Heterogeneous Memory Architectures for Energy-efficient DNN Task Adaptation on Edge Devices

Zirui Fu, Aleksandre Avaliani, Marco Donato
Electrical and Computer Engineering Department
Tufts University

I. Motivation and Introduction

State-of-the-art DNN models are rapidly growing in size and complexity to tackle more advanced intelligent computations, posing challenges to power- and resource-limited mobile systems-on-chip (SoCs) on which the models are stored and executed. In the NLP domain, for example, after the proposal of the transformer architecture [1], [2], the BERT [3], and GPT [4] model families have been making continuous breakthroughs in GLUE [5] and SQuAD [6] benchmarks with exploding model sizes. The $\mathrm{BERT}_{\mathrm{large}}$ model has 340 million parameters [3], leading to a memory footprint of 1.12GB using single-precision data representation format. Such enormous memory requirements prevent the direct deployment of these models directly onto mobile SoCs.

Efforts such as the ALBERT model [7], have been made to minimize the model size while keeping inference accuracy. However, the downside of such models is that they are highly specialized, losing generalization capabilities, and requiring considerable training efforts to recover accuracy when changing inference task. Real-world DNN application scenarios, such as home-owned healthcare devices and personalized smart assistants make use of simultaneous multi-task inference (MTI) to perform complex operations. These combinations of tasks may include paraphrasing, sentimental analysis, and question-answering for applications in the NLP domain and image recognition, action detection, and object tracking in computer vision. Although existing solutions are effective in alleviating the latency, energy, and area costs of running single tasks, achieving real-time MTI requires running computations over multiple variants of the model parameters, which are tailored to each of the targeted tasks. This approach leads to either unrealistic on-chip memory requirements or expensive off-chip memory access to update the model parameters. Additionally, the deployment of multiple tailored copies of the model parameters is not a scalable solution when the number of targeted tasks increases.

II. Methodology

We propose a memory-centric hardware/software co-design optimization to solve these conflicts from multiple directions. Inspired by the approach used in MEMTI for computer vision tasks [8], we extend the use of residual adapters [9], [10] from the ResNet model to the ALBERT model with the goal of achieving MTI operations with low on-chip memory requirements and minimal overheads for additional tasks. The resulting adapter-ALBERT model keeps a copy of the majority of its layers unchanged from a pre-trained vanilla ALBERT model as the **backbone parameters**, while a separate set of task-specific **non-fixed parameters** are fine-tuned and stored for each separate task. This is achieved by fine-tuning a vanilla ALBERT model on a target task, then invoke adapter modules and fine-tune non-fixed parameters for new tasks while the backbone parameters are excluded from back propagation, so that the backbone parameters are only trained once. While it is possible to train adapters for new tasks in-flight, this work assumes backbone and task-specific parameters are pre-trained offline before deployment.

The block diagram of Figure 1(a) shows the structure of the adapter-ALBERT model. The green and yellow blocks represent the non-fixed layers and backbone of the adapter-ALBERT model respectively. While the model's embedding layers are entirely task-agnostic and the classifier layer is entirely task-specific, the transformer layers are partitioned between fixed and adapter parameters to accommodate different tasks and are mapped to a heterogeneous memory architecture (HMA) obtained by combining CMOS-based SRAM and resistive RAM (RRAM).

In particular, we take advantage of RRAM's high storage density and non-volatility and mask its programming cost and endurance limitations by storing fixed layers that do not have to be updated when switching tasks. The task-specific layers are stored in SRAM and updated during inference over multiple tasks. This architecture ensures its scalability for increased number of suitable tasks without influencing backbone model performance.

We further enhance the storage efficiency with pruning, quantization, and bit-mask encoding optimizations. The ability to distribute parameters across different memory technologies provides the additional benefit of selectively introduce fault tolerance across the different layers. In order to maximize storage density while limiting the impact of faults in RRAM bitcells, we used multi-level cell encoding to store non-zero values, while bit-mask matrices, which are more susceptible to faults, are stored using the more robust single-level cell encoding. Figure 1(b) shows the structure of an accelerator architecture based on EdgeBERT using our proposed hetero-

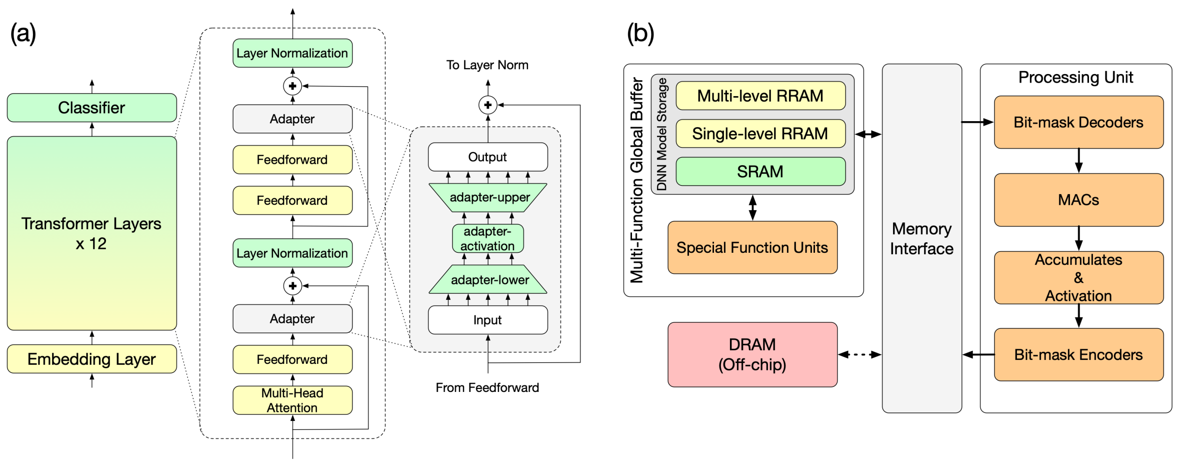

Fig. 1. The overview of (a) the adapter-ALBERT model and (b) the HMA. The colors of the adapter-ALBERT model indicate the backbone parts (yellow) and non-fixed parts (green). The colors of the HMA architecture indicate different roles of components: yellow and green are HMA memory blocks and their colors match the parts of the adapter-ALBERT model; Red DRAM block is off-chip memory; Orange blocks belong to the compute blocks in the accelerator.

geneous memory. We extended NVMExplorer [11] with an accelerator performance model to evaluate the design power, performance, and area of the resulting architecture. We compare layer-by-layer dataflow execution for both adapter-ALBERT and vanilla ALBERT, including multiple design choices of memory array combinations for user-specified optimization targets provided by NVMExplorer.

III. Results and Potential Impact

Our proposed memory architecture, inspired by the introduction of adapter modules has the obvious advantage of reducing data movement costs and maintaining competitive accuracy while performing multi-task inference on the nine GLUE datasets. The fixed, task-agnostic layers are stored in RRAM, offering the opportunity for additional energy savings by powering off the system when idle: Leveraging non-volatility we can reduce the cost of switching tasks to that of loading only the task-specific parameters in SRAM during the power-up stage. This optimization has a profound impact on the execution of multi-task NLP inference on edge devices. For a 3-task MTI scenario, using MNLI, MRPC, and QNLI datasets, our adapter-ALBERT shows clear benefits over the vanilla ALBERT implementation, in which the entire set of transformer parameters and classifier have to be replaced to run on different task. Using 8-bit quantization, adapter-ALBERT requires a memory footprint of 4.5MB (RRAM) and 0.125MB (SRAM), while vanilla ALBERT requires 1.5MB (RRAM) and 2.5MB (SRAM). While the overall on-chip capacity is larger for adapter-ALBERT, we can take advantage of the increased RRAM density compared to SRAM for a net gain of 1.7×, 32×, and 9× in area, energy, and latency respectively.

Looking beyond this specific implementation tailored around the adapter-ALBERT model, our co-design approach can further provide a unified and efficient memory storage to reduce the memory footprint for data-intensive learning algorithms where several DNN models collaborate across modalities and application domains. The key advantage offered by our solution is to provide a more scalable and flexible memory architecture that, by virtue of combining different memory technologies, can hide the limitations of emerging non-volatile memories such as limited endurance, reliability, and write performance. These improvements are supported by carefully selected algorithmic optimizations, which highlights the importance of adopting cross-stack design space exploration in the design of future memory systems, especially when targeting resource-constraint edge devices.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.
[2] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences, 2018.
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.
[4] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.
[5] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2018.
[6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.
[7] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.
[8] Marco Donato, Lillian Pentecost, David Brooks, and Gu-Yeon Wei. Memti: Optimizing on-chip nonvolatile storage for visual multitask inference at the edge. *IEEE Micro*, 39(6):73–81, 2019.
[9] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters, 2017.
[10] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks, 2018.
[11] Lillian Pentecost, Alexander Hankin, Marco Donato, Mark Hempstead, Gu-Yeon Wei, and David Brooks. Nvmexplorer: A framework for cross-stack comparisons of embedded non-volatile memories, 2021.