USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

CLONE: Customizing LLMs for Efficient Latency-Aware Inference at the Edge

Chunlin Tian, Xinpeng Qin, Kahou Tam, Li Li, Zijian Wang, Yuanzhe Zhao, Minglei Zhang, and Chengzhong Xu, *University of Macau*

https://www.usenix.org/conference/atc25/presentation/tian

This paper is included in the Proceedings of the 2025 USENIX Annual Technical Conference.

July 7–9, 2025 • Boston, MA, USA

ISBN 978-1-939133-48-9

Open access to the Proceedings of the 2025 USENIX Annual Technical Conference is sponsored by

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

CLONE: Customizing LLMs for Efficient Latency-Aware Inference at the Edge

Chunlin Tian, Xinpeng Qin, Kahou Tam, Li Li†, Zijian Wang, Yuanzhe Zhao, Minglei Zhang, and Chengzhong Xu

University of Macau

Abstract

Deploying large language models (LLMs) on edge devices is crucial for delivering fast responses and ensuring data privacy. However, the limited storage, weight, and power of edge devices make it difficult to deploy LLM-powered applications. These devices must balance latency requirements with energy consumption and model accuracy. In this paper, we first quantify the challenges of deploying LLMs on off-the-shelf edge devices and then we present CLONE, an in-depth algorithm-hardware co-design at both the model- and system-level that intelligently integrates real-time, energy optimization while maintaining robust generality. In order to maximize the synergistic benefits of these algorithms in always-on and intermediate edge computing settings, we specialize in a 28nm scalable hardware accelerator system. We implement and extensively evaluate CLONE on two off-the-shelf edge platforms. Experiments show that CLONE effectively accelerates the inference process up to 11.92×, and saves energy up to 7.36×, while maintaining high-generation.

1 Introduction

Large language models (LLMs) [19, 80, 96, 109, 132] are reshaping artificial intelligence for their remarkable performance to comprehend human language and handle language-related tasks [120, 129, 136]. Though born from the cloud, deploying LLMs on edge devices such as personal PC, smartphones, robots, and even IoT devices is becoming an important trend [81, 119]. First, on-device LLM can be widely used to support different kinds of applications, including chatbots [6, 114], robotics [61, 162], and autonomous vehicles [144]. Moreover, directly accessing the LLM on the edge not only preserves data privacy but also can provide instant service without relying on a stable internet connection. Despite the potential benefits, deploying LLMs on commercial-off-the-shelf (COTS) edge devices faces stringent space, weight, and power (SWaP) constraints [10, 138] due to the billion-parameter size coupled with intensive computing costs. For instance, the widely used Llama-7B model [132] requires approximately 14GB of memory for inference, even when utilizing 16-bit precision (FP16) format. However, the available RAM on typical edge devices only ranges from 4GB to 12GB [125]. Furthermore, inferring a single token with Llama-7B requires approximately 14 TFLOPs for an 11-token prompt [25], which is roughly 360× the 39 GFLOPs needed by VGG-19 to process a single image with an input resolution of 224 × 224 [36]. Moreover, LLMs' inference process demands substantial energy consumption. For instance, GPT-3 [31] consumes 300 J/response [20] on an NVIDIA A100 GPU, which is 400× the 0.75 J/response required by ResNet-50 [72]. Thus, intensive memory footprint, high computational demands, and significant energy costs pose severe bottlenecks to deploying LLMs at the edge.

Limitation of Prior Arts. To break the SwaP constraints, several optimization techniques have been proposed. Techniques such as model architecture search [54, 60, 88], quantization [26, 55, 57, 91, 143], and pruning [27, 30, 32, 84] help to reduce the size of models while preserving their performance. However, these methods typically focus on optimizing the model itself and may not fully address the system-level trade-offs related to storage and weight. Additionally, recent improvements in LLM compilers and software stacks [56, 99, 113, 128] have facilitated the adoption of co-processors to sustain high performance by leveraging co-processors or near-sensor processing alongside GPUs [4, 44, 57, 74, 74, 86, 107, 111, 118, 122, 148, 159]. However, they incur additional computational/communication overhead that significantly reduces the edge device lifecycle and interferes with the smooth execution of other applications. To address power constraints, Dynamic Voltage and Frequency Scaling (DVFS) [10, 47, 48, 65, 112] has been widely adopted to adjust the power usage of processors by dynamically changing their voltage and frequency. However, most existing approaches are designed for discriminative CNN or RNN and treat the entire network as a black box during tuning. Generative LLMs have not undergone the necessary scrutiny due to the auto-regressive inference schemes and the heterogeneity

of stochastic requests and generated outputs. *Thus, an edge-tailored LLM system that intelligently coordinates model- and system-level optimizations, balancing energy efficiency, latency, and model performance is urgently required.*

Challenge. Designing such a customization system is not straightforward and faces the following critical challenges. First, the "billion-parameter" poses a significant barrier to deployment on resource-constrained edge devices due to the high memory footprint and computation cost. How to customize the LLM according to the hardware profile of a specific edge device to effectively meet the memory constraint and reduce the computing latency while ensuring the generating capability is the first critical challenge. In addition, when deploying customized LLMs on edge devices, LLM-powered applications feature stochastic and dynamic I/O, unlike traditional models with stable inputs. Moreover, tailored layers introduce unique approximation characteristics, while interference from co-running applications during mobile execution can cause runtime variance. Therefore, designing an efficient system-level controller to meet latency targets while optimizing energy consumption is another challenge. Furthermore, model-level (accuracy/latency) and system-level (energy/latency) optimizations in isolation can be naive, often resulting in unnecessary energy consumption or reduced accuracy. Effectively coordinating these optimizations to balance accuracy, latency, and energy efficiency is the third challenge.

In this work, we present CLONE, a comprehensive system that intelligently customizes LLM for efficient latency-aware inference on commercial-off-the-shelf edge devices. It adopts a hierarchical structure to jointly handle model- and system-level optimization in a unified manner to consider real-timeliness, model accuracy, and energy efficiency jointly. Specifically, CLONE consists of two main phases: offline device-specific model tailoring and online latency-aware system optimization. Within the offline tailoring process, for a specific device, CLONE reformulates the tailoring of target LLMs for a specific device as a generative task within a continuous representation space. Using an encoder-evaluator-decoder architecture, it generates optimal pruning configurations through gradient-based optimization, effectively bridging the gap between resource constraints and model performance. To support diverse edge applications [6, 138, 146], CLONE performs parameter-efficient fine-tuning on the customized LLM using multiple plug-and-play Low-Rank Approximation (LoRA) [52] adapters, ensuring adaptability and optimized performance. Once the customized model is ported to edge devices, during online inference, CLONE employs a Mixture-of-Experts (MoE) [59, 131] router to dynamically integrate optimal LoRA adapters, enabling more precise responses to stochastic, complex, or mixed-task end-user requests. At the system level, CLONE applies Dynamic Voltage and Frequency Scaling (DVFS) to the per-token, autoregressive inference process of LLMs, optimizing the supply voltage (V_{DD}) and operating frequency (F_{req}) to minimize energy consumption while meeting real-time constraints. Recognizing the LLM layer characteristics, especially post-pruned uneven parameter, CLONE enhances the granularity of DVFS adjustments at layer boundaries. This provides greater flexibility and adaptability for power-sensitive systems compared to traditional workload-level (whole model) black-box optimizations. While existing vanilla DVFS on edge devices are typically discrete, CLONE introduces a learning-based DVFS approach to reduce budget gaps that maximize system efficiency and model performance. To fully harness the benefits of these technologies, we have developed a specialized 28nm scalable hardware accelerator system. It contains a LoRA Processing Unit (LPU) for hot-swapping adapters, enabling dynamic model performance adjustments through dedicated data paths, and a Special Function Unit (SFU) designed for fine-grained and continuous DVFS adjustments equipped with a fast-switching low-dropout (LDO) voltage regulator and an all-digital phase-locked loop (ADPLL). The LPU supports the runtime MoE router algorithm to handle stochastic requests, eliminating bottlenecks of general-purpose processors and boosting computational efficiency. The SFU implements the learning-based DVFS algorithm to effectively manage runtime variance and swiftly achieve the desired V_{DD} and F_{req}. Specifically, we make the following key contributions:

- We propose CLONE, an efficient software-hardware co-design system for customized LLM inference on resource-constrained edge devices, where the model- and system-level coordinators cooperate to intelligently balance the inference speed and energy efficiency while maintaining robust generative performance.
- We design hierarchical offline/online optimization phases that effectively coordinate customization by addressing static model weights, hardware characteristics, stochastic I/O patterns, and runtime variance. Additionally, a 28nm accelerator with specialized datapaths supports these algorithms for optimized performance.
- To evaluate the effectiveness of CLONE, we conduct extensive experiments based on commercial-off-the-shelf edge devices, representative LLMs and benchmarks.

2 Background and Related Work

2.1 Large Language Models

LLM Architecture. In contrast to traditional deep neural networks (DNNs) and convolutional neural networks (CNNs) [38, 76], which integrate diverse types of layers (e.g., convolutional (CONV), fully connected (FC), recurrent (RC), pooling, etc.) designed for specific tasks, large language models (LLMs) predominantly consist of a uniform stack of transformer decoder layers. For instance, Llama-7B [132] adopts

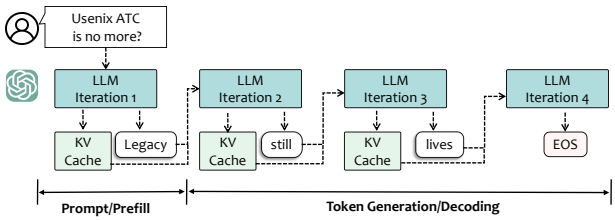


Figure 1: Overview of LLMs autoregressive inference.

a homogeneous architecture composed of 32 identical LlamaDecoderLayers. Each decoder layer encompasses two core components: LlamaAttention and LlamaMLP. Despite the structural uniformity across decoder layers, their contributions to model efficiency and effectiveness vary significantly [26, 137]. Consequently, optimizing the inference execution of LLMs necessitates a detailed analysis of the individual impact of each layer (§3.1).

LLMs Inference. LLMs process a structured sequence involving multiple forward passes through the model to sequentially generate each output token. Figure 1 shows the inference process with a simple example. Typically, this process mainly contains two stages [3, 75, 118]. **1) Pre-fill** takes a prompt sequence and generates the key-value (KV) cache for each Transformer layer of LLM. Upon receiving the prompt *"Usenix ATC is no more?"*, the tokenizer embeds input as tokens, denoted as $X_{in} \in \mathbb{R}^{n \times d}$, where d is the hidden size and n is the length of input token. Then, the LLM handles all input tokens in parallel during a single forward iteration to generate a KV cache. The output of attention is sent to MLP to generate the first output token *"Legacy"*. Large-scale matrix multiplications are required to generate the KV cache, which makes the pre-fill computing intensive. **2) Decoding** utilizes and updates the KV cache to generate tokens step-by-step. Following the generation of the first token, the LLM leverages the KV caches prepared earlier and adds new information to them. The creation of each new token is influenced by the tokens generated before it. During each token generation, for the input $X_{dec} \in \mathbb{R}^{1 \times d}$, attention layers load the previously stored KV cache, and new KV pairs are computed and concatenated to the existing cache. The output of the last decoder layer is sent to the final prediction layer to predict the next token sequentially. It executes iteratively until an End of Sequence (EOS) token is encountered or a predefined termination criterion is met. Unlike traditional models with fixed input formats and structured workflows [38, 76], LLM inputs and outputs are highly non-deterministic. This stems from the diverse, open-ended nature of user prompts, which vary widely in structure, intent, and context [15, 87, 89]. Additionally, autoregressive token generation is inherently probabilistic, driven by sampling methods [50, 111, 115, 132] and variability in training datasets [11, 13, 134], making outputs context-sensitive.

2.2 Bottlenecks of Deploying Edge LLMs

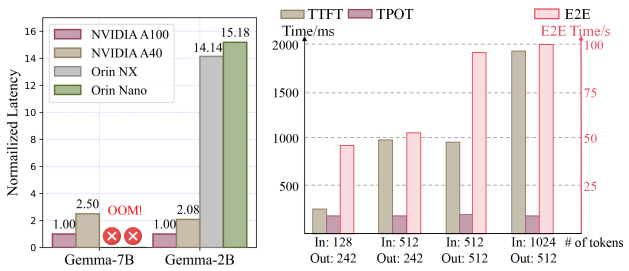
Table 1 lists specifications for server-level and edge-level processors commonly used for ML workloads, highlighting resource collapse. Despite the potential benefits, including privacy preservation and instant responses without depending on a stable internet connection [42, 105], deploying LLMs on the edge faces the following critical bottlenecks.

1) High Memory Footprint. The main contributors of "billion-parameter" LLMs are model weights (memory is occupied by the model parameters) and KV cache (memory is occupied by the caching of self-attention tensors to avoid redundant computation). For example, Llama-7B in 16-bit precision requires approximately 14GB memory ($7B \times$ sizeof(FP16)). Its architecture with 32 layers, 32 heads per layer, and a head dimension of 128 incurs a memory cost of 0.5MB per token, accounting for K and V matrices. Consequently, processing 4096 tokens demands 2GB, limiting the size of models that can be deployed on edge devices with 4–12GB memory [125] and often causing Out-of-Memory (OOM) errors.

Table 1: Popular ML hardware specifications.

GPU Types	Peak Perf.	Memory	Bandwidth	Peak Power
Server-level				
NVIDIA A100	312 TFLOPS	80GB	1935 GB/s	300W
NVIDIA A40	149.7 TFLOPS	48 GB	696 GB/s	300W
Edge-level				
Jetson Orin NX	100 TOPS	16GB	102.4GB/s	25W
Jetson Orin Nano	40 TOPS	8GB	68 GB/s	15W

2) High Inference Latency. Inference latency is a crucial metric for mobile optimization because if the latency of a service exceeds the human-acceptable thresholds (e.g., 33.3 ms for a 30 FPS video frame rate [28, 161] or 50 ms for interactive applications [29, 92]), end-users will abandon the service [161]. To evaluate inference QoE for LLMs, there are three main metrics: 1) *Time To First Token (TTFT)* latency from input to the output of the first token (prefill). Low TTFT means fast response, which is essential for user experience in real-time interactions. 2) *Time Per Output Token (TPOT)* latency to track the auto-regressive process of each token generated serially. TPOT refers to how each user perceives the "speed" of the model. 3) *End-to-end (E2E)* latency, the overall time it takes for the model to generate the full response for a user, denoted as: $E2E = (TTFT) + (TPOT) * N$, where N is the generated token number. Figure 2 (a) illustrates the latency across different processors running the same LLMs, representative Gemma-2B/7B [96], on the Wikitext2 [95] dataset. While Gemma-7B is OOM for edge device. For the same user request on Gemma-2B, the Orin Nano experiences 15.18× more latency than the NVIDIA A100, significantly impacting the Quality of end-user Experience (QoE) [53, 68, 69, 90]. Additionally, LLMs operate with an auto-regressive and probabilistic inference process [121]. Figure 2 (b) shows



(a) Server/edge latency. (b) Latency breakdown.

Figure 2: LLMs inference latency analysis. (a) Latency efficiency of two common LLMs (Gemma-2B/7B) inference use cases over server-level and edge-level processors. (b) Inference latency with different prompt and output tokens.

as prompt sizes increase (128 → 512 → 1024) inference Gemma-2B on Orin NX. We can observe that the number of pre-filled tokens increases significantly (255→1935ms) due to parallel processing capabilities. Conversely, the TPOT remains relatively stable (180∼200ms) due to serial decoding generation. However, the E2E is more than 100s, which fails to serve the end-user demands, indicating opportunities for optimizing practical inference performance.

3) High Energy Consumption. In server-level GPU deployments, it is typically assumed that these units are plugged into wall power due to the substantial energy demands of LLMs inference [4, 64, 74, 107, 111, 141]. However, this assumption does not hold for edge-level devices such as robots and autonomous vehicles, where energy availability is severely restricted [66]. In contrast to conventional on-device tasks, LLM inference consumes an order of magnitude more energy. For example, a Google search driven by a large AI model expends 8.9 watt-hours (Wh) of energy, approximately 30 times the 0.3 Wh required by a standard Google search [124]. Importantly, "Scaling Laws" [21, 63] suggest that as model parameters scale up, there is a corresponding increase in both performance and energy consumption. Transitioning from Gemma-2B to 7B, for instance, boosts accuracy from 42.3% to 64.3% on the MMLU benchmark [46]—a comprehensive evaluation platform—at the cost of tripling energy usage. Consequently, there is a pressing need to optimize the energy efficiency of LLM inference at edge to meet QoE.

3 Motivation

In this section, we present key observations from model-intrinsic dimensions (layer characteristics (§3.1)), stochastic I/O (§3.2)) and system factors (hardware and runtime variance (§3.3)) for practical LLM inference on edge devices. The design space is analyzed across three critical axes: generation quality, latency, and energy efficiency.

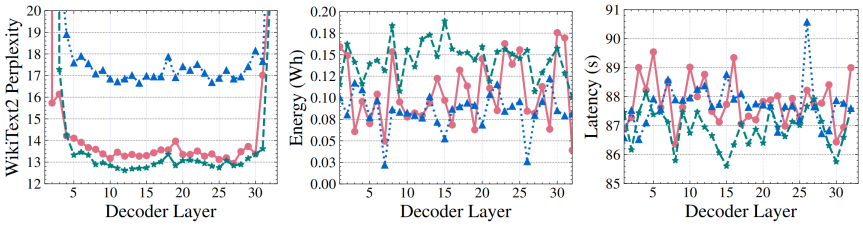
3.1 Heterogeneity of Model Characteristics

In order to investigate the contribution of different decoder layers, we conduct an in-depth analysis of the inherent sensitivity of the stacked transformer architecture across three critical dimensions: generative ability, energy efficiency, and latency. For general-purpose, we employ zero-shot perplexity (PPL) [14, 96, 132], a common metric, to evaluate generative abilities, lower PPL indicates higher model adeptness in predicting sequence tokens. Figure 3 (a) presents the PPL on the WikiText2 [95] dataset after removing specific decoder layers in the Llama-7B, Llama2-7B [133], and Vicuna-7B [160] models. The results show that the front and back layers have higher PPL values than the middle layers. Because the front layers play a critical role in feature extraction, while the back layers significantly influence output generation, both contribute markedly to model performance. while LLMs maintain a homogeneous layer structure, the generative impact varies by layer due to input and output data stream heterogeneity. To evaluate the system effectiveness, we infer LLMs on WikiText2, monitoring energy consumption and duration using CodeCarbon [24]. Figure 3 (b, c) shows that removing different decoder layers yields non-uniform changes in end-to-end energy and latency. This suggests that, beyond nominal parameter counts, each layer contributes heterogeneously to practical resource usage. We attribute this to (i) varying sequence-length–dependent workloads, (ii) hardware-level cache and parallel-sync effects, and (iii) non-linear inter-layer interactions typical of deep networks.

► *Motivation 1*: LLM layers contribute unevenly to effectiveness and efficiency, underscoring opportunities for model customization by pruning non-essential components and system optimization through fine-grained layer-wise tuning.

3.2 Heterogeneity of Stochastic Input/Output

As discussed at §2.1, inference serving systems exhibit significant non-deterministic due to the diversity of input prompts, and LLM auto-regressive output exhibit distinct execution behaviors. **Case study.** As shown in Figure 5 (a), we first illustrate the distribution of prompt input and generated output tokens on an Azure LLM inference services trace [111]. These long-tail patterns and the unpredictability of token generation lengths highlight the need for dynamic resource allocation to manage varying computational demands. **Performance impact.** To analyze the inherent patterns of mixed-task end-user requests, we visualized the task embedding similarities of the Flanv2 dataset [135] using a heatmap, as shown in Figure 4. Each axis enumerates the individual subtasks within the dataset, and each matrix cell quantifies the pairwise correlation between subtasks. The results reveal significant data heterogeneity across tasks. **System impact.** Otherwise, requests of different input and output lengths possess different compute and energy characteristics. As shown in Figure 5



(a) Generative-ability (b) Energy (c) Latency

Figure 3: Layer sensitivity analysis: removing specific layers one-by-one to measure each layer's impact on Llama-7B (●), Llama2-7B (⋆), and Vicuna-7B (▲).

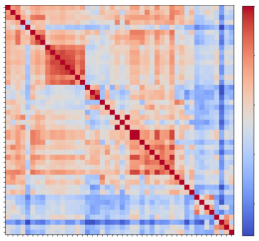
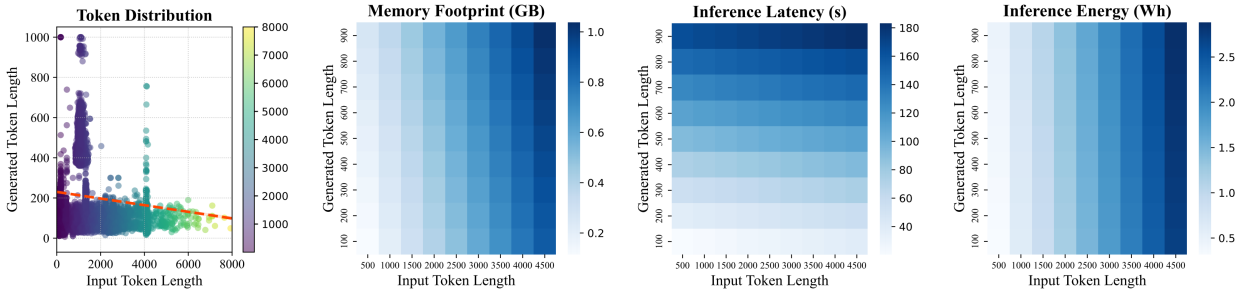


Figure 4: Task embedding similarity heatmap.



(a) Practical inference traces. (b) Memory consumption. (c) Latency analysis. (d) Energy consumption.

Figure 5: Comprehensive analysis of LLM inference system using Gemma-2B on NVIDIA Orin NX. (a) Analysis of practical LLM inference traces., while (b-d) highlight memory, latency, and energy consumption w.r.t. input and generated length.

(b-d), we measure Gemma-2B [96] on Nvidia Orin NX with different input and generated length to evaluate the memory footprint, latency and energy consumption. Memory consumption is influenced by both static model parameters and the dynamic KV cache, which grows as the number of tokens to be processed increases. E2E latency is primarily impacted during the decoding phase, as it is step-by-step, whereas the prefill phase computes all prompts in parallel. Consequently, an increase in generated tokens significantly impacts overall latency. Energy consumption is higher during the prefill phase due to large-scale matrix operations, while the decoding phase, processing one token at a time, has lower power demands. Inefficient system performance degrades QoE, making it critical to meet diverse LLM request demands without compromising output quality.

► *Motivation 2*: Stochastic I/O with mixed-task requests underscores the need for precise inference services to improve model performance. Similarly, significant resource slack suggests potential for fine-grained, LLM-specific system optimizations to improve utilization efficiency.

3.3 Heterogeneity of Runtime Platform

Practical system heterogeneity encompasses performance and energy variations arising from platform diversity (static heterogeneity, Table 1), diverse LLM application demands, and interference from co-running applications (dynamic heterogeneity). Table 2 lists LLM requests from various applications or tasks, each with distinct QoE demands [83, 130, 139, 140, 142, 147, 152, 161]. To investigate the impact of edge device hardware and concurrent running apps, we deployed the Gemma-2B model on two NVIDIA platforms—Jetson Orin NX and Orin Nano—and processed a workload comprising 128 prompt tokens and 242 generated tokens (the first scenario in Figure 2(b)). We profiled the resulting inference behavior, and Figure 6 reports the TTFT, TPOT, and E2E inference latency for both devices, alongside the corresponding performance degradation observed in a foreground web-search application [78, 161]. We find that 1) Due to the varying processing capabilities of different edge devices, the response time to the same end-user request can differ significantly. For instance, when both are in an idle state, the Orin NX, with its superior computational power, provides more timely feedback (TTFT: 255; TPOT: 198 ms) compared to the Orin Nano (TTFT: 268ms; TPOT: 231ms). 2) Resource contention caused by user interaction with the concurrent running app prominently slows down the model inference progress. For instance, the E2E inference latency on Orin NX is 42.6s when there is no user interaction in the foreground. However, it increases to 61.4s with a website searching application running in the foreground. This highlights the impact of latency on user experience, as service abandonment occurs when latency

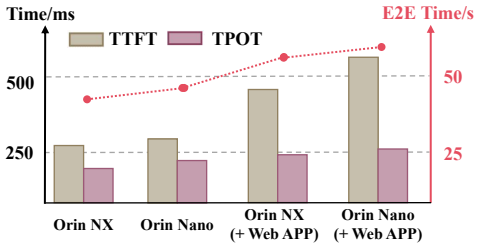


Figure 6: Impact of different hardware devices (Nano/NX) and concurrently running apps (web search) on edge LLMs inference.

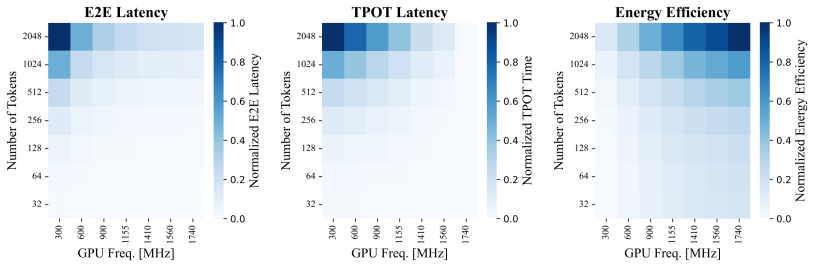


Figure 7: Impact of different token lengths and GPU frequencies on a) end-to-end (E2E) latency, b) time per output token (TPOT), and c) energy efficiency.

Table 2: SLOs of various LLM applications. Time To First Token (TTFT): refers to the latency from input to the first token output (prefill), while Time Per Output Token (TPOT) is the latency for each subsequent token (decode).

LLM Application	Service-Level Objective
Chatbot [5, 108]	Readable TTFT/TPOT
Search Engine [97]	Low TTFT, Medium TPOT
Event Logger [1, 58, 110]	Tolerable TTFT, Medium TPOT
Smart Reply [40, 98]	Low TTFT, Low TPOT
Code Generator [37]	Medium TTFT, High TPOT
Virtual Assistant [7, 39]	Low TTFT, Medium TPOT

exceeds the human-acceptable threshold [83, 161]. Figure 7 demonstrates that higher GPU frequencies reduce E2E latency and TPOT by accelerating token processing (a, b) while supporting energy-efficient control via frequency scaling (c). Dynamically adjusting processor frequencies based on workload size enhances efficiency with minimal performance impact, where the optimal frequency is determined by the LLM architecture and output length. Although DVFS [47, 48, 65] is commonly used to scale voltage and frequency for low-intensity workloads, existing schemes often rely on coarse-grained, black-box, workload-level adjustments with heuristic methods [10, 47, 48, 65, 112].

► ***Motivation 3:*** Hardware type and LLM application SLOs significantly impact end-user QoE, while resource preemption from foreground applications exacerbates runtime system heterogeneity. Accurate runtime estimation of application demands and device capabilities is crucial for guiding DVFS and optimizing LLM inference system efficiency.

4 CLONE: Design

4.1 System Overview

Figure 8 presents the architecture of CLONE which can be mainly divided into the following two phases: 1) offline device-specific tailor and 2) online latency-aware optimization. The overall workflow of CLONE can be represented as

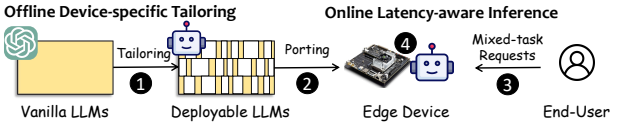


Figure 8: The overview of CLONE, including offline device-specific tailoring and online latency-aware inference.

the following main steps. ❶ CLONE leverages edge hardware profiles and LLM architecture using an encoder-evaluator-decoder framework to reframe LLM compression as a data-driven generative task. This approach addresses memory constraints, meets latency and energy requirements, and preserves generation performance. Plug-and-play LoRA adapters further fine-tune the tailored LLM for diverse edge applications. ❷ The deployable model is then ported onto the edge device. ❸ During runtime inference, to handle stochastic and mixed-task user requests, CLONE integrates a MoE router to dynamically select optimal LoRA adapters, enhancing model performance. ❹ At the system level, CLONE incorporates a learning-based DVFS controller to adjust device voltage (V_{DD}) and frequency ($Freq$) dynamically at layer boundaries during per-token autoregressive inference. This minimizes energy consumption while ensuring target latency is met. Furthermore, in order to fully leverage the synergistic advantages of these algorithms, we develop a specialized 28nm scalable hardware accelerator system designed to further boost energy efficiency through customized hardware integration.

4.2 Offline Device-specific Tailoring

According to *Motivation 1*, LLMs demonstrate considerable parameter redundancy, especially within their intermediate layers. This variance in contribution across layers not only affects performance but also system efficiency, facilitating tailored adaptations for edge deployment. To meet heterogeneous hardware constraints, CLONE redefines LLM pruning as a generative task, surpassing traditional manually designed discrete heuristic optimizations to identify and remove non-essential structures in a system-efficient manner. Specifically,

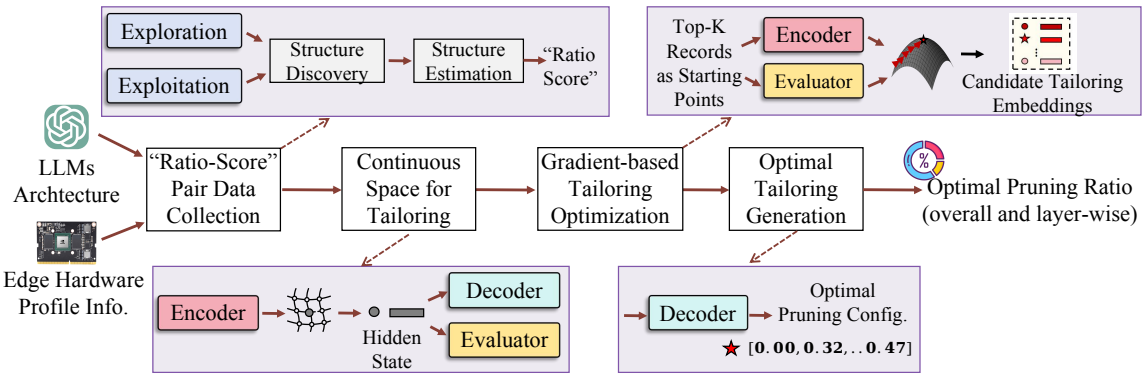


Figure 9: Hardware-aware tailor workflow. Four key components: "ratio-score" data collection, continuous space for tailoring, gradient-based tailoring optimization, and optimal tailoring generation.

it employs a data-driven tailor that identifies the optimal pruning configuration through gradient-based optimization within a continuous representation space. As illustrated in Figure 9, the generative tailor comprises four key modules:

1) *"Ratio-score" Data Collection.* Considering the memory constraints of the edge device and the dimensions of the model to be deployed, we initially establish the overall reduction ratio for the LLMs. For fine-grained optimization, we develop an exploration-exploitation strategy to determine layer-wise level pruning ratios r_i. This approach utilizes heuristic-based pruning methods [8, 30, 94] to produce high-quality ratios and incorporates random pruning ratios as part of the exploration process. In contrast to prior works [45, 55, 102] primarily focus on reducing the model size and computational overhead, CLONE integrates broader criteria including generation ability, inference latency, and energy cost. We define a holistic metric function f to characterize overall performance for a given r_i, aiming for optimization suitable for resource-constrained edge devices:

$$s_i = f(r_i) = \frac{1}{ppl_i} \times \left(\frac{E}{e_i}\right)^{\mathbf{1}(E<e_i)\times\alpha} \times \left(\frac{T}{t_i}\right)^{\mathbf{1}(T<t_i)\times\beta}, \quad (1)$$

where ppl is the zero-shot perplexity [14], quantifies the generative capabilities, with lower values indicating more precise model predictions. T and E denote the latency and energy budgets specific to edge, respectively. t_i and e_i represent the latency and energy consumption for a given ratio r_i. $\mathbf{1}(x)$ is an indicator that returns 1 if condition x holds, and 0 otherwise. Thus, configurations that exceed the thresholds T and E are penalized by developer-specified factors α and β, both set to 2 in our implementation. Finally, we obtain comprehensive ratio-score pairs, denoted as $\mathcal{P} = (r_i, s_i)$.

2) *Continuous Space for Tailoring.* Traditional approaches [8, 30, 94] often employ discrete pruning spaces, resulting in heuristic and incomprehensive pruning ratios. In order to characterize the continuous tailoring optimization space of edge devices, we implement an encoder-evaluator-decoder framework. This architecture includes a single-layer LSTM network [49] functioning as both encoder and decoder, alongside a feed-forward neural network serving as the evaluator. This setup effectively embeds ratio-score pairs (r_i, s_i) into a continuous representation space Θ.

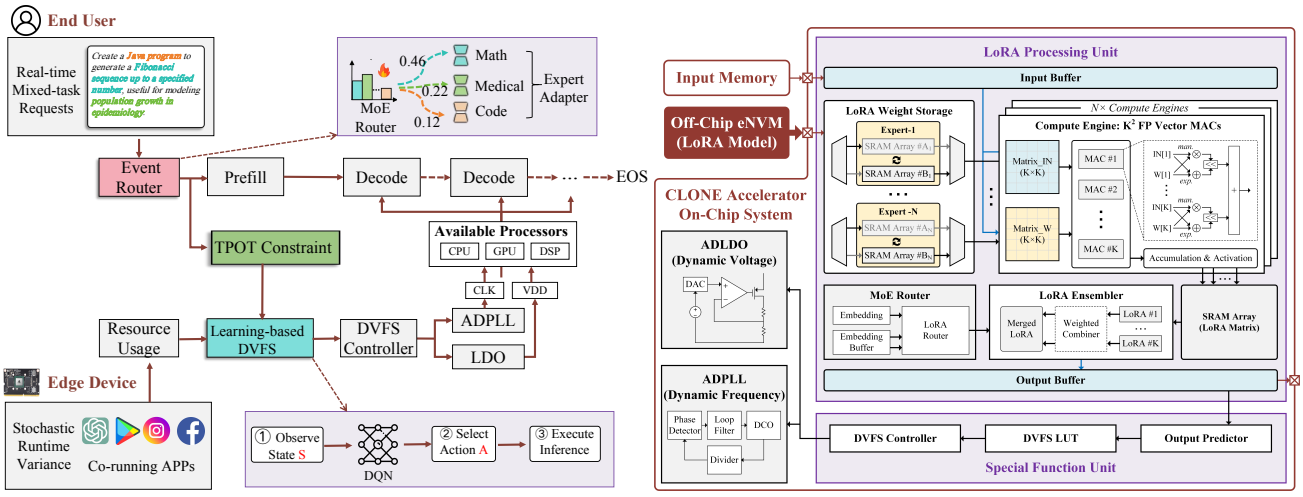
3) *Gradient-based Tailoring Optimization.* With the well-trained representation space Θ, we utilize a gradient-based optimization method to identify the optimal pruning configuration. We select top-K collected pairs serving as starting points to ensure effective initialization. Denoting such starting points as $\mathcal{E}r$, we initiate the optimization process along the gradient direction driven by the evaluator π:

$$\mathcal{E}_r^* = \mathcal{E}_r + \eta \frac{\partial \pi(\mathcal{E}_r)}{\partial \mathcal{E}_r}, \quad (2)$$

where $\mathcal{E}_r^*$ denotes the optimal pruning configuration representation, and η is step size used in the gradient update.

4) *Optimal Tailoring Generation.* Based on the optimal pruning representation $\mathcal{E}_r^*$, the optimal pruning configuration r^* is identified by the trained decoder ξ, denoted as $r^* = \xi(\mathcal{E}_r^*)$. To generate r^* iteratively without pre-specifying the ratios, we employ a beam search strategy [33], allowing for a systematic exploration of potential configurations to achieve the best possible performance. The generation process continues until the stop token $\langle$EOS$\rangle$ is encountered, enhancing the adaptiveness of the model configuration. Finally, remove non-essential groups through structural pruning, guided by the generated optimal pruning ratio.

According to *Motivation 2*, the patterns of user application usage and the inter-dependencies among tasks offer opportunities for optimization in downstream tasks, which can reduce the costs of fine-tuning and improve model performance. Therefore, plug-and-play LoRA adapters are utilized to enhance the tailored LLM generalization capabilities across diverse downstream applications. For n downstream applications, a set of LoRAs, $\Phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$, is initialized, where each LoRA $\phi_i = BA$ is trained per task. The forward computation is:$y' = y + \Delta y = \mathbf{W}_0 x + BAx$, where $y' \in \mathbb{R}^d$ is the output, $x \in \mathbb{R}^k$ is the input, $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$ with



(a) Software (b) Hardware

Figure 10: Online latency-aware inference workflow. **Left (Software)**: CLONE employs a request-wise MoE router for dynamic LoRA configuration to manage layer-wise DVFS, optimizing energy efficiency within the specified latency targets. **Right (Hardware)**: The hardware accelerator system features a request-wise LoRA Processing Unit (LPU) designed for LoRA adaptation, and a Special Function Unit (SFU) tailored for pre-token DVFS at token layer boundaries.

$r \ll \min(d,k)$, and $\mathbf{W}_0$ are frozen LLM parameters. B is initialized to zero and A with Gaussian values.

4.3 Online Latency-aware Inference

Motivation 2 & 3 reveals that current approaches to real-time inference are constrained to workload-level resource adjustments, resulting in energy wastage. It identifies a crucial potential for system optimization via fine-grained management of performance and energy. While offline device-specific tailoring effectively reduces memory constraints and enhances downstream performance, the achieved latency can fluctuate significantly due to diverse user requests and runtime variability, potentially violating real-time latency constraints and increasing energy consumption. As shown in Figure 10, at the model level, a MoE-based router dynamically selects the optimal LoRA adapter based on stochastic user prompts, improving the accuracy of LLM-powered application responses. At the system level, CLONE integrates a token predictor to estimate the output token count, guiding a learning-based DVFS controller to adjust frequency and voltage for each token at layer boundaries. This minimizes per-request energy consumption while adhering to real-time latency targets. Additionally, CLONE embeds these efficient algorithmic enhancements into a 28nm on-chip accelerator architecture.

Request-wise MoE-based Router. In practice, end-user requests cover a diverse range of prompts, each associated with different tasks. Consequently, as shown in Figure 10 (a), an MoE (Mixture-of-Experts) [59, 117] router has been developed to dynamically and effectively merge LoRA modules for each mixed-task prompt. This development significantly extends the plug-and-play capabilities of CLONE. A set of experts $\mathbf{E} = (E_1, E_2, \ldots, E_N)$, where each expert E_i represents a tuned LoRA module. For input x_i, the output y_i:

$$y_i = \mathbf{W_o} x_i + \sum_{j=1}^{N} \omega_{ij} \cdot E_j(x_i), \tag{3}$$

where ω_{ij} modulates the contribution weights of each expert. Unlike traditional methods [131, 150] that incur extra computation and storage from trainable gate functions, we propose a parameter-free soft MoE method leveraging dynamic prompts. Using a sentence-embedding model, Γ (here adopting BGE [17]), the input embedding is formulated as $\Gamma(x)$. For each LoRA module ϕ, the embedding is obtained from randomly selected domain-specific samples k, expressed as $\Gamma(\phi) = \frac{1}{k}\sum_{i=1}^{k}\Gamma(k_{i\cdot\phi})$. To measure the similarity between the LoRA module ϕ and the prompt x, we leverage the cosine similarity, denoted as σ, as follows:

$$\sigma(x,\phi) = cos(\Gamma(x), \Gamma(\phi)), \tag{4}$$

Followed by a softmax function which takes an intermediate token representation as input and combines the output of each expert based on the gating weight $\Omega = (\omega_1, \ldots, \omega_N)$:

$$\Omega = \text{softmax}(\mathbf{s}_x), \tag{5}$$

Learning-based DVFS Controller. To further improve system effectiveness for LLM inference at the edge, a learning-based DVFS is developed to reduce per-generated token energy consumption while satisfying the real-time latency target at the layer-wise level. Among various reinforcement learning

(RL) methods [18, 62, 79, 85, 101], CLONE employs a simple two-layer MLP to efficiently learn policies within an episodic Markov decision process, ensuring minimal latency overhead. RL involves three core components:

State: Edge inference efficiency is heavily influenced by the processor intensity of co-running applications, denoted as S_{pro}. For runtime SLO constraints, inference efficiency is tightly coupled with frontend prefill time TTFT (T_{PRE}) and per-token decoding latency TPOT target (T_{DEC}).

Action: In RL, actions represent the adjustable control parameters of the system. For LLM inference at the edge, we define these actions as the selectable execution settings, specifically the energy-optimal supply voltage (V_{DD}) and the optimal running frequency (F_{req}) for each layer. Provided that the QoS constraints are met, it is feasible to lower the processor frequency, thereby conserving energy.

Reward: In RL, a reward models the optimization objective of the system. We encode energy optimization within the execution target as the reward, $\mathbb{R}_{energy}$, which is calculated using the power model [67, 70]:

$$\mathbb{R}_{energy} = \sum_{f} (P^{f}_{DEC} \times T_{DEC} + P^{f}_{PRE} \times T_{PRE}) \tag{6}$$

where P^{f}_{PRE} and P^{f}_{DEC} , representing the power consumption of the CPU/GPU at each frequency during the prefill and decoding states respectively, are determined through power measurements. These values are subsequently stored in a lookup table (LUT) within CLONE for efficient retrieval.

4.4 Hardware Accelerator System.

As illustrated in Figure 10 (b), acknowledging the limitations inherent in purely software-based approaches, CLONE introduces a scalable 28nm hardware accelerator specifically designed for edge-based LLM inference, effectively translating software optimizations into concrete performance improvements. This system integrates a LoRA Processing Unit (LPU) for dynamic adapter hot-swapping, enabling adaptive model performance optimization via dedicated data paths. Complementing this, a Special Function Unit (SFU) is equipped to perform continuous, fine-grained DVFS adjustments. The LPU executes a request-wise MoE router algorithm to efficiently manage stochastic requests, mitigating the limitations of general-purpose processors and enhancing computational throughput. Unlike conventional LoRA weight stored in on-chip SRAM, which either require frequent reloads from DRAM during wake-up cycles or keep SRAM active, wasting leakage power [73, 77, 90], we utilize an embedded non-volatile memory (eNVM) buffer to retain LoRA modules, eliminating reload overhead upon power-on. Simultaneously, the SFU employs a learning-based DVFS mechanism to adapt to runtime variability, ensuring rapid convergence to the desired voltage (V_{DD}) and frequency (F_{req}) settings. The SFU employs a lightweight predictor and DVFS model implemented as lookup tables (LUTs) for efficient hardware operation and action determination. The generated voltage (V_{DD}) and frequency (F_{req}) are realized using a fast-switching low-dropout (LDO) voltage regulator [9, 16] and an all-digital phase-locked loop (ADPLL) [2, 51], ensuring rapid runtime adjustments. Communication between the LPU and SFU is facilitated via a custom-built bi-directional streaming channel. An AXI splitter arbitrates the control flow of instructions and data bound for the LPU and SFU AXI-slave partitions.

5 EVALUATION

5.1 Experimental Setup

Infrastructure. In order to comprehensively evaluate the efficiency and effectiveness of CLONE, We perform our experiments on two heterogeneous edge devices: Orin Nano [104] and Jetson Orin NX [106]. Table 1 summarizes their specifications. To emulate heterogeneous deployment models, we utilize Llama-7B [132], Llama2-7B [133], Llama2-13B and Vicuna-7B [160] as baseline models. To simulate the heterogeneous performance requirements of multiple tasks at the edge, we employ the Flanv2 [135] dataset, which comprises 46 tasks across 10 domains, serving as the edge downstream dataset. Devices run stochastic web-search to simulate runtime variance. CodeCarbon [24] is utilized to measure runtime duration and the power consumption.

Baselines. To evaluate the effectiveness of CLONE, we compare CLONE with 7 representative approaches, including: ***A. Vanilla:*** (1) *Vanilla* assess the effectiveness of the original LLMs as the baseline criterion. ***B. Model compression:*** (2) *Random* randomly selects certain groups for pruning. (3) *LLM-Pruner* [30] removes connected structures of dependent multi-head attention from LLMs based on gradient information. (4) *ShortGPT* [94] defines the Block Influence metric to guide redundant subset removal. (5) *SliceGPT* [8] replaces each weight matrix in the network with a smaller, dense matrix, thereby reducing the network embedding dimension. ***C. Computation optimization:*** (6) *FlexGen* [118] retains all model parameters on CPU DRAM. During inference, the demand weights are loaded from the CPU to the GPU. ***D. Small size model:*** (7) OpenLLaMA-3B [35], an open reproduction of Llama with 3B parameters, trained on 1T tokens.

Evaluation Metrics. In order to comprehensively evaluate the efficiency and effectiveness of CLONE, we conduct evaluations from three perspectives. ***A.Generation ability.*** We measure the zero-shot perplexity (PPL) analysis on WikiText2 [95] and PTB [93], two of the most commonly used evaluation datasets, to evaluate the generation capabilities of the customized model. Lower PPL values signify stronger generation ability. ***B. General-purpose task solving ability.*** To demonstrate the world knowledge and problem-solving skills of the customized model, we utilize three benchmarks:

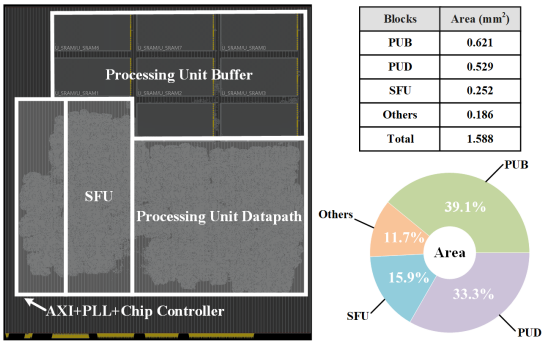


Figure 11: 28nm physical layout and area breakdown of the energy-optimal CLONE accelerator system.

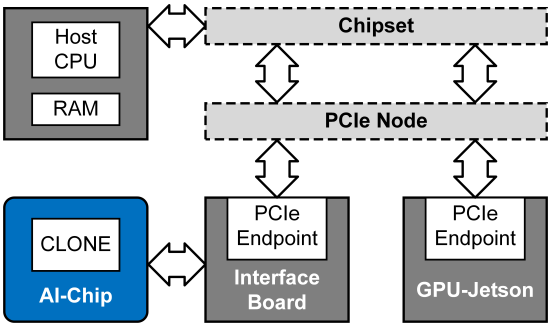


Figure 12: Test system. Integration of the CLONE accelerator into the Jetson platform via a PCIe interface to facilitate high-performance data exchange.

1) BBH [123], which includes 23 challenging tasks such as Q&A, natural language reasoning, and sentiment analysis. 2) MMLU [46], covering 57 tasks across diverse domains like mathematics, history, law, and ethics. 3) Commonsense reasoning tasks used in Llama paper [132], featuring BoolQ [22], PIQA [12], OBQA [100], ARC-c [23], ARC-e [23], WinoGrande [116], and HellaSwag [151]. ***C. System effectiveness.*** We evaluate the system effectiveness of CLONE from two perspectives: latency and energy consumption, which are defined as the duration and energy cost to infer the WikiText2 [95].

Hyperparameter and Reproducibility. For offline device-specific tailoring, we first execute classic approaches for 100 epochs to collect training data, and use 25× randomly shuffled client selection as data augmentation for tailoring training. Both Encoder and Decoder use a single-layer LSTM configuration, while the Predictor has a dual-layer feed-forward setup. Hidden state dimensions are 64 for Encoder and Decoder, and 200 for the Predictor, with all layers having an embedding size of 32. The hyperparameters include a batch size of 1024, a learning rate of 0.001, and $\eta = 0.8$. For optimization, we start with the top 25 model pruning ratio records. The LoRA rank r and the scaling hyperparameter α are set to 8 and 16, respectively, with 3 training rounds.

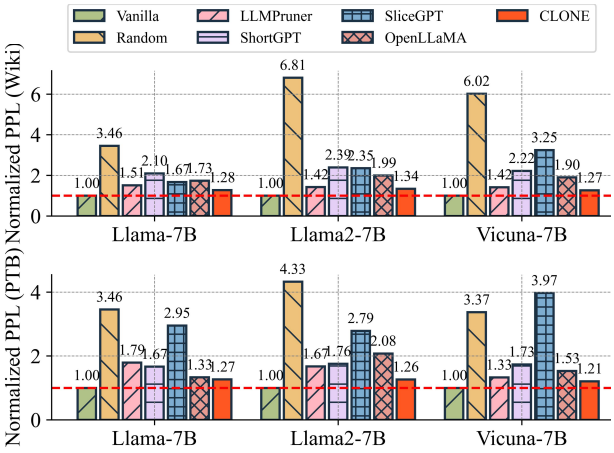


Figure 13: Generation ability comparison of various schemes on WikiText2 and PTB dataset across 3 representative LLMs.

5.2 Test System

Figure 11 presents the layout of the energy-optimal 28nm CLONE accelerator system. After place-and-route, the core footprint is only 1.588 mm^2; the figure also delineates module boundaries and reports the post-layout power- and area-breakdowns. Due to the high cost and time requirements of a full tape-out, the CLONE accelerator was validated through post-layout simulation. The physical layout was generated from Cadence Innovus synthesis results, following a complete design flow: behavioral modeling, RTL design, synthesis, P&R, and physical verification, ensuring simulation fidelity to actual fabrication. Figure 12 illustrates the integration of the CLONE accelerator into the Jetson platform via a PCIe interface to facilitate high-performance data exchange. The data flow initiates with the AI-Chip (CLONE), which interfaces with the Jetson GPU through a PCIe connection. An interface board bridges the PCIe endpoints of the CLONE accelerator and the Jetson GPU. This architecture is supported by the host CPU and RAM, where the CPU handles top-level data flow control, ensuring that requests are efficiently managed and routed. The CLONE is responsible for executing tasks based on user inputs, such as selecting the most efficient LoRA group and performing edge real-time DVFS control to optimize performance. By bypassing system software dependencies, this hardware-centric setup ensures seamless communication and data transfer between the CLONE and the Jetson device, achieving high levels of performance and energy efficiency.

5.3 Evaluation Results and Analysis

Generation Ability. To evaluate the customized model generation ability, we use the WikiText2 and PTB datasets to measure the zero-shot PPL (lower values correspond to stronger text generation capability) across different models on the

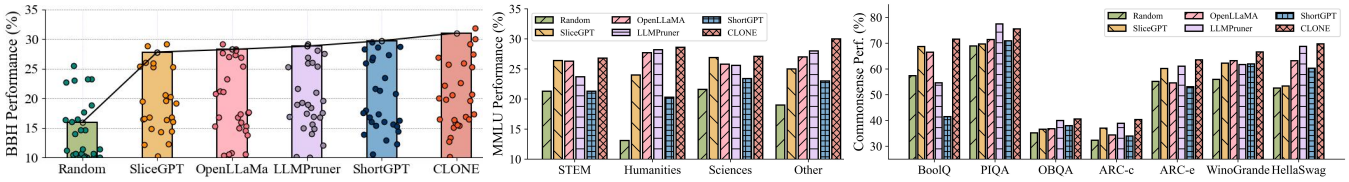
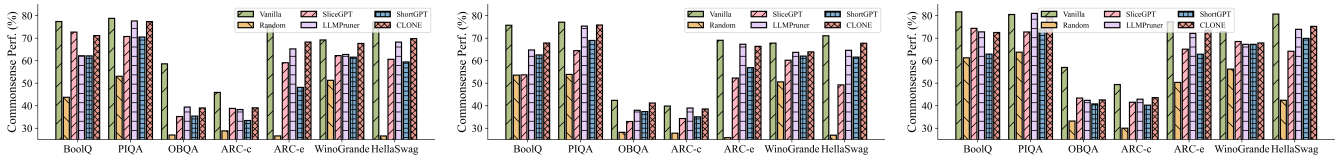


Figure 14: Downstream task performance of different customization approaches of Llama-7B on 3 representative benchmarks, including BBH (zero-shot), MMLU (3-shot), and Commonsense (zero-shot).



(a) Llama2-7B (b) Vicuna-7B (c) Llama2-13B

Figure 15: Zero-shot performance (commonsense) of Llama2-7B, Vicuna-7B and Llama2-13B.

Orin NX. Figure 13 shows the zero-shot PPL values that compare CLONE with the baselines. In order to effectively demonstrate the characteristics of different schemes, we normalize the measured PPL values to the vanilla model. Compared to baselines, CLONE preserves the most generation ability, achieving up to 5.1× generating capacity of *Random* on WikiText2 and up to 3.4× on PTB. This can be attributed to CLONE using PPL as a pruning metric to construct a continuous pruning selection space. It then utilizes gradient-based optimization to identify the optimal pruning configuration that minimally impacts generation ability.

Downstream Task Performance. Figure 14 illustrates the downstream task performance of CLONE across three representative benchmarks (BBH, MMLU, and Commonsense), covering 87 tasks. We observe that CLONE significantly outperforms other baselines across various domains and tasks. Specifically, for BBH, CLONE improves the average accuracy by 15.1% over Random and by 2.37% over others on average; for MMLU, it increases average accuracy by 6.0% over Random and by 2.96% over others on average; for commonsense tasks, CLONE enhances average accuracy by 10.1% over Random and by 6.1% over others on average. The principle as discussed in §3, CLONE implements task-aware tailoring for specific devices by considering end-user usage patterns and task specificity and relevance. It selects appropriate types and quantities of LoRA for fine-tuning. Moreover, we note that baseline performance varies across different downstream tasks. For instance, in BBH, ShortGPT is the sub-optimal model, whereas in Commonsense, LLMPruner is the sub-optimal. This variability stems from the heuristic and incomplete nature of baseline approaches. In contrast, CLONE utilizes a request-wise MoE-based router to effectively manage the properties of diverse real-world requests, resulting in superior LoRA module fusion. Overall, this experiment demonstrates CLONE's practical and robust ability to scale to complex workloads and applications.

Robustness and Scalability. Figure 15 details the zero-shot performance of the pruned model across a variety of downstream tasks, as well as its zero-shot perplexity on the WikiText2 and PTB datasets, utilizing diverse pruning configurations. Our observations confirm that CLONE consistently outperforms established baselines across all metrics.*1) Different types of models*: Comparing Llama2-7B and Vicuna-7B, the results indicate that CLONE exhibits a strong generalization capability, outperforming baselines by an average of 23.85%. *2) Different model sizes*: Evaluating Llama2-7B and Llama2-13B, it is evident from Figure 15 that CLONE maintains 91.13% of the performance of the unpruned (vanilla) models, underscoring its scalability. These results collectively affirm that CLONE is scalable to different LLMs in general, not limited to specific LLMs, so it is not a one-off "software-ASIC".

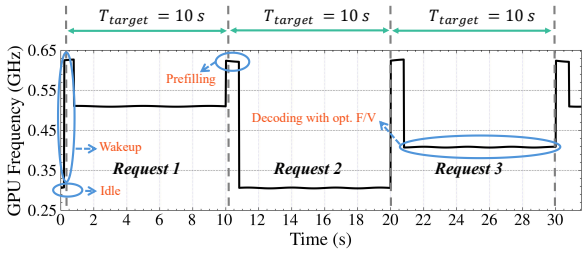


Figure 16: Simulations of LDO dynamic V_{DD} adjustments.

System Effectiveness. Then, we evaluate the system efficiency of CLONE from latency and energy consumption during the LLMs inference process on the WikiText2 dataset at the off-the-shelf devices Orin NX and Nano. As shown

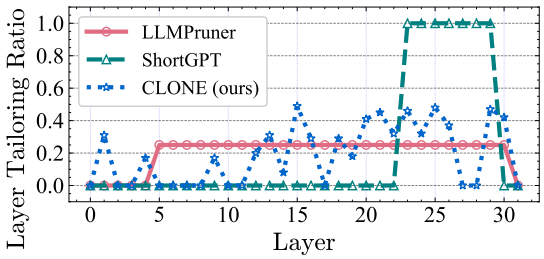


Figure 17: Comparison of the model pruning configuration.

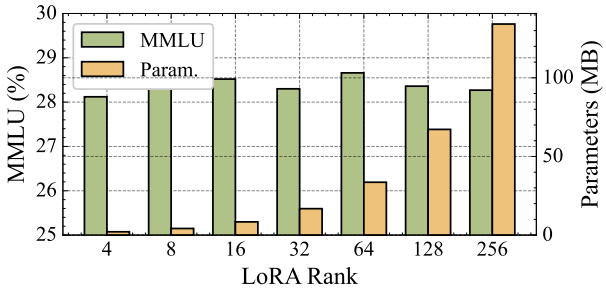


Figure 18: MMLU performance and parameter size across LoRA ranks.

in Table 3, we can see that CLONE effectively speeds up the inference process up to 11.92×, and achieves significant energy saving up to 7.36×. Figure 16 shows the spice-level simulation of the DVFS for runtime inference. There are two possible reasons 1) As shown in Equation 1, CLONE jointly considers the latency and energy budgets as metrics to generate optimal pruning configuration. 2) CLONE decouples prefill and decoding phases, applying request-level DVFS to boost effectiveness. In contrast, FlexGen trades high latency and energy for full-parameter LLMs, while methods like LLMPruner and ShortGPT focus on static, one-off compression without considering system-level optimization.

Table 3: System efficiency (latency and energy) comparison of various schemes to inference WikiText2 dataset.

Method	Energy (Wh)		Latency (s)	
	Jetson NX	Jetson Nano	Jetson NX	Jetson Nano
Random	7.27	8.26	842.40	1145.07
SliceGPT	5.47	7.54	661.65	929.39
OpenLLaMA	5.67	7.70	506.73	662.73
LLMPruner	6.01	6.91	622.92	1023.51
ShortGPT	5.67	8.56	555.14	698.02
FlexGen	21.12	26.04	3166.27	4674.42
CLONE^{-HW}	4.81	5.56	462.72	552.18
CLONE	**3.46**	**3.54**	**322.76**	**392.15**

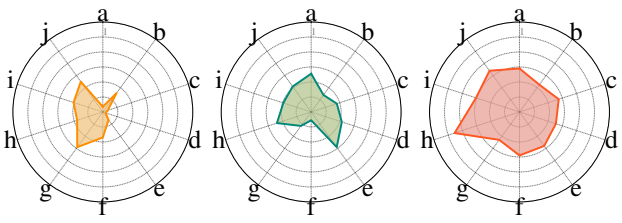
Study of the Generative Pruning Configuration. To examine how CLONE adaptively generates the optimal model pruning configuration, we compare the generative pruning approach to the layer-wise method used by ShortGPT and the uniform direct average method employed by LLMPruner on Llama-7B to illustrate the strategic disparities in their decision-making processes. As presented in Figure 17, LLMPruner remains static from the 5th to 30th layer. In contrast, ShortGPT uses only binary values of 0 or 1. We note that the layer-level pruning configuration determined by CLONE dynamically varies across layers. This adaptability reflects a more nuanced decision-making process, where the contributions of individual layers are recognized as heterogeneous, as discussed in *Motivation 1*. This indicates that CLONE's generative decision-making is more adaptive and tailored to the specific contributions of each layer.

Effectiveness of the Hardware Accelerator. The accelerator boosts efficiency by exploiting dedicated hardware resources; it does not itself alter generation quality or task-specific accuracy. Disabling it forces the workload onto general-purpose compute units, eliminating these hardware-level gains. As Table 3 shows, energy rises from 3.46Wh to 4.81Wh and end-to-end latency from 322.76s to 462.72s without the accelerator CLONE^{-HW}. Even in this regime, our offline, metric-guided generative pruning (Eq. 1) still surpasses all baselines, underscoring its robustness.

Effectiveness of Request-wise MoE Router. Figure 19 illustrates the performance of different LoRA adapter fusion methods during runtime inference for dynamic end-user requests. *w/o MoE* indicates direct averaging of all LoRA modules for all requests, which typically results in suboptimal performance due to the fact that not all stochastic multi-tasks benefit each other. *MoE (Top-1)* means based on the request to select the most similar LoRA for LLM inference. However, from Figure 19, we can see that *w/o MoE* Simply combining LoRAs results in worse performance because not all stochastic multi-tasks are beneficial to the others. Though *MoE (Top-1)* signifies that the most similar LoRA is selected based on the request for LLM inference. Although this approach enhances overall performance, it can still be sub-optimal in practice, as even a single request may involve multiple tasks, and merely considering task differences is insufficient. This observation highlights the critical importance of both the request-wise MoE router in maintaining CLONE performance within diverse data and task demands.

Impact of the Rank of LoRA Adapters. The rank r controls the number of trainable parameters, where larger values of r provide greater adapter capacity at the expense of increased training overhead. As illustrated in Figure 18, performance initially improves with higher ranks but eventually saturates, offering diminishing returns despite the rapidly growing parameter size.

Overhead Analysis. Offline tailoring utilizes a single-layer LSTM encoder-decoder and a dual-layer feed-forward network, adding minimal overhead and enhancing accuracy without affecting online inference. During online inference, the soft MoE router introduces no additional trainable parameters and performs a single probabilistic routing computation.



(a) w/o MoE (b) MoE (Top-1) (c) CLONE

Figure 19: Performance magnitudes of w/o MoE, MoE (Top-1) and CLONE. a-j represent different downstream tasks.

Implemented as a two-layer MLP with under 1K parameters, the DVFS controller imposes negligible overhead relative to billion-parameter LLMs. Both modules first execute concurrently (< 10ms) with the prefill stage (> 100 ms; see Figure 2), then the DVFS decision for token $t+1$ is generated while token t is being decoded, keeping the controller off the critical path. Meanwhile, the incremental energy cost of CLONE is on the milliwatt-hour scale, rendering it negligible compared with the watt-hour-level consumption of full-model inference.

6 Related Work

LLMs are highly memory-, compute-, and energy-intensive [5, 19, 96, 109, 132, 133], driving most "billion-parameter" inference to the cloud [6, 34, 81, 122, 124, 145, 149]. However, as edge devices become more powerful, there is increasing interest in executing LLM inference on the edge [4, 6, 34, 57, 74, 81, 107, 111, 122, 145, 149], which enhances data privacy and enables real-time service delivery. To address edge resource constraints, techniques such as model architecture search [54, 60, 88], quantization [26, 55, 57, 91, 143], pruning [27, 32, 91, 103], and knowledge distillation [41, 71] have been proposed. However, most approaches focus solely on model-level optimization, neglecting system-level trade-offs like storage and weight efficiency. Advances in LLM compilers and software stacks [56, 99, 113, 128] have enabled integration with co-processors and near-sensor processing [4, 43, 57, 74, 107, 111, 118, 122, 148, 155–158], but these often add computational and communication overhead, reducing edge device longevity and hindering concurrent application execution. Existing model customization methods [82, 126, 127, 153, 154] cannot directly address task-agnostic LLMs, where generalization is crucial. The unique characteristics of LLM decoder layers and stochastic outputs present untapped opportunities for hardware optimization. For system-level optimization, DVFS [10, 47, 48, 65, 112] has been widely used to dynamically adjust processor voltage and frequency. However, most DVFS strategies are designed for discriminative models like CNNs and RNNs, treating networks as black boxes. Generative LLMs, with their auto-regressive inference and stochastic prompt variability, remain underexplored in this context.

Edge–cloud collaboration offers a pragmatic middle ground between fully local and fully cloud-based inference. In practice, the edge should remain the first line of execution for latency-critical or privacy-sensitive workloads, running compact SLMs that fit the device's power and memory envelope. When a request exceeds local capacity—e.g., requires deeper reasoning, broader context, or larger knowledge—CLONE can transparently escalate the call to a cloud-resident LLM. This selective offloading preserves real-time responsiveness, keeps private data on-device whenever possible, and amortizes bandwidth and compute costs by invoking the cloud only for the fraction of tasks that truly need it.

7 Conclusion

CLONE integrates offline generative pruning and online latency-aware optimization to enhance edge deployment of LLMs. Offline, it dynamically generates device-specific pruning configurations via a generative framework. Online, CLONE employs a layer-wise DVFS strategy to optimize energy efficiency within latency constraints, using an MoE router with multiple LoRA adapters to support diverse tasks. Supported by a dedicated 28nm hardware accelerator, including specialized units for rapid LoRA adapter switching and fine-grained DVFS control, benchmarks confirm that CLONE significantly improves inference speed and energy efficiency while maintaining robust task performance, ideal for latency-sensitive, energy-constrained edge scenarios.

Acknowledgment

We sincerely thank the anonymous ATC'25 reviewers, and our shepherd, Dr. Joel Wolfrath, for their insightful suggestions. This work is supported in part by the Science and Technology Development Fund of Macau (0107/2024/RIA2), Joint Science and Technology Research Project with Hong Kong and Macau in Key Areas of Nansha District's Science and Technology Plan (EF2024-00180-IOTSC) and the Multi-Year Research Grant of University of Macau (MYRG-GRG2023-00211-IOTSC-UMDF, MYRG-GRG2024-00180-IOTSC). Please ask Dr. Li Li (llili@um.edu.mo) for correspondence.

References

[1] Rewind AI. Rewind: A better way to search your digital life, 2024.

[2] Tutu Ajayi, Sumanth Kamineni, Yaswanth K Cherivirala, Morteza Fayazi, Kyumin Kwon, Mehdi Saligane, Shourya Gupta, Chien-Hen Chen, Dennis Sylvester,

David Blaauw, et al. An open-source framework for autonomous soc design with analog block generation. In *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*, pages 141–146. IEEE, 2020.

[3] Jay Alammar. The illustrated gpt-2 (visualizing transformer language models). *Jalammar. github. io. https://jalammar. github. io/illustrated-gpt2*, 2019.

[4] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C. Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. LLM in a flash: Efficient large language model inference with limited memory. *CoRR*, abs/2312.11514, 2023.

[5] Anthropic. Claude: A family of ai models, 2024.

[6] Apple. Apple intelligence. https://www.apple.com/apple-intelligence/, 2024.

[7] Apple. Apple siri: Virtual assistant, 2024.

[8] Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *CoRR*, abs/2401.15024, 2024.

[9] Suyoung Bang, Wootaek Lim, Charles Augustine, Andres Malavasi, Muhammad Khellah, James Tschanz, and Vivek De. 25.1 a fully synthesizable distributed and scalable all-digital ldo in 10nm cmos. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 380–382. IEEE, 2020.

[10] Soroush Bateni and Cong Liu. {NeuOS}: A {Latency-Predictable}{Multi-Dimensional} optimization framework for {DNN-driven} autonomous systems. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 371–385, 2020.

[11] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In Madeleine Clare Elish, William Isaac, and Richard S. Zemel, editors, *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, pages 610–623. ACM, 2021.

[12] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020.

[13] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

[14] Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, Jennifer C Lai, and Robert L Mercer. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1):31–40, 1992.

[15] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[16] Chaitanya K Chava and José Silva-Martínez. A frequency compensation scheme for ldo voltage regulators. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(6):1041–1050, 2004.

[17] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multilingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

[18] Yonghun Choi, Seonghoon Park, and Hojung Cha. Optimizing energy efficiency of browsers in energy-aware scheduling-enabled mobile devices. In Stephen A. Brewster, Geraldine Fitzpatrick, Anna L. Cox, and Vassilis Kostakos, editors, *The 25th Annual International Conference on Mobile Computing and Networking, MobiCom 2019, Los Cabos, Mexico, October 21-25, 2019*, pages 48:1–48:16. ACM, 2019.

[19] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113, 2023.

[20] Jae-Won Chung, Jiachen Liu, Zhiyu Wu, Yuxuan Xia, and Mosharaf Chowdhury. ML.ENERGY leaderboard. https://ml.energy/leaderboard, 2023.

[21] Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake A. Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew J. Johnson, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Marc'Aurelio Ranzato, Jack W. Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 4057–4086. PMLR, 2022.

[22] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics, 2019.

[23] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[24] codecarbon. Track and reduce co2 emissions from your computing. https://codecarbon.io/, 2023.

[25] Dell. Llama 2: Inferencing on a single gpu. https://infohub.delltechnologies.com/zh-cn/t/llama-2-inferencing-on-a-single-gpu/, 2023.

[26] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[27] Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. Prunerzero: Evolving symbolic pruning metric from scratch for large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[28] Begum Egilmez, Matthew Schuchhardt, Gokhan Memik, Raid Ayoub, Niranjan Soundararajan, and Michael Kishinevsky. User-aware frame rate management in android smartphones. *ACM Trans. Embed. Comput. Syst.*, 16(5s):131:1–131:17, 2017.

[29] Yasuhiro Endo, Zheng Wang, J. Bradley Chen, and Margo I. Seltzer. Using latency to evaluate interactive system performance. In Karin Petersen and Willy Zwaenepoel, editors, *Proceedings of the Second USENIX Symposium on Operating Systems Design and Implementation (OSDI), Seattle, Washington, USA, October 28-31, 1996*, pages 185–199. ACM, 1996.

[30] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Van-*

couver, BC, Canada, June 17-24, 2023, pages 16091–16101. IEEE, 2023.

[31] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

[32] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR, 2023.

[33] Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In Thang Luong, Alexandra Birch, Graham Neubig, and Andrew M. Finch, editors, *Proceedings of the First Workshop on Neural Machine Translation, NMT@ACL 2017, Vancouver, Canada, August 4, 2017*, pages 56–60. Association for Computational Linguistics, 2017.

[34] Daocheng Fu, Xin Li, Licheng Wen, Min Dou, Pinlong Cai, Botian Shi, and Yu Qiao. Drive like a human: Rethinking autonomous driving with large language models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 910–919, 2024.

[35] Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama. https://github.com/openlm-research/open_llama, May 2023.

[36] Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. AI and memory wall. *IEEE Micro*, 44(3):33–39, 2024.

[37] github. GitHub Copilot: Your AI pair programmer. https://github.com/features/copilot.

[38] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[39] Google. Google assistant, 2024.

[40] Google. Ml kit smart reply, 2024.

[41] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[42] Peizhen Guo, Bo Hu, and Wenjun Hu. Mistify: Automating DNN model porting for on-device inference at the edge. In James Mickens and Renata Teixeira, editors, *18th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2021, April 12-14, 2021*, pages 705–719. USENIX Association, 2021.

[43] Pengyu He, Yuanzhe Zhao, Heng Xie, Yang Wang, Shouyi Yin, Li Li, Yan Zhu, Rui P Martins, Chi-Hang Chan, and Minglei Zhang. A reconfigurable floating-point compute-in-memory with analog exponent preprocesses. *IEEE Solid-State Circuits Letters*, 2024.

[44] Pengyu He, Yuanzhe Zhao, Heng Xie, Yang Wang, Shouyi Yin, Li Li, Yan Zhu, Rui Paulo Martins, Chi-Hang Chan, and Minglei Zhang. A 28nm 314.6 tflops/w reconfigurable floating-point analog compute-in-memory macro with exponent approximation and two-stage sharing td-adc. In *2024 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–2. IEEE, 2024.

[45] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1398–1406. IEEE Computer Society, 2017.

[46] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[47] Sebastian Herbert and Diana Marculescu. Variation-aware dynamic voltage/frequency scaling. In *15th International Conference on High-Performance Computer Architecture (HPCA-15 2009), 14-18 February 2009, Raleigh, North Carolina, USA*, pages 301–312. IEEE Computer Society, 2009.

[48] Robert Hesse and Natalie D. Enright Jerger. Improving DVFS in nocs with coherence prediction. In André Ivanov, Diana Marculescu, Partha Pratim Pande, José Flich, and Karthik Pattabiraman, editors, *Proceedings of the 9th International Symposium on Networks-on-Chip, NOCS 2015, Vancouver, BC, Canada, September 28-30, 2015*, pages 24:1–24:8. ACM, 2015.

[49] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[50] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[51] Terng-Yin Hsu, Bai-Jue Shieh, and Chen-Yi Lee. An all-digital phase-locked loop (adpll)-based clock recovery circuit. *IEEE Journal of Solid-State Circuits*, 34(8):1063–1073, 1999.

[52] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv e-prints*, page arXiv:2106.09685, June 2021.

[53] Yi Huang, Zhiyu Chen, Dai Li, and Kaiyuan Yang. CAMA: energy and memory efficient automata processing in content-addressable memories. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2022, Seoul, South Korea, April 2-6, 2022*, pages 25–37. IEEE, 2022.

[54] Yingbing Huang, Lily Jiaxin Wan, Hanchen Ye, Manvi Jha, Jinghua Wang, Yuhong Li, Xiaofan Zhang, and Deming Chen. New solutions on llm acceleration, optimization, and application. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–4, 2024.

[55] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30, 2018.

[56] HuggingFace Team. Transformers documentation, 2024.

[57] Mohamed Assem Ibrahim, Shaizeen Aga, Ada Li, Suchita Pati, and Mahzabeen Islam. Just-in-time quantization with processing-in-memory for efficient ml training, 2023.

[58] Mem Inc. Mem: Your ai-powered assistant, 2024.

[59] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87, 1991.

[60] Ganesh Jawahar, Muhammad Abdul-Mageed, Laks VS Lakshmanan, and Dujian Ding. Llm performance predictors are good initializers for architecture search. *arXiv preprint arXiv:2310.16712*, 2023.

[61] Kabilankb. Robot control using llama: Bridging ai and robotics. Medium, 2024. Oct 13, 2024.

[62] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[63] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[64] Byeongho Kim, Sanghoon Cha, Sangsoo Park, Jieun Lee, Sukhan Lee, Shinhaeng Kang, Jinin So, Kyungsoo Kim, Jin Jung, Jong-Geon Lee, Sunjung Lee, Yoonah Paik, Hyeonsu Kim, Jin-Seong Kim, Won-Jo Lee, Yuhwan Ro, Yeongon Cho, Jin Hyun Kim, Joon-Ho Song, Jaehoon Yu, Seungwon Lee, Jeonghyeon Cho, and Kyomin Sohn. The breakthrough memory solutions for improved performance on LLM inference. *IEEE Micro*, 44(3):40–48, 2024.

[65] Wonyoung Kim, Meeta Sharma Gupta, Gu-Yeon Wei, and David M. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *14th International Conference on High-Performance Computer Architecture (HPCA-14 2008), 16-20 February 2008, Salt Lake City, UT, USA*, pages 123–134. IEEE Computer Society, 2008.

[66] Young Geun Kim, Minyong Kim, and Sung Woo Chung. Enhancing energy efficiency of multimedia applications in heterogeneous mobile multi-core processors. *IEEE Trans. Computers*, 66(11):1878–1889, 2017.

[67] Young Geun Kim, Minyong Kim, Jae Min Kim, Minyoung Sung, and Sung Woo Chung. A novel gpu power model for accurate smartphone power breakdown. *ETRI journal*, 37(1):157–164, 2015.

[68] Young Geun Kim, Joonho Kong, and Sung Woo Chung. A survey on recent os-level energy management techniques for mobile processing units. *IEEE Trans. Parallel Distributed Syst.*, 29(10):2388–2401, 2018.

[69] Young Geun Kim and Carole-Jean Wu. Autoscale: Energy efficiency optimization for stochastic edge inference using reinforcement learning. In *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, October 17-21, 2020*, pages 1082–1096. IEEE, 2020.

[70] Youngsok Kim, Joonsung Kim, Dongju Chae, Daehyun Kim, and Jangwoo Kim. μlayer: Low latency on-device inference using cooperative single-layer acceleration and processor-friendly quantization. In *Proceedings of the Fourteenth EuroSys Conference 2019*, pages 1–15, 2019.

[71] Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models. In *Forty-first International*

Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net, 2024.

[72] Brett Koonce and Brett Koonce. Resnet 50. *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*, pages 63–72, 2021.

[73] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Paul Whatmough, Aleksandra Faust, Sabrina Neuman, Gu-Yeon Wei, David Brooks, and Vijay Janapa Reddi. Automatic domain-specific soc design for autonomous unmanned aerial vehicles. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 300–317. IEEE, 2022.

[74] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace, editors, *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM, 2023.

[75] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace, editors, *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM, 2023.

[76] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[77] Haitong Li, Mudit Bhargava, Paul N Whatmough, and H-S Philip Wong. On-chip memory technology design space explorations for mobile deep neural network accelerators. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[78] Li Li, Xiaorui Wang, and Feng Qin. Energydx: Diagnosing energy anomaly in mobile apps by identifying the manifestation point. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 256–266. IEEE, 2020.

[79] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[80] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models, 2025.

[81] Haicheng Liao, Zhenning Li, Huanming Shen, Wenxuan Zeng, Dongping Liao, Guofa Li, Shengbo Eben Li, and Chengzhong Xu. BAT: behavior-aware human-like trajectory prediction for autonomous driving. *CoRR*, abs/2312.06371, 2023.

[82] Edgar Liberis and Nicholas D Lane. Differentiable neural network pruning to enable smart applications on microcontrollers. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(4):1–19, 2023.

[83] Chaofan Lin, Zhenhua Han, Chengruidong Zhang, Yuqing Yang, Fan Yang, Chen Chen, and Lili Qiu. Parrot: Efficient serving of llm-based applications with semantic variable. In Ada Gavrilovska and Douglas B. Terry, editors, *18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024*, pages 929–945. USENIX Association, 2024.

[84] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. AWQ: activation-aware weight quantization for LLM compression and acceleration. *CoRR*, abs/2306.00978, 2023.

[85] Xue Lin, Yanzhi Wang, and Massoud Pedram. A reinforcement learning-based power management framework for green computing data centers. In *2016 IEEE International Conference on Cloud Engineering, IC2E 2016, Berlin, Germany, April 4-8, 2016*, pages 135–138. IEEE Computer Society, 2016.

[86] Jiahao Liu, Yuanzhe Zhao, Yan Zhu, Chi-Hang Chan, and Rui Paulo Martins. A weak puf-assisted strong PUF with inherent immunity to modeling attacks and ultra-low BER. *IEEE Trans. Circuits Syst. I Regul. Pap.*, 69(12):4898–4907, 2022.

[87] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35, 2023.

[88] Shu Liu, Asim Biswal, Audrey Cheng, Xiangxi Mo, Shiyi Cao, Joseph E Gonzalez, Ion Stoica, and Matei Zaharia. Optimizing llm queries in relational workloads. *arXiv preprint arXiv:2403.05821*, 2024.

[89] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning

can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602, 2021.

[90] Zhi-Gang Liu, Paul N Whatmough, Yuhao Zhu, and Matthew Mattina. S2ta: Exploiting structured sparsity for energy-efficient mobile cnn acceleration. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 573–586. IEEE, 2022.

[91] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. Deja vu: Contextual sparsity for efficient llms at inference time. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 22137–22176. PMLR, 2023.

[92] Daniel Lo, Taejoon Song, and G Edward Suh. Prediction-guided performance-energy trade-off for interactive applications. In *Proceedings of the 48th International Symposium on Microarchitecture*, pages 508–520, 2015.

[93] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330, 1993.

[94] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *CoRR*, abs/2403.03853, 2024.

[95] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[96] Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295, 2024.

[97] microsoft. Your Everyday AI Companion | Microsoft Bing. https://www.bing.com/new.

[98] Microsoft. Azure cognitive services - text analytics: Smart reply, 2024.

[99] Microsoft DeepSpeed Team. Deepspeed: Advancing the science of ai through efficient training of large models, 2024.

[100] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391. Association for Computational Linguistics, 2018.

[101] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[102] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11264–11272. Computer Vision Foundation / IEEE, 2019.

[103] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[104] nano. Nvidia jetson nano, 2023.

[105] Rajiv Nishtala, Vinicius Petrucci, Paul M. Carpenter, and Magnus Själander. Twig: Multi-agent task management for colocated latency-critical cloud services. In *IEEE International Symposium on High Performance Computer Architecture, HPCA 2020, San Diego, CA, USA, February 22-26, 2020*, pages 167–179. IEEE, 2020.

[106] NVIDIA. Nvidia jetson orin - autonomous machines - nvidia. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/, 2024.

[107] Hyungjun Oh, Kihong Kim, Jaemin Kim, Sungkyun Kim, Junyeol Lee, Du-seong Chang, and Jiwon Seo. Exegpt: Constraint-aware resource scheduling for LLM inference. In Rajiv Gupta, Nael B. Abu-Ghazaleh, Madan Musuvathi, and Dan Tsafrir, editors, *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2024, La Jolla, CA, USA, 27 April 2024- 1 May 2024*, pages 369–384. ACM, 2024.

[108] OpenAI. Chatgpt, 2022.

[109] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[110] Otter.ai. Otter: Transcription and note-taking with ai, 2024.

[111] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Aashaka Shah, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. *CoRR*, abs/2311.18677, 2023.

[112] Leonardo Piga, Iyswarya Narayanan, Aditya Sundarrajan, Matt Skach, Qingyuan Deng, Biswadip Maity, Manoj Chakkaravarthy, Alison Huang, Abhishek Dhanotia, and Parth Malani. Expanding datacenter capacity with DVFS boosting: A safe and scalable deployment experience. In Rajiv Gupta, Nael B. Abu-Ghazaleh, Madan Musuvathi, and Dan Tsafrir, editors, *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1, ASPLOS 2024, La Jolla, CA, USA, 27 April 2024- 1 May 2024*, pages 150–165. ACM, 2024.

[113] PyTorch Contributors. Pytorch: An open source machine learning framework, 2024.

[114] Qualcomm Technologies, Inc. Llama-v2-7B-Chat Quantized Model. https://aihub.qualcomm.com/models/llama_v2_7b_chat_quantized, 2024. Qualcomm AI Hub, 2024.

[115] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[116] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

[117] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[118] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.

[119] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 11523–11530, London, UK, 2023. IEEE.

[120] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Kumar Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Schärli, Aakanksha Chowdhery, Philip Andrew Mansfield, Blaise Agüera y Arcas, Dale R. Webster, Gregory S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle K. Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge. *CoRR*, abs/2212.13138, 2022.

[121] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.

[122] Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 590–606, 2024.

[123] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai,

Andrew La, Andrew K. Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakas, and et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *CoRR*, abs/2206.04615, 2022.

[124] stanford. Ai index report. https://aiindex.stanford.edu/report/, 2024. Accessed: 2024-07.

[125] Statista Inc. Mobile ram usage worldwide from 1q-19 to 1q-21 (in gb per device). www.statista.com/statistics/1057679/mobile-ram-usage-worldwide-by-average-size-per-device/, 2021.

[126] Kahou Tam, Chunlin Tian, Li Li, Haikai Zhao, and ChengZhong Xu. Fedhybrid: Breaking the memory wall of federated learning via hybrid tensor management. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, pages 394–408, 2024.

[127] Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul Whatmough, Alexander M Rush, David Brooks, et al. Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 830–844, 2021.

[128] TensorFlow Contributors. Tensorflow: An open source machine learning framework for everyone, 2024.

[129] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

[130] Chunlin Tian, Li Li, Zhan Shi, Jun Wang, and ChengZhong Xu. Harmony: Heterogeneity-aware hierarchical management for federated learning system. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 631–645. IEEE, 2022.

[131] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng-Zhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:9565–9584, 2024.

[132] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[133] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[134] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[135] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652, 2021.

[136] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022.

[137] Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large

language models by equivalent and effective shifting and scaling. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 1648–1665. Association for Computational Linguistics, 2023.

[138] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (MobiCom '24)*, Washington D.C., USA, 2024. Association for Computing Machinery.

[139] Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, et al. Machine learning at facebook: Understanding inference at the edge. In *2019 IEEE international symposium on high performance computer architecture (HPCA)*, pages 331–344. IEEE, 2019.

[140] Yebo Wu, Li Li, Chunlin Tian, Tao Chang, Chi Lin, Cong Wang, and Cheng-Zhong Xu. Heterogeneity-aware memory efficient federated learning via progressive layer freezing. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2024.

[141] Yebo Wu, Chunlin Tian, Jingguang Li, He Sun, Kahou Tam, Li Li, and Chengzhong Xu. A survey on federated fine-tuning of large language models. *arXiv preprint arXiv:2503.12016*, 2025.

[142] Yebo Wu, Chunlin Tian, Jingguang Li, He Sun, Kahou Tam, Li Li, and Chengzhong Xu. A survey on federated fine-tuning of large language models. *CoRR*, abs/2503.12016, 2025.

[143] Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR, 2023.

[144] Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. Llm4drive: A survey of large language models for autonomous driving. *arXiv preprint arXiv:2311.01043*, 2023.

[145] Xinyu Ye, Zhe Wang, Haihao Shen, Yu Luo, and Hanwen Chang. Creating large language models on your laptop. Medium, 2023.

[146] Rongjie Yi, Xiang Li, Weikai Xie, Zhenyan Lu, Chenghua Wang, Ao Zhou, Shangguang Wang, Xiwen Zhang, and Mengwei Xu. Phonelm: An efficient and capable small language model family through principled pre-training. *arXiv preprint arXiv:2411.05046*, 2024.

[147] Wangsong Yin, Rongjie Yi, Daliang Xu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. Elms: Elasticized large language models on mobile devices. *arXiv preprint arXiv:2409.09071*, 2024.

[148] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In Marcos K. Aguilera and Hakim Weatherspoon, editors, *16th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2022, Carlsbad, CA, USA, July 11-13, 2022*, pages 521–538. USENIX Association, 2022.

[149] Jinliang Yuan, Chen Yang, Dongqi Cai, Shihe Wang, Xin Yuan, Zeling Zhang, Xiang Li, Dingge Zhang, Hanzi Mei, Xianqing Jia, et al. Mobile foundation model as firmware. *arXiv preprint arXiv:2308.14363*, 2023.

[150] Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermis, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *CoRR*, abs/2309.05444, 2023.

[151] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics, 2019.

[152] Shichen Zhan, Yebo Wu, Chunlin Tian, Yan Zhao, and Li Li. Heterogeneity-aware coordination for federated learning via stitching pre-trained blocks. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2024.

[153] Yifan Zhao, Hashim Sharif, Vikram Adve, and Sasa Misailovic. Felix: Optimizing tensor programs with gradient descent. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 367–381, 2024.

[154] Yifan Zhao, Hashim Sharif, Peter Pao-Huang, Vatsin Shah, Arun Narenthiran Sivakumar, Mateus Valverde Gasparino, Abdulrahman Mahmoud, Nathan Zhao, Sarita Adve, Girish Chowdhary, et al. Approxcaliper: A programmable framework for application-aware neural network optimization. *Proceedings of Machine Learning and Systems*, 5, 2023.

[155] Yuanzhe Zhao, Pengyu He, Yan Zhu, Rui P Martins, Chi-Hang Chan, and Minglei Zhang. A 28-nm 3.32-nj/frame compute-in-memory cnn processor with layer fusion for always-on applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2025.

[156] Yuanzhe Zhao, Yang Wang, Yuheng Wang, Heng Xie, Yan Zhu, RP Martins, Chi-Hang Chan, Shouyi Yin, and Minglei Zhang. A 28nm value-wise hybrid-domain compute-in-memory macro with heterogeneous memory fabric and asynchronous sparsity manager. In *2025 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–3. IEEE, 2025.

[157] Yuanzhe Zhao, Yuheng Wang, Zijian Wang, Yan Zhu, RP Martins, Chi-Hang Chan, and Minglei Zhang. A reconfigurable 0.69-1.02 nj/classification biomedical ai processor for intelligent health monitoring devices. In *2025 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–3. IEEE, 2025.

[158] Yuanzhe Zhao, Heng Xie, Zijian Wang, Chunlin Tian, Li Li, Yan Zhu, RP Martins, Chi-Hang Chan, and Minglei Zhang. A one-shot floating-point compute-in-memory macro featuring pvt robustness and mismatch tolerance for edge llms. In *2025 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–3. IEEE, 2025.

[159] Yuanzhe Zhao, Minglei Zhang, Pengyu He, Yan Zhu, Chi-Hang Chan, and Rui Paulo Martins. A double-mode sparse compute-in-memory macro with reconfigurable single and dual layer computation. In *IEEE Custom Integrated Circuits Conference, CICC 2023, San Antonio, TX, USA, April 23-26, 2023*, pages 1–2. IEEE, 2023.

[160] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[161] Yuhao Zhu, Matthew Halpern, and Vijay Janapa Reddi. Event-based scheduling for energy-efficient qos (eqos) in mobile web applications. In *21st IEEE International Symposium on High Performance Computer Architecture, HPCA 2015, Burlingame, CA, USA, February 7-11, 2015*, pages 137–149. IEEE Computer Society, 2015.

[162] Ömer Bilgin Bilgili. Ai robotics case - controlling robots with llms (large language models). acrome, 2024. Acrome Robotics Blog, October 23, 2024.