M2R2: EFFICIENT TRANSFORMERS WITH MIXTURE OF MULTI-RATE RESIDUALS

Nikhil Bhendawade,* **Mahyar Najibi, Irina Belousova, Devang Naik**
Apple
{nbhendawade, najibi, ibelousova, naik.d}@apple.com

ABSTRACT

Residual transformations play a crucial role in enhancing the representational depth and expressive power of large language models (LLMs). However, static residual transformation during auto-regressive generation leads to a sub-optimal balance between inference efficiency and generation fidelity. Existing methods such as Early Exiting, Mixture of Depths, Skip Decoding focus on token traversal distance across layers to enable dynamic transformation but overlook the velocity of residual evolution, leading to suboptimal inference efficiency. We introduce *Mixture of Multi-rate Residuals* (M2R2), a framework that dynamically modulates residual velocities to ensure early alignment of intermediate representations. M2R2 shows improvements across *dynamic computing*, *speculative decoding*, and *Mixture-of-Experts* (MoE) architectures. In dynamic computing settings, M2R2 outperforms state-of-the-art distance-based strategies, achieving a superior trade-off between generation metrics and speedup. In self-speculative decoding, M2R2 achieves up to 2.8× speedup on MT-Bench and, in MoE models, up to 2.9× speedup with ahead-of-time expert loading. This positions M2R2 as an effective strategy for mobile resource-constrained deployment.

1 INTRODUCTION

Large Language Models (LLMs) excel in complex NLP tasks by capturing long-range dependencies and modeling intricate linguistic patterns through residual transformations, enhancing representations across layers (Brown et al., 2020; Radford et al., 2019; Vaswani et al., 2017). However, the static nature of residual transformations fails to account for the inherent variability in token complexity, leading to inefficiencies in dynamic compute scenarios (Shen et al., 2021; Garncarek & Snaider, 2021). Approaches such as Early Exiting (Schuster et al., 2022; Varshney et al., 2024; Chen et al., 2023b), Skip Decoding (Del Corro et al., 2023), and Mixture-of-Depth (Raposo et al., 2024) address this by modulating the depth of residual transformations based on token complexity, however these strategies focus solely on token traversal distance, neglecting the critical dimension of residual transformations: residual velocity, the rate at which token representations evolve.

We propose a novel framework, *Mixture of Multi-rate Residuals* (M2R2), which dynamically adjusts the velocity of residual transformations to optimize early alignment of token representations. M2R2 enhances efficiency in setups such as dynamic computing, speculative decoding, and Mixture-of-Experts (MoE) Ahead-of-Time (AoT) expert loading. In speculative decoding (Leviathan et al., 2023; Chen et al., 2023a), speculative draft sampled from accelerated residuals is validated in parallel against base residual streams. In MoE models (Shazeer et al., 2017; Fedus et al., 2022), M2R2 facilitates early expert speculation through accelerated residuals, enabling overlapping computation and memory transfer, reducing latency in expert switching in resource constrained setups.

This paper makes the following contributions: (1) We introduce *Mixture of Multi-rate Residuals*, a framework that achieves significantly better trade-off between generation quality and decoding speedup than state-of-the-art dynamic computing strategies without requiring costly pre-training. (2) We demonstrate that M2R2 outperforms state-of-the-art methods in self-speculative decoding, achieving a 2.8X speedup on MT-Bench. (3) We demonstrate effectiveness of M2R2 for on-device

*Correspondence to: Nikhil Bhendawade ¡nbhendawade@apple.com¿

MoE models with Ahead-of-Time (AoT) Expert Loading, which reduces latency associated with on-demand expert loading by overlapping memory transfers with computation, achieving a 2.9X speedup over traditional expert loading methods (Lepikhin et al., 2020; Fedus et al., 2022).

2 METHOD

Residual connections in transformer architectures play a pivotal role in evolving hidden representations across layers. Each decoder layer transforms the residual stream h_i through multi-head attention ($\mathcal{A}_i$) and MLP blocks ($\mathcal{M}_i$) from interval E_j to E_{j+1}, as expressed by:

$$h_{E_{j+1}} = h_{E_j} + \sum_{i=E_j}^{E_{j+1}-1} \left(\mathcal{A}_i(h_i) + \mathcal{M}_i(h_i + \mathcal{A}_i(h_i))\right), \tag{1}$$

where $\mathcal{A}_i = f(c_i, h_i)$ depends on contextual representation c_i.[1] To examine whether residual transformations can be accelerated across layers, we measured the directional shift in residual states as $1 - \mathcal{C}(h_{i-1}, h_i)$, where $\mathcal{C}$ denotes normalized cosine similarity. This shift is notably higher in the initial layers, gradually decreasing in subsequent layers as shown in fig. 8a. While this enables early exiting methods to accelerate decoding for simpler tokens by approximating full residual transformation with that of initial-layers, these approaches disregard rate of change in residual states. To gain deeper insights into the distance versus velocity aspects of residual transformation, we conducted a comparative study. We compared early exit heads at layer k of the 32-layer Phi3 model with a smaller k-layer model, keeping other parameters consistent. To accelerate residual transitions, the smaller model used low-rank adapters and residual state distillation, aligning its layer i state with that at $4 \times i$ in the full model (Sanh et al., 2019). As shown in fig. 8b, the smaller model achieved faster residual changes and superior next-token prediction accuracy after k layers, outperforming the base model's early exit mechanism (Schuster et al., 2022; Chen et al., 2023b; Varshney et al., 2024). These results reveal a slow residual transformation bias in dense transformers and prompt the question: could accelerating residual velocity across layers enable earlier token alignment?

2.1 MULTI-RATE RESIDUAL TRANSFORMATION

To address the slow residual transformation bias in section 2, we introduce *accelerated residual streams* operating at rate R relative to the base slow residual stream. We pair the slow residual stream h with an accelerated residual stream p, intrinsically biased towards earlier alignment. The accelerated residual transformation from interval E_j to E_{j+1} can be formally represented as:

$$p_{E_{j+1}} = p_{E_j} + \sum_{i=E_j}^{E_{j+1}-1} \left(\hat{\mathcal{A}}_i(p_i) + \hat{\mathcal{M}}_i(p_i + \hat{\mathcal{A}}_i(p_i))\right) \quad \text{where} \quad \hat{\mathcal{A}}_i = \hat{f}(c_i, p_i), \hat{\mathcal{M}}_i = \hat{g}(h_i), \tag{2}$$

where $\hat{\mathcal{A}}_i$ and $\hat{\mathcal{M}}_i$ denote non-linear transformations added by layer i to the previous accelerated residual p_i. Similar to $\mathcal{A}_i$, non-linear transformation $\hat{\mathcal{A}}_i$ attends to the same context c_i but uses a different transformation $\hat{f}$ for accelerating p_{E_j} relative to h_{E_j}. We integrate accelerated residual transformations into the base model using parallel low-rank accelerator adapters with rank $R_p \ll d$, where d is the hidden dimension. This allows the base slow residual stream h_{E_j} to traverse the base layers while the accelerated stream p_{E_j} utilizes parallel adapters (see fig. 1). Both streams are processed concurrently via attention masking, introducing negligible additional latency in memory-bound decoding setups. In compute-bound scenarios, we develop FLOPs optimization methods to minimize FLOPs overhead as described in appendix A.4. We continue further discussion on KV cache sharing and attention dynamics in appendix A.1.4.

[1] Normalization layers are omitted here for brevity.

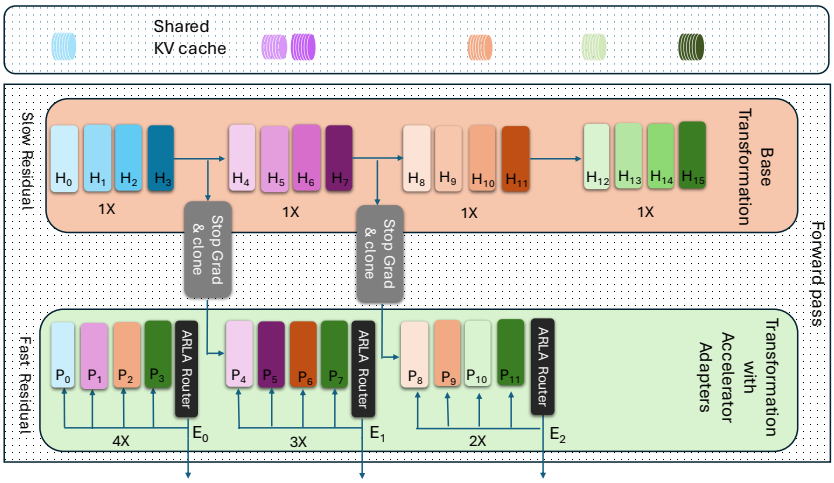


Figure 1: Multi-rate Residuals Framework: Slow residual stream of base model is accompanied by a faster stream that operates at a $2-(J+1)\times$ rate relative to the slow stream, undergoing transformations via accelerator adapters as detailed in section 2.1, where J denotes number of early exit intervals. Colors within the slow and fast residual streams indicate similarity, with matching colors representing the most closely aligned residual states. At the beginning of the forward pass and at each exit point, the accelerated residual state is initialized from the corresponding slow residual state to avoid gradient conflict during training (see appendix A.8). Early exiting decisions are informed by the Accelerated Residual Latent Attention (ARLA) mechanism, described in appendix A.1.2, which evaluates residual dynamics across consecutive exit gates.

2.1.1 Dynamic Computation

Early exiting strategies typically approximate the residual transformation between intermediate layer E_j and the final layer $N-1$ using a linear, context-independent mapping, $\mathcal{T}$, such that $H_{N-1} \approx \mathcal{T}(H_{E_j})$ (Varshney et al., 2024). However, this method is limited by two main factors: the assumption of linearity may not hold uniformly for all tokens, and $\mathcal{T}$ neglects context influence. In contrast, M2R2 addresses these limitations by using context dependent accelerated streams, allowing for faster, more consistent residual shifts and better alignment with final-layer representations (see fig. 6). Beyond alignment, we also enhance the precision of early exit decisions by capturing residual transformation dynamics; see the appendix A.1.1 and appendix A.1.2 for further details.

2.1.2 Self Speculative Decoding

Another way to leverage early alignment is by using accelerated residual streams for speculative token sampling. Since accelerated residuals demonstrate higher fidelity to their slower counterparts, sampling speculative tokens from accelerated stream operating at rate N/k at layer k and verifying them in parallel results in high acceptance rates and enhances the token generation rate. This approach also eliminates the need for a separate draft model, significantly reducing training and memory overhead of maintaining both models in memory as observed in (Chen et al., 2023a; Xia et al., 2023). For more details on self speculative decoding setup, please refer to appendix A.2.

2.1.3 Ahead-of-Time (AoT) Expert Loading

Recent advancements in sparse Mixture-of-Experts (MoE) architectures have enabled token generation efficiency by dynamically activating only a subset of experts per input (Shazeer et al., 2017; Fedus et al., 2022). This selective expert activation approach, while improving efficiency, introduces challenges in pre-loading since expert selection occurs dynamically based on previous layer's output. We address this issue by utilizing accelerated residuals, which allow for the prediction of required experts in advance, mitigating the latency introduced by on-demand expert loading from low bandwidth memory as shown in fig. 2. Ahead-of-Time (AoT) expert loading overlaps with attention and

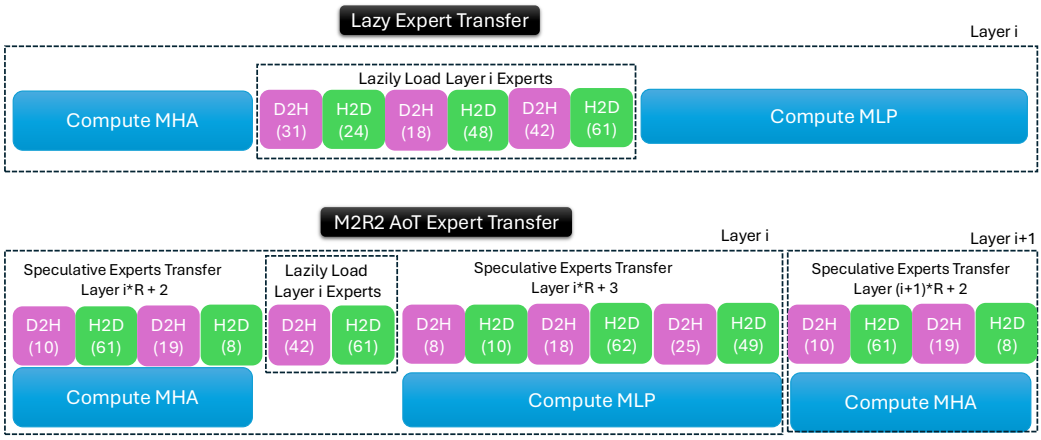


Figure 2: Ahead-of-Time Expert Loading: M2R2 accelerated residual stream predicts experts required for future layers, reducing reliance on on-demand lazy loading. Speculative pre-loading is efficiently overlapped with computation of multi-head attention (MHA) and MLP transformations. Only incorrectly speculated experts are loaded lazily, resulting in faster inference steps and improved computational efficiency. Here, H indicates LBM Host while D indicates HBM Device.

MLP layer computations, substantially reducing idle time for accelerators while expert parameters are being loaded. For more details on the expert loading mechanism, refer to appendix A.3.

2.2 Training

To accelerate residual stream, we train parameters of parallel accelerator adapters. For the dynamic computation (see section 2.1.1), we define the training objective using the following loss function, which combines cross-entropy loss at each exit E_j with distillation loss at each layer i. Loss weights coefficients α_0 and α_1 are employed to balance contribution of corresponding losses.

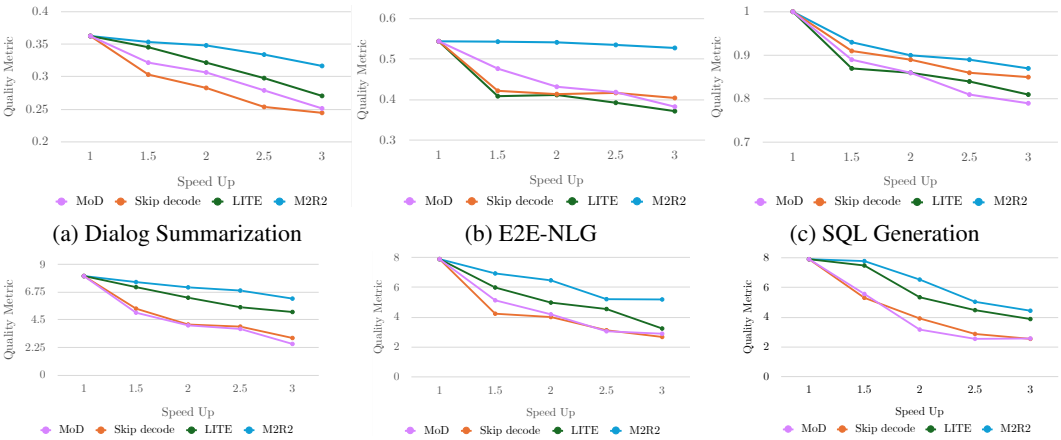
$$L_{\text{m2r2}} = \underbrace{-\alpha_0 \sum_{j=1}^{J} \sum_{t=1}^{T} \log p_\theta \left(\hat{y}_t^{E_j} \mid y_{<t}, x \right)}_{\text{cross-entropy loss}} + \underbrace{\alpha_1 \sum_{i=1}^{E_{J-1}} \sum_{t=1}^{T} \| \mathbf{p}_t^i - \mathbf{h}_t^{((i-E_{j(i)}) \cdot R_i) + E_{j(i)})} \|^2}_{\text{distillation loss}} . \quad (3)$$

where $\hat{y}_t^{E_j}$ denotes the predictions from the accelerated residual stream at layer E_j and time step t, y_t represents the corresponding ground truth tokens, and x indicates previous context tokens. The distillation loss at each layer i is computed by comparing accelerated residuals at layer i with base slow residuals at layer $(i - E_{j(i)}) \cdot R_i + E_{j(i)}$, where R_i denotes the rate of accelerated residuals at layer i while $E_{j(i)}$ represents the most recent gate layer index such that $E_{j(i)} <= i$. J represents the total number of early exit gates (typically set to 4 in dynamic computing setups), N denotes number of hidden layers and E_j denotes layer index corresponding to gate index j and T denotes the sequence length. For self-speculative decoding, as described in section 2.1.2, number of intervals set to $J = 1$ and the rate of residual transformation set to $R_i = N/k$, where the first k layers generate speculative candidate tokens. In case of Ahead-of-Time Expert Loading for MoE models (see section 2.1.3), we set $J = 1$ and the rate of residual transformation to $R_n = 2$.

3 Experiments

Experimental Setup We perform a comprehensive evaluation of our method across instruction-tuning and supervised fine-tuning paradigms, utilizing pre-trained models of varying scales. Our experiments focus on user-facing, supervised fine-tuning tasks for on-device AI assistants, covering Structured API Generation, Text Summarization, and Meaning Representation. These tasks are

evaluated using datasets like TextToSQL (Zhong et al., 2017; Yu et al., 2018), Dialogsum (Chen et al., 2021), and e2e-nlg (Dušek et al., 2020). Additionally, we assess generalizability by training on Alpaca (Touvron et al., 2023) dataset and testing across held-out instruction sets such as Self-Instruct (Wang et al., 2022), Koala (Contributors, 2023), WizardLM (Xu et al., 2023), and MT Bench (Bai et al., 2024). We evaluate our method on various open-source, dense transformer models, such as Phi-3-mini-4k-instruct (3.8B) (Abdin et al., 2024) and Gemma (7B) (Mesnard et al., 2024), as well as the sparse MoE model OlMoE (1B-7B) (Muennighoff et al., 2024). Our baselines include dynamic compute techniques like LITE (Varshney et al., 2024), CALM (Schuster et al., 2022), skip decoding (Del Corro et al., 2023), and Mixture of Depths (MoD) (Raposo et al., 2024). For speculative decoding, we compare our method against draft-target speculative decoding (Chen et al., 2023a) and single-model baselines like Medusa (Cai et al., 2023), DEED (Tang et al., 2024), and LookAhead Decoding (Fu et al., 2023). Metrics include trade-offs between wall-time speedups and generation quality for dynamic compute approaches. For speculative decoding, we focus on wall-time speedups and acceptance rates. In MoE configurations with Ahead-of-Time (AoT) expert loading, we report expert speculation hit rate and decoding speedups. Specific task metrics include Exact Match (EM) for Structured Query, Rouge-LSum for Dialog Summarization (Wolf et al., 2020), and response quality evaluated using GPT-4 (OpenAI, 2023) on human instruction datasets.



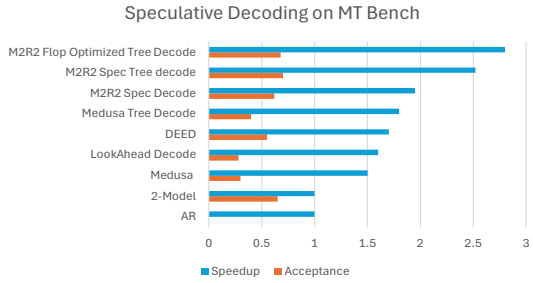
(a) Dialog Summarization (b) E2E-NLG (c) SQL Generation

(d) Self Instruct (e) WizardLM (f) Koala

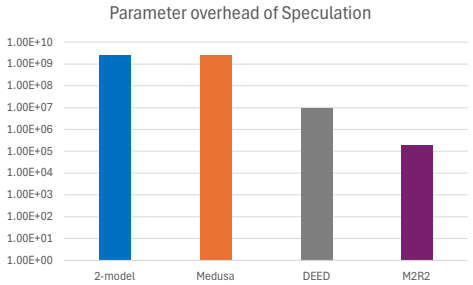
Figure 3: Generation Metric vs Speedup trade-off of different dynamic computing approaches with Phi-3 model on instruction and application specific test sets.

Dynamic Residual Transformation We evaluated LITE (Varshney et al., 2024) and our approach with various early exiting thresholds, examining generation metrics and speedup trade-offs. For Mixture of Depths (MoD) (Raposo et al., 2024), we varied layer capacities to assess performance in relation to speedup. In Skip Decoding (Del Corro et al., 2023), we adapted exit points based on sequence length to achieve similar trade-offs. As shown in fig. 3, M2R2 outperforms other baselines and exhibits consistent improvement in a diverse set of instruction tuning and application specific tasks. Notably approaches that are shown to perform well during pre-training such as Mixture of Depths (Raposo et al., 2024) and Skip decoding (Del Corro et al., 2023) tend to perform poorly during instruction-tuning setups. In appendix A.10 we provide some empirical reasoning that may lead to this suboptimal behavior.

Speculative Decoding Improved early alignment using accelerated residual streams (see fig. 9a) significantly enhances acceptance rates when used for speculative candidate sampling. As shown in fig. 4a, we compare acceptance rates by sampling tokens from the first $k = 4$ layers of Gemma-7B (Mesnard et al., 2024), employing a residual transformation rate $R = N/k$ as noted in section 2.2. While speculative decoding with aligned draft-model (Spector & Re, 2023) achieves high acceptance rates, it does not yield generation speedups due to latency overhead in speculative candidate generation. In contrast, Medusa minimizes speculation overhead but suffers from suboptimal acceptance rates due to the lack of token dependency (Ankner et al., 2024; Bhendawade et al., 2024). Our approach improves on both these approaches as well as DEED (Tang et al., 2024) by leveraging



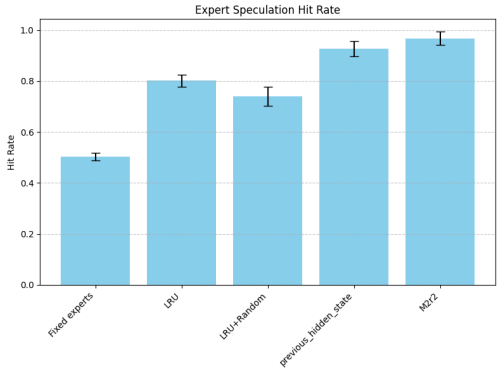
(a) Acceptance rates and speedups of different speculative decoding architectures on Nvidia-A100.



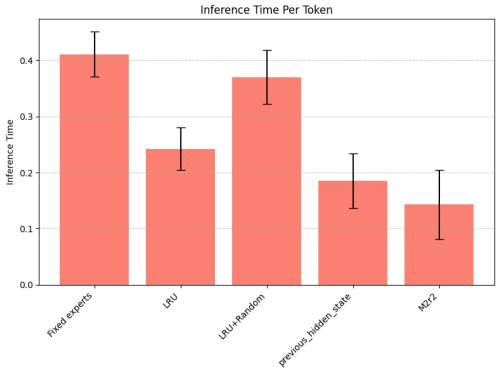
(b) Overhead of trainable parameters of different speculative decoding architectures.

Figure 4: Effectiveness of M2R2 in Speculative Decoding settings.

accelerated residuals, boosting both acceptance rates and generation speedups, as shown in fig. 4. For further details on the speculative decoding experiments and results, see Appendix A.6.2.



(a) Expert speculation hit rate of different expert speculation strategies measured on MT Bench.



(b) Inference latency per decode step using different expert pre-loading strategies measured on MT Bench.

Figure 5: Effectiveness of M2R2 for speculative expert pre-loading on sparse MoE Transformers

MoE Expert Pre-Loading We evaluate decoding latency of MoE models in resource-constrained environments with limited high-bandwidth memory (HBM). When experts are selected that are not stored in HBM, they must be loaded from low-bandwidth memory (LBM), introducing latency. To simulate this scenario on an A100 GPU, we restrict HBM capacity to 8GB, with experts beyond this limit being loaded from LBM. We compare five strategies to optimize compute efficiency and reduce decoding latency: (1) fixed expert caching in HBM, (2) LRU based expert eviction, (3) LRU eviction with random expert preloading, (4) speculative expert preloading based on residual state of previous layer and (5) our proposed method leveraging accelerated residual states for expert preloading. As shown in fig. 5, both our method and approach (4) achieve higher hit rates than other baselines, however our approach achieves most notable reduction in decoding latency. For further details on the experimental setup, results, and additional insights, please refer to appendix A.7.

4 CONCLUSION

In this work, we propose the Mixture of Multi-rate Residuals (M2R2) framework, which optimizes early residual alignment by modulating residual velocities, improving inference efficiency in diverse inference setups. M2R2 outperforms state-of-the-art dynamic computation methods offering better generation metrics to speedup trade-off. Furthermore, it achieves 2.8X speedup in lossless self-speculative decoding setup and 2.9X speedup over traditional MoE inference with lazy expert loading. Overall, M2R2 offers an effective solution for optimizing inference in resource-constrained environments, enhancing both dense transformer and sparse MoE models.

REFERENCES

Marah Abdin et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024.

Megha Agarwal, Asfandyar Qureshi, Nikhil Sardana, Linden Li, Julian Quevedo, and Daya Khudia. Llm inference performance engineering: Best practices., 2023a.

Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. Gkd: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649*, 2023b.

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontanon, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. Colt5: Faster long-range transformers with conditional computation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5085–5100, Singapore, 2023. Association for Computational Linguistics.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa de- coding, 2024.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7421–7454, Bangkok, Thailand, 2024. Association for Computational Linguistics.

Nikhil Bhendawade, Irina Belousova, Qichen Fu, Henry Mason, Mohammad Rastegari, and Mahyar Najibi. Speculative streaming: Fast llm inference without auxiliary models. *arXiv preprint arXiv:2402.11131*, 2024. URL https://arxiv.org/abs/2402.11131.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, and Tri Dao. Medusa: Simple framework for accelerating llm generation with multiple decoding heads. https://github.com/FasterDecoding/Medusa, 2023.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. EE-LLM: Large-scale training and inference of early-exit large language models with 3d parallelism. *arXiv preprint arXiv:2312.04916*, 2023b. Version 3, revised 16 Jun 2024.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. DialogSum: A real-life scenario dialogue summarization dataset. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 5062–5074, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.449. URL https://aclanthology.org/2021.findings-acl.449.

Koala Project Contributors. Koala instruction set documentation, 2023. URL https://github.com/KoalaInstructionSet/koala-docs. Accessed: 2024-11-06.

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference. *arXiv preprint arXiv:2307.02628*, July 2023. URL https://arxiv.org/abs/2307.02628.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts. *arXiv preprint arXiv:2112.06905*, 2022.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computer Speech & Language*, 59:123–156, January 2020. doi: 10.1016/j.csl.2019.06.009.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, April 2020. OpenReview.net. `https://openreview.net`.

Kaya et al. Predictive exit: Prediction of fine-grained early exits for computation- and energy-efficient inference. *arXiv preprint arXiv:2206.04685*, 2023a. URL `https://ar5iv.org/abs/2206.04685`.

Sharma et al. Decoupled early time series classification using varied-length feature augmentation and gradient projection technique. *MDPI Electronics*, 2023b. URL `https://www.mdpi.com/`.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2021.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23:1–39, 2022.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Breaking the sequential dependency of llm inference using lookahead decoding, November 2023. URL `https://lmsys.org/blog/2023-11-21-lookahead-decoding/`.

L. Garncarek and J. Snaider. Dynamic model compression via reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 3000–3010, 2021.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: dynamic bert with adaptive width and depth. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 9782–9793, December 2020.

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

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, January 2024. URL `https://arxiv.org/abs/2401.04088`.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nash Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, 2017. doi: 10.1145/3079856.3080246.

Norman P Jouppi et al. Ten lessons from three generations shaped google's tpuv4i. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021.

Stephen W Keckler, William J Dally, and Brucek Khailany. Gpus and the future of parallel computing. *IEEE Micro*, 31(5):7–17, 2011. doi: 10.1109/MM.2011.89.

Nicholas D Lane and Petko Georgiev. Ai and machine learning acceleration in mobile devices: A survey of architectures, hardware, and algorithms. *IEEE Signal Processing Magazine*, 37(6): 75–84, 2020. doi: 10.1109/MSP.2020.3018112.

Denis Lepikhin, Yi Lee, Hao Xu, Zongwei Chen, Orhan Firat, Yanping Huang, and et al. GShard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Gemma Team: Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. URL https://doi.org/10.48550/arXiv.2403.08295. Computation and Language (cs.CL); Artificial Intelligence (cs.AI).

Xiangru Miao, Gabriel Oliaro, Zhenyu Zhang, Xinyang Cheng, Zhen Wang, Randy Y. Y. Wong, Zhi Chen, Danish Arfeen, Rishita Abhyankar, and Zhihao Jia. SpecInfer: Accelerating generative LLM serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 2023.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, September 2024. URL https://arxiv.org/abs/2409.02060.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Mostofa Ali Patwary, Sheikh Mostakim, Jianwei Huang, and et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15, 2021.

NVIDIA. Cuda c++ programming guide, 2021. Available at: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

OpenAI. GPT-4 Technical Report, 2023. URL https://openai.com/research/gpt-4.

Alec Radford, Jeff Wu, Rewon Child, Dario Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://cdn.openai.com/transcriptions/GPT-2.pdf.

David Raposo, Sam Ritter, Blake Richards, T. Lillicrap, Peter Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, April 2024. URL https://doi.org/10.48550/arXiv.2404.02258. Corpus ID: 268876220.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 17456–17472, April 2022.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Y. Shen, R. Wang, and D. Zhang. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, volume 34, pp. 11778–11789, 2021.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2020.

Benjamin Spector and Chris Re. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Peng Tang, Pengkai Zhu, Tian Li, Srikar Appalaraju, Vijay Mahadevan, and R. Manmatha. Deed: Dynamic early exit on decoder for accelerating encoder-decoder transformer models. In *NAACL 2024*, 2024.

Hugo Touvron, P. Tuan Pham, S. Ravi, and A. Joulin. Alpaca: A strong, affordable instruction-following model. 2023.

Neeraj Varshney and Chitta Baral. Model cascading: Towards jointly improving efficiency and accuracy of nlp systems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 11007–11021, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics.

Neeraj Varshney, Agneet Chatterjee, Mihir Parmar, and Chitta Baral. Investigating acceleration of LLaMA inference by enabling intermediate layer decoding via instruction tuning with 'lite'. In *Findings of the Association for Computational Linguistics: NAACL 2024*, volume Findings of *NAACL*, pp. 3656–3677, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.232. URL https://aclanthology.org/2024.findings-naacl.232. Anthology ID: 2024.findings-naacl.232.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, Joe Davison, et al. Transformers: State-of-the-art natural language processing, 2020. Accessed via https://huggingface.co.

Haoyang Xia, Tianyang Ge, Shiqi Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Lossless speedup of autoregressive translation, 2023. URL https://openreview.net/forum?id=H-VlwsYvVi. Presented at OpenReview.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2246–2251, Online, 2020. Association for Computational Linguistics.

Canwen Xu et al. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.

A Appendix

A.1 Dynamic Computing

A.1.1 Early Exiting Analysis

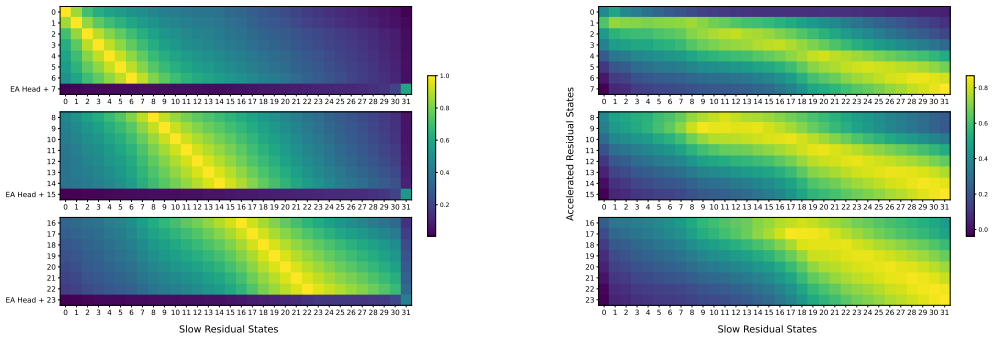
The traditional early exiting methods (Schuster et al., 2022; Chen et al., 2023b; Varshney et al., 2024) approximate the residual transformation from layer E_j to $N-1$ using a linear mapping $\mathcal{T}$, which is expressed as:

$$h_{E_j} + \sum_{i=E_j}^{N-1} (\mathcal{A}_i(h_i) + \mathcal{M}_i(h_i + \mathcal{A}_i(h_i))) \sim \mathcal{T}(h_{E_j}) \quad \text{where} \quad \mathcal{T} \perp c. \tag{4}$$

Here, $\mathcal{A}_i$ and $\mathcal{M}_i$ represent the contributions from multi-head attention and MLP layers, respectively, and $\mathcal{T}$ is independent of the preceding context c. This approximation is constrained by the linearity assumption and the lack of contextual awareness. In contrast, M2R2 addresses these challenges by approximating the slow residual transformation using a faster transformation over fewer layers as follows:

$$h_{E_j} + \sum_{i=E_j}^{N-1} (\mathcal{A}_i(h_i) + \mathcal{M}_i(h_i + \mathcal{A}_i(h_i))) \sim p_{E_j} + \sum_{i=E_j}^{E_{j+1}-1} \left(\hat{\mathcal{A}}_i(p_i) + \hat{\mathcal{M}}_i(p_i + \hat{\mathcal{A}}_i(p_i))\right), \tag{5}$$

where p_{E_j} is initialized from the slow residual state h_{E_j} using an identity transformation at each early exit interval. This leads to smoother residual shifts and improved alignment with the final residual states, as shown in fig. 6b. Moreover, the normalized cosine similarity between accelerated and final residual states is significantly higher, emphasizing the improved alignment compared to traditional methods.



(a) Residual similarity in traditional early exiting

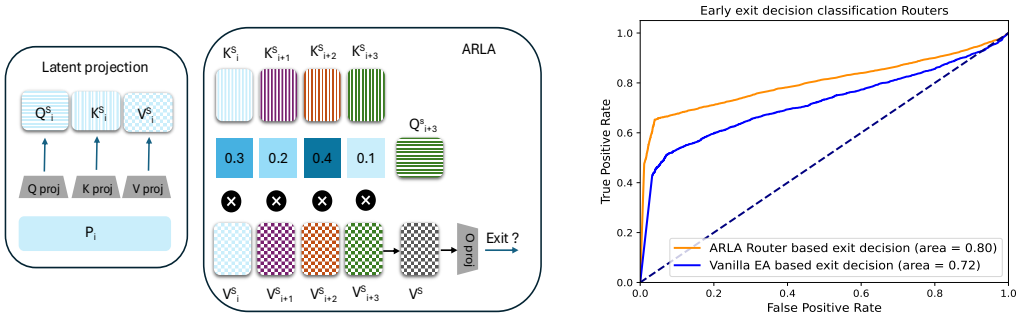
(b) Residual similarity with M2R2

Figure 6: Traditional early exiting approaches approximate the final residual state with context-independent mapping, $\mathcal{T}$, applied on intermediate hidden state, resulting in discontinuities in transformations and lower similarity with final residual state. In contrast, M2R2 progressively enhances residual transformation velocity at each layer, enabling more robust and uniform early alignment.

A.1.2 Accelerated Residual Latent Attention (ARLA)

In the context of residual streams, we observe that the decision to exit at a given layer can be more effectively informed by analyzing the dynamics of residual stream transformations, instead of solely relying on a classification head applied at the early exit interval E_j. To capture the subtle dynamics of residual acceleration, we propose a *Accelerated Residual Latent Attention* (ARLA) mechanism. This approach involves making the exit decision at gate E_j by attending to the residuals spanning

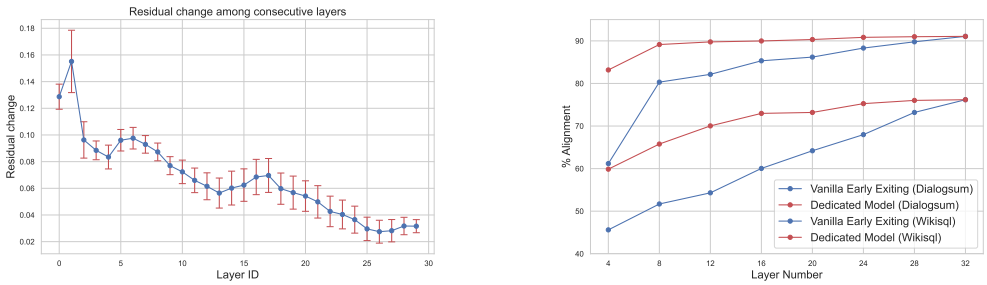
from gate E_{j-1} to E_j, rather than considering only the residual at gate E_j. To minimize the computational overhead associated with exit decision-making, the attention mechanism operates within the latent domain as depicted in fig. 7a. Formally, for each interval $[E_j, E_{j+1}]$, the accelerated residuals are projected into Query $(Q^s_{E_j}, \ldots, Q^s_{E_{j+1}})$, Key $(K^s_{E_j}, \ldots, K^s_{E_{j+1}})$, and Value $(V^s_{E_j}, \ldots, V^s_{E_{j+1}})$ vectors, with latent dimension d^s for Q^s, K^s, and V^s being significantly smaller than that of p.[2] Notably, when the router is allowed to make exit decisions at gate E_j based on residual change dynamics, we observe that the attention is not confined to the residual state at E_j but is distributed across residual states from E_{j-1} to E_j, This broader focus on residual dynamics significantly reduces decision ambiguity in early exits, as demonstrated in Figure 7b, which contrasts routers based on the last hidden state, and the proposed ARLA router.



(a) Accelerated Residual Latent Attention (ARLA): Accelerated residuals between early exit gates are projected into latent domain and attention over residual states within the interval is computed to capture residual dynamics and exit decision is made based on residual saturation.

(b) ROC classification curves of early exit decision strategies using a linear router used on last residual state (Schuster et al., 2022; Varshney et al., 2024; Chen et al., 2023b) and using ARLA approach that considers residual dynamics.

Figure 7: Effectiveness of ARLA in capturing residual dynamics for early exiting decisions.



(a) (b)

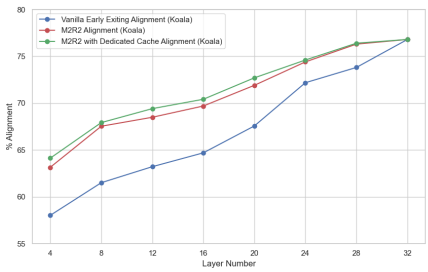
Figure 8: (a) As residual streams propagate through the model, the directional shifts in the residuals become progressively smaller. (b) A smaller model with k layers achieves faster rate of change in residual streams and higher alignment (consistency between token predictions from intermediate layers and the final layer) than base model leveraging early exit mechanisms at layer k. To ensure an equal number of trainable parameters, we inserted LoRa adapters into the smaller model and trained only these adapters, whereas, in early exit-based approach, we trained solely the early exit head.

A.1.3 Early Alignment and ARLA Effectiveness

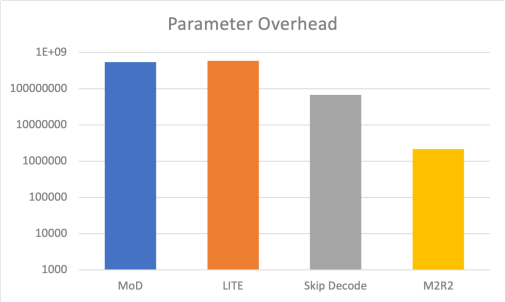
To demonstrate effectiveness of M2R2, we begin by analyzing the alignment of tokens exited at intermediate gates with those exited at the final layer using the Koala instruction set (Contributors, 2023). As shown in fig. 9a, we observe that accelerated residuals achieve significantly higher alignment compared to conventional early exit approaches, such as those proposed in (Schuster et al.,

[2]We use $d^s = 64$ for experiments described in section 3.

2022; Chen et al., 2023b; Varshney et al., 2024). This difference in alignment is particularly pronounced at lower gates, demonstrating that accelerated residual streams more effectively capture the features of the final-layer slow residual stream than applying a projection layer on intermediate slow residuals. Additionally, we find that sharing the KV cache between slow and accelerated residuals does not significantly impact alignment. Cache sharing allows for substantial reductions in runtime memory, and in the experiments detailed in section 3, we share the cache between slow and accelerated residual streams. We also compare ROC curves obtained from confidence scores that are used to make exiting decisions in (Schuster et al., 2022) and our approach. As observed in fig. 7b, ARLA described in appendix A.1.2, is consistently effective in optimally determining decision boundaries than classifier-based routers that operate on latest slow residual state at each gate in (Schuster et al., 2022; Varshney et al., 2024).



(a) Early alignment performance on the Koala test set, comparing traditional early exiting with M2R2, both with and without cache sharing between slow and accelerated residuals.



(b) Parameter overhead across different dynamic compute approaches, highlighting additional trainable parameters in routers, projection layers, accelerator adapters, and ARLA.

Figure 9: Alignment of early exited tokens and trainable parameter overhead associated with different dynamic computing approaches.

A.1.4 M2R2 Caching and Attention Dynamics

To maximize the utility of accelerated residual transformations without introducing dedicated KV caches, we propose a shared caching mechanism between the slow and accelerated streams. This approach minimizes memory costs while preserving alignment benefits (fig. 9a). Specifically, the attention operation for slow residuals, $\text{MHA}(h_t, h_{\leq t}, h_{\leq t})$, is redefined for accelerated residuals as:

$$\hat{\mathcal{A}} = \text{MHA}(p_t, h_{<t} \oplus p_t, h_{<t} \oplus p_t),$$

where p_t reuses the slow residual's KV cache, facilitating the sharing of contextual information without additional caching overhead. Here, $\text{MHA}(q, k, v)$ represents multi-head attention between query q, key k, and value v.

A.2 Self Speculative Decoding

Speculative decoding enhances autoregressive inference by using a draft model to predict tokens and then verifying these predictions in parallel, enabling token generation at a faster rate (Leviathan et al., 2023; Chen et al., 2023a; Xia et al., 2023; Miao et al., 2023). However, it introduces deployment and training complexities due to the need for a draft model trained in alignment with the target model. The simultaneous maintenance of both models in memory further exacerbates this inefficiency (Leviathan et al., 2023).

To overcome these issues, DEED (Tang et al., 2024) propose leveraging the initial layers of the target model itself for speculative token generation using early exiting mechanism. This method reduces overhead by eliminating the need for an additional model while still enabling rapid token generation. However, this method suffers from suboptimal acceptance rates due to the poor approximation of residual transformations (see section 2.1.1, appendix A.1.1) and weak alignment between speculative tokens from intermediate layers and those from the final layer as discussed in appendix A.1.3. Our solution utilizes accelerated residuals to better approximate the final model's output, with a

virtual draft model operating at a rate of N/k, where k is the number of layers used for candidate speculation and N is total number of layers. This improves speculative token generation and acceptance rates, offering a more efficient alternative to traditional speculative decoding methods. For experimental details and further performance analysis, see section 3.

A.3 AHEAD OF TIME EXPERT LOADING

Sparse Mixture-of-Experts (MoE) models, such as those discussed in (Shazeer et al., 2017; Fedus et al., 2022), are designed to activate only a subset of experts per input, which allows for greater efficiency compared to dense models. However, this introduces a challenge in expert selection, as it must occur dynamically based on the output of previous layers. In dense transformer models, pre-loading the next layer's parameters can be done in parallel with the current layer's computation (Narayanan et al., 2021; Shoeybi et al., 2020). This strategy is not directly applicable to MoE models due to the sequential nature of expert selection, which leads to inherent latency during decoding (Lepikhin et al., 2020; Fedus et al., 2022).

To resolve this, our method leverages accelerated residuals that capture essential characteristics of base slower residual states and exhibit high similarity to their final counterparts (see fig. 6b). By employing a $2\times$ accelerated residual, we predict and start pre-loading the necessary experts for layers $2i + 2$ and $2i + 3$ while still processing layer i, thus reducing the latency caused by sequential expert selection fig. 2. In this setup, we utilize a fixed set of accelerator adapters for transforming accelerated residuals, while base slower residuals are processed via an expert routing mechanism. Additionally, our strategy incorporates a Least Recently Used (LRU) caching mechanism, which improves memory efficiency by replacing the least recently used experts with those speculated to be required in subsequent layers. This combined approach of preemptive expert loading and LRU caching leads to substantial improvements over traditional methods, enhancing both memory management and compute efficiency. Evaluation results and detailed compute and memory traces on an A100 GPU are presented in appendix A.7 and fig. 11.

A.4 FLOPS OPTIMIZATION

Naively implemented, M2R2 incurs a higher FLOP overhead compared to traditional speculative decoding and early exiting approaches such as (Cai et al., 2023; Schuster et al., 2022; Tang et al., 2024). However, modern accelerators demonstrate compute bandwidth that exceeds memory access bandwidth by an order of magnitude or more (Agarwal et al., 2023a; Jouppi et al., 2021), meaning increased FLOPs do not necessarily translate to increased decoding latency. Nevertheless, to ensure fair comparison and efficiency in compute bound scenarios, we introduce targeted optimizations.

Attention FLOPs Optimization For medium-to-long context lengths, attention computation dominates FLOPs in the self-attention layer, surpassing the contribution from projection layers. Specifically, matrix multiplications involving queries, cached keys, and cached values scale with $l_{kv} * l_q$ where l_{kv} denotes previous context length and l_q denotes current query length. Since M2R2 pairs accelerated residuals with base slow residuals, a naive implementation results in twice the FLOPs consumption compared to a standard attention layer. To address this, we limit the attention of accelerated residual stream to selectively attend to the top-k most relevant tokens, identified by the slow residual stream based on top attention coefficients[3]. This is possible since slow and accelerated residual streams are processed in same forward pass and accelerated streams have access to attention coefficients of slow stream. Note that, the faster residual stream still retains the flexibility to assign distinct attention coefficients to these tokens. Furthermore, we design the faster residual stream to employ only 8 attention heads, compared to the 32 heads used in the slow residual stream of the Phi-3 model, reducing query, key, value, and output projection FLOPs by a factor of 1/4. fig. 12b indicates effect of using a slicker stream on alignment. As depicted, using $\hat{n}_h = 8$ offers a good trade-off between alignment and FLOPs overhead.

MLP FLOPs Optimization The accelerator adapters operating on the accelerated residual stream are intentionally designed with lower rank than their counterparts in the base model. This reduces

[3]We set to k = 64 and attend to top 64 tokens as identified by the slow residual stream.

FLOP overhead by a factor proportional to $hiddenSize/rank$. Additionally, since the faster residual stream uses only 8 attention heads (compared to 32 in the slow residual stream of Phi-3), the subsequent MLP layers process a smaller set of activations, further reducing FLOPs by another factor of 1/4.

These optimizations significantly reduce the FLOP overhead per speculative draft generation, as illustrated in fig. 12a. Notably, while traditional early-exiting speculative approaches such as DEED require propagating the full slow residual state through the initial layers, incurring substantial computational costs, M2R2 achieves efficient token generation via slimmer residual streams. In contrast, Medusa introduces considerable FLOP overhead due to per-medusa-head computations scaling with $d^2 + dv$[4], whereas M2R2 employs low-rank layers for both MLP and language modeling heads, maintaining computational efficiency. All experiments involving the M2R2 approach, as detailed in section 3, are conducted using these FLOPs optimizations.

A.5 Additional Experimental Details

In this section, we provide supplementary details about the experimental setup and evaluation methodology to complement the overview in the main paper.

Dataset Details. For Structured API Generation, we use `TextToSQL` dataset constructed by combining examples from WikiSQL (Zhong et al., 2017) and SPIDER (Yu et al., 2018), including cases with multi-turn queries and nested joins. For Text Summarization, we use the Dialogsum (Chen et al., 2021) dataset containing multi-speaker scenarios and conversational coherence. The e2e-nlg dataset (Dušek et al., 2020) used for Meaning Representation tasks is contains domain-specific templates to better evaluate the semantic accuracy of generated responses.

Generalization Studies. While the main paper outlines the use of Alpaca (Touvron et al., 2023) for training and several held-out instruction sets for testing, we include additional details about the domain diversity of these instruction sets. For example, Self-Instruct (Wang et al., 2022) covers open-ended questions, Koala (Contributors, 2023) emphasizes conversational nuances, WizardLM (Xu et al., 2023) includes creative problem-solving tasks, and MT Bench (Bai et al., 2024) tests instruction-following in highly technical domains.

A.6 Results in detail

A.6.1 Parameter Overhead

We measure the trainable parameter overhead of M2R2 associated with routers, projection layers, and the ARLA mechanism (see appendix A.1.2). As shown in fig. 9b, our approach achieves significantly lower parameter overhead, as accelerator adapters work effectively with lower ranks (see fig. 10a), and the ARLA latent dimension is substantially smaller (see appendix A.1.2). Parameter overhead in LITE (Varshney et al., 2024) and Skip Decoding (Del Corro et al., 2023) is mainly due to the projection layer, while MoD (Raposo et al., 2024) is dominated by router parameters, including linear projections and binary classifiers at each layer.

A.6.2 Speculative Decoding Speedups

We conducted self speculative decoding experiments using the first $k = 4$ layers of Gemma-7B (Mesnard et al., 2024), employing a residual transformation rate $R = N/k$, where N denotes the number of layers of the base model. The baselines compared include draft-model-based speculative decoding (Leviathan et al., 2023; Chen et al., 2023a), where Gemma-2B (Mesnard et al., 2024) serves as the draft model, and single-model methods such as Medusa (Cai et al., 2023), DEED (Tang et al., 2024), and LookAhead Decoding (Fu et al., 2023).

While draft-target setup shows high acceptance rates, it does not provide speedup due to latency associated with candidate generation since draft model generates candidate speculation autoregressively and the speculative overhead outweigh parallel acceptance gains. Medusa minimizes speculative generation overhead by generating candidates in a non-autoregressive manner but suffers from

[4]Here d denotes hidden state dimension while v denotes vocab size.

suboptimal acceptance rates as it fails to leverage token dependencies (Ankner et al., 2024; Bhendawade et al., 2024). DEED (Tang et al., 2024) generates candidates using only a subset of model layers, reducing candidate generation costs, but lacks sufficient alignment for high acceptance rates.

Our method addresses these issues by using accelerated residuals to improve both alignment and generation speedups as shown in fig. 4a. Furthermore, as shown in fig. 4b, the parameter overhead of our approach for speculative draft generation is significantly lower than the baselines, making it well-suited for resource-constrained environments.

A.7 Ahead-of-Time (AoT) MoE Expert Loading

We evaluate efficacy of Ahead-of-Time (AoT) expert loading in resource-constrained environments, where limited high-bandwidth memory (HBM) restricts the number of experts that can be stored in fast-access memory. When an MoE gate selects an expert that is not in HBM, it must be loaded from low-bandwidth memory (LBM), introducing latency. To simulate this scenario on an A100 GPU, we restrict HBM capacity to 8GB, requiring experts beyond this limit to be loaded from LBM.[5].

We compare five strategies to maximize compute efficiency and reduce latency. The first approach fixes experts in HBM without replacement. In the second strategy, we employ a Least Recently Used (LRU) eviction policy, where the least accessed experts are dynamically replaced when new ones are needed. The third method extends the LRU approach by adding speculative caching and randomly pre-loading experts. The fourth strategy uses speculative loading based on residual states from the previous layer. Finally, our proposed approach, described in section 2.1.3, leverages accelerated residual states to more effectively speculate and pre-load experts in advance. Using the OLMoE model, which has 64 experts per MLP layer, our 8GB HBM capacity allows only 32 experts to be cached in high-speed memory, while the remaining 32 reside in LBM.

As shown in fig. 5, both our method and the approach of speculating experts based on residual state of previous layer achieve significantly higher hit rates compared to the fixed and LRU-based strategies. Our method operates on accelerated residuals at a rate of 2X, initiating speculative pre-loading of experts at layers $2i + 2$ and $2i + 3$ while the GPU kernel is engaged in computing the attention and MLP transformations for layer i. Starting pre-loading earlier proves advantageous, as we observe that miss rates tend to increase in the final layers when using LRU caching strategies (see fig. 10b). If speculative pre-loading is initiated only one layer before the current layer, it often results in insufficient loading time, preventing all the necessary experts for layer i from being fully loaded during the computation of layer $i-1$. By pre-loading ahead, our method ensures that most speculated experts are readily available, thereby reducing latency and improving inference efficiency. While we demonstrate the effectiveness of operating at a 2X rate for initiating expert pre-loading, the optimal extent of early pre-loading necessary to maximize inference performance remains an open question. We leave this exploration for future work.

A.8 Gradient conflict resolution

Traditional early exiting strategies frequently encounter issues related to gradient conflicts (et al., 2023a;b), where multiple exit points induce conflicting gradients during the training phase. This phenomenon leads to optimization instability and challenges in convergence, as gradients computed from divergent branches may not align effectively, and the presence of early exits can perturb the gradient flow, potentially resulting in the incomplete training of lower early exit heads. To illustrate this problem, consider a trainable parameter w_j situated between gates E_j and E_{j+1}. For the loss associated with the early exit at gates $E_{j+1\ldots n}$, the parameter update required in w_j can be expressed as:

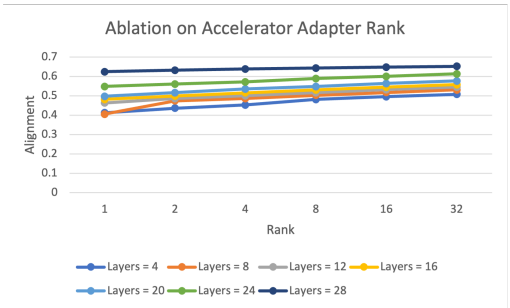
$$\Delta w_j = -\eta \sum_{k=j+1}^{n} \beta_k \frac{\partial L_{E_k}}{\partial w_k} \tag{6}$$

[5]In our experimental setup, GPU DRAM is treated as HBM and disk as LBM, though the setup generalizes to architectures where SRAM serves as HBM and DRAM as LBM, a common design in many accelerators (Jouppi et al., 2017; Keckler et al., 2011; Lane & Georgiev, 2020)

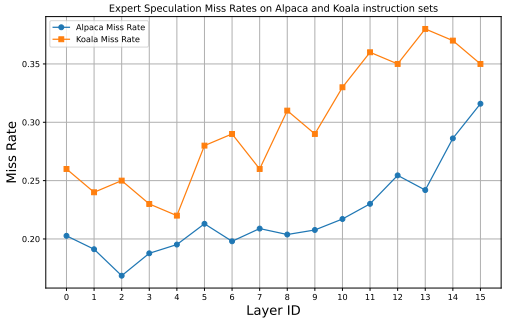
where β_k is the backward transformation coefficient for the gradient from gate E_k to reach parameter w_j and η is the learning rate. Conversely, since accelerated residuals at gate E_j are initialized from slow residuals H_j which are trained with base model parameters that are frozen, gradient propagation is limited to parallel adapter parameters from gate E_j to gate E_{j+1} thus ensuring every parallel adapter parameter is optimized for specific exit as shown in fig. 1. Formally speaking, the update of accelerator adapter parameter w_j within our proposed framework is delineated as:

$$\Delta w_j = \begin{cases} -\eta \hat{\beta}_{j+1} \frac{\partial L_{E_{j+1}}}{\partial w_j} & \text{if } E_j < w_j < E_{j+1} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\hat{\beta}_{j+1}$ is the backward transformation coefficient for the gradient from gate E_{j+1} to reach parameter w_j of accelerator adapter. This formulation mitigates gradient conflicts arising from gradients associated with top gates, thereby enhancing the stability of the optimization process.[6]



(a) Alignment of early-exited tokens with those from the final layer improves as adapter rank increases, but tends to plateau beyond a rank of 8.



(b) Expert speculation miss rates increase in later layers with LRU policy, while initial layers exhibit lower miss rates. Thus, accurate speculative pre-loading benefits later layers more. We leverage accelerated residuals to speculate and pre-load experts for these layers during the computation of earlier layers.

Figure 10: (a) Adapter Rank Ablation on Dialog Summarization (b) Expert speculation miss Rates

A.9 Related Work

The inference speed of large language models (LLMs) is often constrained by the sequential nature of auto-regressive decoding, which necessitates a complete forward pass of the network for each token generated. To mitigate the high inference latency associated with LLMs, various strategies have been proposed to reduce their memory footprint. Techniques such as model quantization (Frantar et al., 2022; Yao et al., 2022; Dettmers et al., 2023), knowledge distillation to smaller models (Gu et al., 2023; Agarwal et al., 2023b), and pruning (Frantar & Alistarh, 2023; Sun et al., 2023) have emerged as effective solutions. However, these strategies often neglect the variational complexity inherent in each token, resulting in a reliance on static computation for all tokens. To better address this issue, several early exiting approaches have been developed to facilitate dynamic computation. These methods focus on terminating residual transformations early for simpler tokens, achieving significant speedups in embedding models (Xin et al., 2020; Hou et al., 2020; Varshney & Baral, 2022). In the context of sequence generation models, techniques like Confident Adaptive Language Modeling (CALM) (Schuster et al., 2022) and Depth-Adaptive Transformers (Elbayad et al., 2020) have effectively employed early exiting by integrating classifiers into the decoder layers. However, these approaches are constrained by key-value (KV) cache mismatches that arise between the training and inference phases, as KV states are not accessible for tokens that are early-exited. To mitigate these limitations, skip decoding (Del Corro et al., 2023) has been introduced. This method allows for bypassing a progressively increasing number of layers based on the token's position in the decoded

[6]For simplicity, we focus on cross-entropy loss in this discussion; however, the same reasoning extends to distillation loss as detailed in section 2.2.

sequence. While this approach effectively circumvents KV mismatches, the pre-defined limitations on the number of bypassed layers can lead to suboptimal generation quality.

Another promising direction involves conditioning residual transformations at each layer through the use of a router. For example, CoLT5 (Ainslie et al., 2023) employs conditional routing to determine whether a token should follow a heavy or light computational pathway for each feedforward layer in encoder-decoder models. Mixture-of-depths (Raposo et al., 2024) builds upon this idea by introducing a predictive router at each layer, which enables efficient inference for conditional computation in decoder-only models. Although conditional routing demonstrates potential during pre-training, as illustrated in section 3, its effectiveness during supervised fine-tuning and instruction tuning remains limited. This restricts the applicability of this technique across a wider array of publicly available pre-trained models.

Speculative decoding (SD) has also emerged as a potent method for accelerating autoregressive inference. Techniques such as the original SD framework (Leviathan et al., 2023; Chen et al., 2023a) utilize a smaller draft model to generate token candidates, which are subsequently validated by the target model, achieving speedups of 2-3x. However, this dual-model approach complicates deployment, as it necessitates hosting both models in memory, which can be resource-intensive in constrained environments. Alternatives like Medusa offer single-model solutions but are limited by their inability to account for token dependencies. In contrast, our approach introduces dependencies between speculative tokens, resulting in more coherent and efficient speculation, thereby achieving higher decoding speedups.

The recent proliferation of Mixture-of-Experts (MoE) language models builds on a long-established concept (Jacobs et al., 1991; Jordan & Jacobs, 1994) of training ensembles of specialized models or "experts," and employing a gating function to select the appropriate expert for a given task. Shazeer et al. (Shazeer et al., 2017) further this idea by developing a sparsely gated Mixture-of-Experts language model. Numerous studies have since explored the application of MoE architectures in Transformer-based models for tasks such as machine translation (Lepikhin et al., 2021), masked language modeling (Fedus et al., 2021), and general-purpose LLMs (Du et al., 2022). Recently, state-of-the-art sparse Mixture-of-Experts models, such as Mixtral-8x7B (Jiang et al., 2024) and OLMoE (Muennighoff et al., 2024), have been released, outperforming their open-source dense transformer counterparts across several benchmarks.

A.10 Discontinuity in Mixture of Depths and Skip Decoding

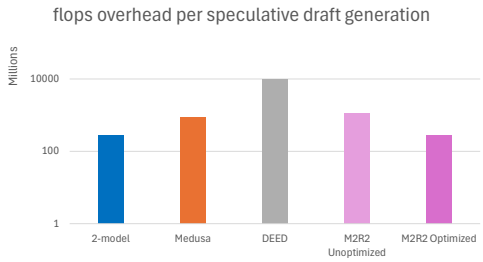
To get a deeper understanding of discontinuity leading to suboptimal performance of architectures like MoD and Skip decoding during instruction tuning and fine-tuning phases, we pass a diverse set of prompts through the models and observe residual stream transformation. Residual streams in pre-trained dense transformers tend to undergo significant changes in both direction and magnitude in first few layers than later layers as depicted in fig. 8a. Since Mixture of Depth (MoD) relies on skipping the residual transformation for some of the tokens determined by router, skipping early transformations makes it harder to obtain final residual state closer to that obtained from full dense transformers even when MoD parameters are explicitly trained to alleviate this discontinuity. Skip decoding on the other hand approximates skipping residual transformation of first few layers with a linear projection while ignoring non-linearities and context, leading to sub-otpimal performance.

A.11 MoE Speculation Continued

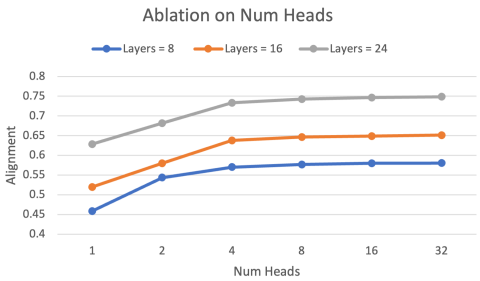
In this section, we detail the expert transfer process between High Bandwidth Memory (HBM) and Low Bandwidth Memory (LBM) on the A100 GPU. We employ CUDA's multi-stream functionality (NVIDIA, 2021) to establish distinct compute and memory-loading streams, both of which operate concurrently during each forward pass. The load stream is scheduled ahead of the compute stream to ensure efficient memory management: while the compute stream processes layer i, the load stream transfers the least recently used experts of layer $2i + 2$ and $2i + 3$ to LBM and loads speculated experts into HBM. This approach leverages the accelerated residual at layer i, which exhibits strong similarity to the slow residuals at layers $2i + 2$ and $2i + 3$, enabling effective expert speculation as shown in fig. 2. Before executing the MLP experts, we verify whether all required experts are available on HBM; if not, the load stream initiates prioritized, on-demand loading for the experts

Figure 11: A100 GPU trace demonstrating overlap of computation and expert transfer between LBM and HBM.

necessary for MLP computation at layer i. Coordination between the load and compute streams is managed using CUDA primitives.



(a) FLOPs overhead of generating speculative draft for different approaches on Gemma-7B. The optimized M2R2 approach incorporates FLOPs optimization techniques described in appendix A.4



(b) Ablation study on attention heads and M2R2 alignment benefits. Using 4 to 8 heads in the accelerated residual stream reduces FLOPs with minimal alignment degradation.

Figure 12: FLOPs overhead of M2R2 and optimization based on Attention head pruning.

Accelerator Adapter Rank Ablation To minimize parameter overhead from accelerator adapters, we conduct an ablation study on adapter rank to identify the optimal rank that achieves strong alignment without substantially increasing parameter load. As illustrated in fig. 10a, a rank of 8 offers an effective trade-off, with alignment performance showing a steep improvement up to rank 8, beyond which the benefit curve begins to plateau.

Prompt for Evaluation of Dynamic Compute Responses

To assess the responses generated by our approach alongside baseline models, we utilize the following prompt for GPT-4 oracle. Note that the baseline and target responses are randomly assigned to either Assistant 1 or Assistant 2 in the template below.

```
Human: You are a helpful and precise assistant for evaluating the
   quality of an answer.
[Question]
{question}
[The Start of Assistant 1's Answer]
{answer_1}
[The End of Assistant 1's Answer]
[The Start of Assistant 2's Answer]
{answer_2}
[The End of Assistant 2's Answer]

We request your feedback on the performance of both AI assistants
   in response to the user question above. Please rate their
   responses based on helpfulness, relevance, accuracy, and level
    of detail.

Assign each assistant an overall score on a scale of 1 to 10,
   where a higher score reflects better performance.

Please provide a single line output with only two values,
   representing the scores for Assistant 1 and Assistant 2,
   respectively, separated by a space.

Assistant:
```