PANGU ULTRA: PUSHING THE LIMITS OF DENSE LARGE LANGUAGE MODELS ON ASCEND NPUS

Pangu Team, Huawei

PanguTech@huawei.com

ABSTRACT

We present Pangu Ultra, a Large Language Model (LLM) with 135 billion parameters and dense Transformer modules trained on Ascend Neural Processing Units (NPUs). Although the field of LLM has been witnessing unprecedented advances in pushing the scale and capability of LLM in recent years, training such a large-scale model still involves significant optimization and system challenges. To stabilize the training process, we propose depth-scaled sandwich normalization, which effectively eliminates loss spikes during the training process of deep models. We pre-train our model on 13.2 trillion diverse and high-quality tokens and further enhance its reasoning capabilities during post-training. To perform such large-scale training efficiently, we utilize 8,192 Ascend NPUs with a series of system optimizations. Evaluations on multiple diverse benchmarks indicate that Pangu Ultra significantly advances the state-of-the-art capabilities of dense LLMs such as Llama 405B and Mistral Large 2, and even achieves competitive results with DeepSeek-R1, whose sparse model structure contains much more parameters. Our exploration demonstrates that Ascend NPUs are capable of efficiently and effectively training dense models with more than 100 billion parameters. Our model and system will be available for our commercial customers.

1 Introduction

Large Language Models (LLMs) have transformed the landscape and our understanding of Artificial Intelligence. Their remarkable capabilities are enabling more and more AI applications, bringing numerous commercial opportunities. Unsurprisingly, teams are racing to push the scaling law to create models with more and more parameters. Although the Transformer [68] structure is a popular choice for large models, it is still debatable whether the models should be sparse or dense. With more than 100 billion parameters, sparse architectures powered by Mixture of Experts (MoE), such as DeepSeek [46, 19], have demonstrated surreal human-like language and thinking abilities [36], which makes sparse models a pupular choice when pushing the limit of LLMs.

At the same time, dense models, such as the Qwen [11, 72], Llama [25], and Gemma [67] series, are currently popular among models with fewer than 100 billion parameters thanks to their strong performance in specific skills and ease of deployment. The parameters in dense models are usually easier to optimize, while the dynamic components in sparse models usually need to turn to additional heuristics for stable training. In addition, the dense model structures at inference time make it easier to optimize system performance due to deterministic parameter usage. In this study, we aim to further explore the potential of dense models at large scales and show the performance of dense models can be on par with state-of-the-art MoE models on diverse tasks.

The numbers of model parameters and layers are two crucial dimensions to release the full potential of dense models. While model parameter count is critical for model performance and plays a central role in scaling laws [38], recent studies [73, 50] suggest that model depth has a significant impact on reasoning capabilities. However, our exploration in those two aspects poses significant challenges in exploring the limits of those two aspects. Deeper models usually introduce unstable training, manifested as spikes in training loss curves. Experimental observations suggest that those spikes can knock our model out of the ideal parameter landscape and cause irreparable damage to the training process. Meanwhile, training hundreds of billions of parameters in dense models requires orchestrating thousands of AI processors, which poses significant system efficiency challenges.

For our exploration, we introduce Pangu Ultra, a dense Transformer architecture with 135 billion parameters and 94 layers. The model setup is at the forefront scale of the top performing dense models [11, 72, 25, 67]. Regarding challenges of training deep models, we hypothesize that the loss spikes are due to gradient fluctuations, which in turn hinder convergence rates and may lead to training divergence. Therefore, we propose two techniques, the depth-scaled sandwich norm and tiny initialization, both of which are designed to maintain stable gradient norms. Specifically, we first replace pre-layer norm [47] with the sandwich norm [20] and scaled initialization values in the post-layer normalization based on the model's depth. This depth-based adjustment helps control the range of gradient fluctuations effectively. In addition, we scale the standard deviation of weight initialization according to the model's width and depth, leading to tiny initialization. These two techniques lead to more stable gradients throughout the training process, eliminating loss spikes during the training of Pangu Ultra, and improving overall model performance.

In practice, we pre-train Pangu Ultra on 13.2 trillion tokens of our built corpus. In the pre-training stage, we use three phrases of data corpus each with a distinct data recipe. The design principles behind three phrases are first to help the model develop knowledge and linguistics, and then to directly equip it with reasoning ability, and finally to boost it on actively learning to reason. The model context window is gradually extended from 4K to 128K. In the post-training stage, we begin with applying efficient supervised fine-tuning (SFT) for a cold start, utilizing a carefully curated set of instruction data. Following this, Pangu Ultra undergoes further optimization through Reinforcement Learning (RL). The overall training of Pangu Ultra is stable in this process.

To handle large-scale model training of more than 100 billion parameters, we utilize a large-scale computing cluster consisting of 8,192 Ascend NPUs and employ a series of system optimization to improve the system efficiency. The primary challenge is minimizing pipeline bubbles [29] at large scales, which arise due to batch size constraints [35]. We take advantage of the typical 4 types of parallelism on our Ascend cluster, that is, Data Parallelism (DP), Tensor Parallelism (TP) [63], Sequence Parallelsim [39] and Pipeline Parallelism (PP) [30, 51]. As the training cluster scales up, the mini-batch size allocated to each DP decreases, leading to an increased pipeline bubble ratio. To mitigate this issue, we employ additional virtual pipeline (VPP) scheduling [52] with fine-grained tuning to ensure load balancing and reduce the PP bubble ratio from 30.45% to 6.8%. The second challenge is to achieve high training efficiency for long sequences. Both attention mask generation and self-attention computation are time- and memory-intensive, particularly for long contexts. We utilize a NPU Fusion Attention (NFA) operator [4, 18, 17] tailored for the Ascend NPUs, which supports reset attention mask scenarios and eliminates the need to construct the attention mask before calling the NFA, thus improving computational efficiency and reducing memory cost. Under the implementation of several fine-grained system optimization, we achieve a Model FLOPs Utilization (MFU) [14] of over 50% when training Pangu Ultra on 8,192 Ascend NPUs.

On public evaluation benchmarks, Pangu Ultra outperforms existing dense LLMs including Llama 405B and Mistral Large 2 123B on almost all major language tasks, and achieves competitive results with sparse models consisting of more than 500 billion parameters. These results indicate the potential of dense model capabilities is still promising to explore. Pangu Ultra also demonstrates that the Ascend NPUs are suitable for exploring the full capabilities of large-scale dense language models.

2 Model Architecture

The basic architecture of Pangu Ultra is similar to Llama 3 [25]. It has 135 billion parameters with a hidden dimension of 12,288, a SwiGLU [60] feed-forward network (FFN) intermediate size of 28,672, and 94 layers. The attention blocks in Pangu Ultra leverage Group Query Attention (GQA) to reduce KV-cache size by incorporating 96 query heads and 8 KV heads.

There are two crucial differences to address the fundamental challenges of training stability and convergence in large dense LLMs. We propose Depth-Scaled Sandwich-Norm to replace the layer normalization and TinyInit for parameter initialization. By integrating these techniques, Pangu Ultra achieves substantial improvements over previous dense models.

2.1 Depth-Scaled Sandwich-Norm

Large-scale dense models typically adopt deeper architectures [22], although MoE models usually scale in width [19]. However, increased depth introduces greater challenges in maintaining training stability. Given the prohibitive cost of pre-training, stable training of large dense LLMs becomes paramount. Pre-Layer

Normalization (Pre-LN) has been found to make back-propagation more efficient for deep Transformers [69], leading to its widespread adoption in Transformer-based large language model (LLM) architectures [22, 11, 19].

However, in models employing the pre-LN structure, the fluctuating output scale of each sub-layer can easily lead to training instability [66]. To address this issue, sandwich-norm [20] applies an layer normalization to each sub-layer's output prior to the residual connection. While the sandwich-norm maintains the scale stability of individual sub-layer outputs, the progressive accumulation of output norms via residual connections across multiple layers may nevertheless lead to training instability.

To mitigate this, we present the depth-scaled sandwich norm, which integrates the sandwich norm with a depth-scaled initialization scheme. The layer normalization regulates layer-wise output magnitudes through trainable gamma parameters, which are initialized with values scaled proportionally to the inverse of network depth. Figure 1 illustrates the differences between the depth-scaled sandwich-norm and pre-norm architectures. The formula of depth-scaled sandwich-norm is

$$\begin{aligned} \mathbf{h} &\leftarrow \mathbf{h} + \text{Norm}(\gamma_{\text{attn}}, \text{ATTN}(\text{Norm}(\mathbf{h}))), \quad \gamma_{\text{attn}} = \frac{c_{\text{attn}}}{\sqrt{L}}, \\ \mathbf{h} &\leftarrow \mathbf{h} + \text{Norm}(\gamma_{\text{mlp}}, \text{MLP}(\text{Norm}(\mathbf{h}))), \quad \gamma_{\text{mlp}} = \frac{c_{\text{mlp}}}{\sqrt{L}}, \end{aligned} \tag{1}$$

where L is the number of layers, c_{attn} and c_{mlp} are set as the initial output standard deviations of the attention layer and feed-forward network (FFN) layer, respectively. For Pangu Ultra, we set c_{attn} to 0.283 and c_{mlp} to 0.432 .

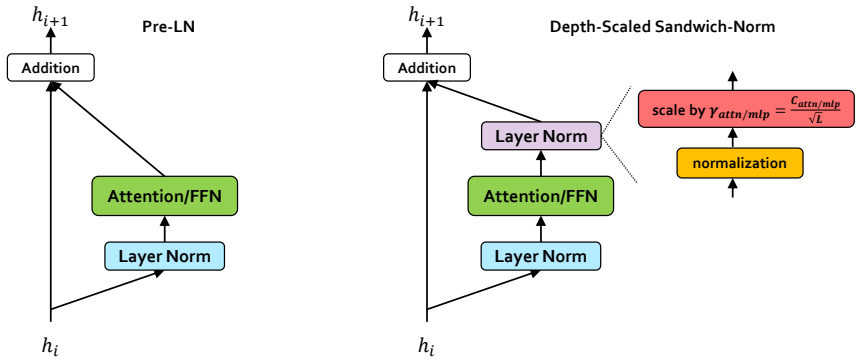


Figure 1: Structure comparison between Pre-Layer Norm (Pre-LN) and Depth-Scaled Sandwich-Norm (DSSN). DSSN applies normalization layers to both before and after the attention and FFN block, while Pre-LN only utilizes one normalization layer. DSSN also employs a depth-scaled initialization schema, which is not in the original sandwich norm.

2.2 Model Initialization

Existing works [53] observe that model initialization plays a crucial role in training stability and performance. Transformer-based LLMs widely adopt small initialization[53], which initialze all the weight with a normal distribution of standard deviation $\sqrt{\frac{2}{5d}}$, where d is the hidden dimension. It's also common practice to scale the weights of residual layers at initialization by a factor of $1/\sqrt{L}$ [57], where L is the number of layers.

Our findings suggest that scaling initialization by both model depth and width, using $\sqrt{\frac{1}{2dL}}$, leads to faster loss convergence and improved performance on downstream tasks. We call this initialization method TinyInit. We hypothesize that TinyInit achieves more consistent parameter scales across the model, which may facilitate optimization and convergence.

Research [66] indicates that embedding layers require different initialization strategies compared to other layers. Specifically, maintaining the standard deviation of embedding weights close to 1 may enhance training

stability. Our experimental results indicate that initializing with a standard deviation of 0.5 achieves good model performance.

2.3 Tokenizer

The design of the tokenizer significantly impacts model performance. An optimal vocabulary balances domain coverage (handling diverse tasks such as text, math, and code) with efficiency (encoding data with fewer tokens). Common methods use Byte-Pair Encoding (BPE) [62] and SentencePiece [40] build vocabularies by directly computing word frequencies across the entire training dataset. However, this approach suffers from domain imbalance, as common domains such as general text dominate the vocabulary, while specialized domains such as math and code remain underrepresented due to their limited data volume.

Pangu Ultra adopts a domain-aware vocabulary strategy. We perform independent frequency analyses across multiple domains including general Chinese, general English, code, and mathematics, generating distinct domain-specific vocabularies. These vocabularies are then merged and de-duplicated to form a unified vocabulary of 153,376 unique tokens, maintaining balanced representation across domains while preserving overall compression efficiency. Table 1 summarizes the detailed token distribution across different domains.

Table 1: Token distribution in the unified vocabulary of Pangu Ultra.

Domain	**Number of Tokens**	**Percentage (%)**
English	68,017	44.35
Chinese	41,053	26.77
Other	30,573	19.93
Latin-based languages	4,507	2.94
Arabic	2,755	1.80
Korean	2,733	1.78
Mathematics	2,139	1.39
Japanese	1,599	1.04
Total	**153,376**	**100.00**

3 Model Training

In this section, we present our training pipeline, which is similar to training state-of-the-art language models, e.g., DeepSeek-V3 [19] and Llama 3 [22]. The training process consists of three main stages: pre-training, long context extension, and post-training. Each stage has specific training strategies and data construction methods to gradually enhance the model capabilities.

3.1 Pre-training Stage

We first introduce the data construction in the pre-training of Pangu Ultra, followed by the details of data verification. Then we elaborate the practical approach for the long context extension. The detailed pre-training hyper-parameters are finally presented.

3.1.1 Data Construction

The pre-training corpus of Pangu Ultra contains high-quality and diverse 13.2T tokens produced by our tokenizer, as stated in Section 2.3. Table 2 shows the pre-training process is structured into three sequential phases: the *general* phase, the *reasoning* phase, and the *annealing* phase. These phases are designed to progressively develop general knowledge and linguistic capabilities, enhance reasoning skills, and further refine knowledge and behavior, respectively. The amount of data used in each phase is 12T, including 7.4T and 4.6T data in two distinct subphases, 0.8T, and 0.4T tokens.

In the initial general training phase, we utilize a corpus focused on developing broad linguistic capabilities and general knowledge. This stage primarily consists of English and Chinese data collected from a diverse range of sources, including web pages, books, encyclopedias, *etc*. Data from the multilingual and various industrial domains is also incorporated. Based on our data quality assessment in Section 3.1.2, we perfer to use higher-quality data in the second sub-phrase than the first.

Table 2: Data recipe of Pangu Ultra pre-training.

Dataset	General	Reasoning	Annealing
General English	54%	14%	21%
General Chinese	13%	6%	20%
Multi-lingual	8%	4%	3%
Instruction	2%	11%	20%
Math	6%	28%	18%
Code	17%	37%	18%

In the second reasoning phase, we increase the proportion of high-quality and diverse mathematical and coding data—raising it to over 60% of the corpus to enhance the reasoning capabilities of Pangu Ultra. The coding data includes both pure code and mixed text-code samples. The math data also involves a lot of English and Chinese texts. Moreover, LLM-generated synthetic data is widely incorporated to enrich the corpus.

The third annealing phrase is designed to help the model consolidate and effectively apply the knowledge and reasoning skills acquired in the previous stages. Therefore, we place greater emphasis on instruction data, which accounts for approximately 20% of the corpus. We curate in-house question banks covering a wide range of topics and construct both short and long chain-of-thought (CoT) responses. These reasoning paths are carefully refined to ensure clarity and logical coherence.

Overall, the pre-training data for Pangu Ultra is carefully designed to ensure high quality, diversity, and minimal redundancy. We assign quality and difficulty labels to the data and adopt a curriculum-based sampling strategy for the reasoning data across all three phases—progressing from simpler examples to more complex ones throughout the training cycle.

3.1.2 Data Quality Assessment

Data quality assessment plays a crucial role in enhancing the overall quality of the data. Training Pangu Ultra employs both rule-based heuristics and model-based evaluation to enhance data quality.

For model-based quality assessment, we leverage the Pangu series as the base model. To better align quality evaluation with human value judgments, we fine-tune the model using a manually annotated dataset. The fine-tuned evaluator is then applied to a large-scale pre-training corpus exceeding 10T tokens. Data samples are scored across multiple dimensions, including cleanliness, fluency, educational value, and richness. These annotated scores are then used in a prioritized sampling strategy, where higher-quality samples are assigned higher sampling probabilities.

To validate the effectiveness of our data quality assessment, we conducted an ablation study using a proxy model with 2.6 billion parameters. Empirical results show that, to achieve comparable performance, the model trained on low-scoring data required 1.6× more tokens than the one trained on high-quality high-scoring data. Therefore, high data quality is important for improving training efficiency.

3.1.3 Pre-training Parameters

Pangu Ultra is trained using AdamW optimizer [48] with a weight decay of 0.1 and epsilon is set to 1×10^{-8}. The momentum parameters are set to $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The gradient clipping norm is set to 1.0. To improve the training stability and overall performance, the pre-training of Pangu Ultra is organized into the following phases:

0T–7.4T tokens The sequence length is set to 4K (RoPE base = 1×10^4). The batch size increase from 1,024 to 1,536 (at 1.2T) and 2,048 (at 1.9T). The increased batch size improves training efficiency and throughput. The learning rate follows a cosine decay from 1×10^{-4} to 1×10^{-5} with 4,000 warmup steps to ensure stable early training.

7.4T–12.0T tokens The sequence length remains at 4K with a batch size of 2,048. The learning rate is fixed at 1×10^{-5} in this phase.

12.0T–12.8T tokens The sequence length increases to 8K (RoPE base = 1×10^5). The batch size is reduced to 1,536. The learning rate decays from 1×10^{-5} to 7.5×10^{-6} using cosine scheduling.

3.2 Long Context Extension

The ability of LLMs to understand long context inputs is critical in long-thinking process and practical applications. In the final stages of pre-training, Pangu Ultra is trained on long sequence data to support a maximum context length of 128K. The training consists of two progressive phases: the first phase expands the context length to 32K, and the second phase further expands it to 128K.

Rotary Position Embedding (RoPE) [64] is the core module for supporting ultra-long input sequences. Existing open-source LLMs typically extend context length by either increasing the base frequency in RoPE [64, 32] or by adopting methods such as YaRN [55, 22, 19]. Our findings show that both methods perform similarly well if the hyper-parameters are correctly chosen, and we adopt the increased base frequency method in Pangu Ultra. To determine the base frequency in RoPE for long-context extension, we evaluate the offline performance of "Needle In A Haystack" (NIAH) with different base frequencies at the target sequence length, and select the one with the best result. This ensures a relatively low initial loss in long-context training. In practice, the selected base frequency for 32K is 1.6×10^6, and for 128K is 2.56×10^7. Detailed hyper-parameters of Pangu Ultra long context training are summarized below:

8K to 32K phase The sequence length is expanded to 32K (RoPE base = 1.6×10^6). The batch size is 384 with a learning rate of 7.5×10^{-6}, matching the final learning rate from the previous post-training stage.

32K to 128K phase The sequence length is further expanded to 128K (RoPE base = 2.56×10^7). The batch size is reduced to 96. The learning rate remains 7.5×10^{-6}.

3.3 Post-training Alignment

In the post-training stage, Pangu Ultra is aligned with human preferences through Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL). This stage focuses on constructing high-quality, diverse instruction data and designing scalable, efficient training strategies.

3.3.1 Post-training Data

In constructing post-training data, we emphasize the data quality, diversity, and complexity. The data pool is curated from a wide range of domains and task types, including general question answering, AI-generated content (AIGC), text classification and analysis, programming, mathematics, logical reasoning, and tool usage. These tasks cover application areas such as finance, healthcare, and public services. Data sources span open-source instruction datasets, real-world industrial queries, and synthetic problems derived from the pre-training corpus.

To promote data diversity, data samples are selected along two orthogonal dimensions, guided by the entropy law [74]: domain and task type. Hierarchical tagging models with varying levels of granularity are used to support balanced data sampling. Data quality is managed through a combination of rule-based validation and model-based validation, which helps eliminate low-quality or ambiguous samples.

To better stimulate the reasoning capabilities of Pangu Ultra, a large portion of the post-training data, approximately six-sevenths, consists of reasoning tasks such as mathematics, coding, and logic. The post-training data covers a range of complexities, with a focus on moderately to highly challenging tasks.

3.3.2 Post-training Strategy

In the post-training stage, Pangu Ultra was first trained with SFT to establish preliminary instruction-following capabilities. Following SFT, we apply RL with outcome-based reward signals to further enhance reasoning, alignment, and instruction-following abilities of Pangu Ultra.

We implement a latency-tolerant reinforcement learning framework optimized for the Ascend infrastructure, which will be detailed in a future report. The framework enables efficient large-scale policy optimization on Ascend. To guide the RL process, we implement a hybrid reward system that provides task-specific feedback for mathematics, coding, and general problem-solving. This hybrid reward system combines deterministic reward signals and model-based evaluations to facilitate stable and efficient policy optimization.

4 Training System

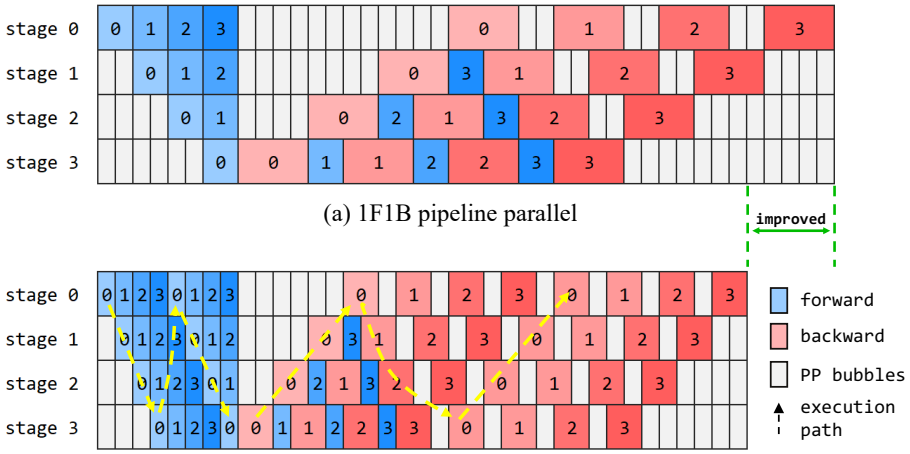
Training our Pangu Ultra with 135B parameters on 13.2 trillion tokens necessitates the need to ensure training stability and efficiency in large-scale computing cluster. In this section, we elaborate the details of our training system from two important perspectives: parallelization strategies and system-level optimization techniques, in Section 4.2 and Section 4.3. Overall, we achieve over 52% Model FLOPs Utilization (MFU) when training Pangu Ultra on 8,192 Ascend NPUs.

4.1 Computing Setup

A computing cluster with 8,192 Ascend Neural Processing Units (NPUs) [5, 6] is deployed to train Pangu Ultra. Each node in the cluster houses 8 NPUs, interconnected via Huawei Cache Coherence System (HCCS) using a full-mesh topology, and each device is equipped with 64GB Memory. Inter-node communication is facilitated through RDMA over Converged Ethernet (RoCE) fabric, leveraging 200 Gbps interconnects for communication between NPUs across different nodes.

4.2 Parallelism Strategies for Model Scaling

In order to scale model training[1], we leverage a combination of different parallelism strategies to distributes the model across multiple NPUs, including Data Parallelism (DP) [43], Tensor Parallelism (TP) [63], Sequence Parallelism (SP) [39], and Pipeline Parallelism (PP) [30, 51]. For Pangu Ultra, 128-way DP with ZERO [58] is performed to reduce the memory cost of model parameters and the associated optimizer states. 8-way TP is applied to leverage the high intra-node bandwidth for efficient activation transfer, while 8-way PP is adopted to utilize inter-node connections, since it only requires transmitting activations at the partition boundaries. However, as mentioned in existing studies [35, 30, 51, 56], pipeline parallelism encounters severe PP bubbles when the training cluster scales up, primarily due to batch size constraints [35]. For one-forward-one-backward (1F1B) PP scheduling, the bubble ratio is defined as $\frac{p-1}{p-1+n}$, where p represents the number of pipeline stages and n denotes the number of micro batches for every DP. The ratio represents the idle time of accelerators, as shown in Figure 2. A large-scale training cluster increases the number of DPs, which in turn reduces the number of micro batches assigned to each DP due to batch size constraints, leading to a significant increase in the bubble ratio. Therefore, minimizing bubble ratio is crucial for improving system efficiency. Under such circumstances, we employ interleaved pipeline-parallel scheduling with 6-way virtual PP stages on each device [52] and manage to reduce it from 30.45% to 6.8%. Through careful tuning of load balancing across PP and VPP stages, we are able to achieve approximately 43% MFU on an 8,192 NPU cluster as a baseline.



(a) 1F1B pipeline parallel

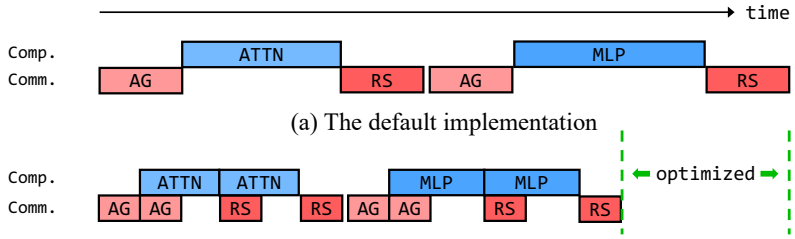
(b) 1F1B interleaved pipeline parallel

Figure 2: Pipeline parallelism and the interleaved pipeline-parallel scheduling.

[1]The training of Pangu Ultra is supported by MindSpeed [8] and Megatron [7, 63] framework, which provides comprehensive parallel strategies and system optimization methods.

4.3 System Optimization

Based on the optimizations outlined in Section 4.2 that achieved 43% MFU, additional system-level enhancements are implemented to push training efficiency to new heights. Through a combination of kernel fusions, context parallelism via subsequence partitioning, data caching and sharing mechanisms, and other refinements, Pangu Ultra benefits from a significant improvement in training efficiency. These comprehensive optimizations enable the system to achieve over 52% MFU, representing a 9% relative improvement compared to the baseline configuration mentioned in Section 4.2.



(a) The default implementation

(b) The MC2 implementation

Figure 3: A Comparison of the default transformer computation and the MC2 method. Note that in actual training, communication and computation tasks are fused into a single kernel in MC2.

4.3.1 Kernel Fusion

Kernel fusion is widely adopted in LLM training to enhance efficiency. It combines multiple operations into a single kernel, reducing the number of data accesses to global memory [17]. During the training phase of Pangu Ultra, key operators are fused, resulting in significant improvements in hardware utilization and overall training efficiency.

MC2 - Merged Compute and Communication Tensor parallelism, when combined with sequence parallelism, introduces All-Gather (AG) and Reduce-Scatter (RS) communication operations for exchanging input and output activations across distributed devices. This approach exhibits a direct dependency between matrix multiplication (MatMul) and AG/RS communications, which fundamentally constrains the overlapping of TP communication with computational workflows. The MC2 is implemented [2, 3] to tackle this challenge by fusing MatMul computations with communication operations. It decomposes large computation and communication tasks into fine-grained subtasks and employs pipelined execution to maximize overlap between communication and computation. Thus, MC2 significantly reduces communication latency and improves hardware utilization (Figure 3).

NPU Fusion Attention Training LLMs with long sequence length suffers from quadratic memory and computational requirements in self-attention mechanisms as sequence length grows. To address these challenges, Flash Attention (FA) has emerged as a standard technique in LLM training owing to its superior performance [18, 17]. Pangu Ultra leverages a self-attention fusion operator, called NPU Fusion Attention (NFA)[9], which is specifically optimized for Ascend NPUs, offering system-level improvements across a wide range of self-attention computation scenarios.

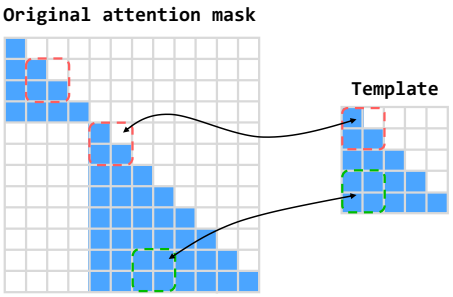
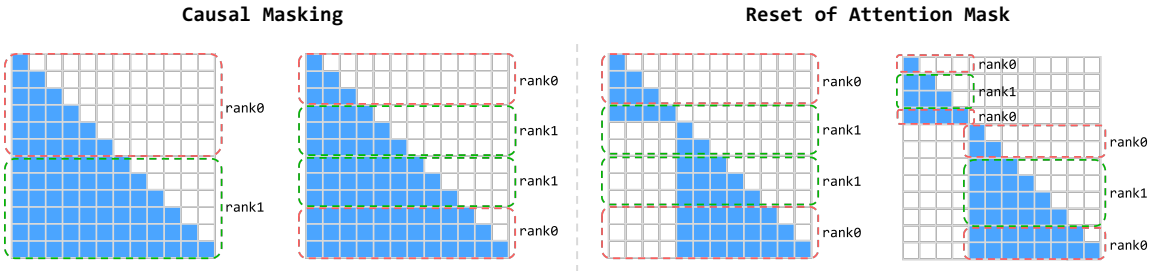


Figure 4: Examples of attention mask compression for the NFA operator.

It is worth mentioning that Pangu Ultra uses a reset attention mask strategy to prevent self-attention between different documents within a sequence. This requires calculating the corresponding attention mask for every sequence, leading to significant memory and computational overhead. To mitigate the time and memory requirements of generating attention masks, the NFA operator employs a mask compression optimization. As shown in Figure 4, NFA utilizes a 2048×2048 causal mask as a template to construct the computational mask within the fusion attention operator. For every iteration, Pangu Ultra retrieves the actual sequence length based on the position of the end-of-document (eod) token, which is then provided as input to the NFA operator to accelerate the computation of self-attention. The detailed usage of NFA is provided in the Ascend documentation [9].

Other Kernel Fusions for Efficiency In addition to MC2 and NPU-optimized fused attention, we also integrate a series of kernel fusion optimizations within key components such as RMSNorm [77], SwiGLU [60], and rotary positional embeddings (RoPE) [64], as well as critical processes including gradient accumulation and PP send/receive communications. These fusion operators are designed to reduce kernel launch and memory access overheads, while maintaining high numerical precision and enhancing overall training performance.



(a) Original (b) Megatron (c) Megatron (d) Ours

Figure 5: Examples of the mechanism of sub-sequence partitioning for context parallelism.

4.3.2 Optimization for Long Context Training

Scaling long-context capabilities is becoming increasingly important for applications such as long document summarization and conversational AI. However, training on long sequences presents several challenges in terms of both time and memory complexity. To improve the efficiency of long-context training, we propose two key strategies, as outlined below.

Sub-Sequence Partitioning for Context Parallelism Context parallelism (CP) is an crucial approach for the training of very long sequences, that divides the input sequence into segments to reduce memory consumption [44, 33]. Yet, with causal masking, simply splitting the sequence into CP chunks results in a severely imbalanced workload for Ring Self-Attention (RSA) [44] (as shown in Figure 5(a)). Megatron-LM addresses this issue by splitting the sequence into $2 \times CP$ chunks, where each rank receives chunks from both the top and bottom, thus balancing the workload within a CP group (Figure 5(b)) [7]. However, this method still results in an imbalanced workload when the attention mask is reset (Figure 5(c)). Therefore, in training with 128k-long contexts, we propose a load-balanced partitioning strategy for CP training, where each rank is responsible for computing two chunks within each subsequence (Figure 5(d)).

Fast Mask Generation and Data Reuse When scaling the training sequence of Pangu Ultra up to 128k, the generation of the attention mask or the calculation of the actual sequence length still incurs a non-negligible performance overhead. Additionally, in the training scenario with reset attention masks, each VPP stage is required to retrieve the corresponding mask or actual sequence length in every iteration, resulting in redundant computations and increased overhead. We optimize these problems by (1) using efficient NPU operators to compute the attention mask, instead of constructing it on the CPU, thus accelerating mask generation and eliminating the need for data transfer between the CPU and NPU, and (2) enabling cross-VPP stage mask sharing, where attention masks are generated by the first stage (VPP0) and shared across different VPP stages on the same rank, thereby avoiding redundant mask computations and memory cost.

5 Results

In this section, we discuss the evaluation results of Pangu Ultra, including pre-training performance and post-training outcomes. In addition, we provide comprehensive ablation studies that exam the model architecture and further discuss the observations of training Pangu Ultra.

5.1 Pre-Training Training Loss Curve

Figure 6 shows the training loss curve of Pangu Ultra during the entire pre-training. Each segment in the loss curve corresponds to one training stage, as described in Section 3.1.3. The loss curves demonstrate consistent descending trends across all training stages. For the second interval, although the descent rate moderated due to a constant learning rate, the performance metrics continued to show steady improvement throughout this interval.

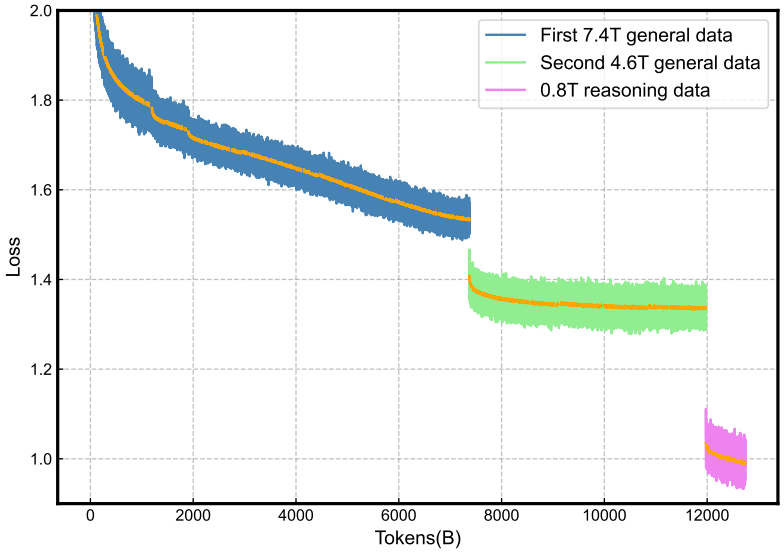


Figure 6: The training loss curve of Pangu Ultra during the pre-training stage.

Zero loss spike As shown in Figure 6, **no loss spikes** occur throughout the entire pre-training process. While such spikes are common in LLM training [66], the absence of them here underscores the importance of our depth-scaled sandwich norm and TinyInit in ensuring stable training. The negative effect of loss spike to the model performance will be further elaborated in Section 5.4.1.

5.2 Pre-Training Stage

Benchmarks We evaluate Pangu Ultra base model across multiple domains using open-source benchmarks, including language understanding, question answering, code generation, and math problem solving. The evaluation mainly uses English and Chinese test sets, with some additional multilingual benchmarks for broader coverage.

- Language understanding: We employ *Hellaswag* [76] and *Winogrande* for contextual reasoning tasks, *DROP* [21], *RACE* [42], and *ARC* [15] series for comprehensive reading comprehension evaluation, along with *PIQA* [12], *Natural Questions* [41] and *TriviaQA* [37] to assess knowledge retrieval.
- Question answering: The assessment includes *C-Eval* [31] for Chinese knowledge, *MMLU* [27] and its advanced variant *MMLU-Pro* [70] for English domain knowledge, supplemented by *BigBenchHard* [65] to evaluate creative problem-solving
- Code generation and understanding: We utilize *HumanEval* [13] and *MBPP* [10] for standard code generation tasks, while *CruxEval* [26] for code understanding and reasoning.

- Mathematical Reasoning : We measure skills with *CMath* [71] and *GSM8K* [16] for fundamental arithmetic and simple problems, *MATH* [28] for advanced mathematical reasoning, and *MGSM* [61] for multilingual math problem solving.

Baselines & Comparison Settings We compare Pangu Ultra against several strong baselines covers both dense models (Qwen2.5-72B, Llama-405B) and MoE architectures (DeepSeek-V3). For base models, the majority of our evaluations employ few-shot inputs, with a minority using zero-shot prompts. We evaluate most benchmarks with gold answers through exact matching, while employing execution-based verification for code generation tasks.

Evaluation Results In Table 3, we compare the pre-trained base model of Pangu Ultra with other leading models. Overall, Pangu Ultra achieves state-of-the-art performance on most general English benchmarks and all Chinese benchmarks. While it trails DeepSeek V3 on code and math-related tasks, it performs competitively on these domains.

A closer examination reveals that Pangu Ultra excels on Chinese benchmarks, surpassing both Qwen 2.5 72B and DeepSeek V3, the current best-performing Chinese model. In addition, when compared to Llama 3.1 405B, Pangu Ultra achieves better scores on most of the challenging benchmarks, while utilizing only about 29% of the training FLOPs required by Llama 405B. These results suggest the effectiveness of our model architecture and the high quality of our training data.

Table 3: Comparison of Pangu Ultra and other representative models across a diverse set of benchmarks for evaluating language, coding and mathematical skills. Bold values represent the best results in each line, and underlined values represent Pangu Ultra is the best among dense models.

	Benchmark (Metric)	**# Shots**	**Qwen2.5 72B Base**	**Llama-3.1 405B Base**	**DeepSeek V3 Base**	**Pangu Ultra Base**
	Architecture	-	Dense	Dense	MoE	Dense
	# Activated Params	-	72B	405B	37B	135B
	# Total Params	-	72B	405B	671B	135B
English	BBH (EM)	3-shot	79.8	82.9	**87.5**	79.1
	MMLU (EM)	5-shot	85.0	84.4	**87.1**	85.4
	MMLU-Pro (EM)	5-shot	58.3	52.8	**64.4**	63.1
	DROP (F1)	3-shot	80.6	86.0	**89.0**	61.0
	ARC-Easy (EM)	25-shot	98.4	98.4	98.9	**100.0**
	ARC-Challenge (EM)	25-shot	94.5	95.3	95.3	**97.0**
	HellaSwag (EM)	10-shot	84.8	89.2	88.9	**99.0**
	PIQA (EM)	0-shot	82.6	85.9	84.7	**98.0**
	WinoGrande (EM)	5-shot	82.3	85.2	84.9	**91.0**
	RACE-Middle (EM)	5-shot	68.1	74.2	67.1	**97.0**
	RACE-High (EM)	5-shot	50.3	56.8	51.3	**97.0**
	TriviaQA (EM)	5-shot	71.9	82.7	82.9	**90.5**
	NaturalQuestions (EM)	5-shot	33.2	41.5	40.0	**52.7**
	AGIEval (EM)	0-shot	75.8	60.6	79.6	**80.4**
Code	HumanEval (Pass@1)	0-shot	53.0	54.9	65.2	**81.1**
	MBPP (Pass@1)	3-shot	72.6	68.4	**75.4**	72
	CRUXEval-I (EM)	2-shot	59.1	58.5	**67.3**	61.8
	CRUXEval-O (EM)	2-shot	59.9	59.9	**69.8**	61.5
Math	GSM8K (EM)	8-shot	88.3	83.5	**89.3**	**89.3**
	MATH (EM)	4-shot	54.4	49.0	61.6	**62.5**
	MGSM (EM)	8-shot	76.2	69.9	**79.8**	75.1
	CMath (EM)	3-shot	84.5	77.3	**90.7**	78.2
Chinese	CLUEWSC (EM)	5-shot	82.5	83.0	82.7	**95.0**
	C-Eval (EM)	5-shot	89.2	72.5	90.1	**90.3**
	CMMLU (EM)	5-shot	89.5	73.7	88.8	**91.7**
	CMRC (EM)	1-shot	75.8	76.0	76.3	**86.0**
	C3 (EM)	0-shot	76.7	79.7	78.6	**99.0**
	CCPM (EM)	0-shot	88.5	78.6	92.0	**93.0**

5.3 Post-Training and Reasoning Capability

Benchmarks We conduct a comprehensive evaluation of the Pangu Ultra's capabilities over reasoning and non-reasoning tasks:

- Sophisticated reasoning tasks encompass three specialized subcategories: mathematical competence measured by *AIME 2024* [49] and *MATH-500*, Coding competition benchmarks *LiveCodeBench* [34] and scientific reasoning task *GPQA Diamond* [59];
- General language comprehension and reasoning capabilities, represented by *MMLU-Pro* [24], *Arena Hard* [45].

Baselines & Comparison Settings We compare Pangu Ultra against strong baselines including GPT-4o-0513, reasoning models DeepSeek-R1, Hunyuan-T1 and large dense models, Qwen2.5-72B-Instruct and Mistral-Large 2. We use Pass@1 averaged over multiple independent runs as the evaluation metric to assess the performance.

Evaluation Results In Table 4, we compare the evaluation results of Pangu Ultra with other baseline models. Pangu Ultra achieves state-of-the-art performance on the reasoning benchmarks including AIME 2024, MATH-500, GPQA and LiveCodeBench, while maintaining strong capabilities in general language comprehension tasks.

When compared to dense LLMs (Qwen and Mistral-Large 2), Pangu Ultra shows particularly significant advantages in reasoning tasks. This superior performance stems from the 0.8T reasoning-focused data used in pre-training (Section 3.1.3). The reasoning-enhanced base model substantially benefits subsequent post-training phases.

Table 4: Comparison of Pangu Ultra models and other representative models across benchmarks. † indicates results from Artificial Analysis [1].

Model	AIME 2024	MATH-500	GPQA Diamond	LiveCode Bench	ArenaHard	MMLU-pro
GPT-4o-0513	9.3	74.6	49.9	32.9	80.4	72.6
Qwen2.5-72B	16.0	83.1	49	27.6	81.2	72.0
Mistral-Large 2†	11.0	73.6	48.6	29.3	-	69.7
Hunyuan-T1	79.8	96.2	69.3	64.9	91.9	**87.2**
DeepSeek-R1	79.8	97.3	71.5	65.9	**92.3**	84.0
Pangu Ultra	**80.8**	**97.4**	**74.2**	**66.5**	91.5	84.4

5.4 Ablation Studies

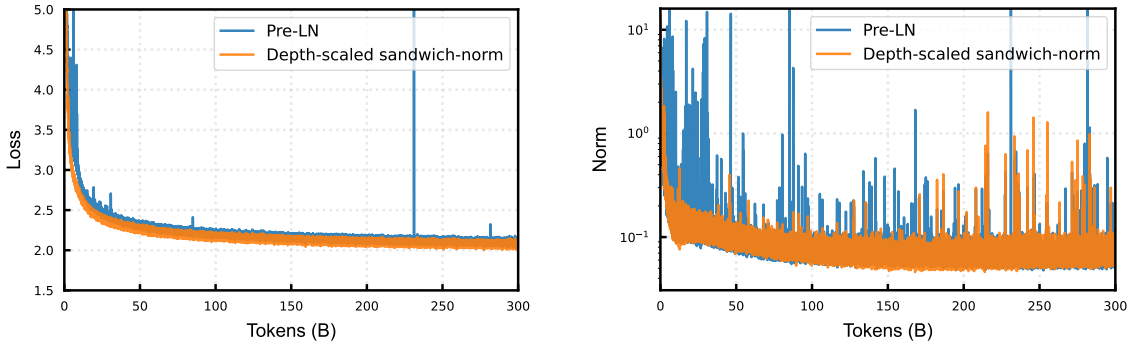
This section presents additional ablation studies of the model architecture and analyzes key training behaviors to facilitate a deeper understanding and discussion of dense LLM training.

5.4.1 Depth-scaled Sandwich-norm

We conducted experiments to validate the effectiveness of depth-scaled sandwich norm compared to pre-norm architectures. Using a dense Transformer model with 13 billion parameters trained on 300 billion tokens with identical hyperparameters for both configurations, we observe significant improvements.

Figure 7 shows the depth-scaled sandwich-norm architecture stabilizes gradient norms and effectively eliminates loss spikes, leading to faster training convergence. We evaluated performance on two composite benchmarks: EN basic, consisting of multiple English benchmarks, and ZH basic, representing Chinese benchmarks. Additional evaluation using LAMBADA [54] (English) and WPLC [23] (Chinese) next-token prediction tasks confirmed the advantage of applying depth-scaled sandwich-norm. The results clearly suggest that preventing loss spikes during pre-training is crucial for optimal model performance.

To further ablate the effect of our depth-scaled factor in RMSNorm initialization, we compare with the plain sandwich-norm that does not have the $\sqrt{L}$ scaling factor in Eq. (1). Here, we use a proxy model containing 1.6



(a) Loss (b) Gradient norm

Figure 7: Pre-training loss and gradient norm for a 13B model using Pre-LN and Depth-Scaled Sandwich-Norm (DSSN). The curves with Pre-LN has significant spikes, which harm the trained model, while the curves of DSSN are much smoother.

Table 5: Performance comparison between Pre-LN and Depth-scaled Sandwich-Norm.

Model	Tokens (B)	EN basic	ZH basic	LAMBADA	WPLC
Pre-LN	300	0.42	0.52	0.675	0.194
Depth-scaled sandwich-norm	300	0.45	0.54	0.693	0.224

billion parameters and 94 layers, which has the same depth with Pangu Ultra. By using this proxy model, we examine the effectiveness of sandwich-norm on training very deep Transformers. In Figure 8, we can observe some loss spikes with the plain sandwich-norm, but our depth-scaled sandwich-norm can be trained smoothly, and attains lower loss.

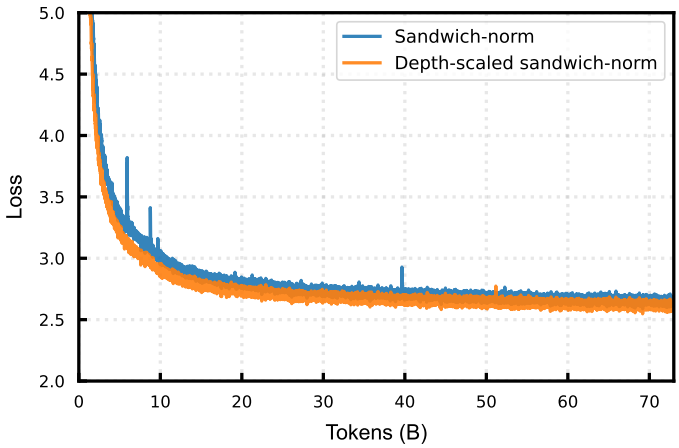


Figure 8: Pre-training loss for a 94-layer 1.6B model using original and depth-scaled sandwich-norm. The original sandwich-norm still suffers loss spikes during training.

5.4.2 Tiny Initialization

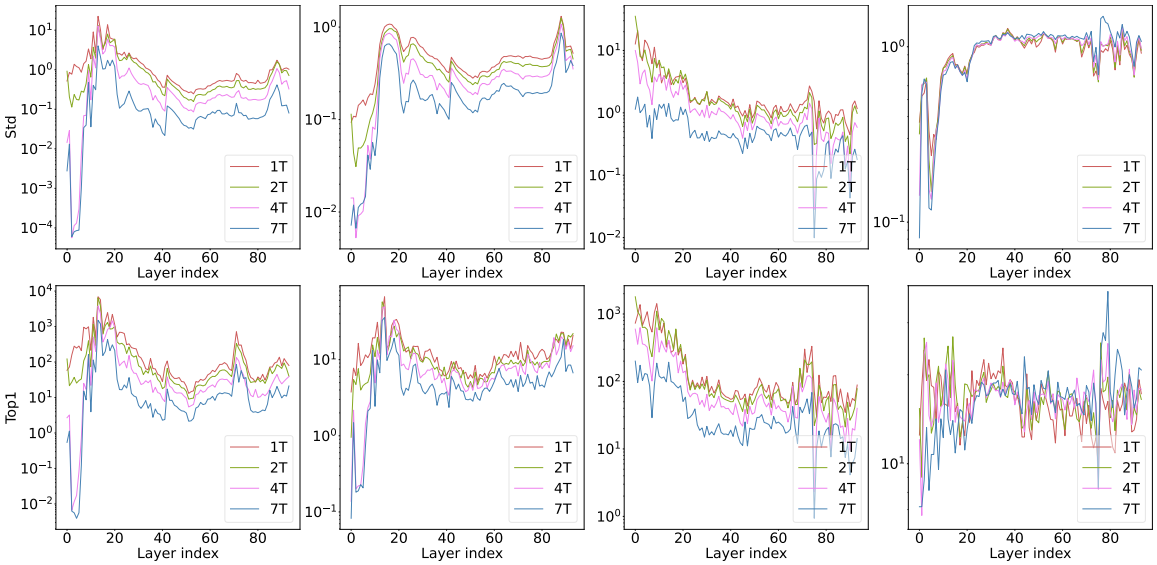
We conduct experiments to study the effectiveness of TinyInit proposed in Section 2.2. After being trained on 102 billion tokens, Pangu Ultra initialized with TinyInit strategy, with standard deviation $\sqrt{\frac{1}{2dL}}$), performs significantly better than the baseline model that utilizes traditional initialization, whose standard deviations are $\sqrt{\frac{2}{5d}}$ and $\sqrt{\frac{2}{5dL}}$. The results are shown in Table 6. BIG-bench (aug) is a test set developed internally through data augmentation of the original BIG-bench, designed to mitigate the impact of test set leakage.

Table 6: Performance comparison of traditional initialization and TinyInit.

Model	Tokens (B)	EN basic	ZH basic	LAMBADA	WPLC	C-Eval	MMLU	BIG-bench (aug)
Baseline	102	0.444	0.538	0.694	0.229	0.476	0.473	0.357
TinyInit	102	0.456	0.537	0.727	0.257	0.524	0.502	0.384

5.4.3 Layer Statistics of Pangu Ultra

Stable activation scale Figure 9 presents the activation patterns of attention and FFN modules across Transformer layers, showing the mean, standard deviation, and top-1 activation values. The activation distributions demonstrate stability, with standard deviations maintaining consistent scales throughout pre-training while preserving a clear layer-wise pattern. Our analysis reveals the presence of "super activations", whose magnitude reaches 10^3 magnitude in shallow layers, a phenomenon consistent with findings in the Llama model [75]. Notably, Figure 9 illustrates that these top-1 activation values progressively decrease with layer depth, indicating that their influence becomes relatively small on the final output.



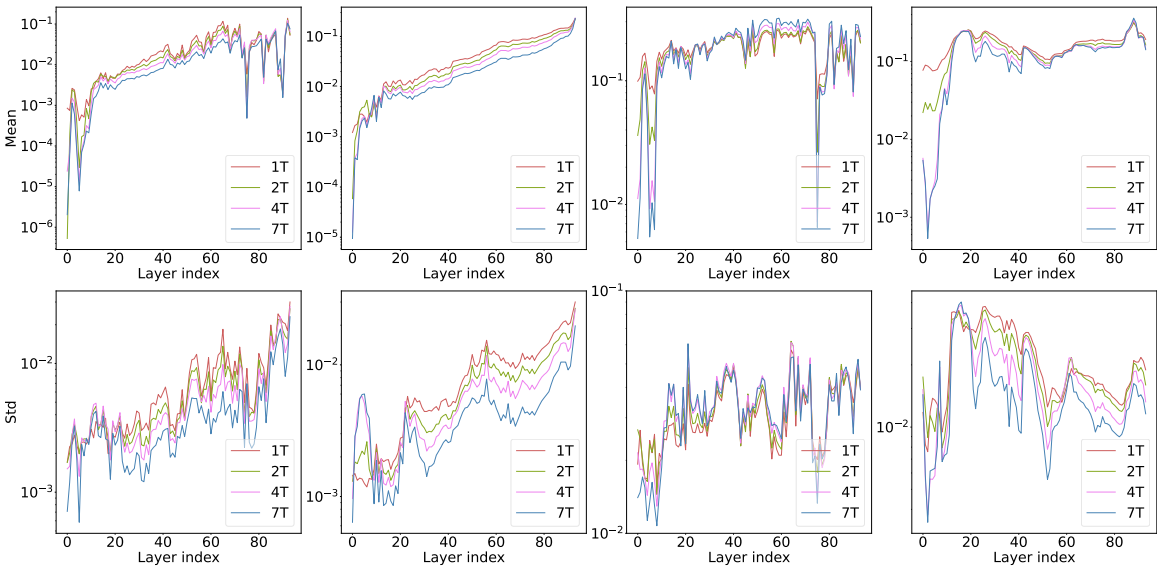
(a) Down projection (b) Up & Gate projection (c) Attention output projection (d) Attention QKV projection

Figure 9: Activation of attention and FFN modules. Mean, standard deviation, and top-1 value of activations are included. Each line represents different training tokens from 1T, 2T, 4T to 7T. The "Std" row shows the stable activation scale across layers. The "Top 1" row shows the existence of the "super activations" in down projection and attention output projection, with magnitudes falling within a reasonable range and comparable to those observed in the LLaMA model [75].

Layer-wise patterns of depth-scaled sandwich norm. Figure 10 presents the distribution of scaling parameters γ across all sandwich-norm layers, revealing several key observations: All four LayerNorm γ parameters exhibit decreasing mean/standard deviation during training, consistent with weight decay effects. Post-norm γ values show layer-dependent patterns: The standard deviation of post-norm γ increases substantially with layer depth. Pre-norm γ maintains relatively constant standard deviation across layers. This pattern suggests an intriguing model behavior: shallow layers rely primarily on residual connections, while deeper layers progressively emphasize transformer layer outputs as the scaling factor γ grows in magnitude.

6 Conclusion

We present Pangu Ultra, a dense language foundation model with 135 billion parameters trained on Ascend NPUs. To address challenges in training large-scale deep models, we propose depth-scaled sandwich-norm, enabling Pangu Ultra to achieve remarkable training stability without significant loss spikes. After being pre-trained on 13.2 trillion tokens and long context extension on 8,192 Ascend NPUs, our model further



(a) Post-norm after attention (b) Post-norm after FFN (c) Post-norm before attention (d) Post-norm before FFN

Figure 10: Distribution of sandwich-norm's γ parameter. Mean and standard deviation are included. Each line represents different training tokens from 1T, 2T, 4T to 7T. There is a clear layer-wise pattern of the two post-norms: the mean and std value of γ increase with depth. Larger post-norm γ indicates deeper layers emphasize more on transformer outputs instead of residual connections.

enhances its reasoning capabilities through Supervised Fine-Tuning and Reinforcement Learning. Extensive experiments lead to the observation that Pangu Ultra not only surpasses state-of-the-art dense LLMs like Llama 405B and Mistral Large 2 but also delivers competitive performance against larger sparse models such as DeepSeek-R1. These results highlight the efficacy of our architectural and systemic optimizations, paving the way for future advancements in scalable and efficient LLM training. In addition, our experience demonstrates that the Ascend NPUs are capable of training dense models with hundreds of billions of parameters.

References

[1] Artificial analysis. https://artificialanalysis.ai/.

[2] Ascend mc2. https://gitee.com/qingfenxiaochong/MindSpeed/blob/master/docs/features/mc2.md.

[3] Ascend mc2. https://www.hiascend.com/developer/techArticles/20240613-1.

[4] Flash attention. https://github.com/Dao-AILab/flash-attention.

[5] Huawei atlas 800t a2. https://e.huawei.com/cn/products/computing/ascend/atlas-800t-a2.

[6] Huawei atlas 800t a2 technical specifications. https://support.huawei.com/enterprise/en/doc/EDOC1100349804/2bf2c017/technical-specifications?idPath=23710424|251366513|22892968|252309113|254184887.

[7] Megatron-lm. https://github.com/NVIDIA/Megatron-LM.

[8] Mindspeed. https://gitee.com/ascend/MindSpeed.

[9] Npu fusion attention. https://www.hiascend.com/document/detail/zh/Pytorch/60RC1/apiref/apilist/ptaoplist_000139.html.

[10] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *ArXiv*, abs/2108.07732, 2021.

[11] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[12] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2019.

[13] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mo Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, Suchir Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.

[14] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

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

[16] Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021.

[17] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

[18] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[19] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng

Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025.

[20] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. Cogview: Mastering text-to-image generation via transformers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19822–19835. Curran Associates, Inc., 2021.

[21] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[22] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[23] Huibin Ge, Chenxi Sun, Deyi Xiong, and Qun Liu. Chinese wplc: A chinese dataset for evaluating pretrained language models on word prediction given long-range context. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[24] Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, et al. Are we done with mmlu? *arXiv preprint arXiv:2406.04127*, 2024.

[25] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[26] Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida Wang. Cruxeval: A benchmark for code reasoning, understanding and execution. *ArXiv*, abs/2401.03065, 2024.

[27] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300, 2020.

[28] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Xiaodong Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *ArXiv*, abs/2103.03874, 2021.

[29] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[30] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 103–112, 2019.

[31] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Fanchao Qi, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *ArXiv*, abs/2305.08322, 2023.

[32] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024.

[33] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models, 2023.

[34] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[35] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, Yulu Jia, Sun He, Hongmin Chen, Zhihao Bai, Qi Hou, Shipeng Yan, Ding Zhou, Yiyao Sheng, Zhuo Jiang, Haohan Xu, Haoran Wei, Zhang Zhang, Pengfei Nie, Leqi Zou, Sida Zhao, Liang Xiang, Zherui Liu, Zhe Li, Xiaoying Jia, Jianxi Ye, Xin Jin, and Xin Liu. Megascale: Scaling large language model training to more than 10,000 gpus, 2024.

[36] Cameron R Jones and Benjamin K Bergen. Large language models pass the turing test. *arXiv preprint arXiv:2503.23674*, 2025.

[37] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *ArXiv*, abs/1705.03551, 2017.

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

[39] Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models, 2022.

[40] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[41] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc V. Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[42] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. Race: Large-scale reading comprehension dataset from examinations. *ArXiv*, abs/1704.04683, 2017.

[43] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: Experiences on accelerating data parallel training. *CoRR*, abs/2006.15704, 2020.

[44] Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, and Yang You. Sequence parallelism: Long sequence training from system perspective. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2391–2404, Toronto, Canada, July 2023. Association for Computational Linguistics.

[45] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, April 2024.

[46] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[47] Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *EMNLP (1)*, pages 5747–5763. Association for Computational Linguistics, 2020.

[48] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[49] MAA. Codeforces. American Invitational Mathematics Examination - AIME 2024, 2024. https://maa.org/math-competitions/american-invitational-mathematics-examination-aime.

[50] William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers, 2025.

[51] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. Pipedream: generalized pipeline parallelism for DNN training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019*, pages 1–15. ACM, 2019.

[52] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. In

Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '21, New York, NY, USA, 2021. Association for Computing Machinery.

[53] Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

[54] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and R. Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *ArXiv*, abs/1606.06031, 2016.

[55] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models, 2023.

[56] Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. Zero bubble pipeline parallelism, 2023.

[57] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[58] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM, 2020.

[59] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[60] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[61] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. *ArXiv*, abs/2210.03057, 2022.

[62] Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. Byte pair encoding: A text compression scheme that accelerates pattern matching. 1999.

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

[64] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[65] Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[66] Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the pre-training of large language models, 2024.

[67] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

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

[69] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. Learning deep transformer models for machine translation. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, Florence, Italy, July 2019. Association for Computational Linguistics.

[70] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max W.F. Ku, Kai Wang, Alex Zhuang, Rongqi "Richard" Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *ArXiv*, abs/2406.01574, 2024.

[71] Tianwen Wei, Jian Luan, W. Liu, Shuang Dong, and Bin Quan Wang. Cmath: Can your language model pass chinese elementary school math test? *ArXiv*, abs/2306.16636, 2023.

[72] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[73] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process, 2024.

[74] Mingjia Yin, Chuhan Wu, Yufei Wang, Hao Wang, Wei Guo, Yasheng Wang, Yong Liu, Ruiming Tang, Defu Lian, and Enhong Chen. Entropy law: The story behind data compression and llm performance. *arXiv preprint arXiv:2407.06645*, 2024.

[75] Mengxia Yu, De Wang, Qi Shan, Colorado Reed, and Alvin Wan. The super weight in large language models. *ArXiv*, abs/2411.07191, 2024.

[76] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019.

[77] Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019.

A Contributions and Acknowledgments

Core Contributors Yichun Yin, Wenyong Huang, Kaikai Song, Yehui Tang, Xueyu Wu, Wei Guo, Peng Guo, Yaoyuan Wang, Xiaojun Meng, Yasheng Wang, Dong Li, Can Chen, Dandan Tu, Yin Li, Fisher Yu, Ruiming Tang, Yunhe Wang

Contributors Baojun Wang, Bin Wang, Bo Wang, Boxiao Liu, Changzheng Zhang, Duyu Tang, Fei Mi, Hui Jin, Jiansheng Wei, Jiarui Qin, Jinpeng Li, Jun Zhao, Liqun Deng, Lin Li, Minghui Xu, Naifu Zhang, Nianzu Zheng, Qiang Li, Rongju Ruan, Shengjun Cheng, Tianyu Guo, Wei He, Wei Li, Weiwen Liu, Wulong Liu, Xinyi Dai, Yonghan Dong, Yu Pan, Yue Li, Yufei Wang, Yujun Li, Yunsheng Ni, Zhe Liu, Zhenhe Zhang, Zhicheng Liu