Appendix

In this appendix, each section provides the following:

- **Section A (Technical Appendix):** detailed hardware and software configurations used in our experiments, including system specifications and common settings.
- **Section B (Parallel Training Methods):** comprehensive background on major distributed training paradigms (Data, Pipeline, and Tensor Parallelism).
- **Section C (Additional Motivation Analyses):** extended motivation analyses across different model scales, datasets, and architectures (GPT-2, ViT-B, LLaMA2-7B, CodeLLaMA-34B, and LLaMA3.2-11B-Vision), confirming (1) the feasibility of bypassing per-block MHA outputs, and (2) the pivotal role of the first MHA output across transformer variants and domains.
- **Section D (Additional Evaluation Results and Analyses):** further ablation studies, analyses on information-dilution mitigation, and inference acceleration results demonstrating how FAL improves both training and inference efficiency while preserving model stability and quality through effective reuse of the first attention output.
- **Section E (Evaluation of Generalizability to Transformer Variants):** evaluation of FAL and FAL+ on diverse transformer variants (e.g., GQA-based, MoE-based, and ViT architectures) to confirm adaptability across attention mechanisms and modalities.

A Technical Appendix

A.1 Hardware & Software

We performed the experiments using PyTorch and Colossal-AI on our server and a public cloud service.

Common Settings

- Version of PyTorch: 2.2.2
- Version of CUDA: 12.3
- Version of Colossal-AI: 0.4.0

System-1

- Operating system: Ubuntu 20.04.6
- CPU: AMD EPYC 7542 32-Core
- GPU: NVIDIA RTX 3090 24GB X 4
- Interconnect: PCIe Gen4 x16 (64GB/s)

System-2

- Operating system: Ubuntu 20.04.6
- CPU: AMD Ryzen Threadripper 3970X 32-Core
- GPU: NVIDIA RTX 4090 24GB X 2
- Interconnect: PCIe Gen4 x16 (64GB/s)

System-3

- Operating system: Ubuntu 20.04.6
- CPU: AMD Ryzen Threadripper 3970X 32-Core
- GPU: NVIDIA RTX A6000 48GB X 2
- Interconnect: PCIe Gen4 x16 (64GB/s)

System-4 (Public Cloud)

- Operating system: CentOS 7.9
- CPU: Intel Xeon Emerald Rapids (Platinum 8558) / 2.10GHz (48-core) / 2 socket
- GPU: NVIDIA H200
- Interconnect: NVIDIA NVLink (900GB/s)

Figure 1 (d)

- System: 4
- GPU#: 8
- Model: GPT-2 774M
- Sequence Length: 1024
- Batchsize: 128

Figure 3 (a), (b), Figure 4 (a), (b) Motivation analyses are done with pretrained GPT-2 model on four different text datasets – WikiText-2 [18], PTB [19], BookCorpus [20], and CC-News [21].

- System: 1
- Model: GPT-2 117M (pretrained: openai-community/gpt2-large [59] from Hugging Face)
- Max Sequence Length: 1024

Figure 6 We employ FlashAttention [41] and mixed-precision training [60] in all experiments to maximize tensor core utilization and overall training efficiency. We benchmark using the largest batch size (in powers of two) supported under each training setting.

- System: 4, 1
- GPU#: [2, 4, 8]
- Model: [GPT-2 774M, 1.5B, 2.5B, 8.3B]
- Batchsize: System4: 64 (774M, 2GPU), 16 (1.5B, 2GPU), 16 (2.5B, 2GPU), 8 (8.3B, 2GPU), 64 (774M, 4GPU), 32 (1.5B, 4GPU), 32 (2.5B, 4GPU), 16 (8.3B, 4GPU), 128 (774M, 8GPU), 64 (1.5B, 8GPU), 64 (2.5B, 8GPU), 32 (8.3B, 8GPU) System1: 4 (774M, 2GPU), 2 (1.5B, 2GPU), 8 (774M, 4GPU), 4 (1.5B, 4GPU)
- Sequence Length: 1024

Figure 7

- System: 1
- GPU#: 2
- Model: GPT-2 774M
- Total Batchsize: 32 (used gradient accumulation)
- Sequence Length: 1024
- Epochs: 1
- Learning rate: 0.0001
- Weight decay: 0.001
- Dropout: 0

Figure 8 (a) We compare GPT-2 and FAL under both the minimum (1) and maximum batch sizes for each GPU setting. We also evaluate the speedup with and without acceleration techniques — specifically, FlashAttention — on each GPU configuration.

- System: [1, 2, 3]
- GPU#: [1]
- Model: GPT-2 774M
- Sequence Length: 1024

Figure 8 (b) We use NVIDIA Nsight Systems [61] to profile GPU performance, including SM utilization, warp occupancy, tensor core usage, and memory bandwidth.

- System: 1
- GPU#: 1
- Model: GPT-2 774M
- Sequence Length: 1024

Table 1 We train each architecture on OpenWebText [38], an open-source replication of the WebText dataset originally used to train GPT-2. The dataset comprises approximately 41.7 GB of text, corresponding to 4 billion tokens. Given our limited computational resources, we use a compute-efficient batch size of 32, which has been shown to be sufficient for stable hyperparameter transfer in μP-based training [62, 63]. To evaluate language understanding performance, we report zero-shot results on the SuperGLUE benchmark [8], which includes BoolQ [42], CB [43], COPA [44], MultiRC [45], ReCoRD [46], RTE [47], WiC [48], and WSC [49]. No finetuning or additional training was performed on any task. CB and ReCoRD are evaluated using F1 score, while the remaining tasks use accuracy.

- System: 1
- Epochs: 1
- GPU#: 4
- Model: GPT-2 774M, 1.5B
- Parallel setting: 2TP/2DP
- Total batchsize: 32 (used gradient accumulation)
- Sequence Length: 1024
- Learning rate: 0.0001
- Weight decay: 0.001
- clip-grad-norm: 1
- embd-pdrop: 0.1

Figure 9 Motivated by Cramming [50], which demonstrated that scaling laws [1] can be observed even under small-scale, fast-training settings, we compare FAL and FAL+ to the standard pre-LN architecture by stacking transformer blocks with depths of 36 (equivalent to GPT-2 774M), 48 (GPT-2 1.5B), and 60. To evaluate scalability, we stack the transformer blocks of a pre-LN masked language model architecture based on BERT-Large [64].

Training settings follow the original Cramming paper, including a budget-based one-cycle learning rate scheduler [65] and batch size ramp-up for 24 hours on 1 GPU. Models of the same scale are trained under identical system configurations.

- System: 1, 2
- GPU#: 1
- Hidden size: 1024
- Intermed size: 4096
- Nonlinear: GELU
- Max sequence length: 128
- Number of transformer block: 36, 48, 60
- Final Batchsize: 8192 (used gradient accumulation)
- Learning rate: 0.0001
- Weight decay: 0.001
- Clip-grad-norm: 1
- Hidden dropout probability: 0.1
- Attention dropout probability: 0.1
- Embedding dropout probability: 0.1

B Parallel Training Methods

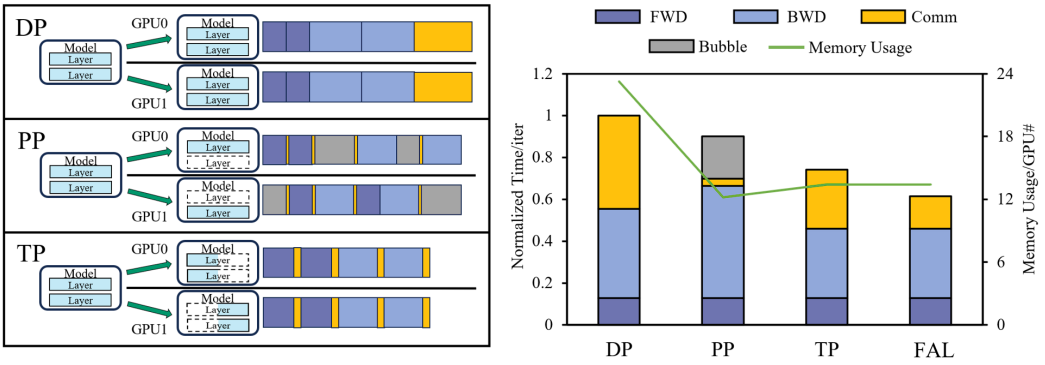


Figure 10: Comparison of Data parallelism(DP), Pipeline Parallelism(PP), Tensor Parallelism(TP). We stack GPT-2 blocks until DP can handle them (number of blocks: 42), using OpenWebText (max sequence length: 1024) for comparison on two NVIDIA RTX 3090 GPUs connected via PCIe.

To address challenges of training the large-scale models, employing multiple distributed GPUs along with parallel training methods [66, 15, 4, 67, 68, 16, 69, 70] has become a common practice. These methods encompass a range of parallelism paradigms, including Data Parallelism (DP), Pipeline Parallelism (PP), and Tensor Parallelism (TP). DP duplicates the entire model across multiple distributed GPUs. Each GPU then trains the duplicated model with different data batches and synchronizes the trained gradients for unified updates [15, 69, 71]. While DP is effective for smaller models, it results in significant memory and communication overhead for larger models as each GPU needs to retain the model duplicates and synchronize the large amount of parameters.

PP and TP have been proposed to address the scalability issue. PP [72, 16] partitions layers of a model across the GPUs. A batch is split into smaller microbatches, and training of different layers is pipelined with the microbatches across the GPUs. However, to ensure consistent weight updates for a particular batch (without being affected by the weight updates from the other batches), GPUs need to synchronize the weight updates of microbatches for every batch. This introduces pipeline bubbles where some GPUs (which process former microbatches of a batch) need to wait for the weight updates from the other GPUs (which process latter microbatches of the batch) delaying the entire training process [72]. TP [4], on the other hand, distributes matrix multiplications within each transformer layer (i.e., MHA and MLP) across the GPUs. Each GPU handles a portion of the matrix multiplications in parallel, without having model duplicates or pipeline bubbles, making it highly effective for large-scale models.

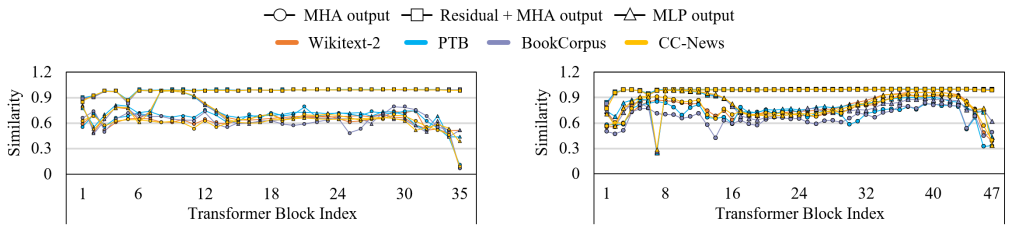
Although TP is receiving much attention recently for its scalability benefit to further enhance memory efficiency and latency with large-scale models [5, 6, 55], its further efficiency is still limited by the communication overhead. Fig. 10 illustrates the train time and memory usage comparison between DP, PP and TP. While TP shows the fastest training time among three methods as it does not require communication of full parameters and pipeline bubbles, frequent communication between GPUs is still required to process synchronized and complete intermediate activations and gradients from MHA and MLP. As a result, a large portion of the training time is devoted to these communications (37.9% of the training time), resulting in a notable decrease in training efficiency.

To further enhance the potential of TP for fast large model training, we propose FAL, which eliminates intra-block data communication by harnessing the output of the MHA in the first (i.e., bottom-most) transformer block for the MLP's inputs, instead of using the output of the MHA in the same block.

C Additional Motivation Analyses

C.1 Motivation Analyses in Different Scale

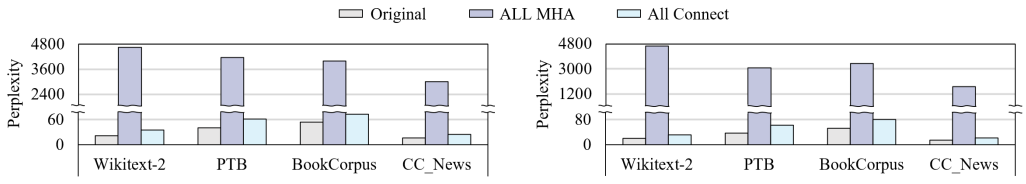
Fig. 11 shows the CKA similarity scores for MHA outputs, MLP inputs (*Residual + MHA*), and MLP outputs across adjacent blocks (conducted on GPT-2 774M and 1.5B). As shown in Fig. 11, even in the case of larger models, the MLP input remains highly similar despite significantly changing MHA



(a) GPT2-L similarity analysis (b) GPT2-XL similarity analysis

Figure 11: CKA-based similarity analysis of GPT-2 774M and 1.5B. across successive Transformer blocks. The x-axis shows the block index, and the y-axis shows the similarity (CKA) between consecutive MHA output, Residual + MHA output (i.e., the MLP input), and the MLP input

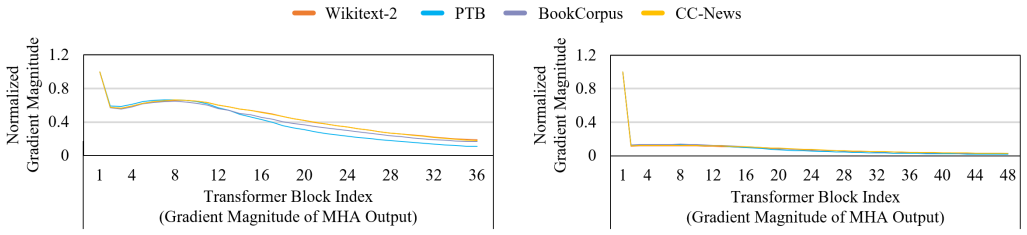
output. This demonstrates that, regardless of the model size, MLP may not require the most recent MHA output (i.e., the output of the MHA within the same block).



(a) GPT2-L connection ablation (b) GPT2-XL connection ablation

Figure 12: Connection ablation results measured by perplexity with GPT-2 774M and 1.5B. “Original” denotes the unaltered model. "All MHA" removes every MHA layer. “All Connect” removes every direct MHA-MLP connection.

Fig. 12 illustrates two scenarios: removing all MHAs (*All MHA*) versus removing all MHA-MLP connections (*All Connect*), measured by perplexity on GPT-2 774M and 1.5B. As expected, removing *All MHA* severely degrades model quality. In contrast, removing *All Connect* recovers a significant portion of the lost performance compared to removing the entire MHAs, and this recovery becomes even more pronounced with larger models (though still not fully reaching the original performance). This suggests that bypassing MHA is a better option for larger models.

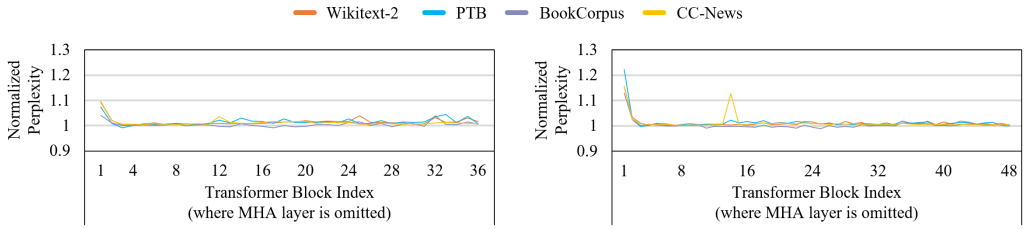


(a) GPT2-L gradient analysis (b) GPT2-XL gradient analysis

Figure 13: Normalized gradient magnitude of the MHA outputs across Transformer blocks in GPT-2 774M and 1.5B for different datasets. the x-axis represents the block index.

Fig. 13 shows the gradient magnitude of each MHA output on larger scale (774M and 1.5B). As shown in Fig. 13, even in the case of larger models, first MHA output consistently exhibits the highest gradient magnitude. This confirms our finding that perturbations in the earliest attention result have a disproportionately large impact on final predictions, regardless of model size.

Fig. 14 shows the perplexity after omitting the MHA from individual transformer blocks with 774M and 1.5 scale GPT-2. As shown in Fig. 14, removing the first attention causes a far larger perplexity increase than removing later layers, verifying the crucial role of the first attention in language modeling. These findings align with the well-known psychological phenomenon of the primacy effect [26], commonly summarized as “first impressions matter.” The primacy effect of the first



(a) GPT2-L layer ablation (b) GPT2-XL layer ablation

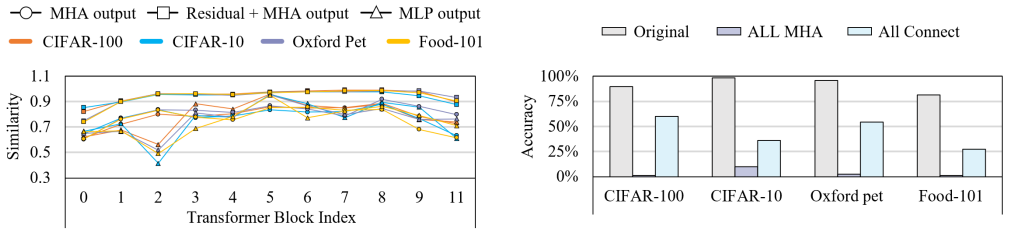
Figure 14: Layer ablation results measured by perplexity with GPT-2 774M and 1.5B. the x-axis indicates the index of the transformer block from which the MHA is omitted.

attention is not limited to a specific model architecture — previous works also identified the prominent impact of the first attention layer across various attention mechanisms and tasks [27, 28, 29].

C.2 Motivation Analyses with Different Task & Model Architecture

Beyond scaling analyses, we further validate our motivation across different tasks and model architectures, including ViT-B (86.6M), LLaMA2-7B, CodeLLaMA-34B, and LLaMA3.2-11B-Vision.

C.2.1 ViT



(a) ViT-B similarity analysis (b) ViT-B connection ablation

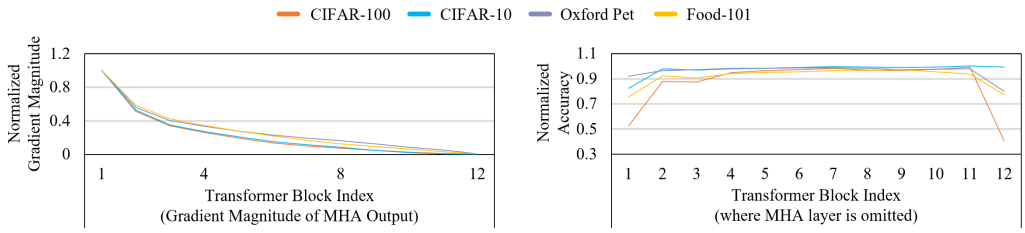
Figure 15: (a) CKA-based similarity analysis of ViT. across successive Transformer blocks. The x-axis shows the block index, and the y-axis shows the similarity (CKA) between consecutive MHA output, Residual + MHA output (i.e., the MLP input), and the MLP input (b) Connection ablation results measured by accuracy. “Original” denotes the unaltered model. "All MHA" removes every MHA layer. “All Connect” removes every direct MHA-MLP connection.

Fig. 15 (a) shows the CKA similarity scores for MHA outputs, MLP inputs (*Residual + MHA*), and MLP outputs across adjacent blocks (conducted on ViT-B). As shown in the Figure, even in the case of vision task, the MLP input remains highly similar despite the MHA output changing significantly. This confirms our findings that MLP may not always require the most recent MHA output (i.e., the output of the MHA within the same block), regardless of the domain.

Fig. 15 (b) illustrates two scenarios: removing all MHAs (*All MHA*) versus removing all MHA-MLP connections (*All Connect*), measured by accuracy on ViT-B. As expected, removing *All MHA* severely degrades model quality. In contrast, removing *All Connect* recovers a large portion of the lost performance compared to removing the entire MHAs, however this recovery becomes smaller with smaller vision models. This suggests that simply bypassing MHA on small-scale vision models may harm their accuracy.

Fig. 16 (a) shows the normalized gradient magnitude of each MHA output in the encoder-based ViT-B [73]. Although the effect is less pronounced than in language models, the first MHA output still consistently exhibits the highest gradient magnitude.

Fig. 16 (b) shows the accuracy after omitting the MHA from individual transformer blocks with ViT-B. As shown in the Figure, removing the first attention causes a far larger accuracy drop than removing later layers (except the last), verifying the crucial role of the first attention in language modeling. The



(a) ViT-B gradient analysis

(b) ViT-B layer ablation

Figure 16: (a) Normalized gradient magnitude of the MHA outputs across Transformer blocks with ViT for different datasets. The x-axis represents the block index. (b) Layer ablation results measured by accuracy with ViT. The x-axis indicates the index of the transformer block from which the MHA is omitted.

importance of the final attention likely stems from the fact that ViT uses a domain-specific classifier based on the output of the final Transformer block [73].

C.2.2 LLaMA

To further validate the generality of our findings across modalities and scales, we extend our motivational analyses to larger models: LLaMA2-7B (language), CodeLLaMA-34B (code generation), and LLaMA3.2-11B-Vision (multilingual vision-language). We use WikiText for LLaMA2-7B, The Stack dataset [74] for CodeLLaMA-34B, and the COCO captioning dataset for LLaMA3.2-11B-Vision.

Table 3: Similarity Analysis Results (Metric: CKA Similarity ± std)

Activation	LLaMA2-7B	CodeLLaMA-34B	LLaMA3.2-11B-Vision
Attn Out	0.60 ±0.14	0.66 ±0.13	0.84 ±0.10
MLP In	0.98 ±0.03	0.99 ±0.03	0.98 ±0.02
MLP Out	0.70 ±0.23	0.80 ±0.18	0.80 ±0.20

Table 3 reports the similarity analysis results, measured by CKA similarity (mean ± std). Across all three models, we observe that the MLP inputs remain highly similar to each other (LLaMA2-7B: 0.98, CodeLLaMA-34B: 0.99, LLaMA3.2-11B-Vision: 0.98), whereas the attention/MLP outputs vary more significantly (0.60–0.84). This analysis shows that the residual path already accumulates sufficient attention signals, making the MLP input less sensitive to the most recent MHA output. The same trend holds even for larger-scale (7B–34B) and multimodal (vision–language) models, extending our earlier motivation analyses 3.1 and supporting the validity of reconfiguring MHA–MLP connections in FAL.

Table 4: Layer Ablation vs. Connection Ablation (Metric: Validation Perplexity)

Setting	LLaMA2-7B	CodeLLaMA-34B	LLaMA3.2-11B-Vision
Original	7.39	1.80	25.12
Remove Layer	5339.99	1484.52	3995.84
Remove Connection	892.12	37.06	504.72

Table 4 reports validation perplexity under two ablations: (1) removing the entire MHA layer, and (2) removing only the direct MHA–MLP connection. Across all three models, removing entire layers leads to catastrophic degradation (e.g., PPL $>$1000), while removing only the connections recovers a substantial portion of performance. Although the connection–removed models still fall short of the original baseline, they consistently perform far better than the layer–removed ones, and the recovery effect becomes more pronounced at larger scales. These results show that reconfiguring MHA–MLP connections causes far smaller degradation than removing entire layers, yet still falls short of fully matching the original performance. A stable alternative signal is needed, which is exactly what FAL provides by reusing the impactful first attention.

Table 5: Gradient Analysis (Metric: Gradient L1 Norm)

Block	LLaMA2-7B	CodeLLaMA-34B
1st	505.99	321.65
2-End avg ±std	85.90 ±102.00	46.24 ±60.39
Ratio (1st/avg)	5.9×	7.0×

Table 5 reports the gradient analysis of MHA outputs in LLaMA2-7B and CodeLLaMA-34B. The first attention block shows a much larger gradient magnitude (505.99 and 321.65) compared to the average of later blocks (85.90 and 46.24, respectively). On average, the first block gradients are $5.9\times$ and $7.0\times$ larger than subsequent ones. These results highlight that the first attention exerts a disproportionately strong influence on final predictions. Thus, reusing this impactful signal in FAL remains well-justified across larger models and diverse tasks.

Table 6: Layer Ablation per Block (Metric: Validation Perplexity)

Block	LLaMA2-7B	CodeLLaMA-34B	LLaMA3.2-11B-Vision
Original	7.39	1.80	25.12
1st	34.37	4.56	40.94
2-End avg ±std	4.37 ±1.37	1.81 ±0.01	24.75 ±1.25
Ratio (1st/avg)	7.9×	2.5×	1.7×

Table 6 reports the effect of ablating individual attention layers in LLaMA2-7B, CodeLLaMA-34B, and LLaMA3.2-11B-Vision. Removing the first attention block causes a far larger degradation in validation perplexity (e.g., 34.37 vs. 7.39 for LLaMA2-7B, 4.56 vs. 1.80 for CodeLLaMA-34B, and 40.94 vs. 25.12 for LLaMA3.2-11B-Vision) compared to ablating later layers, whose impact remains relatively small. The impact of removing the first attention is consistently larger than that of later layers, up to $7.9\times$ in LLaMA2-7B. These results show that the first attention is disproportionately important across scales and modalities. Its removal uniquely destabilizes the model, whereas later attentions contribute far less. This further supports that reusing the first attention in FAL remains well-justified across larger and more diverse models.

D Additional Evaluation Results and Analyses

D.1 Ablation study

We conduct two ablations to verify the effectiveness of reusing the first attention output in FAL and FAL+. Table 7 shows the validation perplexity and training time.

Table 7: Comparison of validation perplexity and training time using Openwebtext dataset (GPT-2 774M)

Model	Perplexity	Training time
GPT-2 774M (Baseline)	17.75	13.2 days
FAL	**17.55**	8.6 days
FAL+	**17.24**	13.2 days
Ablation1	21.34	13.2 days
Ablation2	17.98	8.6 days

Ablation1 (leveraging latest attention). Simple addition of normalized outputs within a block degrades the quality, calling for the necessity of the first attention signal. We apply the same LN + LN structure from FAL using the latest attention output instead of the first attention.

$$X_i + \mathrm{MHA}_i(\mathrm{LN}(X_i)) + \mathrm{MLP}_i(\mathrm{LN}(X_i) + \mathrm{LN}(\mathrm{MHA}_i(\mathrm{LN}(X_i)))) \tag{3}$$

This leads to a significantly higher perplexity (21.34) compared to the baseline (17.75), suggesting that the latest attention output does not provide a stable or beneficial signal for MLP inputs under this reconfiguration.

Ablation2 (removing connection without first). Retaining only the first MHA-MLP connection is not sufficient to retain the first attention signal. We remove all MHA-MLP connections except for the first one.

$$\begin{cases} X_1 + \mathrm{MHA}_1(\mathrm{LN}(X_1)) + \mathrm{MLP}_1\big(\mathrm{LN}(X_1) + \mathrm{MHA}_1(\mathrm{LN}(X_1))\big), & \text{if } i = 1, \\ X_i + \mathrm{MHA}_i(\mathrm{LN}(X_i)) + \mathrm{MLP}_i\big(\mathrm{LN}(X_i)\big), & \text{otherwise.} \end{cases} \tag{4}$$

Although perplexity (17.98) remains comparable to the baseline, it still degrades model quality compared to FAL. This implies that merely connecting the first attention once is not sufficient to maintain the overall performance.

Comparison to FAL and FAL+. FAL reuses the first MHA output in each block. This reuse, aided by an additional LN to balance the first and residual signals, improves perplexity (17.55) and reduces training time (8.6 days). FAL+ augments the original connections with the first attention, achieving an even lower perplexity (17.24) but at a training time similar to GPT-2. These results confirm that properly integrating the first attention output is crucial for both efficiency and model quality.

GPT-2.

$$X_i + \mathrm{MHA}_i(\mathrm{LN}(X_i)) + \mathrm{MLP}_i(\mathrm{LN}(X_i + \mathrm{MHA}_i(\mathrm{LN}(X_i)))) \tag{5}$$

FAL.

$$X_i + \mathrm{MHA}_i(\mathrm{LN}(X_i)) + \mathrm{MLP}_i(\mathrm{LN}(X_i) + \mathrm{LN}(\mathrm{MHA}_1(\mathrm{LN}(X_1)))) \tag{6}$$

FAL+.

$$X_{i+1} = \begin{cases} X_1 + \mathrm{MHA}_1(\mathrm{LN}(X_1)) \\ \quad +\mathrm{MLP}_1\big(\mathrm{LN}(X_1) + \mathrm{MHA}_1(\mathrm{LN}(X_1))\big), & \text{if } i = 1, \\ X_i + \mathrm{MHA}_i(\mathrm{LN}(X_i)) \\ \quad +\mathrm{MLP}_i\Big(\mathrm{LN}(X_i + \mathrm{MHA}_i(\mathrm{LN}(X_i))) + \mathrm{LN}(\mathrm{MHA}_1(\mathrm{LN}(X_1)))\Big), & \text{otherwise.} \end{cases} \tag{7}$$

Ablation with Other Layers. To further verify the benefit of reusing the impactful first attention, we compare FAL+ with variants that reuse the output of other MHA layers (2nd, 3rd, and so on). We adopt FAL+ in a 48-block configuration (see Fig. 9) and train for 500k steps under the same one-cycle schedule with a batch-size ramp-up to 8,192, ingesting 1.02B tokens regardless of hardware speed.

Fig. 17 shows the MLM loss comparison of FAL+, which reuses the first attention, against variants that reuse later-layer attentions. As shown in Fig. 17, reusing later-layer attentions consistently underperforms compared to the first attention. This implies that reusing weaker or less dominant signals is not as effective as leveraging the impactful first attention, confirming that the first attention provides a uniquely stable and beneficial feature for reconfiguration.

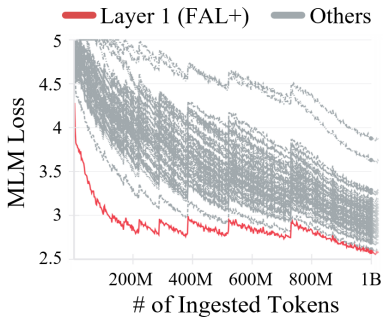


Figure 17: Loss comparison using other MHA's output

D.2 How FAL Mitigates Information Dilution in Deep Transformers

In a standard Pre-LN Transformer, each block processes only its immediate predecessor's output, analogous to unrolling an RNN over depth. As depth grows, early-layer signals must traverse many transformations and risk dilution or loss—much like long RNN sequences forget initial states. FAL breaks pure sequential dependence by feeding the first-layer attention output directly into every later block, akin to self-attention's ability to attend globally. Concretely, we normalize the first MHA output once and then add it to each block's input: $MLP_{input_i} = LN(X_i) + LN(MHA_1(LN(X_1)))$

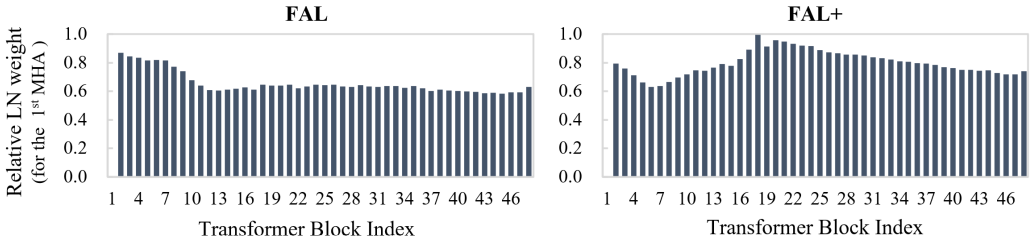


Figure 18: Relative LN scaling parameters connected to the first MHA output, normalized to average LN scaling across layers.

Residual connections alone cannot prevent the first signal from fading: activation variance accumulates layer by layer [9], eventually washing out early-layer information — much like how self-attention's $O(n^2)$ interactions dilute key dependencies over very long sequences [75]. FAL sidesteps this pitfall by reusing normalized first-attention tensor at each block, reinforcing the most salient initial context without extra overhead — echoing cognitive insights that rethinking a first impression can lead to better decisions in deep reasoning [30].

Figure 18 shows the average LN scaling parameters (γ) for the terms connected to the first MHA output, normalized by the average LN scaling across layers. Across depth, both FAL and FAL+ consistently assign non-negligible weights (roughly equivalent to a 0.58:1–1:1 ratio compared to the current block input), indicating that later blocks actively and adaptively incorporate the first-attention signal after training. This dynamic weighting suggests that FAL can alleviate information dilution by adaptively reinforcing the initial signal across depth.

D.3 Inference Acceleration of FAL

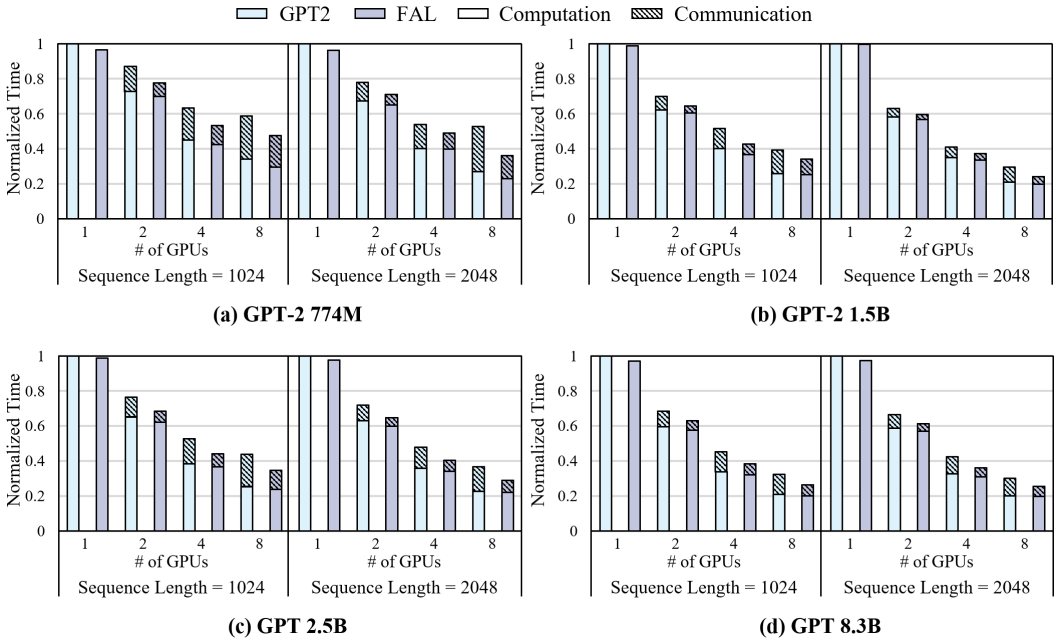


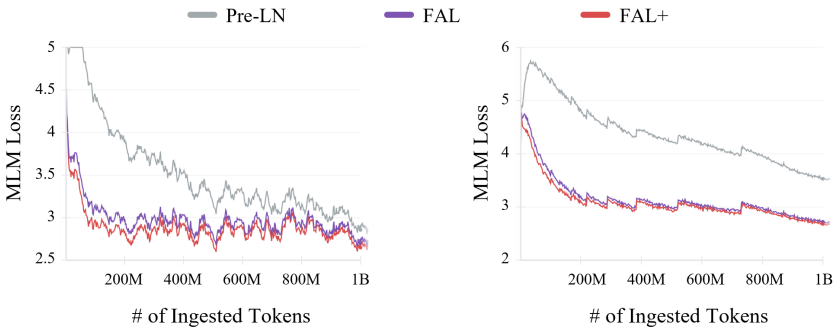
Figure 19: Normalized Multi-GPU Inference Time of GPT-2 and FAL.

TP is often used to accelerate the inference execution, as it can accelerate per-request latency unlike other two parallelism methods (i.e., Data Parallelism and Pipelined Parallelism) [7, 76]. In order to evaluate the inference acceleration that can be further achieved by FAL in TP, we measure the forward step time without gradient calculation which is aligned with the Time To First Token (TTFT) in the inference execution.

Fig. 19 shows the normalized forward step time of GPT-2 and FAL on a multi-GPU server with NVLink (System 4), across model sizes ranging from 774M to 8.3B and sequence lengths of 1024 and 2048. As shown in the figure, tensor parallelism (TP) reduces per-request inference time by utilizing multiple GPUs. With 8 GPUs, TP achieves an average speedup of 56.5% for a sequence length of 1024 (up to 67.6%), and 62.8% for a sequence length of 2048 (up to 70.5%). However, as the number of GPUs increases, the degree of acceleration is limited by the communication overhead between GPUs. In such cases, FAL improves inference performance over GPT-2 by (1) significantly reducing inter-GPU communication and (2) increasing intra-GPU parallelism. Across configurations from 1 to 8 GPUs, FAL reduces inference time by 11.1% on average (up to 31.6%).

E Evaluation of Generalizability to Transformer Variants

E.1 Loss comparison using variants of Multi Head Attention



(a) FAL with GQA **(b) FAL with MoE-Attention**

Figure 20: Loss comparison across different attention mechanisms: Grouped Query Attention (GQA), and MoE-based Attention (MoE-Attention).

FAL and FAL+ can be applied to various Pre-LN based transformer variants, such as LLaMa (with Grouped Query Attention (GQA)), and Switch Transformer (with Mixture of Experts (MoE)) to improve the efficiency and model quality. To evaluate the generalizability of FAL and FAL+ to these variants, we measure token ingestion efficiency before and after applying FAL and FAL+ to GQA-based [77] and MoE-based attention models [58]. We adopt FAL and FAL+ in a 48-block configuration (see Fig. 9). To ensure consistent scheduling across hardware setups, we train for 500,000 steps using a step-based one-cycle learning rate scheduler. The batch size is ramped up to a maximum of 8,192 by step 300,000, resulting in a total of 1.02B tokens ingested regardless of hardware speed.

Fig. 20 (a) shows the comparison of FAL and FAL+ when applied to GQA. Each attention layer uses GQA with two groups. This setup differs from standard Multi Head Attention (MHA) primarily in the key/value projections, and becomes equivalent to MHA when the number of groups equals the number of heads. The results closely resemble those observed with standard MHA. The loss gap between the proposed architectures and the baseline remains consistent, demonstrating that the gains from FAL and FAL+ extend robustly to this efficient attention variant.

Fig. 20 (b) shows the comparison of FAL and FAL+ when applied to MoE-based Attention. We follow the configuration of MoE-based Attention introduced in the Switch Transformer, which was found to be unstable and thus not included in the final architecture. Each expert in the MoE attention has its own query projection and tied key/value projections. One of two experts is activated per attention layer. Unlike the instability observed when using Switch layers in attention, FAL and FAL+ do not suffer from gradient instability. The loss gap between the proposed architectures and the baseline remains consistent, demonstrating that the gains from FAL and FAL+ extend robustly to sparsely activated MoE attention mechanisms.

Table 8: Comparison of validation accuracy using ImageNet dataset (ViT-B 86.6M)

Dataset	ViT (Baseline)	**FAL**	**FAL+**
ImageNet	79.06%	78.76%	**79.20**%

E.2 Accuracy comparison using other kinds of Task

We also evaluate FAL and FAL+ on the Vision Transformer (ViT-B 86.6M) architecture using the ImageNet dataset. As shown in Table 8, FAL slightly reduces accuracy compared to the baseline (79.06% vs. 78.76%), whereas FAL+ achieves a higher accuracy (79.20%). This result indicates that reusing the first attention output can also benefit vision tasks, particularly when combined with the original attention connections as in FAL+.