
FFN Fusion: Rethinking Sequential Computation in Large Language Models

Akhiad Bercovich*

Mohammad Dabbah*

Omri Puny*

Ido Galil

Amnon Geifman

Yonatan Geifman

Izhak Golan

Ehud Karpas

Itay Levy

Zach Moshe

Najeeb Nabwani

Tomer Ronen

Itamar Schen

Elad Segal

Ido Shahaf

Oren Tropp

Ran Zilberstein

Ran El-Yaniv

NVIDIA, {abercovich, mdabbah, opuny, relyaniv}@nvidia.com

Abstract

We introduce *FFN Fusion*, an architectural optimization technique that reduces sequential computation in large language models by identifying and exploiting natural opportunities for parallelization. Our key insight is that sequences of Feed-Forward Network (FFN) layers, particularly those remaining after the removal of specific attention layers, can often be parallelized with minimal accuracy impact. We develop a principled methodology for identifying and fusing such sequences, transforming them into parallel operations that significantly reduce inference latency while preserving model behavior. Applying these techniques to Llama-3.1-405B-Instruct, we create Llama-Nemotron-Ultra-253B-Base (Ultra-253B-Base), an efficient model that achieves a $1.71\times$ speedup in inference latency and $35\times$ lower per token cost while maintaining strong performance across benchmarks. Most intriguingly, we find that even full transformer blocks containing both attention and FFN layers can sometimes be parallelized, suggesting new directions for neural architecture design.

1 Introduction

Large language models (LLMs) have emerged as one of the most transformative technologies of our time, revolutionizing how we approach artificial intelligence and computation. From powering sophisticated virtual assistants [16] to enabling breakthrough scientific research [1, 41], these models have evolved from academic curiosities [35] into indispensable tools that are reshaping entire industries. This transformation has been driven by rapid scaling to hundreds of billions of parameters [33, 2, 9], enabling unprecedented capabilities in reasoning, generation, and complex problem-solving [5, 15]. However, this extraordinary growth in scale and capability comes at a critical cost: the computational demands of running these models have become a fundamental bottleneck. As these models push the boundaries of what is possible with artificial intelligence, their deployment costs and resource requirements severely limit their accessibility, creating an urgent need for innovations that can make their capabilities widely available.

Research in LLM runtime optimization explored diverse approaches to managing computational demands. Traditional techniques like quantization [7, 12, 8, 47]—which reduce memory footprint and accelerate inference through lower-precision arithmetic—and pruning [26, 18, 17, 28, 11]—which

removes redundant parameters—have become standard tools. A more recent innovation, Mixture-of-Experts (MoE) [38, 23], has demonstrated remarkable potential by dynamically activating only a small subset of model parameters during inference. The DeepSeek-V3 architecture [6] pushes this approach to new extremes, employing 256 expert FFN modules at each layer while activating only 8 experts per token (plus one shared expert), effectively achieving the capabilities of a much larger model while maintaining reasonable computational costs through sparse activation. However, each of these approaches faces distinct challenges: quantization encounters precision-accuracy trade-offs at small bit numbers, pruning struggles to identify significant additional redundancy without compromising accuracy performance (and pruning can be harnessed to efficiency mainly when it is structured), and MoE architectures, while efficient for single-token inference or for very large batches, do not provide optimal throughput at small and intermediate batch sizes due to under-utilization of compute resources, as discussed in Appendix E. Given the above limitations, there is an urgent need for complementary approaches that can unlock new dimensions of efficiency improvement while maintaining the simplicity and predictable scaling characteristics of dense architectures.

In this work, we introduce two major contributions that fundamentally advance the state of LLM efficiency. First, we present FFN Fusion, a novel architectural optimization technique that challenges the conventional sequential nature of transformer computation, and is illustrated in Figure 1. By identifying and exploiting patterns of computational independence in FFN layers, our approach enables parallel execution across multiple GPUs while preserving model functionality. This parallelization is particularly effective on modern GPU nodes, where tensor-parallel implementations often suffer from synchronization delays between consecutive layers. By concentrating the computation into fewer layers and reducing cross-device communication, our method significantly improves hardware utilization. Our findings reveal that substantial portions of LLM computation can be parallelized with minimal accuracy impact, complementing existing runtime optimization techniques like pruning and quantization. Second, we demonstrate the practical impact of these insights through Ultra-253B-Base, a state-of-the-art 253B parameter model derived from Llama-3.1-405B-Instruct (henceforth Llama-405B), derived using FFN Fusion and attention pruning. This model not only showcases the scalability of our approach, but also achieves remarkable efficiency gains while maintaining or exceeding its parent model’s capabilities across a comprehensive suite of benchmarks.

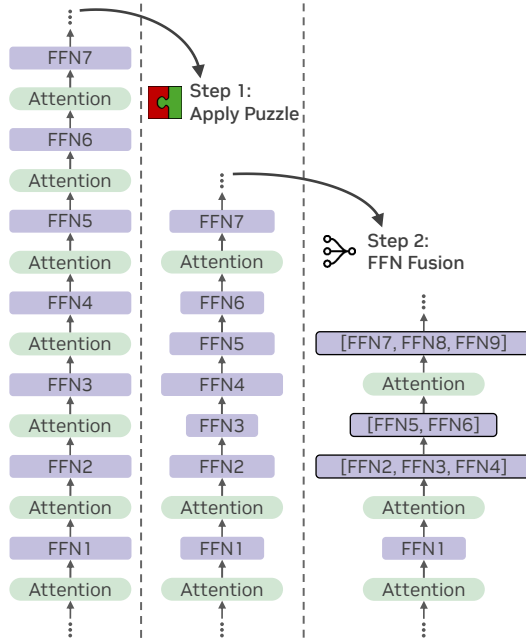


Figure 1: An overview of our FFN Fusion approach. Step 1: We apply Puzzle to partially remove FFN layers and remove entire attention layers. Step 2: We fuse consecutive FFN layers into a single wide FFN layer. Fusion is denoted using the bracket notation $[\text{FFN}^i, \dots, \text{FFN}^{i+k}]$.

The applicability of FFN Fusion rests on a fundamental insight about modern LLM architectures. Recent work has shown that LLM display structural redundancy [43, 31, 10, 49, 36, 25, 14], particularly attention mechanisms, which can be selectively removed with minimal accuracy impact [3, 19], often leaving models with extended sequences of consecutive FFN layers. We demonstrate that these FFN sequences exhibit surprisingly low inter-layer dependencies, enabling a novel transformation: multiple sequential FFN layers can be fused into a single, wider layer that enables simple parallel execution. This transformation is particularly powerful, eliminating synchronization points while allowing for more efficient hardware utilization without requiring architectural redesign (as visualized in Figure 2). Through analysis and empirical validation, we argue for the conditions under which this fusion preserves model behavior, showing that these conditions are commonly satisfied in practice, especially in larger models where the potential efficiency gains are most significant. Most surprisingly, our preliminary investigations suggest that even complete transformer blocks—containing both attention and FFN components—can sometimes be parallelized, pointing to potential new directions in neural architecture design (see Appendix B). Leveraging these insights, we develop Ultra-253B-Base, derived from Llama-405B through a combination of attention pruning and FFN Fusion. This model achieves remarkable efficiency gains while maintaining or exceeding its parent model’s capabilities:

- A $1.71\times$ speedup in inference latency and $35\times$ lower per-token cost at batch size 32.
- State-of-the-art performance on key benchmarks, including 84.92% on Arena Hard, 86.58% on HumanEval, 87.54% on MMLU Instruct, 72.25% on MMLU-Pro, and 9.19 on MT-Bench.
- Improving memory footprint with a $2\times$ reduction in kv-cache memory, and reduced parameter count from 405B to 253B.

Our key contributions are:

- The discovery that some Feed-Forward Networks in transformer architectures can often be parallelized with minimal accuracy loss, challenging the conventional wisdom that sequential processing is necessary for these layers.
- The creation of Ultra-253B-Base, a powerful 253B parameter model that matches or exceeds Llama-405B’s capabilities while offering substantially improved efficiency, to be publicly released upon acceptance.
- Preliminary evidence that even complete transformer blocks—containing both attention and FFN components—can sometimes be parallelized, opening new possibilities for architectural innovations in large language models.

2 Preliminaries

Transformer-based LLMs. The structure of LLMs is typically based on the widely used transformer architecture [42]. It is generally composed of a series of sequential blocks, denoted as $f^1, \dots, f^m$, each containing an attention layer and an FFN layer. The attention layers capture the contextual relationships between input tokens, while the FFN layers apply transformations independently to each token. Given an input $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n represent the sequence length and d the embedding dimension, a *transformer block* $f : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times d}$ is defined by the following equations:

$$g(\mathbf{X}) = \mathbf{X} + \text{Attention}(\eta_1(\mathbf{X})) \quad (1)$$

$$f(\mathbf{X}) = g(\mathbf{X}) + \text{FFN}(\eta_2(g(\mathbf{X}))), \quad (2)$$

where η_k is a token-level normalization module, defined by the equation $\eta_k(x_i) = \frac{x_i}{\|x_i\|} \odot s_k$ with $x_i \in \mathbb{R}^d$ being the i -th token, $s_k \in \mathbb{R}^d$ being a learnable scale factor, and $\odot$ is the element-wise product. The FFN layer used in the models we experiment with is the *SwiGLU* module [37], which is defined as follows:

$$\text{SwiGLU}(\mathbf{X}) = (\sigma(\mathbf{X}W_2^T) \odot \mathbf{X}W_1^T)W_3^T, \quad (3)$$

where σ is the SiLU activation [20] function, $W_1, W_2 \in \mathbb{R}^{d_h \times d_e}$ and $W_3 \in \mathbb{R}^{d_e \times d_h}$. Here, d_e represents the embedding dimension, and d_h denotes the hidden dimension of the FFN. Other components of the LLM include an embedding function at the model’s input, which maps tokens to vectorized embeddings, an additional normalization function, and a linear function at the output, which normalizes the final block outputs and projects them to the vocabulary dimension.

Puzzle. Puzzle [3] is a neural architecture search (NAS) framework that optimizes a trained LLM for inference efficiency. Starting with a pre-trained model, it prunes or reconfigures each transformer block—often reducing or removing attention layers—while preserving model quality through a distillation process. In particular, applications of Puzzle often show that many attention layers can be removed with minimal accuracy loss, thus leaving sequences of FFNs uninterrupted by attention. Additionally, certain FFN layers experience significant channel pruning, leading to a reduction in their hidden dimension. Step 1 in Figure 1 illustrates a hypothetical outcome of this process.

3 FFN Fusion

Several optimization techniques reduce the number of attention layers to improve inference efficiency [3, 19]. An attention-removed transformer block is defined by the equation

$$\hat{f}(\mathbf{X}) = \mathbf{X} + \text{FFN}(\eta_2(\mathbf{X})). \quad (4)$$

Given a consecutive sequence of attention-removed blocks $\hat{f}^i, \dots, \hat{f}^{i+c}$ we define the parallel version of this sequence $\hat{f}^{[i, i+c]}$ as

$$\hat{f}^{[i, i+c]}(\mathbf{X}) = \mathbf{X} + \sum_{j=0}^c \text{FFN}^{i+j}(\eta_2(\mathbf{X})), \quad (5)$$

where FFN^i is the FFN layer of block i and η_2 is taken from the last layer in the sequence. While in the sequential form the input for every FFN depends on the output of the previous layers, in this form all the FFN layers share the same input, which makes their computation independent and enables execution in parallel across different GPUs. Another useful property of this formulation is that equation 5 is equivalent to equation 4 with a single wider FFN when the weights are concatenated.

Theorem 3.1. *Let $n \in \mathbb{N}$, and let $\text{FFN}^1, \dots, \text{FFN}^n$ be a sequence of FFN functions, where the weights of FFN^i are W_1^i, W_2^i, W_3^i . Then, the sum of these FFN functions (equation 5) is equivalent to a single FFN function FFN^* , with the weight matrices given by:*

$$\begin{aligned} W_1^* &= [(W_1^1)^T, \dots, (W_1^n)^T]^T \\ W_2^* &= [(W_2^1)^T, \dots, (W_2^n)^T]^T \\ W_3^* &= [(W_3^1), \dots, (W_3^n)], \end{aligned}$$

where $[\cdot, \dots, \cdot]$ denotes the concatenation of matrices along the second axis and the dimensions of the matrices are $W_1^*, W_2^* \in \mathbb{R}^{nd_h \times d_e}$, $W_3^* \in \mathbb{R}^{d_e \times nd_h}$.

The theorem (and proof in Appendix A) are written for the simple case where d_h is equal for all the FFNs. The extension for the case where the hidden dimension is different for each layer is straightforward. While written in the context of Equation 3, the theorem holds for other common FFN variants [37, 39]. See a visualization in Figure 7.

Efficiency motivation and analysis. LLMs are ubiquitously designed in sequential blocks, with the block sizes and the number of blocks increasing as the models grow in size. For bigger models, parallelization techniques such as tensor parallel (TP) are utilized to split work across GPUs and reach acceptable inference latencies. However, this strategy does not result in linear latency improvements with the number of GPUs applied. The first reason, is the communication time required for the all-reduce that follows each TP block. The second reason is more subtle: GPUs are at their best for very large operations. As each atomic *General Matrix Multiplication* (GEMM) operation becomes smaller, low level overheads (GPU wave quantization) become more apparent and take a larger portion of the latency budget. Therefore, increasing the computation per GPU (by increasing the size of a block) while reducing the number of synchronizations needed (by using fewer blocks) is an effective strategy for low latencies in a TP setting (see Figure 2). Drawing from this motivation, we attempt to fuse many sequential blocks/FFNs in LLMs into fewer, larger blocks/FFNs. Each single reduction in the depth of the computational graph removes one unit of time spent on synchronization, and the bigger fused blocks also enable operating at higher TP numbers with enough computation assigned to each GPU. Figure 6 shows the effects described in this analysis using measurements of different model architectures in the same TP setting.

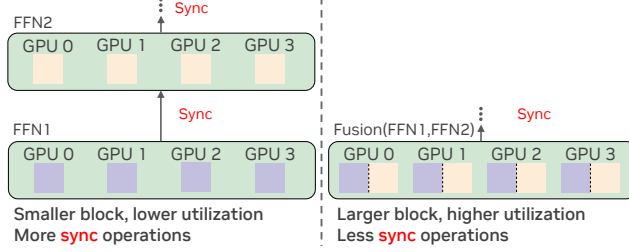


Figure 2: FFN Fusion helps reduce latency by increasing GPU utilization and by reducing syncs.

Pairwise block dependency. It is reasonable to hypothesize that not every transformer block in a sequential model is equally dependent on all its predecessors. In particular, a given block may depend only on a subset of the blocks that precede it. To investigate this, we perform a pairwise dependency analysis between blocks. Let us define $h(\mathbf{X}) = f(\mathbf{X}) - \mathbf{X}$ as the contribution of f to $\mathbf{X}$, and similarly, h^j as the contribution of f^j . We define $\tilde{h}_i^j$ as the contribution of block j when block i is removed from the model. We construct a dependency matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$ by computing the following cosine distance:

$$\mathbf{M}_{ij} = \text{CosineDist}(h^j(\mathbf{X}), \tilde{h}_i^j(\mathbf{X})).$$

That is, $\mathbf{M}_{ij}$ quantifies the dependency of block j on block i . Thus, a small cosine distance indicates that dropping block i has little effect on block j , suggesting relative independence—a characteristic that can be exploited for increasing parallel computation. Conversely, a large cosine distance corresponds to a strong dependency, implying that sequential processing is more critical to maintain performance. We illustrate this approach in Figure 3 by constructing a dependency matrix, $\mathbf{M}$, for Ultra-253B-Base (prior to FFN Fusion; see Section 4). This matrix visually encodes the interdependencies among the model’s layers: **darker blue hues** indicate weaker dependencies—signaling promising opportunities for FFN Fusion—while **darker red hues** denote strong dependencies that would hinder parallelization or fusion. For example, Figure 3 shows that all layers in the model depend on layers 0 and 5, causing their entire rows to be colored in **dark red hues**. The **dark blue** region, marked with a dashed square, shows a sequence of FFNs with low interdependency were selected for FFN Fusion. Due to GPU memory constraints, we are unable to fuse them all at once, so we divide them into fusion sequences based on the maximum size that fits within our devices. Additionally, we applied this metric to explore *block parallelization* (Appendix B), aiming to parallelize general LLM blocks rather than just FFNs. We constructed the dataset that was used for this evaluation following *Distillation Mix* [3].

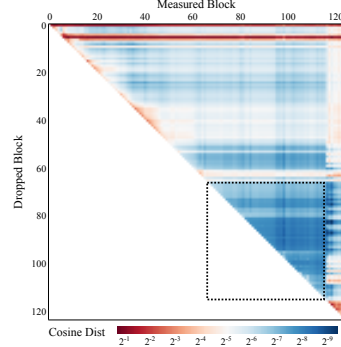


Figure 3: Block-wise dependency heatmap of Ultra-253B-Base before applying FFN Fusion (log-scale). Each coordinate (i, j) encodes how much block j depends on block i , measured by the cosine distance between $h^j(\mathbf{X})$ and $\tilde{h}_i^j(\mathbf{X})$.

4 Producing Large-Scale Models with FFN Fusion

In this section, we describe how FFN Fusion is applied at large scale to transform a 405B-parameter Llama-based model into our more compact Ultra-253B-Base model. Specifically, we show how identifying and fusing long sequential feed-forward blocks reduces depth without sacrificing accuracy. We utilize a lightweight refinement KD and alignment phase to ensure the fused model retains or even improves upon the performance of its larger predecessor. Finally, we present a comprehensive evaluation of the resulting model, demonstrating significant speedups and strong results on standard benchmarks. We first run the standard Puzzle search on Llama-405B, specifying that the derivative model must obtain a $1.5 \times$ latency speedup and fit within a single NVIDIA $8 \times \text{H100}$ node (640 GB total), and in a single B100 GPU (192 GB). This yields a 253B-parameter model whose overall configuration is shown in Appendix D. Many attention layers were removed, resulting in 50 blocks contiguously arranged without interleaved attention layers.

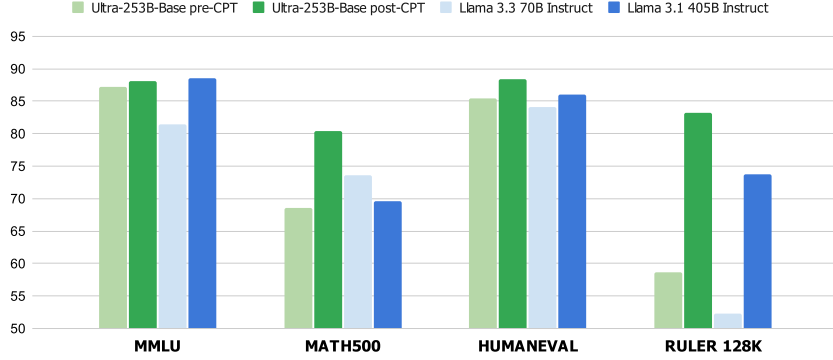


Figure 4: Comparison of Ultra-253B-Base before and after applying an additional CPT.

FFN Fusion. Next, we apply FFN Fusion (§3) to 49 of the 50 consecutive FFN layers (See Section 5.3 for ablation on leaving the last layer). Due to per-GPU memory limits, we split the layers into four sequences [66, 73], [74, 85], [86, 100], [101, 114], each fused into a single FFN. Before fusion, the baseline model achieved 84.23 on MMLU and 8.83 on MT-Bench. Remarkably, after fusing all 49 layers—layers that, if *removed* entirely, would damage performance severely—the model still maintains similar MMLU and MT-Bench scores of 82.76 and 8.35 before any additional training.

Additional Training. To recover performance, we used KD as described in [3]. This involved a multi-stage distillation process: 54B tokens at 8k context, followed by 5B tokens each at 16k and 32k, and finally 0.8B tokens at 128k. The KD process improved MMLU and MT-bench scores to 85.17 and 9.10, respectively. Further optimization was explored through two methods. First, inexpensive alignment via RLHF [45]. Table 1 demonstrated that Ultra-253B-Base surpassed Llama-405B’s capabilities, particularly on the Arena Hard benchmark. We applied only a longer continual pretraining (CPT), without alignment, following the initial KD. This involved 73B tokens at a context length of 8k and another 15B tokens at a context length of 258k, and also yielded strong performance, even before instruction-based tuning (Figure 4). Even after removing nearly half of the model’s attention layers, we still improved its long-context performance, outperforming Llama-405B on RULER-128K [22].

Table 1: Comparison of Ultra-253B-Base and its parent Llama-405B after applying FFN Fusion, KD, and alignment.

Metric	Llama-405B	Ultra-253B-Base
MMLU Instruct [32]	86.53	87.54
MMLU-Pro [44]	71.92	72.25
Arena Hard [29]	72.56	84.92
HumanEval [4]	85.97	86.58
MT-Bench [50]	9.06	9.19

Efficiency Improvements. Ultra-253B-Base achieves a $1.71\times$ speedup in user latency (Table 2), a $35\times$ lower per-token cost at batch size 32, and reduced memory footprint with half the attention layers and 253B parameters (down from 405B). Ultra-253B-Base breaks the efficient frontier on a single H100 node, offering a state-of-the-art accuracy and latency under the similar hardware constraints (see Figure 5). Table 2 details the user latency (tokens/second) achieved by Llama-405B, by 253B model (with and without FFN Fusion), and by Llama-3.3-70B, all under identical tensor parallel settings on a single $8\times$ H100 node. Notably, Ultra-253B-Base is $1.71\times$ faster than the parent for single-user decoding. On NVIDIA H200, its rate increases to 90.05 tokens/second. With speculative decoding [27]—using Llama-3.2-1B-Instruct as a draft model with no extra training—Ultra-253B-Base reaches 202 **tokens/s** on H200. This speculative decoding latency is averaged over all MT-Bench completions. Notably, as TP size increases, the efficiency of the fused FFN layers will increase further, and thus ready to benefit from improving hardware designs, such as GB200 NVL72 nodes. Overall, Ultra-253B-Base demonstrates that FFN Fusion coupled with the Puzzle algorithm can greatly reduce a model’s depth at large scale.

Table 2: User latency under Tensor Parallel (TP) 8 on a single H100 node. A higher tokens/second value indicates lower latency for a single user.

Model	Tokens/Second
Llama-405B	41.44
253B PreFusion	63.38 ($1.53\times$)
Ultra-253B-Base	70.92 ($1.71\times$)
Llama-3.3-70B	94.03

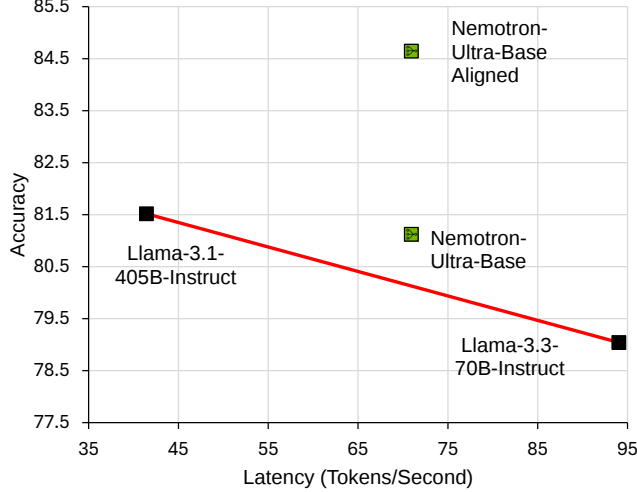


Figure 5: Accuracy vs. latency performance of Ultra-253B-Base. Latency is measured on a single NVIDIA H100 node with tensor parallel (TP) 8, running in FP8. The red line represents the efficient frontier, highlighting models with the best accuracy-to-throughput tradeoff. Accuracy = (MT-Bench $\times$ 10 + MMLU + MMLU-Pro + Arena Hard + HumanEval)/5.

5 Additional Empirical Studies

In this section, we present empirical study to validate our approach. In Section 5.1 we present results for FFN Fusion on a 70B model. Next, in Section 5.2, we examine the consequences of removing FFN layers rather than fusing them, demonstrating that these layers are vital to preserving model quality. In Section 5.3, we investigate the sensitivity of certain FFNs to fusion. Section 5.4 demonstrates that the FFN Fusion phenomenon is applicable to a variety of models, spanning different scales and families. Finally, in Section 5.5 we provide a hypothesis as to why FFN Fusion works.

5.1 FFN Fusion in 70B scale

We consider a derivative of Llama-3.1-70B-Instruct created using the Puzzle algorithm [3], which reduces the model to 49B parameters while matching the original accuracy. The Puzzle process prunes attention in specific layers, leaving two main sequences of consecutive FFN layers: layers 42–51 (10 layers) and layers 53–70 (18 layers). We evaluate FFN Fusion at four progressively increasing levels of intensity, each reducing the number of FFN layers more than the previous:

Table 3: Evaluation of FFN Fusion. The Fusion column indicates how many FFN layers were replaced by the corresponding number of new layers.

Model	Fusion	MMLU	MT-Bench	Accuracy
Baseline	-	80.73	8.87	84.71
Step 1	26 $\rightarrow$ 12	80.64	8.72	83.92
Step 2	26 $\rightarrow$ 6	80.29	8.54	82.84
Step 3	26 $\rightarrow$ 3	80.39	8.30	81.69
Step 4	26 $\rightarrow$ 2	79.98	8.25	81.24

- **Step 1:** Fuse adjacent pairs of FFNs within each sequence of consecutive FFNs.
- **Step 2:** Merge neighboring pairs from Step 1 to form longer fused blocks of length 4 or 5.
- **Step 3:** Fuse the entire first sequence (layers 42–50) into a single FFN, and split the second sequence (53–69) into two fused blocks.
- **Step 4:** Fuse the entire second sequence (53–69) as well, reducing each main FFN run to a single layer.

Similarly to the Ultra-253B-Base, we chose to exclude the last FFN in each sequence. We perform no additional training when applying these fusions. Table 3 reports each variant’s performance on MMLU and MT-Bench, as well as a combined score Accuracy = (MMLU + 10 $\times$ MT-Bench)/2. We observe that step-1 fusion reduces depth with only a 1% overall accuracy drop, while step-4 fusion (collapsing each sequence to a single layer) sees roughly a 4% drop.

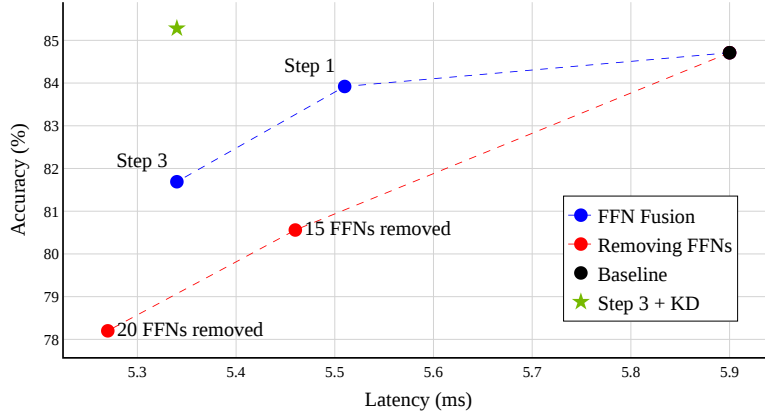


Figure 6: Accuracy vs. Latency for FFN Removal vs. Fusion.

5.2 Removing FFNs vs. FFN Fusion

An intuitive alternative to fusing FFNs is to *remove* the same FFN layers outright. However, as shown below, large-scale removal generally leads to significant accuracy degradation, whereas fusion preserves representational capacity by retaining all parameters in a single, parallel module. To determine which FFNs to drop, we rely on Puzzle’s block-importance scores and remove the least important FFNs first. We restrict removal only to the FFN-fusion regions. For the 49B model, we observe a clear advantage for fusion over removal. Figure 6 compares the accuracy-latency trade-off for two removal levels (15 or 20 FFNs removed) versus the two fusion steps. As we gradually remove more FFNs, we decrease latency but see larger accuracy drops. By contrast, fusing step 3 yields a 10% latency improvement with only a 3.5% accuracy drop. Removing 15 or 20 FFNs yields comparable or slightly higher latency gains (7.5% or 11%), but at steeper accuracy declines (5% or 8.7%). Moreover, a brief knowledge-distillation phase (25B tokens) on the step-3 fused model actually surpasses the baseline (MMLU: 80.56, MT-Bench: 9.00).

5.3 The Final FFN in Each Sequence is Sensitive to Fusion

With the 49B model, we observe that fusing the final FFN in a long sequence often degrades accuracy more than fusing earlier FFNs. Below, we summarize an ablation study to highlight this phenomenon. Table 4 examines various ways of fusing the two main FFN sequences (layers 42–51 and 53–70) in the 49B model. The first part of the table compares fusing the entire second sequence ([53, 69] vs. [54, 70]) and the entire first sequence ([42, 50] vs. [43, 51]) in isolation, indicating that layer 70 is especially sensitive. In the second part, we fuse both sequences simultaneously. When either layer 51 or layer 70 is included in these large fused blocks, we see additional performance drops. Notably, skipping the final FFN (e.g. fusing [42, 50], [53, 69]) yields higher MT-Bench scores than including layer 51 or 70. The final FFN in each attention-removed sequence appears uniquely important to the model’s representations. While most layers can be safely fused, incorporating the last FFN often triggers a significant accuracy drop. Consequently, omitting this final FFN from the fused groups is typically a more reliable choice for efficient fusion with minimal performance loss.

Table 4: Evaluating the impact of fusing the final FFN in 49B scale. Fusing these final layers often causes more accuracy loss.

Sequence	MMLU	MT-Bench
No Fusion	80.73	8.87
[53, 69]	80.57	8.49
[54, 70]	79.89	8.12
[42, 50]	80.49	8.39
[43, 51]	80.46	8.42
[42, 50], [53, 69]	79.98	8.25
[42, 51], [53, 69]	80.05	7.64
[43, 51], [54, 70]	79.92	7.38
[42, 51], [53, 70]	79.89	7.30

5.4 Additional Models

This section explores FFN Fusion on additional models, evaluating its effectiveness across different scales and model families. We applied FFN Fusion to Llama-3.1-8B-Instruct [13] and Mistral Large 2 following a similar procedure as with our 253B and 49B models. Specifically, we used the Puzzle algorithm on both models to remove a portion of their attention layers. Following this, we fused consecutive sequences of FFNs. For Mistral Large 2, we removed 27 attention layers that resulted in 2 sequences of 14 and 13 consecutive FFN layers. We fused each sequence (excluding the last FFN) to a single FFN layer. For Llama-3.1-8B-Instruct (32 layers), we removed 8 consecutive attention layers. We fused the sequence (excluding the last FFN) to a single FFN layer. We also conducted a short KD training on the 8B model over 20B tokens and compared the results to the original model. Table 5 provides a clear demonstration of FFN Fusion’s generalizability, showing its effectiveness across additional models.

Table 5: FFN Fusion impact on Mistral Large 2 and Llama 3.1 8B Instruct.

Fusion	MMLU	MT-Bench
Mistral Large 2		
$\times$	80.56	7.16
$\checkmark$	80.06	6.76
Llama 3.1 8B Instruct		
$\times$	68.91	8.28
$\checkmark$	68.22	7.99

5.5 Fusion Explainability

In this section, we aim to further explain the phenomenon of FFN Fusion. We examine the functional structure of LLMs that allows for the fusing of layers, specifically focusing on the consecutive FFN layers after attention removal. First, we rewrite the token-level normalization equation in an alternative form: $\eta_k(x_i) = D_{s_k} x_i / \|x_i\|$, where we replace the normalization parameters $s_k \in \mathbb{R}^d$ with a diagonal matrix $D_{s_k} \in \mathbb{R}^{d \times d}$, with s_k on the diagonal. This reformulation allows us to “push” the matrix into its corresponding module—either attention or FFN—and consider equations 1 and 4 as functions of the token direction $\vec{x}_i = x_i / \|x_i\|$, while ignoring the magnitude. Figure 11 (a and b), Similar to the measurements in [30], shows the cosine distance between $\mathbf{X}$ and $f(\mathbf{X})$ for each layer of both models. The plot reveals that the FFN fusing areas—[42, 51], [53, 70] for the 49B model and [66, 115] for the 253B model—exhibit lower cosine distance compared to other regions in the model. Assuming that small changes in the input to an FFN layer lead to small changes in its output, we can conclude that fusing the FFNs with low cosine distance (except for the last one) will not cause significant changes to the model. Specifically, by fusing the FFN layers, we are altering the input for each layer, but the directional difference between the original input (according to the original model’s structure) and the new input (resulting from fusion) remains small, and the outputs should remain similar under the smoothness assumption. Another experiment (Table 6) that strengthens this claim shows that even if we reverse the FFNs order instead of fusing them, we get similar performances. The results in Figure 11 (c and d) present the relationship between the layer input $\mathbf{X}$ and $h(\mathbf{X}) = f(\mathbf{X}) - \mathbf{X}$. The small change in token directions may be linked to the low cosine distance between $h(\mathbf{X})$ and $\mathbf{X}$, but they are nearly orthogonal (with a distance of around 0.95 for both models across all layers). However, a more suitable explanation comes from the ratio $\|h(\mathbf{X})\| / \|\mathbf{X}\|$, which is smaller in the fused regions. This suggests that $h(\mathbf{X})$ is small compared to $\mathbf{X}$ that it does not significantly affect the direction of $\mathbf{X}$. Although this section can also serve as motivation for FFN removal, we demonstrated in Figure 6 that removing these layers significantly harms the model’s performance, to a much greater extent than fusion does. Both metrics were evaluated using the *Distillation Mix* dataset.

Table 6: Reverse order vs Fusion experiment on Puzzle-49B.

Sequence	Fusion Accuracy	Reverse Accuracy
[42, 49]	83.74	83.52
[53, 60]	83.24	83.25
[63, 67]	84.22	84.10
[63, 70]	84.00	83.88

6 Concluding Remarks

This work introduces FFN Fusion, a novel optimization technique that demonstrates how sequential computation in large language models can be effectively parallelized. Through comprehensive experiments at scales from 49B to 253B parameters, we showed that sequences of FFN layers can be fused with minimal impact on model capabilities. Our work revealed couple of key insights: (1) FFN layers in attention-removed regions exhibit remarkably low inter-layer dependencies, suggesting these

computations may be more independent than the sequential architecture implies; and (2) empirically, final FFN layers in parallelizable sequences show higher sensitivity to fusion—an observation that points to potentially meaningful structural patterns in how these models process information.

The practical impact of our work is demonstrated through Ultra-253B-Base, which achieves a $1.71\times$ speedup in inference latency compared to its parent Llama-405B model, while requiring $35\times$ lower per-token cost at batch size 32. Most remarkably, these efficiency gains come with minimal degradation in model capability—in fact, Ultra-253B-Base matches or exceeds its parent’s performance across key benchmarks despite using only 253B parameters compared to the original 405B.

Our findings open several promising research directions. First, the clear patterns in inter-layer dependencies we observed could provide a new lens for model interpretability research. The ability to identify which FFN layers operate independently versus those that require sequential processing may offer insights into how these models structure and transform their internal representations. Second, our preliminary finding that even full transformer blocks can sometimes be parallelized suggests possibilities for new architectural designs explicitly optimized for parallel execution. Third, future work could extend FFN Fusion to models incorporating Mixture-of-Experts (MoE) layers. Finding a way to fuse MoE layers while maintaining sparse activation patterns and efficient expert routing could lead to efficiency improvements. Finally, the strong performance of FFN Fusion at larger scales raises intriguing questions about the relationship between model size and natural parallelization.

7 Societal Impact

The inference efficiency achieved through FFN Fusion, as demonstrated by its impact on operational costs in Section 4, significantly improves the accessibility of large language models. By reducing these costs, a wider array of users and organizations can now better afford to deploy and leverage these powerful technologies, effectively dismantling economic barriers.

References

- [1] M. R. AI4Science and M. A. Quantum. The impact of large language models on scientific discovery: a preliminary study using gpt-4, 2023.
- [2] R. Anil, S. Borgeaud, Y. Wu, J. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, D. Silver, S. Petrov, M. Johnson, I. Antonoglou, J. Schrittwieser, A. Glaese, J. Chen, E. Pitler, T. P. Lillicrap, A. Lazaridou, O. Firat, J. Molloy, M. Isard, P. R. Barham, T. Hennigan, B. Lee, F. Viola, M. Reynolds, Y. Xu, R. Doherty, E. Collins, C. Meyer, E. Rutherford, E. Moreira, K. Ayoub, M. Goel, G. Tucker, E. Piqueras, M. Krikun, I. Barr, N. Savinov, I. Danihelka, B. Roelofs, A. White, A. Andreassen, T. von Glehn, L. Yagati, M. Kazemi, L. Gonzalez, M. Khalman, J. Sygnowski, and et al. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023.
- [3] A. Bercovich, T. Ronen, T. Abramovich, N. Ailon, N. Assaf, M. Dabbah, I. Galil, A. Geifman, Y. Geifman, I. Golan, N. Haber, E. Karpas, R. Koren, I. Levy, P. Molchanov, S. Mor, Z. Moshe, N. Nabwani, O. Puny, R. Rubin, I. Schen, I. Shahaf, O. Tropp, O. U. Argov, R. Zilberstein, and R. El-Yaniv. Puzzle: Distillation-based nas for inference-optimized llms, 2024.
- [4] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.
- [5] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue,

- B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- [6] DeepSeek-AI, A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J. L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Wang, J. Chen, J. Chen, J. Yuan, J. Qiu, J. Li, J. Song, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Xu, L. Xia, L. Zhao, L. Wang, L. Zhang, M. Li, M. Wang, M. Zhang, M. Zhang, M. Tang, M. Li, N. Tian, P. Huang, P. Wang, P. Zhang, Q. Wang, Q. Zhu, Q. Chen, Q. Du, R. J. Chen, R. L. Jin, R. Ge, R. Zhang, R. Pan, R. Wang, R. Xu, R. Zhang, R. Chen, S. S. Li, S. Lu, S. Zhou, S. Chen, S. Wu, S. Ye, S. Ye, S. Ma, S. Wang, S. Zhou, S. Yu, S. Zhou, S. Pan, T. Wang, T. Yun, T. Pei, T. Sun, W. L. Xiao, and W. Zeng. Deepseek-v3 technical report. *CoRR*, abs/2412.19437, 2024.
- [7] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Gpt3. int8(): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332, 2022.
- [8] T. Dettmers and L. Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pages 7750–7774. PMLR, 2023.
- [9] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Rozière, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. M. Kloumann, I. Misra, I. Evtimov, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.
- [10] A. Fan, E. Grave, and A. Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.
- [11] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.
- [12] E. Frantar, S. Ashkboos, T. Hoefer, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.
- [13] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra,

A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. van der Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. de Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimploukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhennde, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand, A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardi, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C.-H. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E.-T. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, Guangyi, Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I.-E. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J.-B. Gaya, J. Marcus, J. Tang, J. Chan, J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A. L. Silva, L. Bell, L. Zhang, L. Guo, L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev, N. Dong, N. Cheng, O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng,

- S. Cho, S. Virk, S. Subramanian, S. Choudhury, S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar, V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, Yu, Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma. The llama 3 herd of models, 2024.
- [14] A. Gromov, K. Tirumala, H. Shapourian, P. Gloriosi, and D. A. Roberts. The unreasonable ineffectiveness of the deeper layers, 2024.
- [15] X. Guan, L. L. Zhang, Y. Liu, N. Shang, Y. Sun, Y. Zhu, F. Yang, and M. Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking, 2025.
- [16] Y. Guan, D. Wang, Z. Chu, S. Wang, F. Ni, R. Song, L. Li, J. Gu, and C. Zhuang. Intelligent virtual assistants with llm-based process automation, 2023.
- [17] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.
- [18] B. Hassibi and D. Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.
- [19] S. He, G. Sun, Z. Shen, and A. Li. What matters in transformers? not all attention is needed, 2024.
- [20] D. Hendrycks and K. Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.
- [21] Hive-Digital-Technologies. <https://huggingface.co/datasets/H-D-T/Buzz-V1.2>.
- [22] C.-P. Hsieh, S. Sun, S. Krizan, S. Acharya, D. Rekes, F. Jia, Y. Zhang, and B. Ginsburg. Ruler: What’s the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.
- [23] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mixtral of experts, 2024.
- [24] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.
- [25] V. Lad, W. Gurnee, and M. Tegmark. The remarkable robustness of llms: Stages of inference?, 2024.
- [26] Y. LeCun, J. Denker, and S. Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.
- [27] Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR, 2023.
- [28] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.
- [29] T. Li, W. Chiang, E. Frick, L. Dunlap, T. Wu, B. Zhu, J. E. Gonzalez, and I. Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *CoRR*, abs/2406.11939, 2024.
- [30] Z. Liu, J. Wang, T. Dao, T. Zhou, B. Yuan, Z. Song, A. Shrivastava, C. Zhang, Y. Tian, C. Re, and B. Chen. Deja vu: Contextual sparsity for efficient llms at inference time, 2023.

- [31] P. Michel, O. Levy, and G. Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.
- [32] Nvidia, :, B. Adler, N. Agarwal, A. Aithal, D. H. Anh, P. Bhattacharya, A. Brundyn, J. Casper, B. Catanzaro, S. Clay, J. Cohen, S. Das, A. Dattagupta, O. Delalleau, L. Derczynski, Y. Dong, D. Egert, E. Evans, A. Ficek, D. Fridman, S. Ghosh, B. Ginsburg, I. Gitman, T. Grzegorzec, R. Hero, J. Huang, V. Jawa, J. Jennings, A. Jhunjhunwala, J. Kamalu, S. Khan, O. Kuchaiev, P. LeGresley, H. Li, J. Liu, Z. Liu, E. Long, A. S. Mahabaleshwarkar, S. Majumdar, J. Maki, M. Martinez, M. R. de Melo, I. Moshkov, D. Narayanan, S. Narenthiran, J. Navarro, P. Nguyen, O. Nitski, V. Noroozi, G. Nutheti, C. Parisien, J. Parmar, M. Patwary, K. Pawelec, W. Ping, S. Prabhumoye, R. Roy, T. Saar, V. R. N. Sabavat, S. Satheesh, J. P. Scowcroft, J. Sewall, P. Shamis, G. Shen, M. Shoeybi, D. Sizer, M. Smelyanskiy, F. Soares, M. N. Sreedhar, D. Su, S. Subramanian, S. Sun, S. Toshniwal, H. Wang, Z. Wang, J. You, J. Zeng, J. Zhang, J. Zhang, V. Zhang, Y. Zhang, and C. Zhu. Nemotron-4 340b technical report, 2024.
- [33] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.
- [34] G. Penedo, H. Kydlíček, L. B. Allal, A. Lozhkov, M. Mitchell, C. Raffel, L. von Werra, and T. Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *CoRR*, abs/2406.17557, 2024.
- [35] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019. Accessed: 2024-11-15.
- [36] H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429, 2023.
- [37] N. Shazeer. Glue variants improve transformer, 2020.
- [38] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [39] D. R. So, W. Mañke, H. Liu, Z. Dai, N. Shazeer, and Q. V. Le. Primer: Searching for efficient transformers for language modeling, 2022.
- [40] L. Soldaini, R. Kinney, A. Bhagia, D. Schwenk, D. Atkinson, R. Authur, B. Bogin, K. R. Chandu, J. Dumas, Y. Elazar, V. Hofmann, A. H. Jha, S. Kumar, L. Lucy, X. Lyu, N. Lambert, I. Magnusson, J. Morrison, N. Muennighoff, A. Naik, C. Nam, M. E. Peters, A. Ravichander, K. Richardson, Z. Shen, E. Strubell, N. Subramani, O. Tafjord, E. P. Walsh, L. Zettlemoyer, N. A. Smith, H. Hajishirzi, I. Beltagy, D. Groeneveld, J. Dodge, and K. Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research. In L. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15725–15788. Association for Computational Linguistics, 2024.
- [41] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic. Galactica: A large language model for science, 2022.
- [42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.
- [43] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.
- [44] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, T. Li, M. Ku, K. Wang, A. Zhuang, R. Fan, X. Yue, and W. Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *CoRR*, abs/2406.01574, 2024.
- [45] Z. Wang, A. Bukharin, O. Delalleau, D. Egert, G. Shen, J. Zeng, O. Kuchaiev, and Y. Dong. Helpsteer2-preference: Complementing ratings with preferences. *CoRR*, abs/2410.01257, 2024.

- [46] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Huggingface’s transformers: State-of-the-art natural language processing, 2020.
- [47] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.
- [48] Z. Xu, F. Jiang, L. Niu, Y. Deng, R. Poovendran, Y. Choi, and B. Y. Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *CoRR*, abs/2406.08464, 2024.
- [49] M. Zhang and Y. He. Accelerating training of transformer-based language models with progressive layer dropping. *Advances in neural information processing systems*, 33:14011–14023, 2020.
- [50] L. Zheng, W. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

A Proof of Theorem 3.1

Proof. The proof will proceed by induction. For the base case, we consider two FFN layers, FFN^1 and FFN^2 . We will demonstrate that $\text{FFN}^* = \text{FFN}^1 + \text{FFN}^2$. The induction step follows naturally from the associative property of addition.

$$\begin{aligned}
(\text{FFN}^1(\mathbf{X}) + \text{FFN}^2(\mathbf{X})) &= (\sigma(\mathbf{X}(W_2^1)^T) \odot \mathbf{X}(W_1^1)^T)(W_3^1)^T + (\sigma(\mathbf{X}(W_2^2)^T) \odot \mathbf{X}(W_1^2)^T)(W_3^2)^T \\
&= [\sigma(\mathbf{X}(W_2^1)^T) \odot \mathbf{X}(W_1^1)^T, \sigma(\mathbf{X}(W_2^2)^T) \odot \mathbf{X}(W_1^2)^T] (W_3^*)^T \\
&= ([\sigma(\mathbf{X}(W_2^1)^T), \sigma(\mathbf{X}(W_2^2)^T)] \odot [\mathbf{X}(W_1^1)^T, \mathbf{X}(W_1^2)^T]) (W_3^*)^T \\
&= (\sigma(\mathbf{X}(W_2^*)^T) \odot \mathbf{X}(W_1^*)^T)(W_3^*)^T \\
&= \text{FFN}^*(\mathbf{X})
\end{aligned}$$

where $[\cdot, \cdot]$ is the concatenation of matrices along the second dimension. □

B Block Parallelization

In all previous experiments, we focused on fusing sequences of FFNs in attention-removed layers (see Section 3). In that setting, the homogeneous structure of FFN-only layers permits straightforward weight concatenation and fusion. However, when considering full block parallelization—where entire Transformer blocks, each comprising both an attention module and an FFN, are run in parallel—such fusion is not possible. This is because each full block consists of two distinct components, with the attention output explicitly directed to its paired FFN. Running entire blocks completely in parallel is not currently natively supported by heavily optimized inference frameworks such as TensorRT-LLM or vLLM [24]. Nevertheless, one can envision assigning each full block to a different GPU, thereby maximizing parallelism and potentially achieving significant speedups. We use our block-wise dependency analysis to examine the dependency structure between blocks and identify candidate groups whose outputs are relatively independent. Such groups are ideal for parallelization with minimal accuracy loss and could enable higher degrees of tensor parallelism—each full block operating on its own set of GPUs—thereby improving inference throughput. In our experiments, we implement these full block parallelization strategies using a more flexible environment (e.g., HuggingFace Transformers [46]), albeit with reduced optimization for large-scale deployments. A thorough investigation of the actual speedups in specialized frameworks is left for future work.

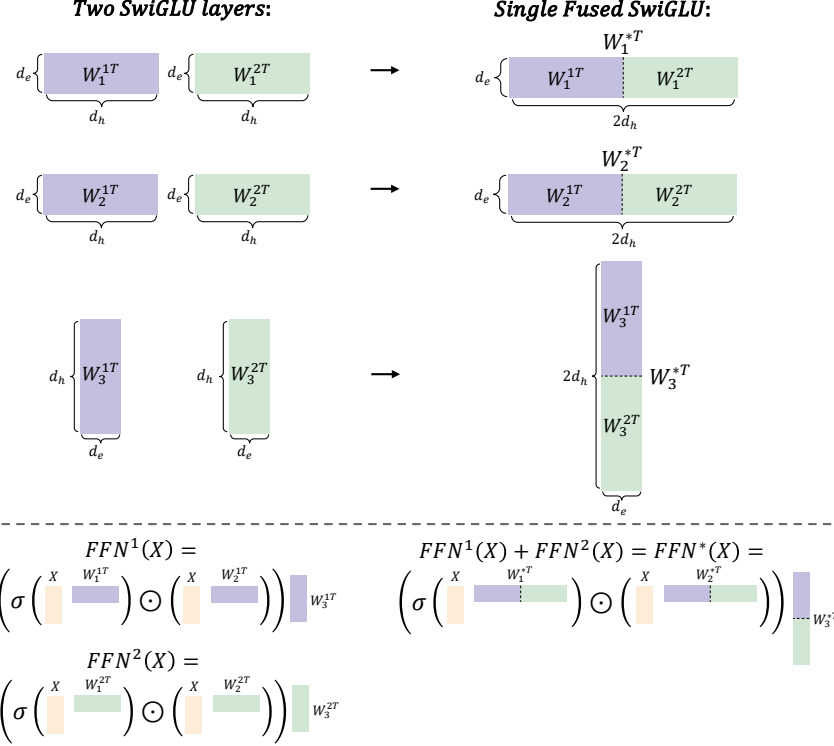


Figure 7: A visualization of FFN Fusion applied to SwiGLU. Two FFNs (left) are fused into a single FFN (right).

Method. We first compute the *block-wise dependency matrix* exactly as in Section 3. We then search for *candidate sequences* of 4 consecutive blocks to fuse. The choice of this number of blocks is mostly related to hardware constraints. Arbitrary sizes will be considered in our future work. For each such sequence $[i, i+3]$, $i \in [m-4]$, we extract the corresponding $M^{[i, i+3]} \in \mathbb{R}^{4 \times 4}$ submatrix from the block dependency matrix and compute two statistics:

$$M_{\max}^{[i, i+3]} = \max_{k, j \in [4]} M_{kj}^{[i, i+3]}, \quad M_{\text{sum}}^{[i, i+3]} = \sum_{k, j \in [4]} M_{kj}^{[i, i+3]}$$

A lower $M_{\max}$, and M_{sum} indicates weaker dependencies among those blocks, suggesting they may be more amenable to parallelization. We exclude any blocks that lack attention (shaded regions in Figure 8b) because our focus here is on *full* Transformer blocks that contain both attention and FFN modules. We employ a simple *greedy algorithm* to choose the best subsequences:

1. Choose the length-4 sequence starting index $i^* = \arg \min_i M_{\max}^{[i, i+3]}$.
2. If the argmin is not unique, break the tie using $i^* = \arg \min_i M_{\text{sum}}^{[i, i+3]}$ (restricted to the set from step 1).
3. Parallelize the blocks according to the chose sequence $[i^*, i^*+3]$.
4. Remove any overlapping sequences.
5. Repeat until no valid sequences remain.

The motivation behind this algorithm is to avoid choosing sequences with high dependency pairs. The sum tie breaking is due to step-like characteristic of $M_{\max}$ statistic (see Figure 8b). Table 7 reports the downstream performance (MMLU and MT-Bench) after fusing the sequences chosen by 4 algorithm steps.

Results. From Table 7, we observe that parallelizing the first and second sequence of four blocks leads to only a modest drop in MMLU, but once the third sequence is added, the decline becomes much more pronounced. This suggests that full block parallelization is more challenging than FFN Fusion. These observations are supported by the block-wise dependency heatmap and the submatrix statistics (Figure 8), both of which indicate stronger inter-block dependencies and higher $M_{\max}$ and M_{sum} values in full block sequences compared to attention removed sequences. In addition, Figure 11 shows that the full transformer blocks alter significantly the tokens directions.

Table 7: Comparison of MMLU and MT-Bench scores across different parallelization strategies .

Model Name	MMLU	MT-Bench
49B model (no fusion)	80.73	8.87
Parallel Sequences		
[38, 41]	80.51	8.67
[38, 41] , [71, 74]	80.04	8.67
[38, 41] , [71, 74] , [32, 35]	77.86	8.20
[38, 41] , [71, 74] , [32, 35] , [26, 29]	56.12	6.62

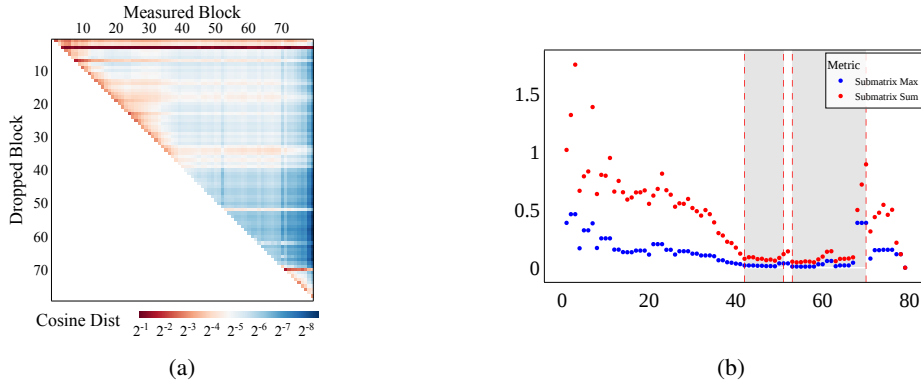


Figure 8: (a) Block-wise Dependency Heatmap of the 49B model (log-scale). Darker blue hues indicate weaker dependencies, darker red hues indicate stronger dependencies. (b) $M_{\max}$ and M_{sum} values for 4-Block Sequences of the 49B model. Lower values indicating more promising candidates for parallelization.

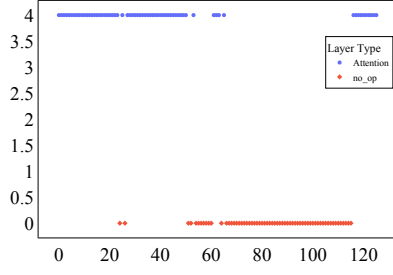
C Datasets

For applying Puzzle throughout our experiments, we used the same dataset mixture used in [3], termed *Distillation Mix*. This mixture includes source code repositories, Wikipedia articles, books, news websites, and several other domains. The dataset comprises 224 billion tokens collected from three public datasets: FineWeb [34], Dolma [40], and Buzz-V1.2 [21]. For Ultra-253B-Base KD training we used the same data reinforced with synthetic data generated with Llama-405B, following the [48] approach, to help further align Ultra-253B-Base with its parent model.

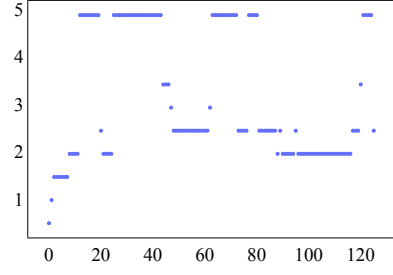
D Ultra-253B-Base and 49B Architecture Overview

In this section, we detail the configuration of each block in the Ultra-253B-Base model and highlight several unique design choices introduced by the Puzzle framework on top of Llama-405B. Notably, the model employs variable FFN widths, with scaling multipliers ranging from 0.4875 up to 4.875 across its 126 blocks. This variability allows the model to dynamically adjust capacity at different depths, balancing performance with computational efficiency. Furthermore, it includes a sequence of 50 consecutive layers that bypass the attention mechanism (denoted by `no_op`), dedicating these layers entirely to FFN processing and creating an ideal region for FFN Fusion.

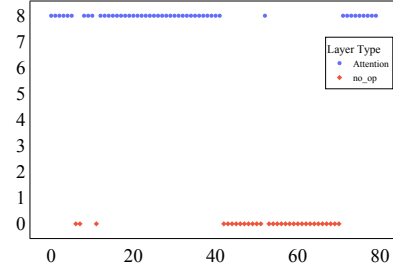
We observe a similar pruning pattern in the 49B model, which is derived from the Llama-3.1 – 70B model using the Puzzle framework. Like its larger counterpart, the 49B model also features variable FFN multipliers and blocks where attention is removed, although at a smaller scale. Figure 9 compares both architectures.



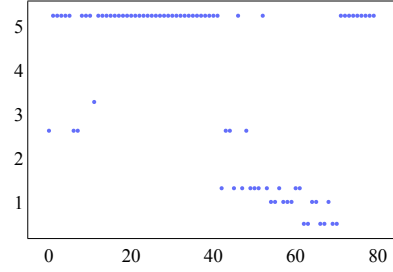
(a) Ultra-253B-Base Attention Layout



(b) Ultra-253B-Base FFN Layout



(c) 49B Attention Layout



(d) 49B FFN Layout

Figure 9: A 2×2 overview of Ultra-253B-Base (top row) and 49B model (bottom row). Subfigures (a) and (b) illustrate the attention and FFN configurations, respectively, for the 253B model. Subfigures (c) and (d) show the corresponding layouts for the 49B model. Both architectures feature variable FFN widths and regions where attention has been removed, although at different scales.

E MoE Inference Performance

MoEs show great promise in reducing inference costs and are currently spearheaded by DeepSeek-V3 [6], a highly sparse MoE with a great performance. However, MoEs have their own problems in inference.

E.1 Problems with Small Blocks

An underlying problem for MoEs paradoxically arises from the small size of their sub-modules. Smaller layers incur large latency than expected for two reasons:

- **GPU utilization** - low level GPU overheads such as wave quantization produce latency problems that become stronger as the module becomes smaller.
- **Communication overhead** - The All-Reduce operation that is typical for tensor parallelism creates an overhead for communication. For small modules this overhead increases in proportion.

MoE’s experts are smaller than dense MLPs and hence suffer more strongly from the above effects. In addition the routing mechanism of MoEs is another small module that suffers accordingly. As a result, dense models scale better with the number of GPUs while MoEs reach an early saturation. It is worth to notice that this is in complete contrast to our FFN Fusing method that creates larger modules that parallelize better, as visualized in Figure 2.

E.2 Batch Size Effects

MoEs behave very well for very large batch sizes. For such large workloads load balancing issues are less dominant and low level overheads become negligible. However, smaller batch sizes are not as efficient. Concretely, we define an efficient size of a batch through the perspective of analyzing a linear layer. The performance of linear layers varies significantly with batch size. For small batches, performance is limited by the time needed to read weights from DRAM (I/O bottleneck). For large batches, performance becomes compute-bound, limited by the hardware’s maximal FLOPs/sec. Notably, the latency of a linear layer remains the same for batch sizes 1 and 64, as both are dominated by weight loading time. This makes small batches extremely inefficient. While other factors exist, we consider a linear layer to operate efficiently once the number of tokens in the batch approaches 64, and inefficient otherwise. MoEs require all experts to be fully utilized for high throughput. In practice, the total batch is split across experts, significantly reducing the effective number of tokens each expert sees. For example, with a sparsity factor of $1/32$ (as in [5, 6]), a batch of 2048 tokens (64×32) is needed to ensure each expert processes around 64 tokens. The case is more extreme in Llama 4 Maverick, where a $1/128$ sparsity factor requires a batch of 8192 tokens (64×128) to reach this threshold. These batch sizes are often impractical—especially in generation mode—leaving MoEs to operate in a low-efficiency regime, where linear layers are under-utilized. We can thus conclude that MoEs only really live up to their promise for very large batch sizes. For more commonly used intermediate batch sizes they suffer from bad scaling with the number of tokens, larger low level overheads, and worse parallelization scaling than dense models.

F Further Pairwise Block Dependency Analysis

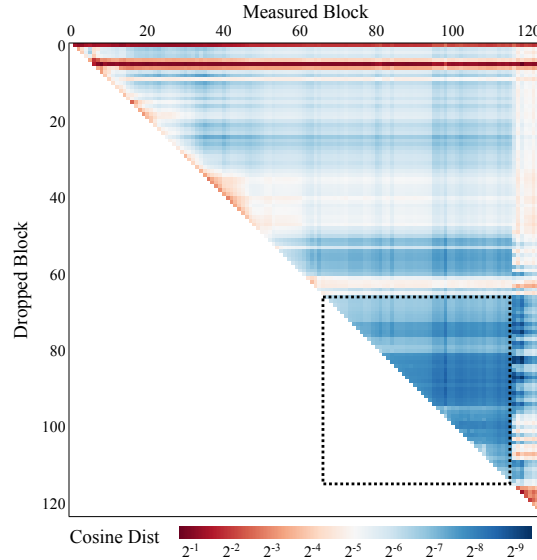


Figure 10: Replication of the block-wise dependency heatmap for Ultra-253B-Base PreFusion, shown here for convenience. Each coordinate (i, j) represents the cosine distance between $h^j(\mathbf{X})$ and $\tilde{h}_i^j(\mathbf{X})$, quantifying how much block j depends on block i . *Darker blue* indicates weaker dependencies, while *darker red* indicates stronger dependencies. The dotted box marks an attention-removed region with generally low dependency values, suggesting high parallelization potential.

Figure 10 offers a more detailed view of the block-wise dependency structure in Ultra-253B-Base PreFusion. Below, we highlight several notable observations:

- **Overall Color Scale.** Each entry (i, j) in the heatmap corresponds to the cosine distance between the outputs of block j when block i is dropped versus when the model is intact. A *dark red* hue signifies a strong dependency, meaning that omitting block i substantially alters block j . Conversely, *dark blue* indicates a weak dependency, suggesting that dropping block i does not notably affect block j .

- **Attention-Removed Region (Dotted Box).** The dotted box highlights a region of consistently low dependency values, where most blocks exhibit *minimal* mutual influence. This area arises from attention pruning in those layers and is particularly well-suited for FFN Fusion, since layers here can be more easily parallelized without significantly harming accuracy.
- **Crucial Early Blocks.** Some blocks in the upper rows of the matrix (toward the top of the y-axis) display consistently *red* cells across many columns. This indicates that certain early layers have a strong influence on a wide range of subsequent blocks, making them less amenable to parallelization.
- **Sub-region with Sequential Reliance.** A noticeable diagonal band of higher values appears in some sub-regions, implying that each block in that band strongly depends on its immediate predecessor. Such sequential reliance reduces the potential for parallelization in those areas, since omitting any single block significantly affects the next one.
- **Global Sinks at Later Layers.** As we move toward deeper layers (near the bottom-right corner), some blocks again show stronger dependencies, acting as “global sinks” that consolidate information from many earlier blocks. Although the importance of certain blocks may wane in mid-network regions, it can resurface in the final layers.

In summary, the dependency matrix reveals both strongly and weakly coupled sub-regions. High-dependency zones require careful consideration to avoid accuracy loss if parallelization is attempted. Conversely, low-dependency zones, such as the attention-removed region, offer a promising avenue for efficient FFN Fusion.

G Fusion Explainability

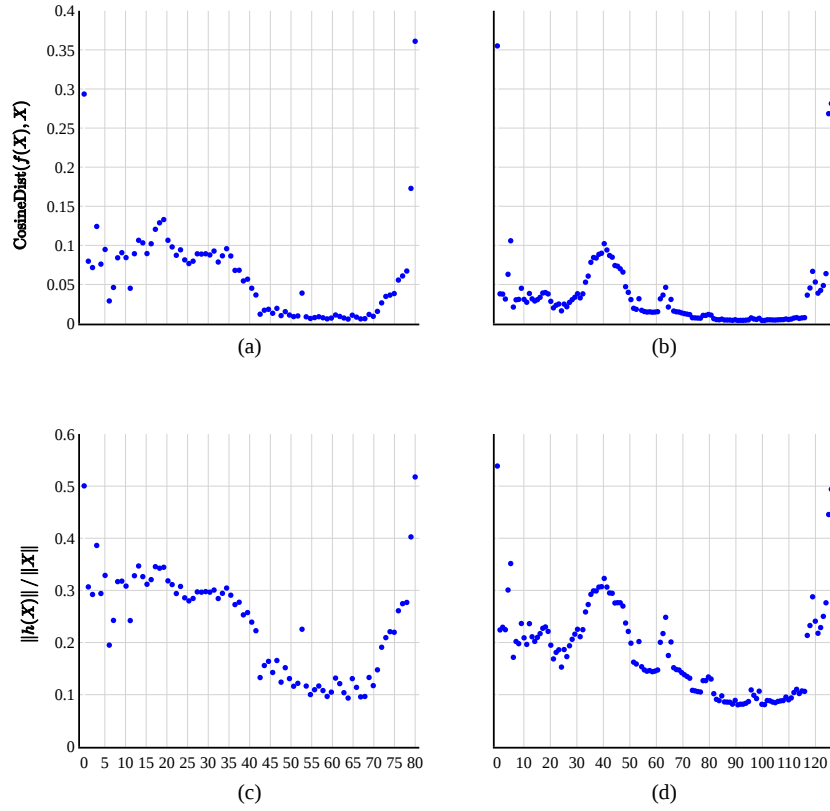


Figure 11: Per-layer metrics. Upper row is the cosine distance between $f(\mathbf{X})$ and $\mathbf{X}$ for the (a) The 49B model and (b) Ultra-253B-Base model. Bottom row represents the ratio between $h(\mathbf{X})$ and $\mathbf{X}$ for the (c) The 49B model and (d) Ultra-253B-Base model.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [Yes]

Justification: The claim introduced in the abstract and introduction are supported by theoretical claims (Theorem 3.1) and empirical evidence (Sections 4 & 5)

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The authors indicate that the suggested method is mainly relevant in cases where there is long a sequence of consecutive FFN blocks.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Theorem 3.1 proves the equivalence between a parallel computation of FFN blocks and our consolidate formulation.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The steps that were taken to create the results are fully described in our method & results sections.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [NA]

Justification: We will release code upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: The paper describes all the details regarding the main method (FFN Fusion). Training methods were based on previous work.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[NA\]](#)

Justification: Error bars are irrelevant for the experiments in the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.