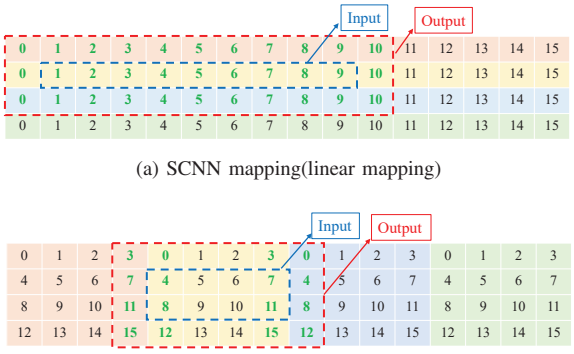


Fig. 1: An example of memory conflict

B. Memory Hierarchy Design for Convolution

We divide a layer's activations into a four layers: whole activation, tile, block and group. The whole activation is divided into tiles because the total workload for stereo matching network cannot be stored on-chip within acceptable memory size. Notice that the tile size is an important design parameter to be explored because it affects the halo ratio (additional computation at the tile overlap). Next, a tile is divided into several blocks to reduce memory conflict. Each block is split into four groups following a checker board pattern to support strides of 2 efficiently, which are commonly used in stereo NNs.

Fig. 2 shows an example of mapping a 16 × 4-tile to 16 banks. Given 50% input sparsity, eight inputs are on average



(a) SCNN mapping(linear mapping)

(b) Block mapping

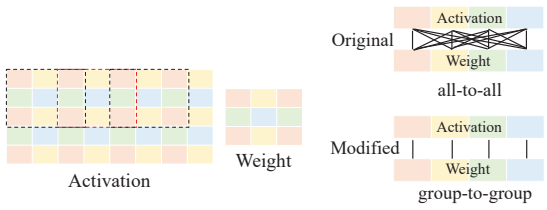
Fig. 2: An example of mapping 16 × 4 - tile to 16 banks with (a) the linear mapping and (b) the block-based mapping. Input and corresponding output ranges are marked in blue and red dashed boxes, respectively. A conflict happens while outputs are mapped to the same number with different colors.

required to obtain four non-zero inputs in a cycle. SCNN processes input in linear order and employs linear mapping (Fig. 2 (a)), which maps input and output data in a linear pattern. Inputs in a tile are compressed together and directly mapped to input banks linearly. For the output, the index of the accumulation bank is

$$(W_{tile} \cdot y + x) \mod N_{banks}. \quad (1)$$

Linear mapping may cause lots of conflicts because outputs in the same cycle have a high possibility to map to the same bank at different addresses as shown in Fig. 2. The origin of this is the mismatch of the 1-d linear mapping and the 2-d convolution kernel. In contrast, block mapping processes input in linear order within a block and maps input and output in several 4 × 4 blocks (Fig. 2 (b)), which can map outputs that span in 2 dimensions in different blocks. On the other hand, block mapping can make greater distance between the same number. Hence, memory conflict can be reduced. However, the block mapping cannot support 2-strided convolution which is frequently used in stereo NN to reduce resolution. Hence, we propose a memory mapping approach based on block mapping.

Convolutions with stride 2 are frequently used in stereo neural network for down-sampling. Conventional SCNN cannot support this operation, because it needs a special dataflow. If convolution with stride 2 is directly applied on SCNN, three quarters of the output are useless because a unit-strided convolution is operated and produces a full output that is 4 times larger than the intended down-sampled output. We design a new memory mapping approach to exploit the stride-2 computation pattern and eliminate the useless operations. As shown in Fig. 3(a), every color of weights is only multiplied with the same color of activations. We can divide activation and weight in checker board pattern into 4 groups and perform a group-to-group operation instead of an all-to-all one as shown in Fig. 3(b). Hence, no excess computations are performed with chess mapping and group-to-group operation.



(a) Stride 2 convolution (b) Operation

Fig. 3: The proposed chess memory mapping technique.

We apply modifications to the dataflow for both convolution configuration to enhance PE utilization. For stride 2 convolution processed in weight stationary order with chess mapping, we first fetch 4 groups of weights into registers and feed a group of 4 non-zero inputs in a cycle. Then, the weights in the same group as the activations are directed to multipliers. Hence, only products that contribute to final outputs are performed (Fig. 3). In addition, we change the group of activations every other cycle and keep the 4 weights in different groups in the buffer to reduce memory conflicts and reuse weights. Notice that memory conflicts can be reduced because adjacent groups of activations are likely to be located in the same window so that output will be directed to the same location. For unit-strided convolutions, conflicts may increase with chess mapping because the output range spans a larger area if 4 non zero inputs in the same group are fetched (Fig. 4). However, we observe that diagonal groups can be merged because if we process two merged groups alternately in adjacent cycles with any fixed group, output may map to same group in same cycle and different group in adjacent cycle. Hence, we can get the same range of outputs with block mapping by merging diagonal groups, which can have a higher PE utilization than SCNN's mapping. In addition, we can also reduce adjacent cycle conflict. The final mapping of ofmap is shown in Fig. 5.

C. De-convolution architecture design

Deconvolution is a common component in stereo NN, which contributes 38.2% percent of total convolution [10]. The standard process starts with ifmap upsampling by zeropadding followed by a convolution. The ifmaps that have been upsampled are not efficient on a general accelerator but suitable for sparse computation. As shown in [10], a kernel can be divided into 4 sub-kernels, which is same as our 4 groups in chess mapping. The property that output may contribute to 4 independent subset of output space can reduce memory conflict. By staggering same groups of weights in different cycle, PE utilization can be enhanced. The overall dataflow is shown in Fig. 6.

D. The proposed architecture

The overall architecture is based on SCNN [22]. To reduce conflict, a conflict unit is added in every accumulation bank in a PE as presented in Fig. 7(a). The conflict unit (Fig. 7(b)) consists of a conflict detection unit and a conflict buffer. The address of $F \times I$ products are compared and clustered in the conflict detection unit. Clustered data is then stored in the conflict buffer to be stored in accumulator bank in the successive cycles. There are 32 Accumulate (Acc.) banks in the proposed architecture as same as SCNN [22].

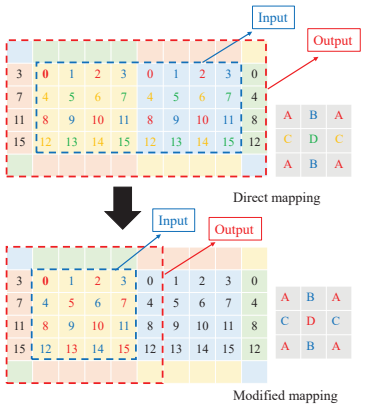


Fig. 4: Modified chess mapping on unit-strided convolution. Different groups of inputs are marked in different colors. For direct mapping, inputs are divided into 4 groups. For modified mapping, inputs are divided into 2 groups. i.e. green groups is merged into red group and yellow group is merge into blue group.

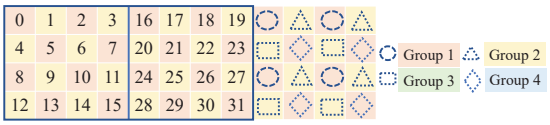


Fig. 5: Block-based chess mapping: all data is labeled into 4 groups. Every 8×4 block is mapped to 32 banks.

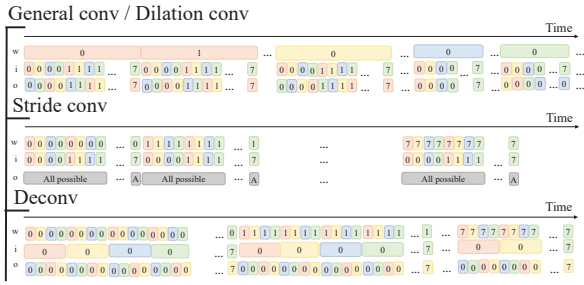


Fig. 6: Data flow for all operation

IV. Experimental Results

We implement our design in TSMC's 40 nm technology. We test these architectures with KITTI 2015 validation set [21] and the error refers to the 3-pixel error. Effects of each techniques on frame rate are listed in Table I. As the results

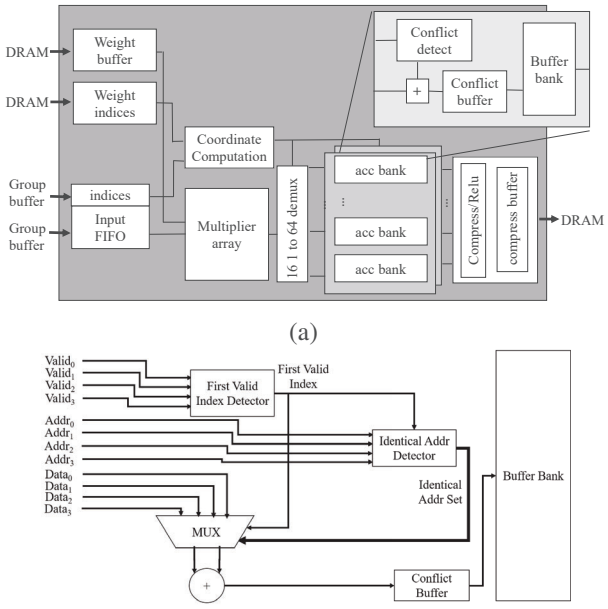


Fig. 7: (a)The proposed PE architecture which the conflict controlling unit is shown in the top right corner and (b) the detailed conflict detection unit.

indicate, applying chess mapping on 2-strided convolution and deconvolution has 1.23× speed up on overall performance with only one quarter of layers. The complete proposed architecture achieves a 2.1× speedup.

We compare our result with systolic array based accelerator [18] ("-Acc" suffix), ASV [10] and SCNN [22]. All accelerator are configured to 1024 multipliers and run in 400 MHz. As shown in Fig. 8, our architecture has an average of 2x speed up over SCNN across different degree of sparsity. Notice that exploiting sparsity can have better potential of speed up and outperform dense accelerator with 95% sparsity.

TABLE I: Operation ablation study

Conv	Strided Conv	Deconvolution	FPS	Speed up
			4.94	1.00
✓			8.39	1.70
✓	✓		9.62	1.95
✓	✓	✓	10.36	2.10

The specification of the proposed architecture in comparison with SCNN is listed in Table II. The proposed architecture can have higher throughput than SCNN [22] since SCNN suffers from the memory conflict in convolution, strided convolution and deconvolution.

In this work, we try to map the data flow in an efficient processing pattern that can fully utilize the computing resources. Although we find that bandwidth is also an important factor of overall performance, which is idealized in our work, there are some work that target on reducing bandwidth [17] which can

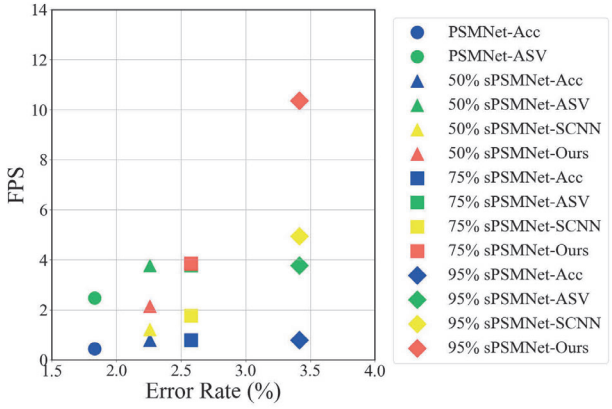


Fig. 8: Relations between frame rates and error rates.

TABLE II: Hardware implementations comparison.

	SCNN	SCNN (Implemented)	Proposed
Task	Classification	Stereo	Stereo
Technology	TSMC 16 nm	TSMC 40 nm	TSMC 40 nm
Frequency (MHz)	1000	400	400
MAC	1024	1024	1024
Input Size	224 × 224	960 × 540	960 × 540
Frame rate (fps)	N/A	4.94	**10.36**
Supported operation	convolution (s=1)	convolution (s=1)	**convolution (s=1,2) deconvolution**
3-px error (kitti2015)	N/A	3.85	3.85
On-chip Mem. (kB)	1000	1393	1161

be incorporated with our work. An interpolation technique is proposed by ASV [10] to speedup the computation, which is able to cooperate with our architecture. The proposed architecture achieves 17 % SRAM reduction by replacing input stationary with weight stationary. Noted that the size of Acc. banks is excluded in SCNN [22], which is included in our re-implemented version.

V. Conclusion

An architecture that can support large weight sparsity is proposed. The memory conflict slow down the speed of sparse neural network accelerator. In this paper, we proposed a chess mapping in accordance with the kernel's shape to reduce memory conflict. Furthermore, the architecture can process stride-2 and de-convolution efficiently. Compared to SCNN, the proposed architecture offers 2.1× speed-up and 17% SRAM reduction with 13% computational overhead.

References

[1] Jorge Albericio, Patrick Judd, Tayler Hetherington, Tor Aamodt, Natalie Enright Jerger, and Andreas Moshovos. Cnvlutin: Ineffectual-neuron-free deep neural network computing. *ACM SIGARCH Computer Architecture News*, 44(3):1–13, 2016.

[2] M. Alwani, H. Chen, M. Ferdman, and P. Milder. Fused-layer cnn accelerators. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, 2016.

[3] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[4] Huixiang Chen, Mingcong Song, Jiechen Zhao, Yuting Dai, and Tao Li. 3d-based video recognition acceleration by leveraging temporal locality. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 79–90, 2019.

[5] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGARCH Computer Architecture News*, 42(1):269–284, 2014.

[6] Y. Chen, T. Yang, J. Emer, and V. Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019.

[7] 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, 2016.

[8] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. Dadiannao: A machine-learning supercomputer. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622. IEEE, 2014.

[9] Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam. Shidiannao: Shifting vision processing closer to the sensor. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 92–104, 2015.

[10] Yu Feng, Paul Whatmough, and Yuhao Zhu. Asv: accelerated stereo vision system. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 643–656, 2019.

[11] V. Gokhale, A. Zaidy, A. X. M. Chang, and E. Culurciello. Snowflake: An efficient hardware accelerator for convolutional neural networks. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2017.

[12] 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. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

[13] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Kartik Hegde, Rohit Agrawal, Yulun Yao, and Christopher W Fletcher. Morph: Flexible acceleration for 3d cnn-based video understanding. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 933–946. IEEE, 2018.

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[17] Chao-Tsung Huang, Yu-Chun Ding, Huan-Ching Wang, Chi-Wen Weng, Kai-Ping Lin, Li-Wei Wang, and Li-De Chen. ecnn: A block-based and highly-parallel cnn accelerator for edge inference. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 182–195, 2019.

[18] 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*, pages 1–12, 2017.

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

[20] HT Kung, Bradley McDanel, and Sai Qian Zhang. Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 821–834, 2019.

[21] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[22] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally. Scnn: An accelerator for compressed-sparse convolutional neural networks. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 27–40, 2017.

[23] Marc Riera, Jose-Maria Arnau, and Antonio González. Computation reuse in dnns by exploiting input similarity. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 57–68. IEEE, 2018.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] Mingcong Song, Jiechen Zhao, Yang Hu, Jiaqi Zhang, and Tao Li. Prediction based execution on deep neural networks. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 752–763. IEEE, 2018.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[27] Po-Hsiang Yu, Sih-Sian Wu, Jan P. Klopp, Liang-Gee Chen, and Shao-Yi Chien. Joint pruning quantization for extremely sparse neural networks, 2020.

[28] Shijin Zhang, Zidong Du, Lei Zhang, Huiying Lan, Shaoli Liu, Ling Li, Qi Guo, Tianshi Chen, and Yunji Chen. Cambricon-x: An accelerator for sparse neural networks. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE, 2016.