Core^M: Accelerating Deep Learning Inference on Multi-Chip Architecture

Yifan Yang
MIT CSAIL
yifany@mit.edu

Ruicong Chen
MIT MTL
raychen@mit.edu

Zhen Guo
MIT RLE
zguo0525@mit.edu

***Abstract*—Deep Learning (DL) has many successes in image classification and natural language processing. However, real-time/high performance inference is still a challenge for the DL community, limiting the developments of latency sensitive and performance hungry systems like autonomous driving [20], drones, augmented reality/virtual reality (AR/VR). Efforts in DL accelerator have been made to scale single chip design to multi-chip efficiently to further improve inference performance. To this end, we propose Core^M, a scalable multi-chip accelerator architecture based on weight stationary data flow. We augmented the memory hierarchy of single chip architecture by adding another level of cache, and further explored the multi-chip design space in three design dimensions, ranging from core organization, buffer size and memory hierarchy bypass in Core^M. Evaluations show that our design can scale the performance of a single chip accelerator linearly and offers a detailed design space exploration of Core^M.**

I. Introduction

Deep Learning has made great progresses over many applications, such as image classification [9], natural language processing [2], [22], as well as decision-making [3], [11], [19]. However, there are still issues in its application in autonomous driving [20], drones, and augmented reality/virtual reality (AR/VR). These applications, though exciting, are also challenging as they require real-time, low-latency DL inference that runs on edge devices [1], [14], where a set of design rules for energy consumption and chip-footprint are strictly imposed [7]. To enable those latency-sensitive DL applications, detailed architecture study on latency-awared and energy-constrained DL accelerator is needed.

Currently, there are two architecture approaches to achieve the low latency requirement. First is the wafer-scale design, such as Cerebras, that integrates trillions of transistors on a large chip [12]. The cost of which, however, is extremely high due to the low yield rate for large die. Therefore, it's not economically viable for mass production. The Chiplets is another promising solution to scale up the architecture. AMD has proposed ZEN2 architecture in its latest CPU [17] and achieved great commercial success with decent cost and performance trade-off.

Inspired by these commercial successes, we choose to work on a multi-chip architecture that runs DL inference in a synergistic manner. To complete such task, several questions in the architecture need to be answered. For instance, how to arrange the spacial dimension of the PE (processing element) array, how to effectively use all the processing resources for high utilization and high throughput, how to implement memory hierarchy between different chips to reduce or even avoid stalling cycles due to intra-chip communication bottlenecks, how to map inputs and model in different chips given the architecture resources and workloads, and how the performance varies from different neural network architecture. These questions and challenges are rooted in system level of parallelization, resources scheduling and data management, since the overall compute, storage, and communication resources are limited [4], [8], [16], [18].

Therefore, we propose Core^M, a multi-chip DNN accelerator that thoroughly evaluates the impact of multi-chip architecture on running real-time deep learning inference. We explored a variety of design choices from the hardware perspective, ranging from different computation resources and memory hierarchy configurations. One key insight of Core^{M}is that we add another level of memory hierarchy in the multi-chip architecture to improve the performance and relax the map space. In addition, we investigated from algorithm/software perspective by imposing different workload/network models in the design. Our project provided insights to facilitate hardware designers in building large scale DL accelerator. The realization of such design may further enable drones, augmented reality/virtual reality (AR/VR) applications, and autonomous driving vehicles.

II. Core^M Architecture

Fig. 1 shows the architecture of Core^M. We extended the single chip weight stationary dataflow design to multi-chip, which consists of a Core array with multiple computation core. Each core is a 16x16 PE mesh, and each PE has a private register file attached to it. A private multi-banked SRAM serves one core and all private SRAMs are connected to a single global shared SRAM. Compared with single chip architecture, an additional private cache level is added to Core^M. Therefore, Core^{M}is a 4-level memory hierarchy design: PE scratchpad (register file) -> private cache (SRAM) -> shared cache (SRAM) -> main memory (DRAM).

In the single chip design with 16x16 PE array, input channel dimension (C) is spatially mapped to X dimension of PE array and output channel (M) is spatially mapped to Y dimension accordingly. The weights are projectively mapped to the 2D PE array, and could be further reused if additional batches are available. Thus, the dataflow of this single chip architecture is weight stationary. We extend this weight stationary dataflow

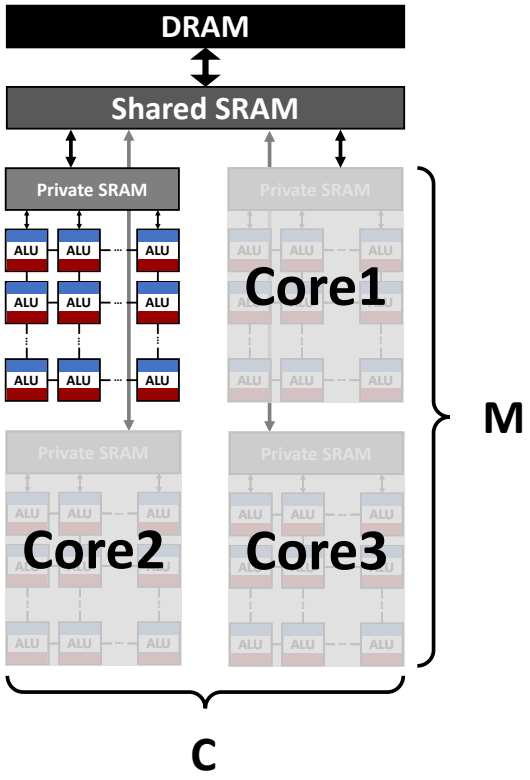

Fig. 1. Example Core^M 2x2 Architecture

to Core^M multi-chip settings: the entire system is treated as a giant PE array without core boundary. C and M dimensions are mapped to X and Y respectively. For example, in Fig. 1, we show an example design with 4 cores and a 2x2 core topology. Both the X and Y dimension has 2x16=32 PEs. Therefore, we spatially map the weights of 32 input channels and 32 output channels to our system. Our design is general so that other types of dataflow , e.g. output stationary, could also be mapped with ease.

Three design parameters are embedded in Core^M:

1) **Core Organization:** We modeled a 4 core design. The computational cores can be arranged into 2x2 (shown in Fig. 1), 1x4 and 4x1 configurations. Different arrangement yields the same area in size but wtih different mapping. Because of the spatial mapping scheme described above, 1x4 configuration outperforms 4x1 in layers that has more input channels than output channels.
2) **Buffer Size:** The SRAM size of the single core architecture is 16k. Thus, in our 4-core architecture we scaled the shared SRAM size by 4x to 64k. In addition, we added a new level of memory hierarchy, namely private SRAM, in the multi-core architecture. We referenced the recent Intel CPU (core i7 9750H) cache size and found that the shared L3 cache is around 8x larger than the private L2 caches. Therefore, we selected 8k as the size of the combined private SRAM size (each is 2k).
3) **Memory Hierarchy Bypass:** We added the per-core private cache to improve the performance and energy efficiency of Core^M architecture by exploiting more data reuse on the private cache level. Higher level of memories including DRAM and global SRAM hold all of the data types, weights/input activations/output activations, while the private SRAM only holds the weight. Further, we explored the possibility of holding the input/output activations in the private SRAM to understand the trade off of bypassing certain memory hierarchy.

We used a 2-by-2 four-chip configuration as the default multi-chip baseline, the private SRAM size in-between global buffer and local PE scratch pad is set to 2kb and it only keeps weights, bypassing inputs and outputs. The choice of baseline is in that a 2-by-2 geometry does not favor specific type of workload and mapping, and thus in theory, it should be the most balanced configuration in the four-core architecture. Furthermore, buffering weights in private SRAM would yield the best performance in our weight stationary design.

III. Evaluation

A. Experimental Setup

1) Run-time Environment: we used Timeloop [13]/Accelergy [21] to evaluate the performance and energy cost of our design. The experiments were conducted on three separate machines, with macOS 10.13.0 High Sierra, Windows 10 pro Enterprise/Education and Windows 10 Home. The virtualization of our infrastructures are based on docker-desktop and docker toolbox. The docker-compose file was provided in the project repo. No estimation plug-in was added in our docker. Each experiment was run once with timeout parameter set to 15000 and victory-condition set to 1000 in mapper.yaml. We manually terminated the program if a 30 minutes timeout is reached.

2) Evaluation Metrics: we reported the throughput (MACs/cycle), energy efficiency (pJ/MAC) and utilization of Core^M design. Since the above three metrics are largely decoupled with layer dimension, we conducted the experiment on one Core^M configuration by sweeping across all of the selected workload and report the mean throughput, energy efficiency and utilization. Unless stated, all of the results below are averaged on all of the workloads.

3) Workload Selection: we chose four representative network architectures: **AlexNet**, **VGG01**, **SqueezeNet**, and **MobileNet v1**. AlexNet scales the insights of LeNet into a larger network architecture that enables its ability to learn complex objects and object hierarchies [10]. We included it in the evaluations given its technical contribution and historical significance. VGG01 was the first to use multiple 3x3 convolution in sequence that emulates the effect of larger receptive field [15]. We included it in the evaluations to see how the system performs on a complex network with large amount of weights. SqueezeNet is adaptable to model compression techniques, such as quantization and pruning, to less than 0.5MB in memory [6]. We included it in the evaluations to see how the system performs under low memory requirement.

MobileNet v1 used depthwise separable convolution to reduce the model size and complexity, which efficiently trade off between latency and accuracy [5]. We included it in the evaluations to see how well the system handles special CNN layers. Within each network model, we selected four representative CNN layers to cover the entire distribution of the workload, including norm cases and edge cases. The detailed layer parameters can be found in Table I.

TABLE I
NETWORK LAYER DIMENSION

Layer	Filters	InChn	OutChn	Strides
AlexNet layer1	11×11	3	96	4
AlexNet layer3	3×3	256	384	1
AlexNet layer4	3×3	384	384	1
AlexNet layer5	3×3	384	256	1
VGG01 layer1	3×3	3	64	1
VGG01 layer4	3×3	256	256	1
VGG01 layer6	3×3	512	512	1
VGG01 layer7	3×3	512	512	1
SqueezeNet layer1	7×7	3	96	2
SqueezeNet layer21	1×1	64	256	1
SqueezeNet layer22	3×3	256	512	1
SqueezeNet layer23	1×1	512	64	1
MobileNet layer3	1×1	32	64	1
MobileNet layer4	3×3	64	64	2
MobileNet layer5	1×1	64	128	1
MobileNet layer6	3×3	128	128	1

B. Comparison with Baseline 1x1

We first compare Core^M base configuration (2x2 core configuration, 2k dedicated private weight buffer) with a 1x1 core design (w/o private cache, 16k global SRAM buffer) to show the scalability of Core^M. As shown in Fig. 2, both baseline and Core^M achieves theoretical peak throughput on MobileNet. This is because the channel sizes in MobileNet are strict multiple of Core^M's PE array dimension (32x32). Therefore, timeloop is able to find the mapping with 100% utilization. On average, Core^M achieves a speedup of 4.15x over single chip accelerator, which is larger than their area difference (4x). The reason of this super-linear scalability is that with the private buffer added, the search space for Core^M is enlarged. Furthermore, larger PE array enables more spatial parallelism to be exploited. Timeloop is able to find such new mapping in this relaxed search space to surpass the single design by more than 4x, which ensures Core^M's scalable design.

In terms of energy, we will first address the outlier, i.e. VGG01, and then explain the general case. We observed that Core^M yields more than 2x degradation on energy efficiency compared to the 1x1 system. The reason is that timeloop finds Core^M an extremely energy consuming mapping for VGG layer01 which only has 3 input channels. This mapping has a slightly larger utilization (43.75%) than the second most optimal mapping where as it consumes 12.8x more energy (245pJ/MAC vs 19pJ/MAC). The base 1x1 system suffers from the same problem. If we tradeoff a little utilization (and throughput) with energy efficiency and choose the second optimal mapping for layer1, Core^M can achieve an average energy efficiency of 16.5pJ/MAC, while the single chip system achieved 16.2pJ/MAC. Therefore, the revised average energy efficiency of Core^M and baseline 1x1 system are 12.1pJ/MAC and 11.8pJ/MAC respectively. Core^M slightly degrades energy efficiency by 2.5%. There are two factors in adding private buffer that affect Core^M's energy efficiency when scaling the single chip system to multi-chip. The first factor is the increased buffer size, which allows more weight reuse and reduces data movement between global buffer and computation core, therefore achieving higher energy efficiency. The second factor is the increased memory hierarchy level, which reduces the energy efficiency in weight fetching/replacing. With new level of hierarchy, each weight need to be fetched from global buffer to core's private buffer before loading to the local register. whereas in the single chip system, weights fetching/replacing only need to access the SRAM buffer once. The additional memory hierarchy level leads to memory access latency increase. From our evaluations we can conclude that the second factor dominates in MobileNet while the first factor is more prominent in AlexNet, VGG and SqueezeNet. On average the two factors balances with each other and the energy efficiency should be slight degraded when moving from single chip to multi-chip architecture due to the second factor.

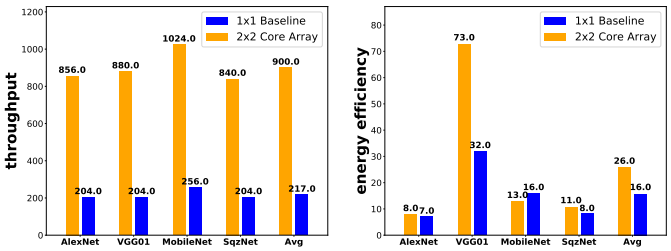

Fig. 2. Comparison with 1x1 Baseline

C. Sensitivity of Design Parameters

We set 2x2 core organization, 2k private cache and private cache keeps weights as the base configuration and explore the effect of the three design parameters to the performance (throughput, energy efficiency, utilization) of Core^M. All of the results shown below are normalized to the base configuration.

We evaluate the performance on 1x4, 4x1 and 2x2 system. Fig. 3 shows the throughput (higher is better), energy efficiency (lower is better) and utilization (higher is better) of the 3 systems. The 2x2 and 1x4 system yields similar performance whereas 4x1 system has less utilization/throughput (93%) and less energy efficient (1.16x worse). This is because we choose the first layer as the corner case layer that only has 3 input channels (C=3). Our spatial mapping scheme maps PE array x dimension to C (which is small). The 4x1 system has 64 PEs along the X dimension which results in the greatest wastes of the computing array. Therefore its utilization is the lowest. This also explains why 1x4 system performs as well to the 2x2

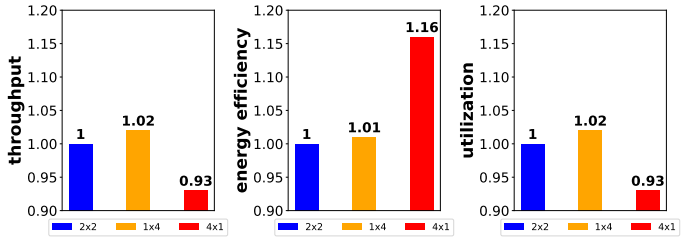

Fig. 3. Sensitivity on Different Core Organizations

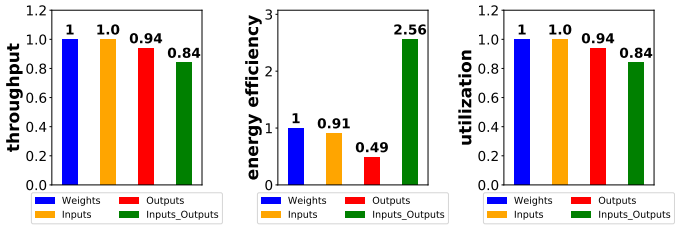

Fig. 5. Sensitivity on Different Private SRAM Bypass Configurations

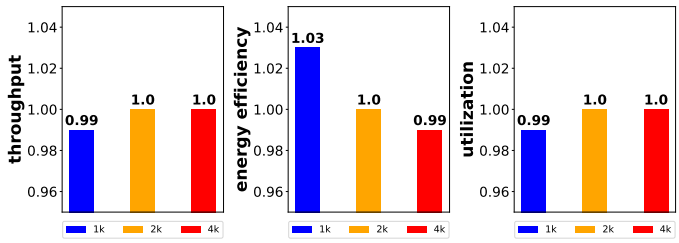

Fig. 4. Sensitivity on Different Private SRAM Size

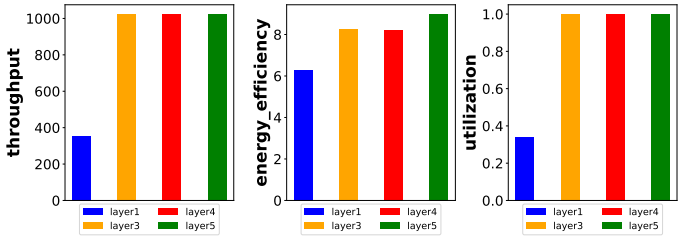

Fig. 6. Evaluation on 4 representative layers of AlexNet

system. Our layer selection is a little biased towards smaller input channel size. If tested on more layers, we believe the 2x2 system has the best flexibility of handling variable input/output channel size.

Fig. 4 shows the tradeoff between performance and private core SRAM size. Compared with 4k cache, the 2k configuration has the same throughput and utilization. The energy efficiency is slightly compromised (1%) because larger cache can exploit more data reuse. But overall 2k is large enough to exploit the benefits of private caches. Compared with 1k cache, the throughput and utilization of 2k Core^M is larger and it achieves a 3% improvement on energy efficiency. Though the performance difference is minor, during the experiment we observe that timeloop fails to find valid mappings on several layers using 1k cache configuration. Only if we relax the termination condition can it be successful. If we generalize the test to more layers, it might be harder for timeloop to find valid mappings for 1k caches because smaller buffer means less data reuse opportunities can be exploited. Therefore, the results justifies our choice of 2k per core cache.

Finally we explore the different data types the private SRAM can hold and present the results in Fig. 5. From the utilization and throughput perspective, keeping a single data type in private buffer yields similar performance because timeloop can produce more regular loop ordering on the private buffer level. Keeping both input and output activations yields lowest throughput because it generally has larger search space and may prioritize the optimization for energy efficiency.

Keeping both the inputs and outputs yields a 2.5x improvement over the base configuration. This is because in the PE design, input and output activation each only has one register storing the data. Fetching from global SRAM costs lots of energy. If we buffer both the inputs and outputs at the private buffer level (small, fast and energy efficient), we can reduce the energy consuming memory accesses. Keeping the outputs yields the worst energy efficiency (50%). Depending on the tiling, output might generally have less reuse (in large scale architecture because larger tile is supported) so that keeping output cannot save too many cycles and energy. This tradeoff might only emerge with large scale architecture.

D. Per-layer Performance Breakdown

We further show the per-layer performance breakdown of AlexNet and SqueezeNet.

Fig. 6 shows the performance on 4 selected AlexNet layers. Except the first layer, the other three layers achieves the theoretical peak throughput of 1024MAC/cycle and 100% utilization of Core^M. This suggests that timeloop is able to find an optimal mapping for the proposed architecture. The first layer has low utilization and throghput because of its small input channel size as mentioned in the previous subsection. Because of the small input layer, less input activations are needed to be replicated and sent to multiple cores. Therefore, it's more energy efficient.

Similar trend is also observed on SqueezeNet in Fig. 7 where the first layer is more energy efficient and the rest achieves peak throughput and utilization. Another observation on SqueezeNet is that layer23 and layer21 have better energy efficiency than layer22. This is because layer 21 and 23 use 1x1 convolution kernel and layer 22 uses 3x3. Smaller kernel indicates that the core private SRAM can hold more weights and therefore exploits more data reuse to save more energy.

IV. Conclusion and Future Work

In this work, we propose Core^M, a large scale multi-chip DNN accelerator. Core^M augments the traditional memory hierarchy by splitting the on-chip SRAM buffer into a global shared buffer and multiple per-chip private SRAM. We

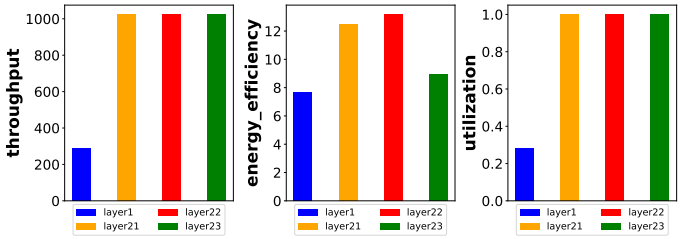

Fig. 7. Evaluation on 4 representative layers of SqueezeNet

further identify three design parameters in Core^M, namely core organization, buffer size and memory hierarchy bypass. Evaluations on four representative DNNs show the proposed architecture can yield 4x throughput improvement over the 4x smaller single chip design while maintaining the same energy efficiency. We also show various tradeoffs of Core^M and per-layer performance breakdown.

For future work we plan to evaluate Core^M's performance on different dataflows (output stationary, row stationary, etc.). Inter-chip direct communication is not modeled in our design (and not supported by timeloop), which adds a new design dimension in multi-chip accelerator design.

REFERENCES

[1] M. R. Abdelhamid, R. Chen, J. Cho, A. P. Chandrakasan, and F. Adib, "Self-reconfigurable micro-implants for cross-tissue wireless and batteryless connectivity," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[2] G. Attardi, "Deepnl: a deep learning nlp pipeline," in *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, 2015, pp. 109–115.

[3] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[4] R. Chen, H. Kung, A. Chandrakasan, and H.-S. Lee, "A bit-level sparsity-aware sar adc with direct hybrid encoding for signed expressions for aiot applications," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2022, pp. 1–6.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[7] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131 543–131 558, 2019.

[8] J. Keuper and F.-J. Preundt, "Distributed training of deep neural networks: Theoretical and practical limits of parallel scalability," in *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*. IEEE, 2016, pp. 19–26.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] Z. Liang, G. Zhang, J. X. Huang, and Q. V. Hu, "Deep learning for healthcare decision making with emrs," in *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2014, pp. 556–559.

[12] S. K. Moore, "Huge chip smashes deep learning's speed barrier," *IEEE Spectrum*, vol. 57, no. 01, pp. 24–27, 2019.

[13] A. Parashar, P. Raina, Y. S. Shao, Y. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. S. Emer, "Timeloop: A systematic approach to DNN accelerator evaluation," in *IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2019, Madison, WI, USA, March 24-26, 2019*. IEEE, 2019, pp. 304–315. [Online]. Available: https://doi.org/10.1109/ISPASS.2019.00042

[14] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, 2019.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[17] D. Suggs, M. Subramony, and D. Bouvier, "The amd "zen 2" processor," *IEEE Micro*, vol. 40, no. 2, pp. 45–52, 2020.

[18] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 328–339.

[19] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, 2018.

[20] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[21] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2019, Westminster, CO, USA, November 4-7, 2019*, D. Z. Pan, Ed. ACM, 2019, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ICCAD45719.2019.8942149

[22] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.