5 Parallel Prism: A topology for pipelined implementations of convolutional neural networks using computational memory

Martino Dazzi[1, 2], Abu Sebastian[1], Pier Andrea Francese[1], Thomas Parnell[1], Luca Benini[2] and Evangelos Eleftheriou[1]
[1] IBM Research – Zurich, Switzerland, [2] ETH Zurich, Switzerland
{daz, ase, pfr, tpa, ele}@zurich.ibm.com
lbenini@iis.ee.ethz.ch

Abstract

In-memory computing is an emerging computing paradigm that could enable deep-learning inference at significantly higher energy efficiency and reduced latency. The essential idea is to map the synaptic weights corresponding to each layer to one or more computational memory (CM) cores. During inference, these cores perform the associated matrix-vector multiply operations in place with O(1) time complexity, thus obviating the need to move the synaptic weights to an additional processing unit. Moreover, this architecture could enable the execution of these networks in a highly pipelined fashion. However, a key challenge is to design an efficient communication fabric for the CM cores. In this work, we present one such communication fabric based on a graph topology that is well suited for the widely successful convolutional neural networks (CNNs). We show that this communication fabric facilitates the pipelined execution of all state-of-the-art CNNs by proving the existence of a homomorphism between one graph representation of these networks and the proposed graph topology. We then present a quantitative comparison with established communication topologies and show that our proposed topology achieves the lowest bandwidth requirements per communication channel. Finally, we present a concrete example of mapping ResNet-32 onto an array of CM cores.

1 Introduction

Training and inference of Deep neural networks (DNNs) is a computationally intensive task and has motivated the search for novel computing architectures targeting this application [2, 10, 11, 17, 3, 8]. However, one of the primary reasons for the computational inefficiency, namely the need to shuttle millions of synaptic weight values between the memory and processing units, remains unaddressed. In-memory computing is an emerging computing paradigm that addresses this challenge of processor-memory dichotomy. The essential idea is to map a DNN on multiple crossbar arrays of memory devices that communicate with each other. A layer of the DNN can be implemented on (at least) one crossbar, in which the weights of that layer are stored in the charge or conductance state of the memory devices at the crosspoints. The propagation of data through that layer is performed in a single step by inputting the data on the crossbar rows and obtaining the result of the associated matrix vector multiply operation at the columns. These operations are performed in place by exploiting the physical attributes of the memory devices. The results are then passed through the neuron nonlinear function and input to the next layer. The neuron nonlinear function is typically implemented at the crossbar periphery using analog or digital circuits. There are several recent demonstrations of

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

in-memory computing using SRAM [16, 9, 1, 15], NOR Flash [12] as well as resistive memory devices [18, 6, 14].

However, the ability to perform matrix-vector multiplication in constant time, as well as the one-to-one mapping between synaptic weights and memory devices, creates a new challenge for designers of such accelerators. If we want to perform computation in a *pipelined* manner, the bottleneck becomes how quickly we can communicate activations from one layer/computational memory (CM) core to another. This clearly depends on both the architecture of the neural network model itself, i.e. how the different layers are connected, as well as the topology of the communication fabric of the accelerator device, i.e. how long it takes to transfer activations between physically distinct layers. To manufacture a general-purpose inference engine, the challenge would be to design a communication topology that supports low-latency execution for a wide range of neural network architectures.

Contributions We attempt to address exactly the above problem with a focus on convolutional neural networks (CNN) architectures due to their popularity in a range of tasks. Specifically, our contributions are:

- We propose a novel graph topology (5 Parallel Prism, or 5PP) for pipelined execution of CNNs that maximally exploits the physical proximity of computational units.
- We prove theoretically that all state-of-the-art neural networks featuring feedforward, residual, Inception-style and dense connectivity are mappable onto the proposed topology.
- We present a quantitative comparison of 5PP with existing communication topologies and demonstrate its advantages in terms of latency, throughput and efficiency.
- We give a detailed example of how to map ResNet-32 onto an array of CM elements.

A CNN can be implemented in CM in a pipelined fashion if there exists communication channels that match the connectivity of the network. If this is not the case, the activations from one CM core need to be transferred through neighboring cores in order to reach their destination, adding unnecessary cycles to the execution schedule. As the execution happens in a pipelined fashion, every core would then systematically stall at every cycle until the slowest data transfer is completed, severely impairing the throughput. Mathematically:

Definition 1.1. *Given a communication fabric with topology $\mathcal{F}$ and the directed graph representation $\mathcal{C}$ of a CNN, with vertices representing convolutional layers and edges representing activations directed toward the direction of computation, then the CNN is executable in a pipelined fashion on $\mathcal{F}$ if there exists a homomorphism $h : \mathcal{C} \to \mathcal{F}$.*

As the communication fabric will have to provide enough connection overhead to accommodate a diverse set of networks, the homomorphism will generally be injective. In the next section, we present one such graph representation, $\mathcal{C}$, of CNNs.

2 A Consolidated Graph Representation of CNNs

Typically, CNN architectures are represented as directed graphs, with vertices representing convolutional layers and directed edges representing activations directed toward their next layers. Nevertheless, CNN representations are not coherent with their physical implementation, some examples being input images or concatenation operations themselves being represented as one vertex of the network, or pooling operations being at times represented along convolutions in the same vertex and at times separately. Based on these arguments, we present a consolidated graph representation of CNNs that follow the following five design rules:

R1. Vertices are identified solely with convolutional layers. Any other operation, such as pooling or addition for the residual path, is treated as pre- or post-processing of the convolution.

R2. We do not distinguish between input and output of a vertex. Vertices are either connected or not connected. This is illustrated in Fig. 1**a**.

R3. Edges that may make the graph non-simple are removed. This is illustrated for the ResNet architecture in Fig. 1**b**.

R4. Concatenation does not imply any operation on the data, thus it cannot be represented as a vertex in the graph. Given this assumption, the concatenation of *m* vertices being fed into *n* others is equivalent to a complete bipartite graph $k_{m,n}$ as in Fig. 1**c**.

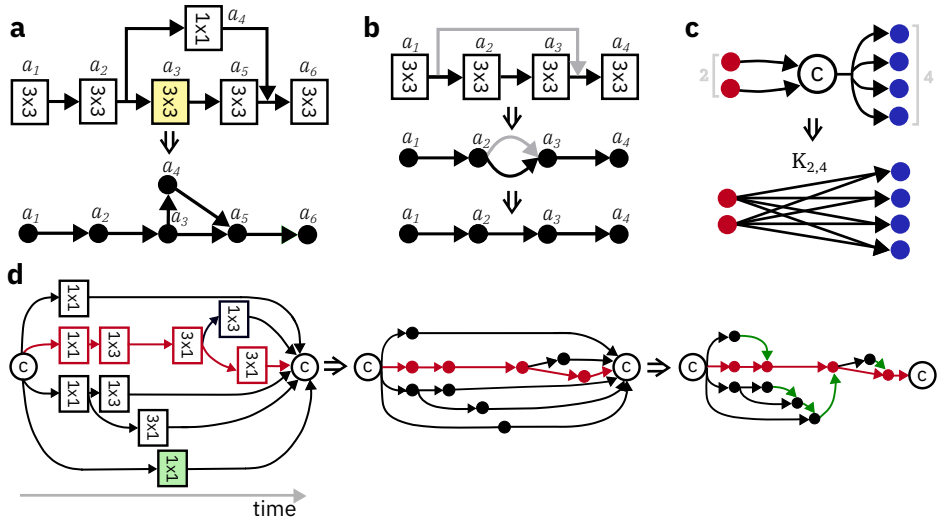

Figure 1: Illustration of representation rules. Yellow layer in **a.** represents a layer with stride = 2; green layer in **d.** represents a layers to which 3×3 max pooling is previously applied.

R5. A series-parallel (s-p) graph such as Inception can have some parallel path with latency much smaller than the critical path (see Fig. 1**d**, with the nodes laid out in order of latency and the critical path in red) that are unable to reach their physical destination unless they *hop* to the neighboring vertex as the critical path is executing. We require that such activations hop to their nearest temporally subsequent node (green arrows in Fig. 1**d**).

3 A New Communication Fabric: 5 Parallel Prism

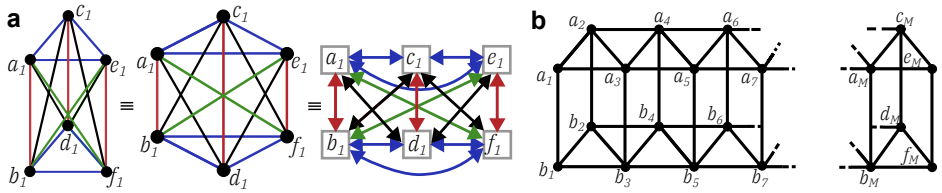

Figure 2: **a.** Unit graph, from left to right: our 3D representation, its canonical graph theory representation as a complete graph K_6, and its physical implementation, with vertices representing computational units and edges representing bidirectional links. **b.** 5 Parallel Prism topology with N vertices $\{a_1, a_2, ..., a_M, b_1, b_2, ..., b_M, c_M, d_M, e_M, f_M\}$; $M = \lceil (n-6)/2 \rceil + 1$. All diagonal edges of the prisms are omitted for clarity.

Here we propose a topology $\mathcal{F}$, referred to as 5 Parallel Prism (5PP), for the communication fabric. 5PP can be thought of as a graph constructed from the coalescing of multiple smaller graphs, referred to as "unit graphs". Figure 2**a** portrays the unit graph as an undirected complete graph K_6. The physical implementation of the unit graph in Fig. 2**a** clearly depicts how we maximize the use of spacial proximity in the interconnection of CM units with non-negligible physical size: The basic communication infrastructure takes place between a 2-by-3 neighborhood, which is the way of packing six computational units on a meshgrid that yields the shortest intra-unit maximum distance. The construction of an N-vertice 5PP starts from a disjoint union of $M = \lceil (n-6)/2 \rceil + 1$ unit graphs. A complete algorithmic description of the construction of the 5PP is provided in Appendix A.

4 Mapping CNN architectures onto 5PP

Here we establish the existence a homomorphism between the consolidated graph representation of four different state-of-the-art CNN architectures and the 5PP topology via an *iterative H-coloring* method [5].

4.1 Feedforward and ResNet topologies

The existence of a homomorphism between a standard, pre-2014 feedforward connection is banal, as in their graph representation this connectivity is a path, and there always exists a homomorphism, e.g.$\{a_1, b_1, a_2, b_2, a_3, b_3..., a_n\}$, that maps a path.

Residual networks (ResNet) are relatively deep networks that employ skip connections or shortcuts to jump over certain layers [4].

Theorem 4.1. *Any ResNet architecture can be H-colored onto the 5PP.*

Proof Sketch. ResNet are series-parallel graphs with maximum out-degree d^o of the vertices equal to 2 for three adjacent vertices, which is reached for a residual path across layers with different strides, where one resampling layer is required as shown in Fig. 1**a**. This kind of pattern is always mappable inside the unit graph by definition of complete graph K_6. A complete mathematical argument is provided in Appendix B.1. □

4.2 DenseNet topologies

Dense connectivity as proposed by [7] sees a sequence of densely connected layers, all with the same channel depth, where every node receives as input the concatenation of all its preceding layers. The DenseNet CNN comprises dense blocks connected in series.

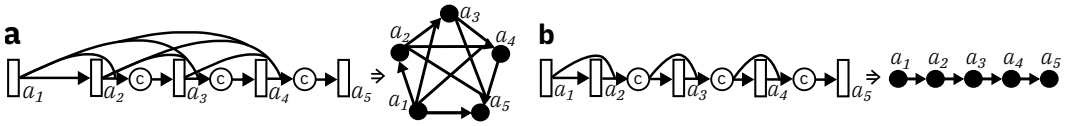

Figure 3: **a.** Representation of dense connectivity as a complete graph. **b.** Representation of dense connectivity as a path graph.

Theorem 4.2. *Any DenseNet topology can be H-colored onto the 5PP by using the maximum number of edges possible.*

Proof Sketch. To represent densely connected layers as a graph, there are two possible representations, based on whether activations are communicated to the subsequent layers before or after data aggregation implied by the concatenation operation. In the former case, in Fig.3**a**, the activations that are moved between layers are the output activations of each layer, and *n* densely connected layers are equivalently *n* vertices with an edge connecting each pair of vertices: that is, a complete graph K_n. Physically, these edges represent channels that communicate the same number of activations. Conversely, in the latter case in Fig. 3**b**, the activations communicated between layers are the input and output activations of each layer and, after application of R2 and R3, in the same fashion as depicted in Fig. 1**a**,**b**, the dense connectivity streamlines into a path connectivity. Physically, these edges represent channels where the number of activations that are communicated increases linearly in the direction of execution. As our proposed topology is a sort of path connection of complete graphs K_6, we adopt both representations in Fig. 3 to distribute the communication and obtain the minimum bandwidth requirements for the physical channels. A complete mathematical argument is provided in Appendix B.2. □

4.3 Inception-style topologies

Theorem 4.3. *Inception v1,2,3,4 and Inception ResNet v1,2 are H-colorable on the 5PP.*

Proof Sketch. Inception blocks feature a source concatenation node connecting to a maximum of four parallel branches in all Inception architectures; Note that in our graph representation of Inception networks, as discussed in design rule 4, we choose to hop all data from layers prior to the longest latency path through the nearest temporally subsequent layer. Although this simple criterion allows implementation of the Inception blocks of all Inception networks, data from one vertex can be hopped through any temporally subsequent vertex and can be considered a design parameter to optimize power consumption or maximum bandwidth of the links. We can prove the H-colorability of the

single Inception blocks with property 3 based on these patterns. A complete mathematical argument is provided in Appendix B.3. □

5 Quantitative Evaluation of 5PP

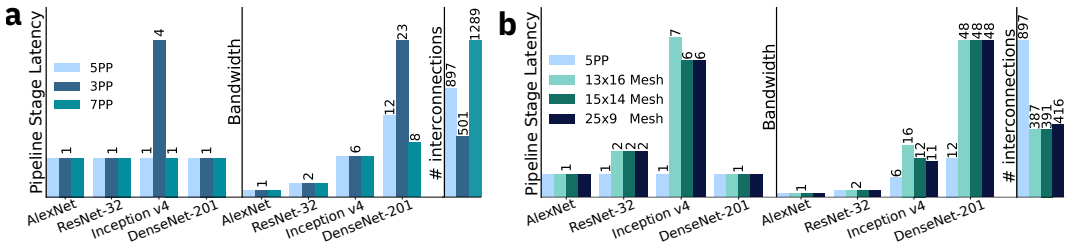

Figure 4: Bar plots of pipeline stage latency and bandwidth requirements for four state-of-the art CNNs on **a.** topologies built in a similar fashion as the 5PP and **b.** well-established topologies.

This section explains why, among the possible complete graphs employed as unit graphs for building the proposed topology, complete graph K_6 and the corresponding topology 5PP constitute the optimal choice. We also quantitatively compare 5PP with well-established communication topologies.

Evaluation metrics. The core criterion that defines optimality is the latency of the single pipeline stage. When a homomorphism cannot be established, the latency of the single pipeline stage would be limited by the additional computational cycles required for data movement. In Fig. 4, pipeline stage latency is measured in number of computational cycles. The bandwidth of the links is determined by the CNN layers with the greatest channel depth. We thus normalize the bandwidth requirements by the quantity $(C_{max} \cdot bits_{act})/T$ with C_{max} maximum number of channels per layer in the network, $bits_{act}$ the accuracy of the activations in bits and T the computational cycle.

Variations on 5PP. Figure 4**a** shows a comparison of these metrics between the 5PP and two topologies built in the same fashion as the 5PP from complete graphs K_4 (3PP) and K_8 (7PP), with the intent of showing the trend for topologies built on complete graphs with a lower and higher number of edges. In terms of latency, 5PP never stalls the pipeline. It is not possible to establish a homomorphism for every class of networks built on $K_n, n < 5$, resulting in the overall higher pipeline stage latency as is the case for 3PP. Conversely, 5PP is a subgraph of any topology built on $K_n, n > 5$, meaning the *H*-coloring still holds and the pipeline is never stalled. For DenseNet, the higher the number of edges, the more effectively one can distribute the data movement, yielding lower bandwidth requirements. Indeed, distributing the communication of activations as described in Section 4 yields the minimum bandwidth requirement: If communication of the activations requires bandwidth *k*, *d* densely connected layers can be made to communicate synchronously on a topology built on a unit graph K_n requiring maximum bandwidth $\left[k + k \cdot \Theta\left(\lceil (d-n)/(n-2) \rceil\right)\right]/T$, where Θ is the Heaviside step function and T the computational cycle. These lower requirements come at the price of instantiating multiple additional edges that have no advantage in terms of latency. In Fig. 4**a**, the 7PP does perform 1.5× better than 5PP solely on the bandwidth requirement for DenseNet-201, at the cost of 1.44× more physical links overall.

Existing topologies. We now compare the performance of our topology with long-established communication topologies [13] on the metrics defined in Section 3. We consider 2D meshes with different aspect ratios as the prior art in communication fabric topologies. Note that mapping onto these topologies is based on the same hypotheses as mapping onto the 5PP shown in Section 3. Figure 4**b**. gives the bar plot rendition of pipeline stage latency and bandwidth requirements measured in the same fashion as in Fig. 4**a**. In terms of latency, path-connected networks (AlexNet) and DenseNet in its path representation (Fig. 3**b**) can be executed without breaking the pipeline. Conversely, this shows quantitatively how the absence of an *H*-coloring between ResNet-32 and Inception v4 impairs execution of the network by stalling the pipeline to allow communication between non-adjacent vertices. With regard to bandwidth, thanks to the DenseNet representation described in Section 4, 5PP significantly lowers bandwidth requirements 4× with respect to the best 2D mesh performance. Although the better performance overall in 5PP costs a greater number of interconnections, their increase(2.31×) is significantly lower than the best factors of improvement in latency(7×) and bandwidth(4×).

6 Case study: ResNet-32 on a CM array

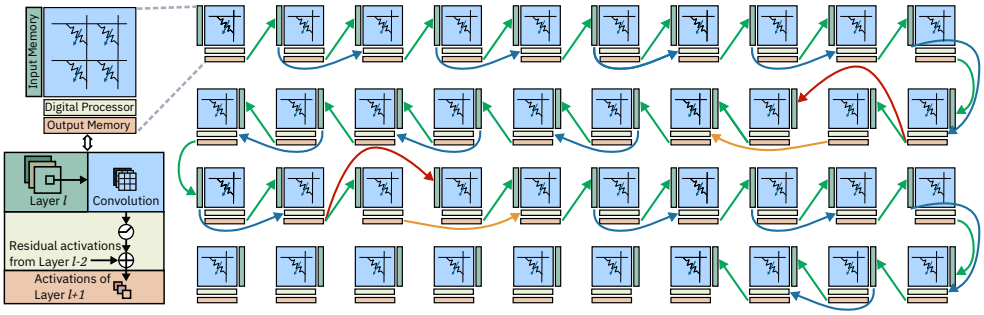

Figure 5: Mapping of ResNet-32 on a 4-by-10 array of CM cores. Each layer is mapped onto one core. Arrows indicate the communication of data between cores. Green arrows denote the propagation of activations to the adjacent layer. Blue arrows represent data movement arising from residual connections. Orange arrows indicate re-sampled residual connections, and red arrows represent communication to residual re-sampling layers.

The physical mapping of a ResNet-32 [4] for CIFAR-10 on an array of CM cores, based on the 5PP communication fabric, is depicted in Fig. 5. The network features 3 layer levels with channel depths equal to 16, 32, and 64, whereas the input image size is 32×32 pixels. The CM cores are rendered as boxes with their associated input (left) and output (bottom) memory. The links active in this implementation verify the homomorphism discussed in section 4.1. Each core also comprises modest digital processing capability to scale the crossbar outputs, e.g. to apply the batch normalization, to apply the activation functions and perform the residual addition. We assume 576×576 for the crossbar size per CM core, meaning the vector-matrix multiplication with a matrix of size 576×576 can be performed in one computational cycle. During execution, the elements of the crossbar hold one convolutional weight each, while the input memory will store the pixel neighborhood required for the convolution. The result of one dot product of the convolution, i.e. a single activation across the entire channel depth, is stored in the output memory. Table 1 reports design parameters and performance and area metrics based on hardware measurement and software simulations. As mentioned in Section 2, the communication channels can communicate indistinctly to both the input memory (to communicate the standard feedforward activations) or to the output memory (to communicate the activations used for residual addition). The dataflow occurs row-wise starting from the core located at the top left. Whenever one set of activations is computed by a core, it is communicated to the core assigned the subsequent layer and stored in its input memory. Computation on one core begins when it has received sufficient activations to perform the dot product corresponding to its own assigned layer. The proposed topology ensures that the pipeline is never stalled irrespective of the bandwidth. However, the most efficient implementation would be having sufficient bandwidth for all data transfer to occur in parallel with the computation cycle. Conversely, if the bandwidth is not sufficient, a constant communication overhead must be added to each computational cycle.

Weights Accuracy $[bits]$	**Activations Accuracy** $[bits]$	**Links Data Rate** $[Gbps]$	**Computational Cycle** $[ns]$
4	**8**	**5**	**100**
Tops/W	**Single Classification Latency** $[\mu s]$	**Throughput** $[Images/s]$	**Area**$[mm^2]$
26.4	**52**	**38600**	**0.0125[Digital Core] + 0.6[Analog array]**

Table 1: Design parameters and resulting performance and area metrics.

7 Conclusions

We introduced 5 Parallel Prism (5PP), an interconnection topology for executing CNNs on an array of Computational Memory cores. We then proved the executability of ResNet, Inception, and DenseNet networks on the proposed communication fabric by proving the existence of a homomorphism between a consolidated graph representation of the CNNs and 5PP. Moreover, we validated the efficiency of our approach by comparing the proposed topology to various 2D meshes in terms of inference latency as well as bandwidth requirements per communication channel. Finally, we provided a case study with the physical mapping of ResNet-32 on an array of CM cores. We estimate a throughput of $38k\ Images/s$ thanks to the unstaggered pipelining offered by our topology. The presented work is a significant step toward developing general-purpose DNN accelerators based on in-memory computing.

References

[1] A. Biswas and A. P. Chandrakasan. CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 54(1):217–230, 2019.

[2] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun. NeuFlow: A runtime reconfigurable dataflow processor for vision. In *CVPR Workshops*, pages 109–116, 2011.

[3] B. Fleischer, S. Shukla, M. Ziegler, J. Silberman, J. Oh, V. Srinivasan, J. Choi, S. Mueller, A. Agrawal, T. Babinsky, et al. A scalable multi-TeraOPS deep learning processor core for AI training and inference. In *Proceedings of the IEEE Symposium on VLSI Circuits*, pages 35–36. IEEE, 2018.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48(1):92–110, 1990.

[6] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, et al. Memristor-based analog computation and neural network classification with a dot product engine. *Advanced Materials*, 30(9):1705914, 2018.

[7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[8] Z. Jia, M. Maggioni, B. Staiger, and D. P. Scarpazza. Dissecting the NVIDIA Volta GPU architecture via microbenchmarking. *arXiv preprint arXiv:1804.06826*, 2018.

[9] Z. Jiang, S. Yin, M. Seok, and J.-s. Seo. XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks. In *2018 IEEE Symposium on VLSI Technology*, pages 173–174. IEEE, 2018.

[10] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12. IEEE, 2017.

[11] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li. Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 553–564. IEEE, 2017.

[12] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, and D. B. Strukov. High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays. *IEEE transactions on neural networks and learning systems*, 29(10):4782–4790, 2017.

[13] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis. An analysis of on-chip interconnection networks for large-scale chip multiprocessors. *ACM Transactions on Architecture and Code Optimization (TACO)*, 7(1):4, 2010.

[14] A. Sebastian, I. Boybat, M. Dazzi, I. Giannopoulos, et al. Computational memory-based inference and training of deep neural networks. In *Proceedings of the IEEE Symposium on VLSI Circuits*, 2019.

[15] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma. A 64-Tile 2.4-Mb in-memory-computing cnn accelerator employing charge-domain compute. *IEEE Journal of Solid-State Circuits*, 54 (6):1789–1799, 2019.

[16] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville. In-memory computing: Advances and prospects. *IEEE Solid-State Circuits Magazine*, 11(3): 43–55, 2019.

[17] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In *Advances in neural information processing systems*, pages 7675–7684, 2018.

[18] C.-X. Xue, W.-H. Chen, J.-S. Liu, J.-F. Li, W.-Y. Lin, W.-E. Lin, J.-H. Wang, W.-C. Wei, T.-W. Chang, T.-C. Chang, et al. 24.1 a 1Mb multibit ReRAM computing-in-memory macro with 14.6 ns parallel MAC computing time for cnn based ai edge processors. In *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 388–390. IEEE, 2019.

A 5PP Construction

The construction of an N-vertice 5PP starts from a disjoint union of $M = \lceil (n-6)/2 \rceil + 1$ unit graphs and is described by a pseudocode in Algorithm 1, with the resulting topology depicted in Fig. 2**b**. To represent the overall topology, all diagonal edges of the prisms are omitted for clarity. . The construction uses a vertex identification operation that keeps the graph simple as described in Eq. 1, where $u \cdot v$ is the vertex identification operation, *G* is the graph to which it is applied and *V(G)*, *E(G)*, the set of its vertices and edges.

$$u \odot v = \{u \cdot v \mid u, v \in V(G), \exists \text{ at most } 1 \ (u \cdot v, x) \in E(G), \forall x \in V(G)\} . \tag{1}$$

Algorithm 1 5 Parallel Prism Construction

% The initial disjoint union of unit graphs has vertices: $\{\tilde{a}_1, \tilde{b}_1, ..., \tilde{e}_1, \tilde{f}_1, ..., \tilde{a}_M, ..., \tilde{f}_M\}$
% It creates a 5PP with vertices $\{a_1, ..., a_M, b_1, ..., b_M, c_M, d_M, e_M, f_M\}$
% The first unit graph comprises the first 6 vertices of the 5PP

$\{a_1, b_1, c_1, d_1, e_1, f_1\} = \{\tilde{a_1}, \tilde{b_1}, \tilde{c_1}, \tilde{d_1}, \tilde{e_1}, \tilde{f_1}\}$;
for $j = 1 : (M-1)$ **do**
    $a_{j+1} = c_j \odot \tilde{a}_{j+1}$;
    $b_{j+1} = d_j \odot \tilde{b}_{j+1}$;
    $c_{j+1} = e_j \odot \tilde{c}_{j+1}$;
    $d_{j+1} = f_j \odot \tilde{d}_{j+1}$;
    $e_{j+1} = \tilde{e}_{j+1}$;
    $f_{j+1} = \tilde{f}_{j+1}$;
end for

B Homomorphism Proofs

The process of verifying the existence of the homomorphism is often referred to as *H*-coloring [5]. We propose an *iterative H-coloring* method that verifies the *H*-coloring of the vertices of the directed graph in their order from initial to final. That is, at the i-th iteration of verifying the *H*-coloring, vertex v_i in graph $\mathcal{C}$ is colored on a vertex in graph $\mathcal{F}$ such that there are enough edges to connect it to whatever previous vertices $v_{0,...,i-1}$ it is connected to. The homomorphism is proven if all vertices of $\mathcal{C}$ have been successfully *H*-colored onto $\mathcal{F}$ at the last iteration. Note that the definition of a homomorphism between a directed and an undirected graph is justified by the fact that, in our representation of the communication fabric, undirected edges represent bi-directional communication channels, which can be assigned either direction when the CNN is mapped onto it.

Generally speaking, the proof of existence of a homomorphism between two arbitrary graphs is an NP-hard problem for any non-bipartite graph. Leveraging the regularity of graphs representing CNNs, we present some properties of the 5PP that facilitate verification of the existence of a homomorphism with the topologies of state-of-the-art CNNs.

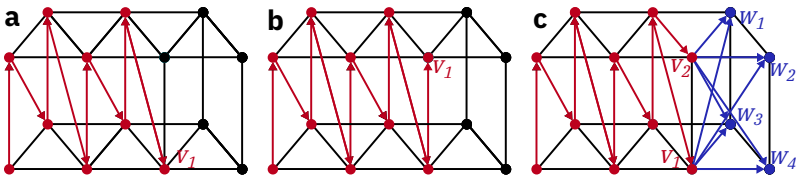

Figure 6: **a.** Example of *H*-coloring of an odd-vertices path. Vertex v_1 can have maximum $d_o = 5$. **b.** Example of *H*-coloring of an even-vertices path. Vertex v_1 can have maximum $d_o = 4$. **c.** Example of *H*-coloring of an even-vertices tree. *H*-coloring is then continued with complete bipartite graph $K_{2,4}$ between vertices $v_{1\sim 2}$ and $w_{1\sim 4}$.

We define an *even(odd) H*-coloring as an *H*-coloring that colors an even(odd) number of vertices. Furthermore, we define the out-degree d^o of a vertex *v* of a directed graph as the number of outgoing

edges from *v*. Figure 6 illustrates three *H*-colorings that are representative of the two main properties of the 5PP:

P1. By construction, every vertex belongs to at least one complete graph K_6. Given an odd(even) *H*-coloring of the 5PP $\{a_1, b_1, ..., a_n, b_n, ..., a_N\}, \forall n \leq N$, vertex a_N has possible maximum d^o =5(4). The maximum number of vertices accessible in parallel in our topology gives it its name.

P2. Given any *H*-coloring of the 5PP $\{a_1, b_1, ..., a_n, b_n, ..., a_N\}, \forall n \leq N$, one can always continue the coloring with a complete bipartite graph $K_{m,n}$ with $(m + n) \leq 5$. Furthermore, if the *H*-coloring is odd(even), the coloring can be continued with a bipartite graph with $(m + n) = 6$ only for configurations with *n* and *m* odd(even) numbers.

B.1 Proof of Theorem 4.1

In the ResNet architecture, given R2, we assimilate, without loss of generality, standard residual connections to connections from the input memory of one vertex to the output of the next (in Fig. 1**b**, a_2 to a_3, thus merging to the feedforward connection from a_2 to a_3). This requires the shortest connections possible to implement the residual connection, merging with the feedforward connection between two adjacent vertices. Let a graph represent a ResNet topology according to the rules in Section 2. Such a graph is a series connection of paths (R3, Fig. 1**b**) and resampling layers (R2, Fig. 1**a**) in that order. We prove a homomorphism by *H*-coloring the two elements in sequence. Any path can be iteratively *H*-colored in 5PP as proved above, ending with a certain parity. Then, the resampling layer is *H*-colored, which is equivalent to coloring a complete graph K_3, represented in Fig. 1**a** by vertices a_3, a_4 and a_5. Regardless of the parity of the coloring before the resampling layer, according to P1 there are always three uncolored vertices on which to color one K_3; there are also always enough edges, as K_3 is always a subgraph of K_6 for any three vertices belonging to K_6. Note that any other implementation of the residual connections would have yielded the same results in terms of maximum d^o in the graph and number of adjacent vertices required.

B.2 Proof of Theorem 4.2

Let a graph represent an *n*-layer DenseNet topology. Assume there exists a homomorphism that maps its layers $\{v_1, ..., v_n\}$ to a sequence $\{a_1, b_1, a_2, b_2, ..., b_m\}$ on 5PP using all edges, ordered as in Fig. 2**b**. If the number of densely connected vertices is ≤ 6, they all belong to the same K_6 $\{a_1, b_1, ..., a_3\}$, and the *H*-coloring follows from Fig. 3**a**. If the number of densely connected layers is > 6, the complete connection of all individual instances of K_6 is used, but there are vertices that do not belong to the same K_6. For example for eight layers, vertices b_4 and a_4 are connected through a complete graph to $\{a_2, ..., b_3\}$, but not to a_1 and b_1. The communication of activations from a_1 and b_1 to vertices b_4 and a_4 is thus distributed among the communication from the four vertices $\{a_2, ..., b_3\}$ belonging to the same K_6 as b_4 and a_4 as described in Fig. 3**b**. In general, for an arbitrary number *n* of densely connected layers, the *i-th* vertex, where i is an even(odd) number > 6 and able to communicate with the previous 5(4) vertices through the communication fabric. The edges between these 5(4) vertices and v_i communicate the output activations of the previous 5(4) layers, plus the activations from vertices $\{v_1, ..., v_{i-6(5)}\}$.

B.3 Proof of Theorem 4.3

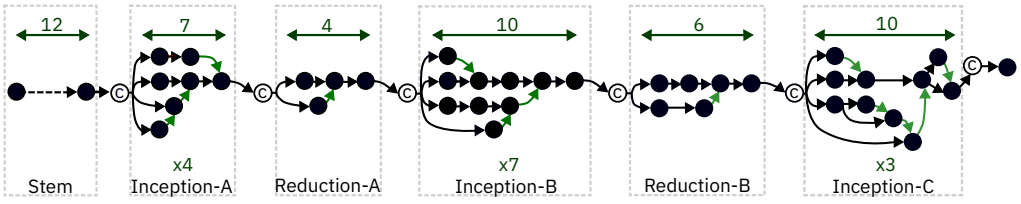

Figure 7: Inception v4 architecture represented according to R1/4 featuring 17 Inception blocks connected through a series of complete bipartite graphs $\{K_{2,2}$(In Stem, not shown in fig.), $K_{1,4}(\times 4), K_{1,2}, K_{1,4}(\times 7), K_{1,2}, K_{1,4}, K_{2,4}, K_{2,1}\}$.

Inception-style architectures feature a less regular, broader spectrum of connections with respect to the previous examples, comprising a sequence of s-p graphs ("Inception blocks") connected in series through concatenation nodes. The number of parallel branches always tapers in the direction of the destination concatenation node and all vertices between source and destination nodes have an out-degree lower than or equal to 2. Let a graph represent one Inception topology. Inception topologies are made up of Inception blocks connected by concatenation nodes. Inception blocks are s-p graphs with four or less parallel branches and vertices with d^o of at most 2. We first prove the coloring of the individual Inception blocks. From P1, every vertex in a 5PP belongs to at least one K_6. Given a maximum of four parallel branches, they are always colorable on a plane with vertices $\{a_n, b_n, a_{n+1}, b_{n+1}\}$, all belonging to a single K_6 with vertices $\{a_n, b_n, a_{n+1}, b_{n+1}, a_{n+2}, b_{n+2}\}$. As $\{a_n, b_n, a_{n+1}, b_{n+1}\}$ can be chosen belonging to the same K_6, every vertex will have possible d^o equal at least to the number of uncolored vertices in that K_6, which is equal to 2. What remains to be proven is the colorability of the concatenation nodes through which they are connected. From P3, complete bipartite graphs with $m + n < 5$ are always colorable. This condition applies to all concatenations in Inception architectures, which are therefore mappable.