E0294: SYSTEMS FOR MACHINE LEARNING

Assignment 5

Author
Pritam Trilochan Gouda
CSA, IISc
April 18, 2024

Question 1

ISAAC enables ML systems that are orders of magnitude more efficient than its nearest competitor architecture, a feat achieved through a novel architecture that integrates memory and computation functions. Implemented in ISAAC's design, so-called in-situ processing functionality allows for direct execution of multiply-accumulate (MAC) operations at memory units, thereby eliminating the latency and energy penalties unavoidably generated through data transfers in conventional architectures. With eDRAM buffers both storage and computational workhorses in each ISAAC tile, direct execution of MAC operations is made possible in memory by exploiting its architecture redesign.
Each tile in the ISAAC array has processing at a fine level of detail — a bespoke In-situ Multiply-Accumulate (IMA) unit for every operating unit. These are connected to one another via memristor crossbars. The reasoning behind the placement of these components is to allow the number of switching elements to be fixed and to eliminate their transport and placement before use. The memristors in these crossbars act like little switches, which change their resistance according to the level of electrical current, so making them excellent devices for emulating general computing functions. Having the memristor switches available intrinsically enhances the system's computational density, which is important for operating computational-intensive matrix operations that lie at the heart of neural network algorithms.
In terms of computational and power efficiency, ISAAC goes well beyond DaDianNao. Specifically, although ISAAC is much more complex in terms of structural compared with DaDianNao, it is still much more efficient in terms of both computational and power efficiency with 478.8 GOPS/s $\times$ mm^2 and 383.7 GOPS/W vs DaDianNao's 286.4 GOPS/s $\times$ mm^2 and 360.7 GOPS/W. These two metrics represent how many operations of ISAAC can be conducted in the same space and how ISAAC uses energy more efficiently.
Another example of ISAAC's sophistication involves mitigating the resolution of the ADC with a clever use of bit-serial computation and sophisticated data encoding. In a sense, this allows some relaxation when it comes to the resolution of the ADC, so power optimisation is more easily achieved without compromising the precision of the overall computation. This is very important because ML computation is power hungry, and that's a big obstacle to scaling up and deploying ML systems in environments where energy-efficiency is significantly important.
However, ISAAC lacks an optimal design. The biggest flaw is that the computational resources of the neural network's final stages are only used in a moderately efficient manner. This indicates that design improvements should be made so that all phases of neural network operation fully exploit the architecture's computational power.
In closing, ISAAC is a pioneering ML hardware project, exhibiting a phenomenal promise of scaling bandwidth to complete 14.8 times more arithmetic work compared with DaDianNao. Efficiency is crucial for improving throughput for large-data systems that implement neural networks and power big-data and AI efforts.
From this, it predicts how far it will travel before encountering an object — information that predicts when the spike output of the memory part of the circuit will be needed to maintain its orientation relative to the world in front of it. In other words, ISAAC's novel fusion of memory and computation in a single energy-efficient architecture outlines a path

forward for the next generation of computational architectures. As ML systems grow in scale and become embedded in our world, ISAAC's success will show us the way forward — how we still need creative new solutions that balance robust performance with renewable energy practices, recognising that this will be an even greater necessity as we move toward a data-driven future.

The practical consequences of ISAAC's architecture for hardware accelerators beyond ML will only be greater. In addition to facilitating a quantum mechanical kind of incremental learning, it demonstrates the viability of performing the most complex computations inside the smallest power envelope possible, which is crucial for the hardware ecosystem as increasingly demanding high-performance computing (HPC) becomes both more powerful and more eco-friendly. It defines what a new category of machine learning accelerator could look like, of which our ISAAC would be just the first example harbinger, but certainly not the last. With this, we have taken the first glimpse into the outer reaches of those engines of intelligence that may one day create the AI of life, if not beyond.

Question 2

The ReMaGN architecture represents a major leap in computational paradigms for GNN training with an on-chip ReRAM-based computation. The proposed approach primarily facilitates an efficient execution of GNN operator's step-specific operations, like vertex updates and edge-aggregation, which lies at the core of GNN learning. ReMaGN performs these operations with the help of two major V-layer and E-layer computations. The vertex-centric (V-layer) and edge-centric (E-layer) computations in Graph Neural Networks (GNNs) can be represented as follows:
For the vertex updates, the ReLU activation function is applied after matrix multiplication of the previous layer's features with the weight matrix for the current layer:

$$Y_v^l = \text{ReLU}(X_v^{l-1} \cdot W^l) \tag{1}$$

For the edge aggregations, the feature vectors are updated by multiplying with the adjacency matrix:

$$X_v^l = A \cdot Y_v^l \tag{2}$$

where A denotes the adjacency matrix encapsulating the graph structure, and W^l is the weight matrix for the l^{th} layer.
Besides using hierarchical 3-D mesh topology as a base for its high-performance architecture, ReMaGN exploits a special 3-D toroidal Network-on-Chip (NoC), capable of high-speed data communication, which is at the core of the system. 3-D mesh network topology of the NoC reduces the number of hops between the PEs, thus avoiding the significant communication latency. This characteristic is of utmost importance for GNN processing, because their communication pattern initiates many-to-one-to-many data transfer.
One of the most interesting aspects of ReMaGN is the judicious use of reduced-precision encoding that allows the model to exchange and store data with high efficiency, yet without sacrificing predictive accuracy. Adopting a 16-bit fixed-point format (fewer bits equals a smaller data footprint), combined with a sophisticated stochastic rounding, does reduce the quantity of stored data, but not the quality of the underlying arithmetic.
This sensible prescription of decreased-precision is not mere speculation but is backed up by actual measurement. The benchmarks compare ReMaGN with the latest GPUs and with the ReGraphX architecture, against different datasets. First shown in Figures 13 and 14 in the paper, ReMaGN's 128x128 Crossbar turns out to be the best compromise configuration, with crossbars of smaller size passing worse than larger, as seen in storage efficiency and energy usage. Figure 13: Energy usage per bit access (blue). Note the logarithmic scale on the Y-axis. Figure 14: Storage efficiency of network, 1000 bits/pixel on a 128x128 image.colorFigure 15: Energy usage scaling for 16x16, 32x32, 64x64 and 128x128. For comparison, standard Graphics Processing Unit (GPU) usage is indicated.The precise control over individual bits provides weak scaling, an simultaneously achieving an energy footprint that improves by orders of magnitude, with an expertly optimised circuit design. By the same token, this kind of computational mechanism enables responsive, environmentally friendly, and massively parallel, scalable computing.
By employing these detailed experiments with datasets like Reddit and Amazon2M, precise performance configurations in ReMaGN presented a significant ability to demon-

strate energy-efficiency and execution times that were way beyond the GPUs benchmarks. Therefore, e.g., the effectiveness of ReMaGN-8 configuration reveals that a reduced precision has very little or no effect on accuracy whatsoever, but savings in terms of computational speed and energy consumption are obviously big.
Moreover, the network's performance is impacted by the architecture, which the developers of ReMaGN have taken into consideration. Its groundbreaking training procedure lets the model process a substantial number of graph divisions in parallel, thus avoiding the memory problems that often complicate the analysis of big, complex graphs. With this strategy, we achieve two main aims: first, the training of GNNs does not just become feasible but also high-throughput, owing to the partitioning of the graph, which is very suitable when the large memory footprint of the GNNs is to be managed.
The research is devoted to the examination of the impact of various hyperparameters configurations on the system efficiency. For example, the different effects of batch sizes are studied, showing that a bigger batch size worsens the model's accuracy. This has paramount importance especially when it comes to the case where there is a scarcity of computation resources in practice.
To round the discussion up, ReMaGN presents a brand-new approach to GNN design, incorporating half-precision computing as well as a 3-D Network-on-Chip (NoC) as an effective solution to high-performance and energy-efficient platform. The main property of Grotzai's algorithm is its time execution speedup and energy savings, which are so well demonstrated that it has no competition with existing GNN training methods. The present results, including for academic research and industrial applications, put forward the usage broad adoption of ReMaGN, which will have the whole the landscape of the graph-based machine learning transformed.

Question 3

part(a):

I have attached the ipynb file

part(b):

Given that:

- Each floating-point number is typically stored using 16 bits (for `tf.float16`).
- The *d_model* value denotes the size of the embeddings and hence the width of the data vector associated with each token.
- The sequence length (*seq_len*) is the number of tokens in the input sequence.

The total number of bits B communicated between two layers for each token is:

$$B = d\_model \times 16 \text{ bits}$$

For the entire sequence of length *seq_len*, the total T is:

$$T = B \times seq\_len = d\_model \times 16 \times seq\_len \text{ bits}$$

If the embedding size increases from one layer to the next, say from *d_model_1* to *d_model_2*, and assuming *seq_len* remains constant, the number of bits communicated to the layer with the larger embedding size is increased proportionally:

$$T_{\text{increase}} = (d\_model\_2 - d\_model\_1) \times 16 \times seq\_len \text{ bits}$$

To provide a visual representation, you could graph the relationship between *d_model* and T to show how the total number of bits scales with the embedding size.