ICCAD: G: Machine Learning Algorithm and Hardware Co-Design Towards Green and Ubiquitous AI on Both Edge and Cloud

Author: Haoran You\textsuperscript{1} \quad Advisor: Yingyan (Celine) Lin\textsuperscript{1}
\textsuperscript{1}Georgia Institute of Technology, Atlanta, GA \quad *haoran.you@gatech.edu

1 PROBLEM AND MOTIVATION

The escalating complexity of state-of-the-art machine learning (ML) models is marked by their expanding parameters and the substantial floating-point operations (FLOPs) required. For instance, within a brief three-year period, model sizes have surged from 1 billion to an astounding 1000 billion parameters [42], exemplified by OpenAI’s GPT-3, which possesses 175 billion parameters. Training such a model could exceed 4 million GPU hours, require storage across 16 NVIDIA 40GB A100 GPUs, and take several seconds to process a single inference request [3]. There is a significant research gap between such powerful yet large-scale ML models and the limited resources available on edge or cloud computing devices as illustrated in Fig. 1. The challenges associated with this gap not only affect the feasibility of training and serving these models in cloud data centers [6, 23–25] but also limit their applicability on resource-constrained edge devices [22, 30], including smartphones, AR/VR devices, and internet of things (IoT) sensors.

![Figure 1: The research gap between ML model complexities and the resource limitations of edge and cloud devices. Left: Exponential increase in model sizes; Right: Diverse edge and cloud platforms.](image)

To tackle these issues, as shown in Fig. 2, we introduce a co-design strategy that integrates algorithm development with hardware acceleration, fostering a synergy aimed at producing efficient ML systems. These systems are designed to optimize pairs of models and accelerators, alongside streamlining both training and inference processes. The co-design principle has demonstrated its effectiveness across a variety of machine learning model architectures and applications, ranging from convolutional and graph neural networks (CNNs and GNNs) [13, 16] to computer vision (CV) and natural language processing (NLP) Transformers [7, 26], as well as eye-tracking models [39]. This principle applies whether the models are deployed on edge devices like AR/VR or within cloud data centers.

2 BACKGROUND AND RELATED WORK

Efficient Algorithm. Numerous model compression techniques are proposed to reduce the sizes of ML models, including but not limited to pruning [8, 12, 33–35], quantization [9, 27, 45], low-rank decomposition [40, 44], knowledge distillation [11], etc. However, most compression techniques fail to achieve their theoretical performance due to the introduced irregularities and lack of consideration of date movements and scheduling overhead, which usually costs much more than the computation itself. For example, unstructured pruning leads to irregular distributions of non-zeros; low-bit or mixed-bit quantization is not well-supported by commercial GPUs and also introduces bit-level irregularity. These limitations underscore the need for hardware-oriented considerations and co-design.

Algorithm-Hardware Co-Design. Driven by the shortcomings of purely algorithmic optimization, the co-design of algorithms and hardware pushes forward the boundaries of the accuracy-efficiency trade-off on the Pareto frontier [17, 31, 43]. This progress is attributed to the incorporation of hardware attributes, which mitigates scheduling complexities and minimizes overhead associated with data movements. The strategy of algorithm-hardware co-design has been successfully implemented in a variety of ML models and applications. For instance, GCoD [31] optimizes graph neural networks, MELOPPR [18] enhances algorithms for page ranking, ViT-CoD [38] applies to vision Transformers, and EyeCoD [39, 41] is tailored for augmented and virtual reality (AR/VR) applications. These implementations demonstrate the broad applicability and significant impact of co-design in improving both the performance and efficiency of computing systems across different domains.

Hardware-Aware Algorithm. In the realm of cloud computing, a significant number of users depend on GPU-based infrastructure and CUDA environments to handle demanding computational tasks [6]. Given that the hardware specifications are relatively fixed, there is a critical need to adaptively optimize algorithms to fully leverage the capabilities of GPU hardware. This necessity has spurred the development of hardware-aware algorithms for accelerating the training and inference of large vision or language models (LVMs/LLMs). For instance, Mishra et al. leverage the 2:4 sparsity supporance of tensor cores to design sparse deep networks [21]. Lin et al. and You et al. propose tailored solutions for edge devices with MCUNet [19] and ShiftAddNet [30], respectively. Bolya et al. employ token merging to compress data-level redundancy, which is well supported by GPUs [2]. These strategies are essential for optimizing algorithms on GPUs, ensuring that fixed hardware does not hinder the performance and scalability of advanced ML models.
3 APPROACH AND UNIQUENESS

Approach Overview. In light of the identified research gap between the increasing complexities of powerful ML models and the restricted capabilities of edge or cloud computing devices, our research endeavors to bridge the gap through the adoption of novel algorithm-hardware co-design approaches. Specifically, our focus will be on the development of efficient ML algorithms and their corresponding hardware accelerators, tailored for both edge and cloud computing environments. This focus is categorized into four key research areas. As depicted in Fig. 4, for edge computing, our exploration is divided into two primary areas: (1) the integration of algorithm and hardware co-design and (2) the development of hardware-aware algorithms. For cloud computing, our attention shifts to two additional areas: (1) the design of efficient ML models and (2) the efficient training and inference techniques on GPUs. Accordingly, we aim to develop four interrelated research thrusts to synergize efficient ML models, training and inference algorithms, and hardware accelerators. These efforts aim to create a unified and efficient ML ecosystem tailored for both edge and cloud platforms. This holistic strategy is crucial for optimizing computational resources and managing data flow on both edge and cloud.

3.1 Algorithm-Hardware Co-Design

Our unique approach is to co-design both the algorithm and hardware to constitute an efficient ML system that accelerates an array of ML models and applications (see Fig. 3). Notably, we have verified the effectiveness of this collaborative synergy across multiple domains, including GNNs [31], Transformers [38], and AR/VR applications [39], as elaborated below:

- GCoD [31] targets to accelerator GNN inferences with extremely (>99.9% sparsity) sparse graphs. On the algorithm level, GCoD integrates a split-and-conquer GCN training strategy that polarizes the graphs to be either denser or sparser in local neighborhoods without compromising the model accuracy, resulting in graph adjacency matrices that (mostly) have merely two levels of workload and enjoy largely enhanced efficiency and thus ease of acceleration. On the hardware level, GCoD integrates a dedicated two-pronged accelerator with a separate engine to process each of the aforementioned denser and sparser workloads, further boosting the overall utilization and acceleration efficiency.

- ViTCod [38] inherits the spirit to polarize sparse attention maps to be either denser or sparser fixed patterns. Moreover, it further integrates a lightweight and learnable auto-encoder module to trade the dominant high-cost data movements for lower-cost computations. On the hardware level, ViTCod has a dedicated accelerator to coordinate the enforced denser or sparser workloads for reduced computations and boosted utilization, while integrating on-chip encoder and decoder engines for much-reduced data movements.
3.2 Hardware-Aware ShiftAddNet

Powerful ML models suffer from large inference and training costs due to dense multiplications. For example, a single forward pass of the ResNet50 [13] model requires 4G FLOPs of computations and training requires $10^{18}$ FLOPs [35]. There is an opportunity to address this issue: the multiplication can be replaced by bitwise shifts and adds. Our profiling shows that such hardware primitives result in up to 31x unit energy reduction over multiplications [30]. To this end, we aim to develop a hardware-aware ShiftAddNet which reparameterizes ML models with efficient shift and add operations.

- ShiftAddNet [30] utilizes the two complementary bitwise shift and add operations to explicitly parameterize ML models. As shown in Fig. 5 (a), we quantize weights to powers of two to construct shift layers; we replace traditional multiplication with additions for add layers and devise custom backpropagation rules to ensure convergence. Such reparameterization enables finer-grained control of the model’s capacity, leading to a better accuracy and efficiency trade-off.

- ShiftAddNAS [32] further advocates hybrid networks that combine the strengths of both multiplication-based and efficient shift and add operators. It features two key innovations: (1) a hybrid search space that includes both types of operators to develop accurate and efficient neural networks; and (2) a novel weight sharing strategy that accommodates different operators’ weight distributions (e.g., Gaussian for convolutions vs. Laplacian for add operators), significantly reducing supernet size and improving network quality.

- ShiftAddViT [37] aims to design ShiftAdd-based Vision Transformers (ViTs) for achieving end-to-end speedups on GPUs without requiring training from scratch. As depicted in Fig. 5 (b), for attentions, all MatMuls among queries, keys, and values are reparameterized using additive kernels, after mapping queries and keys to binary codes. The remaining MLPs or linear layers are reparameterized using a mixture of experts (MoE) framework that treats multiplication and its primitives, such as shifts, as experts. In this framework, a router is fine-tuned alongside the entire model to direct tokens to the appropriate expert based on their importance, with more crucial tokens requiring higher precision and selecting the multiplication expert. We utilize TVM to implement and optimize the customized shift and add kernels for practical hardware deployment on GPUs.

3.3 Efficient Linearized Transformers

Transformers, either ViTs [7] or NLP Transformers [3, 25, 26], have shown impressive performance but still require a high computation cost as compared to CNNs [13]. One reason is that Transformers measure global similarities through their attention mechanism, which has quadratic complexity relative to the number of input tokens [15, 28]. This limits their performance when scaling up...
parameters. Existing efficient Transformer adopt local attention or linear attention, which sacrifice Transformers’ capabilities of capturing either global or local context.

Our unique solution to this issue is a framework dubbed **Castling-ViT** [40], which trains ViTs using both linear-angular attention and masked softmax-based quadratic attention, but then switches to only using linear-angular attention during inference. As illustrated in Fig. 6, our Castling-ViT framework employs angular kernels to measure query-key similarities through spectral angles. We simplify this process with two techniques: (1) a novel linear-angular attention mechanism that decomposes angular kernels into linear terms while discarding high-order residuals; and (2) the use of two parameterized modules, a depthwise convolution and an auxiliary masked softmax attention, to approximate high-order residuals and enhance learning of global and local information. The softmax attention masks are regulated to gradually reduce to zero and thus incur no overhead during inference. We also test the applicability of these concepts on NLP Transformers to validate their effectiveness.

### 3.4 Early-Bird Training and Inference

As illustrated in Fig. 7, the traditional efficient training and pruning process consists of three steps: (1) fully training the dense models; (2) pruning trained models to obtain a smaller model; and (3) fine-tuning this smaller model (i.e., winning ticket) to restore accuracy. The core problem lies in simplifying this complex process and reducing the costs of training and inference. Our unique “early-bird (EB)” training is expected to largely reduce the associated costs of foundational models on cloud platforms, as exemplified below.

Our proposed **EB Train** [34] reveals a unique “early-bird” phenomenon, where the small and efficient models (i.e., early-bird tickets) are identified during the early stages of the dense model training. Building on this insight, we propose an EB training strategy, which only trains large models for a few epochs and then switches to training the pruned smaller models, as shown in Fig. 7. Moreover, we propose a mask distance metric that can be used to identify EB tickets with a low computational overhead, without needing to know the true winning tickets that emerge after the full training. Such concept is widely adopted in the literature to accelerate the training and inference of Transformers [5], generative models [14], GNNs [36] etc. Our follow-up work, SuperTickets [33], extends the concepts to simultaneously search for efficient neural network architectures and subnetworks within a supernet. Additionally, our subsequent study, Spline EB Tickets [29], offers a theoretical proof supporting the concept of early-bird tickets.

### 4 RESULTS AND CONTRIBUTIONS

#### 4.1 Algorithm-Hardware Co-Design

Adopting our co-design approaches leads to substantial latency savings and efficiency improvements as compared to strategies that optimize only algorithms or hardware. For example, GCoD and ViTCoD achieve speedups of 2.5× and 6.8×, respectively, compared to the prior state-of-the-art (SOTA) baselines AWB-GCN [10] and Sanger [20]. Notably, EyeCoD marks a pioneering work as our compact eye tracking system achieves >300 FPS throughput, a large improvement over the prior SOTA solution with 30 FPS throughput.

#### 4.2 Hardware-Aware ShiftAddNet

We explored the reparameterization of CNNs [30], ViTs [37], and hybrid models [32] with both multiplication and multiplication-free operations. For CNNs, we trained the reparameterized ShiftAddNet from scratch, which aggressively cuts over 80% of the energy cost of training and inference, while maintaining or surpassing the accuracies of existing CNNs and other multiplication-less models. For ViTs, we finetuned the model after reparameterizing pre-trained ViTs with shift and add as illustrated in Fig. 5, notably reducing latency by up to 5.18× on GPUs without sacrificing the accuracy of the original ViTs on various 2D and 3D vision tasks.

#### 4.3 Efficient Linearized Transformers

We apply our proposed linearized attention to both ViTs [40] and NLP Transformers (even LLMs). For ViTs, our Castling-ViT achieves up to a 1.8% higher accuracy or 40% MACs reduction on classification and 1.2 higher mAP on detection under comparable FLOPs, as compared to ViTs with vanilla softmax-based attentions. For NLP Transformers, we find that most linear attentions, originally designed for encoder-based LLMs or ViTs, are ill-suited for autoregressive decoder-based models. We then augment our linear attention by integrating masked depthwise convolution to prevent information leakage, reducing perplexity by up to 6.67 and doubling generation speeds after integrating the speculative decoding [4].

#### 4.4 Early-Bird Training and Inference

Our efficient EB train method using EB tickets [35] can achieve energy savings ranging from 5.8× to 10.7×, while maintaining or improving accuracy compared to the most advanced SOTA training methods for CNNs [8, 13]. Our follow-up studies consistently demonstrate speed improvements for training Transformers [5], GNNs [36], and generative models [14], demonstrating a promising and easily adopted method for tackling the often cost-prohibitive deep network training and inference process on cloud data centers.

### ACKNOWLEDGEMENT

We are grateful for the fruitful collaboration, support, and inspiration from Yingyan Lin, Chaojian Li, Yonggan Fu, Huihong Shi, Yipin Guo, Yang Zhao, Luke Zhang, Cheng Wan, Zhongzhi Yu, Shunyao Zhang, Shang Wu, Jiayi Yuan, Zhanyi Sun, Zhihan Lu, Zijian Zhou, Xu Ouyang, Baopu Li, Ziyun Li, Zishen Wan, Tong Geng, Ang Li, Yue Wang, Xiaohan Chen, Zhangyang Wang, Sicheng Li, Richard Baraniuk, Vivek Boominathan, Ashok Veeraraghavan, Yunyang Xiong, Xiaoliang Dai, Bichen Wu, Peizhao Zhang, Haoqi Fan, Peter Vajda, Yichao Fu, Zheng Wang, Amir Yazdanbakhsh, etc.