Adding a new dimension to Large Language Scaling Laws and it's implications for the edge

Preslav Aleksandrov, Meghdad Kurmanji, Nicholas Lane

Abstract. We present the Autoregressive Block-Based Iterative Encoder (`AbbIE`), a novel recursive generalization of the encoder-only Transformer architecture designed for efficient and scalable deployment in resource-constrained environments. `AbbIE` introduces a dynamic, compute-scalable inference mechanism that enables performance tuning based on task complexity, making it well-suited for edge scenarios where computational budgets are variable or limited. At training time, `AbbIE` achieves lower perplexity than standard Transformer baselines and demonstrates strong generalization to arbitrary iteration lengths. Unlike other iterative or latent-space reasoning models, `AbbIE` requires no specialized data or training protocols. Crucially, it achieves superior parameter and token efficiency by trading memory for compute, addressing a core bottleneck in scaling large language models (LLMs). This approach offers up to an **18% gain in zero-shot in-context learning** and a **5% improvement in perplexity** over traditional methods. `AbbIE` matches baseline performance at a single iteration and gracefully scales with more iterations at inference, outperforming both `Std` and `Depth` across a range of ICL and language modeling benchmarks. These results validate `AbbIE` as a practical and flexible alternative to conventional Transformers. **Impact.** Mobile and embedded devices are fundamentally constrained by memory bandwidth and capacity, making the deployment

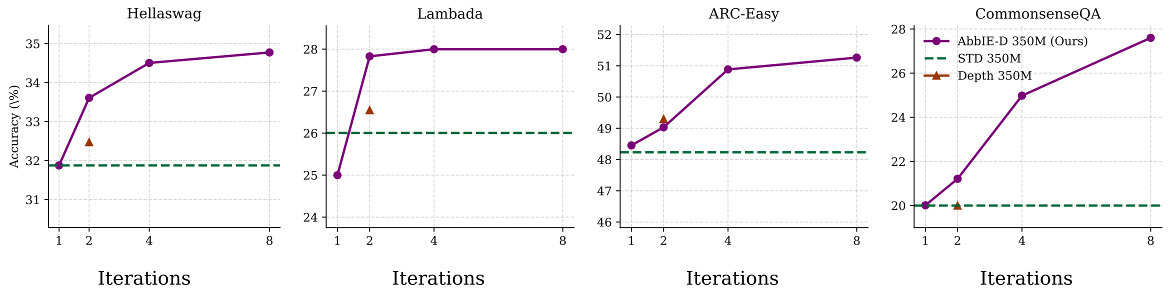

Figure 1: ICL task performance across 4 benchmarks (HellaSwag, LAMBADA, ARC-Easy, CommonsenseQA) with increasing inference iterations. `AbbIE-D` consistently outperforms `Std` and `Depth`.

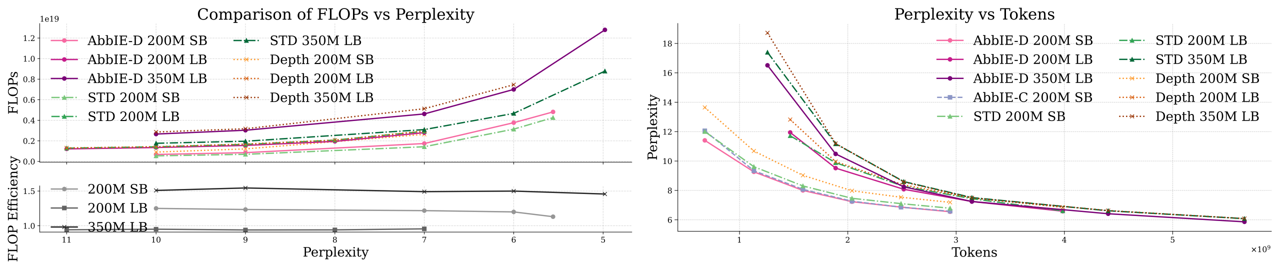

Figure 2: **Left: FLOPs vs. perplexity.** `AbbIE` requires more compute during training than `Std` and `Depth` (Geiping et al.), but its compute efficiency improves over time. The negative slope of FLOP efficiency indicates that longer training runs will eventually match or surpass baseline efficiency. **Right: Perplexity vs. token count.** `AbbIE` adheres to standard scaling laws while consistently outperforming `Std` and `Depth` in perplexity, demonstrating higher sample efficiency.

of large language models (LLMs) challenging despite improvements in compute throughput [1]. As Moore's Law slows and memory improvements plateau, these constraints are becoming a critical bottleneck for on-device intelligence [2]. `AbbIE` enables improved performance by only scaling test-time compute, thus adding a new dimension to Large Language Scaling Laws. In Fig. 1, we show that scaling test-time compute leads to significant improvements in ICL task performance. Notably, `AbbIE` is the only method that scores above the random baseline on the CommonsenseQA task, further demonstrating the value of scalable inference. Unlike other recurrence-based models such as `Depth` [3], `AbbIE` matches the performance of the standard transformer (`Std`) [4] at just one iteration. This makes `AbbIE` viable even in settings where inference-time scaling is not available or affordable. As shown in Fig. 2, `AbbIE` requires more training compute than `Std`; however, the FLOP efficiency—defined as the ratio of the training FLOPs required by `Std` to those required by `AbbIE` —has a negative slope. This indicates that with long training runs (typical in modern pipelines), `AbbIE` approaches or surpasses `Std` in training efficiency. Moreover, `AbbIE` follows standard scaling laws in perplexity as token count increases, confirming both its stability and its compatibility with existing pretraining setups. Crucially, recurrent methods like `AbbIE` benefit mobile and edge devices more than any other platform. These devices are strictly memory-bound and unable to accommodate larger models, but can often afford additional compute via longer inference runtimes. AbbIE transforms this limitation into an advantage, allowing smaller devices to achieve higher model quality without changing hardware. This unlocks entirely new capabilities—such as on-device reasoning, local assistants, and robust offline NLP—and accelerates the timeline for mobile LLM deployment by at least several years compared to what would otherwise be possible through hardware improvements alone.

References

[1] Deepak Kumar and Udit Satija. On-chip memory optimization for deep learning inference on resource-deficient devices. *IEEE Transactions on Circuits and Systems for Artificial Intelligence*, 2(1):14–24, 2025. doi: 10.1109/TCASAI.2024.3523858.

[2] Luis Ceze, Mark D. Hill, and Thomas F. Wenisch. Arch2030: A vision of computer architecture research over the next 15 years, 2016. URL `https://arxiv.org/abs/1612.03182`.

[3] Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025. URL `https://arxiv.org/abs/2502.05171`.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL `https://arxiv.org/abs/1706.03762`.