ARCEE TRINITY LARGE TECHNICAL REPORT

Varun Singh[1] Lucas Krauss[1] Sami Jaghouar[2] Matej Sirovatka[2] Charles Goddard[1]
Fares Obeid[2] Jack Min Ong[2] Jannik Straube[2] Fern[1] Aria Harley[1] Conner Stewart[1]
Colin Kealty[1] Maziyar Panahi[1] Simon Kirsten[2] Anushka Deshpande[1] Anneketh Vij[1]
Arthur Bresnu[1] Pranav Veldurthi[1] Raghav Ravishankar[1] Hardik Bishnoi[1] DatologyAI
Mark McQuade[1] Johannes Hagemann[2] Lucas Atkins[1]

[1]Arcee AI [2]Prime Intellect

Abstract

We present the technical report for Arcee Trinity Large, a sparse Mixture-of-Experts model with 400B total parameters and 13B activated per token. Additionally, we report on Trinity Nano and Trinity Mini, with Trinity Nano having 6B total parameters with 1B activated per token, Trinity Mini having 26B total parameters with 3B activated per token. The models' modern architecture includes interleaved local and global attention, gated attention, depth-scaled sandwich norm, and sigmoid routing for Mixture-of-Experts. For Trinity Large, we also introduce a new MoE load balancing strategy titled Soft-clamped Momentum Expert Bias Updates (SMEBU). We train the models using the Muon optimizer. All three models completed training with zero loss spikes. Trinity Nano and Trinity Mini were pre-trained on 10 trillion tokens, and Trinity Large was pre-trained on 17 trillion tokens. The model checkpoints are available at `https://huggingface.co/arcee-ai`.

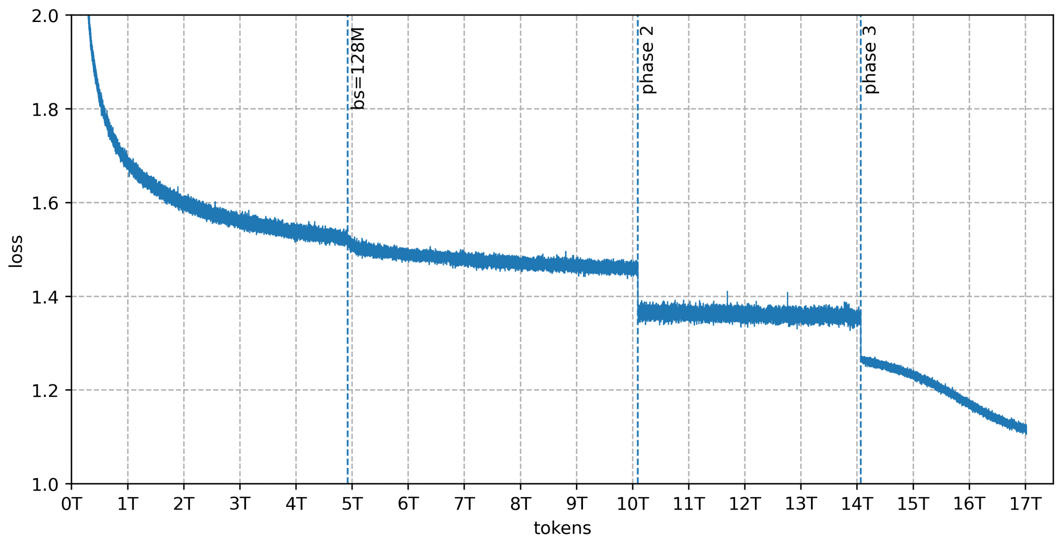

Figure 1: The training loss graph for Trinity Large, with no sub-sampling or smoothing. For clarity, we indicate where the batch size was increased to 128M (134,217,728) tokens, as well as the points where we switch data mixtures.

1 Introduction

Large language models (LLMs) are rapidly evolving from use in broad-coverage chatbot applications to general-purpose components for software systems, with deployments spanning document understanding and generation, retrieval-augmented knowledge work, and workflows in which they serve as long-running agents. Tool execution abilities to interact with external environments, code understanding and writing, and long-context management have proven to be key capabilities for LLMs to be deployed effectively. Beyond capabilities, there is also a growing need for inference-time efficiency as workflows and contexts over which the LLMs are required to operate on grow larger and larger. Sparse Mixture-of-Experts (MoE) [Shazeer et al., 2017] models have emerged as a prominent way for companies to scale up their largest models while being much more efficient and economical to train [DeepSeek-AI et al., 2025a, GLM-4.5 Team et al., 2025, Kimi Team et al., 2025a, Xiaomi LLM-Core Team et al., 2026]. In addition to this, recent models incorporate modified attention module designs such as linear attention variants [Kimi Team et al., 2025b, MiniMax et al., 2025, NVIDIA et al., 2025b] or sparse attention variants [DeepSeek-AI et al., 2025b, MiniCPM Team et al., 2025, Zhang et al., 2026], aiming to scale context and throughput without prohibitively increasing training or inference cost. Additionally, the rise of reasoning models [OpenAI et al., 2024, Guo et al., 2025] has led to inference-time compute scaling as a new paradigm for scaling model performance, enabling models to think via intermediate output for tens to hundreds of thousands of tokens before returning a final answer. These long output chains, often coupled with large inputs that the models need to reason over, further underscore the need for models to be fast and efficient at inference time.

At the same time, the requirements of real-world adoption extend beyond raw benchmark performance. Deployment settings commonly require organizational and regulatory considerations, motivating models that can be audited, hosted, and adapted within environments completely owned by the organizations. In particular, enterprise deployments frequently require clarity about data provenance, licensing, and jurisdictional controls, and therefore benefit from open-weight foundations that can be owned and operated without reliance on opaque third-party checkpoints.

This report presents Trinity Large, the largest member of the Trinity family of open-weight Mixture-of-Experts (MoE) language models. The Trinity family also includes Trinity Nano and Trinity Mini, which were developed as smaller form factors to provide immediately usable open models as well as serve as steps in our scaling ladder that validate the data pipeline, architecture, training recipe, and infrastructure required for Trinity Large.

The Trinity family's architecture is strongly informed from downstream use-cases, primarily the need for efficient training and inference. We adopt an extremely sparse Mixture-of-Experts layer and interleaved local and global attention [Yang et al., 2025] for efficient inference, and additionally adopt gated attention for stronger context comprehension capabilities. Additionally, the models were trained with the Muon optimizer [Jordan et al., 2024a], which enables a larger critical batch size and has higher sample efficiency than the widely used AdamW optimizer [Loshchilov and Hutter, 2019]. Trinity Nano and Trinity Mini were pre-trained on 10 trillion tokens, and Trinity Large was pre-trained on 17 trillion tokens. Additionally, training stability was a key concern in the design of the architecture, due to the limited time and compute availability for the project.

Trinity Large has 400B total parameters, with 13B activated per token, and is trained on a large mixed corpus combining curated web-scale data and synthetic data. In the remainder of this report, we describe the design decisions that govern architecture, training, and post-training, and we evaluate Trinity Large Base as well as Trinity Large Preview across a wide range of standard benchmarks.

2 Architecture

The Trinity models are decoder-only sparse Mixture-of-Experts (MoE) [Shazeer et al., 2017] transformer models [Vaswani et al., 2017], with no biases on the linear layers. We train a tokenizer using BPE [Shibata et al., 1999] with a vocabulary size of 200K. We discuss the components of the architecture below, including the details of our expert balancing scheme.

2.1 Attention

Our chosen attention mechanism combine multiple modifications to the standard Multi-Head Attention (MHA) [Vaswani et al., 2017] setup that have been shown to be effective in previous work. We combine grouped-query attention (GQA) [Ainslie et al., 2023], QK-normalization (QK-norm) [Henry et al., 2020], gated attention

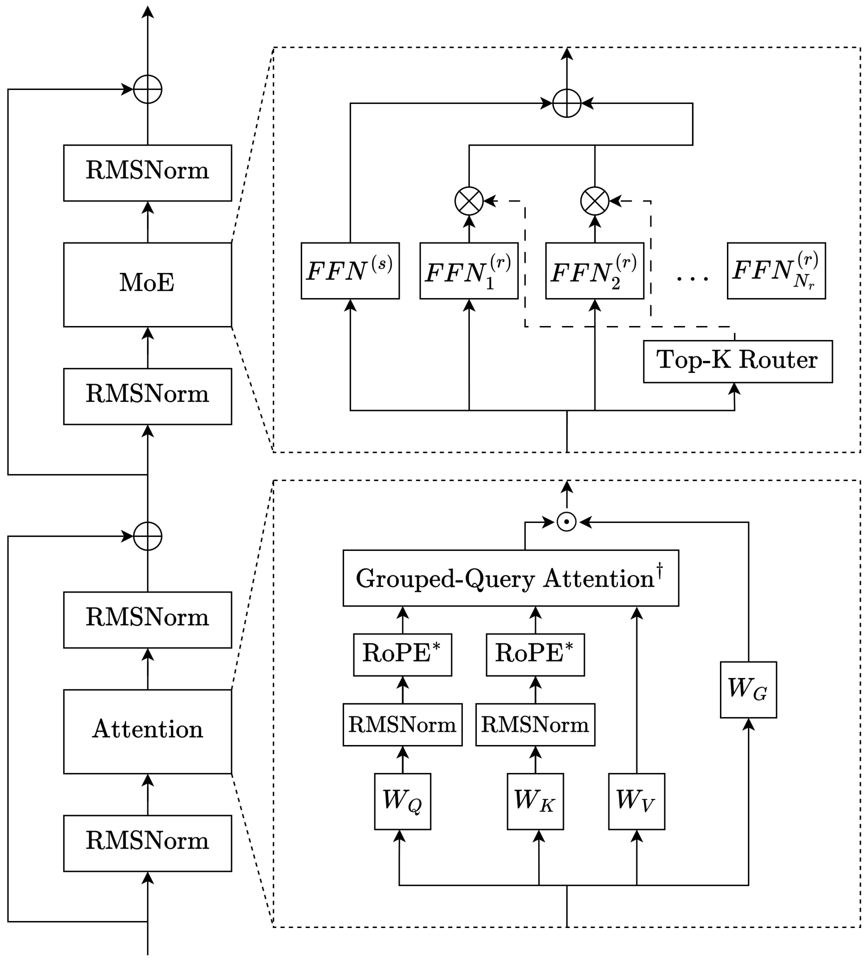

Figure 2: The architecture of the Trinity model family.
* RoPE is only present in local layers.
† The grouped-query attention has a sliding window for the local layers.

[Qiu et al., 2025], and a 3:1 local/global layer pattern, with rotary positional embeddings (RoPE) [Su et al., 2023] and sliding window attention (SWA) [Jiang et al., 2023, Gemma Team et al., 2024b, Beltagy et al., 2020, Child et al., 2019] in the local layers, and no positional embeddings (NoPE) (explored by Kazemnejad et al. [2023]) in the global layers. The local/global arrangement closely follows the results of Yang et al. [2025].

Let d denote the model dimension, h_q the number of query heads, h_{kv} the number of key/value heads, and $d_h = d/h_q$ the per-query-head dimension. Let $\mathbf{x}_t \in \mathbb{R}^d$ be the input at token position t in a given attention layer.

QK-normalization. We compute the queries, keys, and values as linear projections of the input and apply RMSNorm to queries and keys (QK-norm) before scaled dot-product attention:

$$\mathbf{q}_{t,i}^0 = W_i^Q \mathbf{x}_t, \qquad i \in \{1, \ldots, h_q\}, \tag{1}$$

$$\mathbf{k}_{t,j}^0 = W_j^K \mathbf{x}_t, \qquad j \in \{1, \ldots, h_{kv}\}, \tag{2}$$

$$\mathbf{v}_{t,j}^0 = W_j^V \mathbf{x}_t, \qquad j \in \{1, \ldots, h_{kv}\}, \tag{3}$$

$$\mathbf{q}_{t,i} = \mathrm{RMSNorm}\big(\mathbf{q}_{t,i}^0\big)\,, \tag{4}$$

$$\mathbf{k}_{t,j} = \mathrm{RMSNorm}\big(\mathbf{k}_{t,j}^0\big)\,. \tag{5}$$

where $W_i^Q \in \mathbb{R}^{d_h \times d}$ and $W_j^K, W_j^V \in \mathbb{R}^{d_h \times d}$. We choose to use QK-Norm primarily for training stability reasons, especially due to our use of the Muon optimizer, as prior work has identified growing maximum attention logit values to be more of a concern when using Muon as opposed to AdamW (Liu et al. [2025]; Kimi Team et al. [2025a]). Furthermore, GLM-4.5 Team et al. [2025] report that QK-Norm effectively stabilizes the range of maximum attention logit values through a full training run.

Local/global attention pattern. All models use a 3:1 ratio of local:global attention; they have three local (SWA) attention layers with RoPE, followed by one global layer without positional embeddings (NoPE), with this structure repeating for the full depth of the model. This follows the results of Yang et al. [2025], though we use QK-norm in our final setup. We choose to use this configuration primarily for effective long-context performance as well as for large efficiency gains at longer sequence lengths. The combination of this scheme and gated attention allowed the model to recover performance in a comparatively lower number of steps when training at longer sequence lengths, and also resulted in observed length extrapolation for Trinity Large.

Let w be the local window size. The queries and keys used in the scaled dot-product attention operation are:

$$\widehat{\mathbf{q}}_{t,i} = \begin{cases} \mathrm{RoPE}(\mathbf{q}_{t,i}), & \text{local layer,} \\ \mathbf{q}_{t,i}, & \text{global (NoPE) layer,} \end{cases} \tag{6}$$

$$\widehat{\mathbf{k}}_{t,j} = \begin{cases} \mathrm{RoPE}(\mathbf{k}_{t,j}), & \text{local layer,} \\ \mathbf{k}_{t,j}, & \text{global (NoPE) layer,} \end{cases} \tag{7}$$

with valid attention positions:

$$\mathcal{S}_t = \begin{cases} \{\, s \mid \max(1, t-w+1) \le s \le t \,\}, & \text{local layer,} \\ \{\, s \mid 1 \le s \le t \,\}, & \text{global layer.} \end{cases} \tag{8}$$

Scaled dot-product attention with GQA. During attention, GQA maps each query head i to a key/value head index

$$j(i) \;=\; \left\lceil i \cdot \frac{h_{kv}}{h_q} \right\rceil,$$

such that multiple query heads share the same key/value head. This has been shown to roughly match the performance of MHA while drastically reducing KV-cache size.

For each query head i, we attend over keys/values from the shared KV head $j(i)$:

$$\alpha_{t,i,s} = \mathrm{Softmax}_{s \in \mathcal{S}_t} \left(\frac{\widehat{\mathbf{q}}_{t,i}^{\top} \widehat{\mathbf{k}}_{s,j(i)}}{\sqrt{d_h}} \right), \tag{9}$$

$$\mathbf{o}_{t,i}^{\mathrm{sdpa}} = \sum_{s \in \mathcal{S}_t} \alpha_{t,i,s}\, \mathbf{v}_{s,j(i)}. \tag{10}$$

Gated attention. Following a configuration from Qiu et al. [2025], we apply an elementwise gating to the scaled dot-product attention output before the output linear projection:

$$\mathbf{g}_t = \sigma\big(W^G \mathbf{x}_t\big), \tag{11}$$

$$\mathbf{g}_{t,i} = \mathrm{split}_{h_q}(\mathbf{g}_t)_i, \tag{12}$$

$$\widetilde{\mathbf{o}}_{t,i} = \mathbf{o}_{t,i}^{\mathrm{sdpa}} \odot \mathbf{g}_{t,i}, \tag{13}$$

$$\mathbf{u}_t = W^O\, [\widetilde{\mathbf{o}}_{t,1}; \widetilde{\mathbf{o}}_{t,2}; \ldots; \widetilde{\mathbf{o}}_{t,h_q}]. \tag{14}$$

where $\sigma(\cdot)$ is the sigmoid, $\mathrm{split}_{h_q}(\cdot)$ partitions $\mathbf{g}_t \in \mathbb{R}^d$ into h_q contiguous vectors in $\mathbb{R}^{d_h}$, and $W^G \in \mathbb{R}^{d \times d}$, $W^O \in \mathbb{R}^{d \times d}$ are the gate and output projections, respectively. In Qiu et al. [2025], gated attention was identified to reduce attention sinks, reduce overly large activations, improve performance on evaluations, and improve long-sequence generalization. Furthermore, gating appears to stabilize training and reduce the occurrence of loss spikes during the training process, which is also a key motivation for our adoption.

2.2 Mixture-of-Experts

Our Mixture-of-Experts layers follow the DeepSeekMoE [Dai et al., 2024] design, with fine-grained routed experts and an always-active shared expert. We use the SwiGLU [Shazeer, 2020] activation function as the nonlinearity. For Trinity Large, in order to facilitate better throughput, we opted to use coarser-grained experts. We also replace the first k MoE layers with dense layers to stabilize early representations. We use sigmoid routing following Wang et al. [2024a], normalizing the router scores before using them for gating, and we also apply the gating scores to the output of each expert.

MoE formulation. Let $\mathbf{u}_t \in \mathbb{R}^d$ denote the MoE input at token t. The output of the MoE module $\mathbf{h}'_t$ is computed as:

$$\mathbf{h}'_t = \mathbf{u}_t + \sum_{i=1}^{N_s} \mathrm{FFN}_i^{(s)}(\mathbf{u}_t) + \sum_{i=1}^{N_r} g_{i,t}\, \mathrm{FFN}_i^{(r)}(\mathbf{u}_t)\,. \tag{15}$$

where $\mathrm{FFN}_i^{(s)}(\cdot)$ denotes the i-th shared expert and $\mathrm{FFN}_i^{(r)}(\cdot)$ denotes the i-th routed expert.

Routing. We use normalized sigmoid routing scores instead of softmax-based routing, which allows for more stable router logits. We also follow Wang et al. [2024a] in using the sum of the routing scores and expert bias to determine the Top-K experts, but using the routing scores without expert bias as gating scores for the outputs of the routed experts, since the expert bias is updated in a decoupled way. Let $\mathbf{e}_i \in \mathbb{R}^d$ be the router vector for routed expert i:

$$s_{i,t} = \sigma\big(\mathbf{u}_t^\top \mathbf{e}_i\big)\,, \qquad i \in \{1, \ldots, N_r\}. \tag{16}$$

The Top-K experts are selected as follows:

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \mathrm{TopK}\Big(\{s_{j,t} + b_j\}_{j=1}^{N_r},\, K_r\Big)\,, \\ 0, & \text{otherwise,} \end{cases} \tag{17}$$

$$g_{i,t} = \frac{g'_{i,t}}{\sum_{j=1}^{N_r} g'_{j,t}}. \tag{18}$$

where b_i is the expert bias, and $g_{i,t}$ are the gating scores derived from the router scores $s_{i,t}$.

Auxiliary-loss-free load balancing. For Trinity Mini and Trinity Nano, we used the standard formulation of auxiliary-loss-free load balancing [Wang et al., 2024a], with an additional re-centering of the expert bias updates. We maintain a bias vector $\mathbf{b} = [b_1, \ldots, b_{N_r}]$ that is updated in a decoupled fashion. Let n_i be the number of tokens routed to expert i in the current step, and $\bar{n}$ the mean load:

$$\bar{n} = \frac{1}{N_r} \sum_{i=1}^{N_r} n_i, \tag{19}$$

$$\Delta b_i = \gamma \cdot \mathrm{sign}(\bar{n} - n_i)\,, \tag{20}$$

$$b_i = b_i + \Delta b_i, \tag{21}$$

$$b_i = b_i - \frac{1}{N_r} \sum_{j=1}^{N_r} b_j. \tag{22}$$

where γ is the bias update speed.

Soft-clamped Momentum Expert Bias Updates (SMEBU) load balancing. For Trinity Large, we replace the per-step sign-based expert bias update with a $\tanh$ soft-clamped magnitude-aware update. First, we compute the normalized per-expert violation, in order to make the update scale independent of sequence-length and batch size, and apply $\tanh$ soft-clamping:

$$v_i = \frac{\bar{n} - n_i}{\bar{n}}, \tag{23}$$

$$\tilde{v}_i = \tanh(\kappa\, v_i)\,, \tag{24}$$

where κ is a tunable scale. We center the update around zero, and additionally use a momentum buffer to smooth expert bias updates over time:

$$\Delta b_i = \lambda \, \tilde{v}_i, \tag{25}$$

$$\Delta b_i = \Delta b_i - \frac{1}{N_r} \sum_{j=1}^{N_r} \Delta b_j, \tag{26}$$

$$m_i = \beta \, m_i + (1 - \beta) \, \Delta b_i, \tag{27}$$

$$b_i = b_i + m_i, \tag{28}$$

where λ is the load-balance learning rate, β is the momentum factor, and $\mathbf{m} = [m_1, \ldots, m_{N_r}]$ is the maintained momentum buffer.

When training Trinity Large, our initial runs faced router instability issues, and as one method of stabilizing the run, we examined the load-balancing method. Under the assumption that the ideal expert bias value is a fixed value, we note that the standard aux-free load balancing cannot precisely converge on that value, as each local update under the $\mathrm{sign}(\cdot)$ operator is always $\pm\lambda$. We hypothesize that this is a cause of instability for MoE training with the standard aux-loss-free objective. As the total number of experts increases, the per-layer norm of the bias step also increases. Viewing this as an optimization process, as this nears its local minima, the update step is biased towards oscillation, and can never fully settle.

We alleviate this using $\tanh$ as a continuous relaxation of the discrete update that can converge to 0. We replace the $\mathrm{sign}(\cdot)$ operator with $\tanh$ as a smooth approximation, and add a tunable scale κ to control saturation speed. We believe that bounded updates are important to maintaining network stability; preliminary tests using a linear, unclamped update quickly reduced MaxVio early on in training but resulted in instability later in training. Additionally, consistent with the hypothesis that behavior near convergence is one of the causes of instability in MoE aux-loss-free load balancing, and that noise near a local minima should be roughly decorrelated over time, we introduce momentum as a form of noise dampening. This is analogous to momentum SGD reducing variance in noisy gradient updates. For our Trinity Large run, we use SMEBU with $\lambda = 5 \times 10^{-4}$, $\beta = 0.5$, and $\kappa = 2$.

Sequence-wise auxiliary loss. In addition to aux-loss-free balancing, we use a complementary sequence-wise load balance loss to promote balance within a sequence, following the formulation from DeepSeek-AI et al. [2025a]:

$$\mathcal{L}_{\text{Bal}} = \alpha \sum_{i=1}^{N_r} f_i P_i, \tag{29}$$

$$f_i = \frac{N_r}{K_r T} \sum_{t=1}^{T} \mathbb{1}\Big(s_{i,t} + b_i \in \mathrm{TopK}\Big(\{ s_{j,t} + b_j \}_{j=1}^{N_r}, \, K_r \Big) \Big), \tag{30}$$

$$\tilde{s}_{i,t} = \frac{s_{i,t}}{\sum_{j=1}^{N_r} s_{j,t}}, \tag{31}$$

$$P_i = \frac{1}{T} \sum_{t=1}^{T} \tilde{s}_{i,t}, \tag{32}$$

where α is a small coefficient, T is the sequence length, and $\mathbb{1}(\cdot)$ is the indicator function.

2.3 Normalization

For layer normalization, we use a simplified depth-scaled sandwich norm [Yin et al., 2025, Ding et al., 2021, Kim et al., 2025]. Both the input and output of the module are normalized:

$$\mathbf{y}_\ell = \mathbf{x}_\ell + \mathrm{RMSNorm}_\ell^{(2)}\Big(\mathcal{M}_\ell\big(\mathrm{RMSNorm}_\ell^{(1)}(\mathbf{x}_\ell) \big) \Big), \tag{33}$$

where $\mathbf{x}_\ell$ and $\mathbf{y}_\ell$ are the input/output of layer ℓ, and $\mathcal{M}_\ell(\cdot)$ is the sublayer module (attention, FFN, or MoE).

We depth-scale the RMSNorm gain parameters for the second RMSNorm in each layer. Let L be the total number of layers and $\gamma(\cdot)$ denote the RMSNorm gain; we initialize:

$$\gamma\Big(\mathrm{RMSNorm}_\ell^{(1)}\Big) = 1, \tag{34}$$

$$\gamma\Big(\mathrm{RMSNorm}_\ell^{(2)}\Big) = \frac{1}{\sqrt{L}}. \tag{35}$$

We also apply RMSNorm before the language modeling head:

$$\mathbf{z} = \mathrm{RMSNorm}_{\mathrm{LM}}(\mathbf{h}_L). \tag{36}$$

2.4 Initialization

All trainable parameters are initialized from a zero-mean truncated normal distribution, with width-scaled standard deviation:

$$\theta \sim \mathrm{TruncNormal}\big(0,\ \sigma^2;\ [-3\sigma,\ 3\sigma]\big)\,, \tag{37}$$

$$\sigma = \frac{0.5}{\sqrt{d}}, \tag{38}$$

where d is the model dimension. This roughly follows the findings in Takase et al. [2025], which recommends setting $\sigma = \sqrt{\frac{2}{5d}}$. We note that $\frac{0.5}{\sqrt{d}}$ aligns with the initialization scale for DeepSeek-V3 [DeepSeek-AI et al., 2025a], as $\frac{0.5}{\sqrt{7168}} = 0.006$.

During the forward pass, we scale the embedding layer's activations by $\sqrt{d}$, also following Takase et al. [2025]:

$$\mathbf{e}_t = \sqrt{d}\, E(\mathrm{tok}_t). \tag{39}$$

We also note that both the Grok-1 and the Grok-2 model checkpoints on HuggingFace [xAI, 2024, 2025] have `embedding_multiplier_scale` set to $\sqrt{d}$. Additionally, the first two generations of the Gemma models [Gemma Team et al., 2024a,b] refer to this multiplier as a normalizer in their modeling code for HuggingFace's `transformers` library [Hugging Face, 2018].

3 Pre-training

3.1 Data

All pretraining data were curated by DatologyAI. Trinity Nano and Trinity Mini were each trained on the same 10 trillion token mix, and Trinity Large was trained on 17 trillion tokens of a distinct 20 trillion token mix. Both mixes were organized into three phases. The 10 trillion mix was distributed across 7T tokens in phase 1, 1.8T tokens in phase 2, and 1.2T tokens in phase 3. The 20 trillion mix was distributed across 13T tokens in phase 1, 4T tokens in phase 2, and 3T tokens in phase 3. The 17 trillion tokens that Trinity Large was trained on were sampled proportionally from the 20T mix (i.e. a 13:4:3 ratio across the three phases). Each mix was designed to target both general English capabilities as well as programming, STEM, and reasoning skills. The 10 trillion mix reuses the AFM-4.5B dataset and adds substantially more math and code. The 20T Trinity Large mix uses state-of-the art programming, STEM, reasoning, and multilingual data curation, targeting Arabic, Mandarin, Japanese, Spanish, German, French, Italian, Portuguese, Indonesian, Russian, Vietnamese, Hindi, Korean, and Bengali.

A key component of this curation is large-scale synthetic data generation. DatologyAI generated over 8 trillion tokens of synthetic web, code, and STEM data. For the web, approximately 6.5 trillion tokens of synthetic data were generated using rephrasing approaches that build on top of BeyondWeb [Maini et al., 2025]. Briefly, high-quality seed documents were selected from web corpora and then diverse generation strategies were used to rephrase these documents, including format transformation (e.g. converting web content into question–answer pairs), style modification (e.g. enhancing pedagogical tone), and content restructuring to improve information density and accessibility. Similar approaches were used to generate approximately 1 trillion tokens of synthetic multilingual data.

Approximately 800 billion tokens of high quality synthetic code data were also generated. In a similar fashion to web, relevant high-quality code files were selected for rephrasing and enhancement using a breadth of approaches that maximize diversity and relevance to task and style.

Efficiently and reliably generating over 8 trillion tokens of synthetic data requires robust, scalable infrastructure. The DatologyAI curation stack utilizes Ray and vLLM on Kubernetes to support heterogeneous clusters, various GPU types, and seamless scaling.

The data mixes for the Trinity models all performed midtraining by shifting the data mix in 3 phases towards higher-quality and domain specific data as discussed in [Blakeney et al., 2024, Hu et al., 2024, Team OLMo et al., 2025, Allal et al., 2025]. This included both shifting the data mix away from general web to more code, math, and science over the course of the phases as well as shifting to higher quality and more relevant data within these categories, e.g. within web, code, and STEM.

To our knowledge, this represents one of the largest publicly documented efforts in synthetic data generation for pretraining, demonstrating that carefully curated synthetic data can be produced at the multi-trillion token scale required for frontier model development. The effectiveness of the curation approach as a whole is reflected in the downstream evaluations presented in Section 5.1, where the Trinity models demonstrate strong performance across the targeted capability domains.

3.2 Data Preparation

Our pre-training data has three phases. Each phase consists of the same mixture, but the distribution of math and code data is boosted in later phases, in order to increase the final performance of the model. We tokenize documents on the fly and use sequence packing to construct token sequences with the remaining tokens from the document stream accumulated in a buffer for the next sample, with multiple dataloader workers used to interleave documents across batches. For Trinity Nano and Trinity Mini, we do not use intra-doc attention masking, but Trinity Large was pre-trained using intra-doc attention masking. During the pre-training of Trinity Large, we were concerned that our method of on-the-fly tokenization could cause longer documents to remain in consecutive batches for too long, due to insufficient interleaving. To reduce this inter-batch correlation, we introduce a document buffer mid-way through the Trinity Large training run, right before beginning phase 3.

One standard approach to sequential packing for on-the-fly tokenization, as implemented in frameworks like TorchTitan [Liang et al., 2025], continues documents that span multiple times the sequence length into subsequent batch items. While straightforward, this can become problematic when document lengths follow a lognormal distribution, as long documents introduce unbalanced domain biases at the minibatch level. This creates overhead in the learning objective as noise from step-to-step distribution imbalances must be corrected at every training step. We note that this is a consequence of sequential packing, and cannot simply be avoided by shuffling input documents.

We hypothesize that imbalanced minibatches have particular impact in at least two scenarios: (i) in limited-batchsize settings, and (ii) in regimes where the model becomes more data efficient per step. In both settings, informally, if we treat each unbalanced minibatch as an ideally-representative sample from its own target data generator, we can see that we effectively optimize a different target distribution per step. This not only causes additional overhead, as the network must “unlearn” the small domain imbalances every step, but also introduces a source of long-tail noise due to the lognormal distribution of sequence lengths. Under asymmetric losses like cross-entropy, small deviations can result in larger loss fluctuations. Additionally, as models grow larger, their step-to-step data efficiency tends to increase [Kaplan et al., 2020], which we hypothesize makes them more vulnerable to the step-to-step fluctuations in the data distribution. We hypothesize that this is one potential contributor to training instability as models grow larger.

To reduce this intra-batch correlation, we introduce a new method called the Random Sequential Document Buffer (RSDB). The Random Sequential Document Buffer operates as follows: after tokenizing a document, we load the entire token sequence as an entry in the RSDB, and initialize a read head at the 0th index. We continue tokenizing and inserting until the RSDB is full. When populating a single sequence buffer within a minibatch in the dataloader, until the sequence buffer is full, we randomly sample an index of the document in the RSDB, and read tokens from the document at the position of the read head into the sequence buffer. Document read head positions are updated the current position in the document after every step. If the sequence buffer is full, we return it, if not, we randomly select another document index and continue to read tokens into the sequence buffer. This process is continued until the sequence buffer is full. The buffer is refilled in order to keep a sufficient number of active documents.

For runtime efficiency, we set the internal buffer size to twice the user-specified buffer value, and refill the buffer to this larger value as soon as the buffer reaches the user-specified value. We also purge old documents and load new documents in bulk during this refill step. We found this optimization significantly improved dataloader performance. For phase 3 of training for Trinity Large, we used an RSDB with a buffer size of 8192 per GPU (user-specified 4096), with 4 workers per GPU. We split the RSDB evenly across each worker to keep the total size of the RSDB independent of worker counts.

The process addresses the previously mentioned issues in several ways. First, it reduces fragmentation by not pre-chunking documents, which could result in more fragmentation with an online algorithm, instead dynamically packing them at runtime. Second, it retains truly random selection across documents, better respecting IID sampling properties with respect to the data distribution. Third, it is memory read/write efficient, allowing for an efficient implementation that does not bottleneck the training process.

As an efficient way to quantify batch imbalance during training, and the efficacy of our method, we introduce a metric called Batch Heterogeneity, (BatchHet), defined as the difference between the maximum microbatch loss and the average loss for each training step:

$$\mathrm{BatchHet}_t = \max_i \mathcal{L}_i^{(t)} - \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}_i^{(t)} \tag{40}$$

where $\mathcal{L}_i^{(t)}$ is the loss of microbatch i at training step t, and $\bar{\mathcal{L}}^{(t)} = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}_i^{(t)}$ is the mean loss across all M microbatches at that step.

In small scale internal experiments, we observe that this metric correlates strongly with gradient norm instability. In a small internal experimental setup, enabling RSDB reduced BatchHet by 46% and improved loss over a sequential packing baseline. Additionally, the gradient norm was markedly stable, with a kurtosis of 14.6 for the grad norm of the RSDB-enabled network, as opposed to the kurtosis of 187 of the baseline's grad norm. Matching the step-to-step loss variance of the RSDB-enabled network required roughly a $2\times$ increase in the batchsize of the baseline network, due to increased data inefficiency. However, even with the loss-variance-matched batchsize, the baseline network had significantly higher step-to-step outliers in the average loss value when compared to the RSDB-enabled networks. A batchsize increase of $7\times$ was required in order for the BatchHet of the baseline to match the BatchHet of the RSDB-enabled network. We note that these improvements happen without dropping any tokens.

During development of the RSDB, we noted significant enough performance gains from it that we decided to integrate it during phase 3 of the Trinity Large training run instead of waiting for a later training run. While the data distributions between phase 2 and phase 3 make direct comparison difficult, the overall effect was notable: BatchHet reduced by a factor of 4.23x, and step-to-step variance reduced by a factor of 2.4x (see Figure 1), a significant improvement when compared to the default packing strategy. We note that training runs without the RSDB exhibit much higher values in the higher-order moments of the running loss distribution, which we believe to correlate with network instability during training.

3.3 Infrastructure

We trained Trinity on GPU clusters managed by Prime Intellect. We use a modified version of TorchTitan [Liang et al., 2025] as our training framework. For efficient fused cross-entropy and z-loss, we use kernels from the Liger Kernels [Hsu et al., 2025] project. For larger memory savings during context extension and supervised fine-tuning, we use Cut Cross-Entropy [Wijmans et al., 2025].

Trinity Nano and Trinity Mini were trained on clusters of 512 H200 GPUs, and Trinity Large was trained on a cluster of 2048 B300 GPUs. We train Trinity Nano and Trinity Mini with a Hybrid Sharded Data Parallel (HSDP) configuration, where there are multiple replicas of the model and Fully-Sharded Data Parallel (FSDP) [Zhao et al., 2023] is used within the replica group. For Trinity Large, we additionally employ Expert Parallelism (EP) within a GPU node for efficient Mixture-of-Experts Layers.

For all models, we set the FSDP group size to 128. For Trinity Large, we set the EP group size to 8. For the context extension of Trinity Large, we use a context parallelism degree of 4.

We use a modification of the Muon implementation in the Dion repository [Ahn et al., 2025] for efficient distributed Muon. We orthogonalize the gradient for the experts in a batched fashion, without flattening

Table 1: Model configurations for the Trinity model variants.

	Trinity Nano	Trinity Mini	Trinity Large
Transformer layers	56	32	60
Initial dense layers	2	2	6
Model dim (d_{model})	1024	2048	3072
FFN intermediate dim	3072	6144	12288
Attention heads (h_q)	8	32	48
Per-head dim (d_h)	128	128	128
KV heads (h_{kv})	2	4	8
Local window size	2048	2048	4096
Pre-training seq len	4096	4096	8192
MoE shared experts	1	1	1
MoE routed experts	128	128	256
Activated experts / token	8	8	4
Route scale	2.826	2.826	2.448
Expert size	256	1024	3072
Initialization (trunc normal σ)	0.016	0.011	0.009

the gradient tensor prior to orthogonalization. This also simplifies the expert-parallel implementation, since gradients do not need to be gathered across the expert-parallel group.

During the training of Trinity Large, when testing out versions of the PyTorch 2.10 nightly, we came across an issue where early versions did not fully support B300s, but later versions had a performance regression for both B200s and B300s. We conducted a binary search across nightly versions to find a package that worked well for the training setup. Later in the run, we upgraded to a recent PyTorch 2.11 nightly, giving us roughly a 5% speedup.

We designed the training stack to recover quickly from hardware failures. Since Trinity Large was trained on brand new B300 systems and is among the first large scale runs on this hardware, we expected more frequent failure rates during early deployment. In practice, we observed intermittent GPU faults, including recurrent XID errors, which were gradually mitigated through updated firmware bundles. On the systems side, we use failover nodes and prioritize fast restarts through dataloader optimizations and high performance NFS, significantly reducing recovery time after training restarts. We also run continuous heartbeat monitoring that triggers alerts if no training step completes within a three minute window, allowing us to quickly detect stalled jobs and intervene.

3.4 Hyperparameters

3.4.1 Model Configurations

Trinity Nano. We set the number of Transformer layers to 56 with 2 initial dense layers, and the model dimension to 1024 with FFN intermediate dimension 3072. For attention, we use 8 heads with per-head dimension 128 (h_q=8, d_h=128) and 2 KV heads (h_{kv}=2). We use local/global attention with sliding-window size for the local layers set to 2048 and pre-training sequence length 4096. For each MoE layer we use 1 shared expert and 128 routed experts, activating 8 experts per token, with route scale set to 2.826. The expert size was set to 256. We initialize all trainable parameters from a truncated normal distribution with standard deviation 0.016.

With Trinity Nano, we aimed to explore the effectiveness of high depth at producing a powerful small language model, following the improved performance of Falcon-H1-1.5B-Deep over Falcon-H1-1.5B [Zuo et al., 2025].

Trinity Mini. We set the number of Transformer layers to 32 with 2 initial dense layers, and the model dimension to 2048 with FFN intermediate dimension 6144. For attention, we use 32 heads with per-head dimension 128 (h_q=32, d_h=128) and 4 KV heads (h_{kv}=4). We use local/global attention with sliding-window size for the local layers set to 2048 and pre-training sequence length 4096. For each MoE layer we use 1 shared expert and 128 routed experts, activating 8 routed experts per token, with route scale set to 2.826. The expert size was set to 1024. We initialize all trainable parameters from a truncated normal distribution with standard deviation 0.011.

Trinity Large. We set the number of Transformer layers to 60 with 6 initial dense layers, and the model dimension to 3072 with FFN intermediate dimension 12288. For attention, we use 48 heads with per-head dimension 128 (h_q=48, d_h=128) and 8 KV heads (h_{kv}=8). We use local/global attention with sliding-window size for the local layers set to 4096 and pre-training sequence length 8192. For each MoE layer we use 1 shared expert and 256 routed experts, activating 4 experts per token, with route scale set to 2.448. The expert size was set to 3072. For Trinity Large, we opted to activate 4 routed experts and make each expert larger, reducing the granularity, due to throughput requirements. We also greatly increase the sparsity to maximize model capacity while keeping a low active parameter count. We initialize all trainable parameters from a truncated normal distribution with standard deviation 0.009.

3.4.2 Training Hyperparameters

When training all three models, we use the Muon optimizer [Jordan et al., 2024a] for hidden layers and the AdamW optimizer [Loshchilov and Hutter, 2019] for the embedding and output layers. Unlike Liu et al. [2025], we choose not to rescale the RMS of the Muon updates to match the RMS of the AdamW updates. Instead, we use the learning rate adjustment rule:

$$\mathrm{lr}_{\mathrm{adjusted}} = \mathrm{lr}\sqrt{\max\left(1, \frac{\mathrm{fan}_{\mathrm{out}}}{\mathrm{fan}_{\mathrm{in}}}\right)}. \tag{41}$$

following Jordan et al. [2024b]. Empirically, we observe that this adjustment enables optimal learning rate transfer when scaling model width, for Muon.

Trinity Nano. We use a linear warmup of 2000 steps. The peak learning rates are 1.0×10^{-3} for Muon and 3.0×10^{-4} for AdamW. We train with global batch size 4096 at sequence length 4096, and increase the global batch size to 8192 when scaling from a 256-GPU cluster to a 512-GPU cluster to improve hardware utilization. After warmup, we apply a linear decay to zero during the decay stage. For context extension, we use a linear re-warmup to $\frac{1}{10}$ of the peak learning rate followed by a linear decay back to zero. We train to 256k context for inference at 128k.

Trinity Mini. We use a linear warmup of 2000 steps. The peak learning rates are 1.0×10^{-3} for Muon and 2.0×10^{-4} for AdamW. We train with global batch size 4096 at sequence length 4096, and increase the global batch size to 8192 when scaling from a 64-GPU cluster to a 512-GPU cluster to improve hardware utilization. After warmup, we apply a linear decay to zero during the decay stage. For context extension, we use a linear re-warmup to $\frac{1}{10}$ of the peak learning rate followed by a linear decay back to zero. We train to 128k context for inference at 128k.

Trinity Large. We use a linear warmup of 2000 steps. The peak learning rates are 8.0×10^{-4} for Muon and 2.0×10^{-4} for AdamW. We train with global batch size 12288 at sequence length 8192, and increase the global batch size to 16384 after crossing 4.9T tokens to improve throughput, roughly following MiniMax et al. [2025]. During the decay phase, we use a cosine decay to $\frac{1}{10}$ of the peak learning rate, i.e., to 8.0×10^{-5} (Muon) and 2.0×10^{-5} (AdamW), which allows us to proceed into context extension without re-warming. For context extension, we continue cosine decay from $\frac{1}{10}$ to $\frac{1}{20}$ of the peak learning rate, i.e., from 8.0×10^{-5} to 4.0×10^{-5} (Muon) and from 2.0×10^{-5} to 1.0×10^{-5} (AdamW). We train to 256k context for inference at 512k.

3.5 Context Extension

The Trinity architecture utilizes interleaved local/global attention, with the local attention layers pre-trained at a sliding window size set to half of the context window of global layers. In initial context extension experiments, we compared adjusting the local layers (with different values for the RoPE base frequency θ) along with the global layers against adjusting only the global layers. Adjusting only the global layers resulted in a much quicker loss recovery, aligning with the findings in Yang et al. [2025], and further suggesting that the local and global layers learn complementary roles. Additionally, not extending the window size of the local layers allows for more efficient inference, so we choose to follow this approach to extend the context window of the models. As a result, the context extension process was extremely smooth.

Due to our time and compute constraints, we used a rapid iterative approach for testing context extension effectiveness, using the Multi-Key Needle-in-a-haystack (MK-NIAH) task from RULER [Hsieh et al., 2024] at the target context length as a proxy metric. This allowed us to rapidly evaluate checkpoints and make decisions about what to adjust for the next iteration. Consistent with Gao et al. [2025] and NVIDIA et al. [2025a], we saw that training at a sequence length longer than the target context window enables better performance at the target context window. When extending the context window of Trinity Nano, training at a sequence length of 128K (the target context window) yielded an MK-NIAH (@128K) score of 0.38, whereas training at a sequence length of 256K yielded an MK-NIAH (@128K) score of 0.548. Iterating on and improving our selection of dataset and hyperparameters yielded a final MK-NIAH (@128K) score of 0.864 for Trinity Nano. Due to resource constraints, we trained Trinity Mini at 128K for a target of 128K, but found that it performed significantly stronger compared to training Trinity Nano at 128K alone. Our final iteration on Trinity Mini, trained at a sequence length of 128K, achieved a final MK-NIAH (@128K) score of 0.888.

This effect of model size affecting ease of context extension continued for Trinity Large. We trained Trinity Large at a sequence length of 256K, targeting a final context window size of 256K, and the model achieved an MK-NIAH (@256K) score of 0.994. Furthermore, upon evaluating it at 512K, we even found that it achieved an MK-NIAH (@512K) score of 0.976, despite not being trained at that sequence length. Additionally, even evaluating it at 1M, it achieved an MK-NIAH (@1M) score of 0.42, suggesting that it would be possible to push later versions of the model to have strong performance at a context window of 1M. Additionally, we did not find any improvements from progressively increasing the context window size as opposed to training directly at the longest sequence length, beginning from the final pre-trained checkpoint.

Our long-context extension dataset comprises approximately 117 billion tokens across 35.6 million documents. Following established best practices, we blend samples from our original pretraining distribution with targeted long-context sources. For the pretraining component, we apply a length-biased sampling strategy across all three phases of the pre-training corpus: sampling probabilities scale with document length, from a 1% floor up to 90% for documents reaching 128K tokens. This approach yields a subset that remains broadly representative of the source distribution while substantially enriching the proportion of longer sequences. Inspired by Allen AI's Dolma 3 Longmino mix for OLMo 3 [Olmo et al., 2025], we incorporate substantial OCR-derived PDF data, including their olmOCR [Poznanski et al., 2025] dataset and HuggingFace's FinePDF-edu [Kydlíček et al., 2025], which provide naturally long, high-quality documents. We also regenerate the ProLong datasets [Gao et al., 2025] at full sequence length rather than the published 512K/64K truncations; whole-repository code concatenations from this source proved particularly valuable for both extended-context generalization and cross-file code understanding. The mix is rounded out with instruction-style data (FLAN [Chung et al., 2022], math, and code) drawn from our original AFM context-extension recipe to preserve few-shot and instruction-following capabilities, alongside AutoMathText [Zhang et al., 2025] and curated arXiv and textbook sources from ProLong.

4 Post-Training

We conducted the post-training for Trinity Large on the same cluster as the pre-training. Because we allocated most of our cluster time to pretraining, we were constrained to a relatively light post-training phase. As a result, the model we report here, Trinity-Large-Preview, is best viewed as a preliminary release rather than a fully post-trained model. In the next iteration, we plan to build upon this foundation with further, more extensive post-training.

Dataset construction. We build our instruction-tuning mix from a blend of public instruction data and custom data. The custom portion combines human-written prompts with synthetic instructions generated by stronger teacher models. These are then filtered for quality, formatting, and length. We also lean heavily into agentic coding supervision: a large share of our coding subset comes from trajectories collected through agentic harnesses. In particular, we use OpenCode [Anomaly, 2026] as a harness to run multi-step coding tasks and record full interaction traces (edits, tool calls, and test outcomes), so the model learns the edit-run-test loop rather than only a final completion.

Supervised fine-tuning. For the supervised fine-tuning stage, we also use our modified version TorchTitan as the training framework. This allows us to use a similar parallelism setup to our context extension phase, though without requiring context parallelism. We use the Cut Cross-Entropy kernel to reduce memory requirements. To increase training efficiency, we pre-tokenize samples offline and during training, we use greedy packing to process samples in parallel. We train at a sequence length of 64K for the supervised fine-tuning stage. When tokenizing samples, we truncate any samples longer than the target sequence length, and during training, we pad sequences to the full sequence length after packing. We run supervised fine-tuning using a linear warm-up for the learning rate, up to $\frac{1}{10}$ of the peak pre-training learning rate. This is followed by linearly decaying the learning rate to zero.

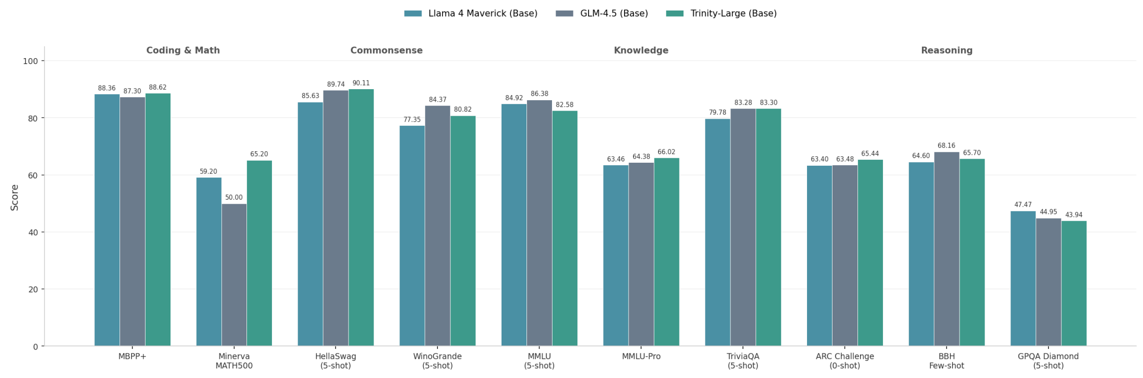

Figure 3: A comparison of Trinity Large Base to other similar open-weight base models.

Table 2: Trinity Large Base performance on our base model evaluation suite.

Evaluation	Score
MBPP+	88.62
Minerva MATH500	65.20
HellaSwag (5-shot)	90.11
WinoGrande (5-shot)	80.82
MMLU (5-shot)	82.58
MMLU-Pro (5-shot)	66.02
TriviaQA (5-shot)	83.30
ARC Challenge (0-shot)	65.44
BBH (few-shot)	65.70
GPQA Diamond (5-shot)	43.94

Reinforcement learning. After supervised fine-tuning, we run a short RL stage using prime-rl [Prime Intellect Team et al., 2025] as our training framework on the same 2048 B300 cluster. We use prime-rl's asynchronous setup, where vLLM-backed workers generate rollouts while a separate distributed trainer applies updates with FSDP2. The environments we use follow the `verifiers` [Brown, 2025] API. We prefer verifiable rewards when tasks have objective checks (for example, strict answer-format validation), and fall back to a learned reward model for prompts without clean ground truth.

5 Trinity Large Evaluation Results

5.1 Capability Benchmarks

We evaluate Trinity Large Base on a standard benchmark suite spanning coding/math (MBPP+ [Liu et al., 2023], Minerva MATH500 [Lewkowycz et al., 2022]), commonsense (HellaSwag [Zellers et al., 2019], WinoGrande [Sakaguchi et al., 2019]), knowledge (MMLU [Hendrycks et al., 2021], MMLU-Pro [Wang et al., 2024b], TriviaQA [Joshi et al., 2017], ARC Challenge [Clark et al., 2018]), and reasoning (BBH [Suzgun et al., 2022], GPQA Diamond [Rein et al., 2023]). The evaluation results are reported in Table 2, and comparisons to other open-weight base models are shown in Figure 3. We note that Trinity Large Base achieves competitive scores with GLM 4.5 Base, despite having $4\times$ higher degree of sparsity and a roughly $2.5\times$ lower active parameter count.

We also present evaluations for our instruct-tuned model, Trinity Large Preview, in Table 3. Our suite of instruct-tuned model benchmarks includes MMLU, MMLU-Pro, GPQA Diamond, SimpleQA [Wei et al., 2024], and AIME25.

Table 3: Trinity Large Preview performance on our instruct model evaluation suite.

Evaluation	Score
MMLU	87.21
MMLU-Pro	75.25
GPQA Diamond	63.32
SimpleQA	23.92
AIME25	24.36

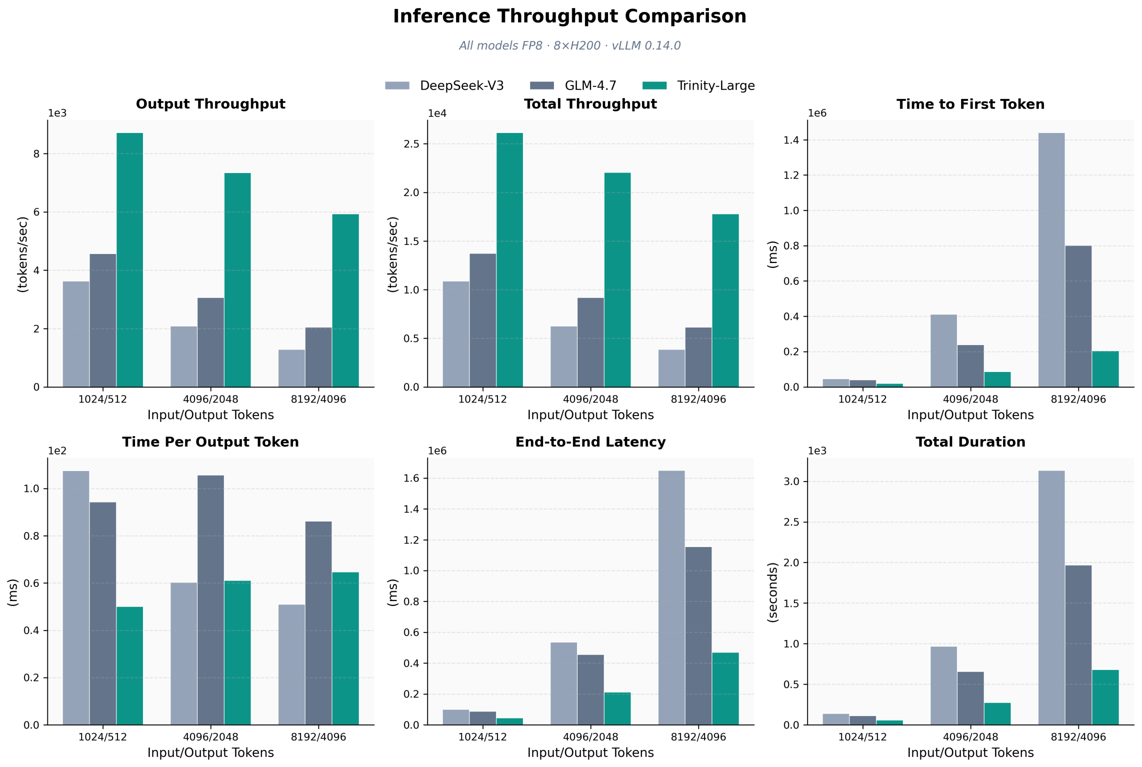

Figure 4: Throughput comparison of models. All tests were done with models quantized to FP8, using vLLM, on 8xH200.

5.2 Inference Benchmarks

To benchmark inference performance, we use vLLM with all models quantized to FP8. Measurements were taken on an 8xH200 node. The measurements can be seen in Figure 4. The design of Trinity Large with its extreme sparsity and interleaved local/global attention results in strong performance.

6 Discussion

When training Trinity Large, we experienced some initial instability. On our first few runs, the loss decreased as expected and expert utilization looked roughly balanced early on. After some progress, routing behavior drifted and expert load became increasingly uneven, eventually resulting in collapsed experts. MaxVio (defined below) [Wang et al., 2024a] would remain stable and then experience a sudden climb as the experts collapsed. When this happened, the loss plateaued, and evaluation improvements also plateaued, far from expected behavior. We attempted a few targeted changes: lowering the learning rate as a general method of stabilization, removing weight decay from normalization parameters, and increasing the post-norm ϵ value to reduce numerical sensitivity in normalization. Each of these changes improved behavior for a while, but did not prevent recurrence of the same failure mode. MaxVio continued to diverge and the loss trend did not recover to a stable downward trajectory. Since we had very limited time, we applied six changes at once, all targeted at increasing the stability of the run:

1. We adopted our new load balancing method SMEBU, which we had briefly tested at very small scale.
2. We disabled our MXFP8 kernels for linear layers and grouped GEMMs, falling back to BF16.
3. We adopted z-loss with a small weight to stabilize training. [Wortsman et al., 2023, Team OLMo et al., 2025]. Initially, we planned to use a weight of 1×10^{-4}. However, due to a bug in the training code, the weight was being applied incorrectly, and was falling back to a z-loss weight of 0. Out of concern due to the continued growth of the maximum logit value, we attempted to introduce z-loss in the middle of training (following Marin Community [2025]), and had to reduce the weight to 1×10^{-6}, as larger values destabilized the network. This effectively stabilized the trend in maximum logit as well as mean logits.
4. We swapped from a purely aux-loss-free load balancing strategy to including a sequence-wise aux loss with a small weight (1×10^{-4}), following DeepSeek-AI et al. [2025a], GLM-4.5 Team et al. [2025], and Xiaomi LLM-Core Team et al. [2026].
5. We chose to increase the number of dense layers from 3 to 6, to further stabilize representations.
6. We chose to adopt intra-document masking to prevent token positions from attending to token positions from a different document, to reduce noise in the learning objective.

With these fixes applied, MaxVio stopped diverging, expert utilization remained balanced throughout the run, and loss continued to smoothly converge to a lower value. Because the fixes were introduced together to unblock training, we did not have time to run controlled ablations to attribute stabilization to any individual change.

MaxVio [Wang et al., 2024a] measures the relative worst-case expert overload, defined as

$$\mathrm{MaxVio} = \frac{\max_i \, \mathrm{Load}_i - \overline{\mathrm{Load}}}{\overline{\mathrm{Load}}}. \tag{42}$$

where $\overline{\mathrm{Load}}$ is the mean expert load, and Load_i is the load on expert i for $1 \leq i \leq N_r$.

7 Conclusion and Future Work

In this report, we introduced the Trinity family of open-weight language models, culminating in Trinity Large, a model with 400B total parameters and 13B active per token. We also introduced Trinity Nano and Trinity Mini as smaller form factors and scaling-ladder validation points. At the largest scale, training stability and expert utilization balance were critical focus points.

We identify two promising directions for future work, both for us and the community at large. First, we believe that enabling greater sparsity in models overall will be the key to efficient scaling. Specifically in the case of MoEs, improved load balancing and routing will be important in allowing us to scale to greater sparsity while keeping training stable. Secondly, we view large-batch training as another key for efficient scaling. Algorithmic improvements that can push the critical batch size higher (and maintain sample efficiency and stability at scale) will directly translate into faster training and better utilization of modern hardware as the number of GPUs scales (when model parallelism is not needed).

ADDITIONAL CONTRIBUTORS

ARCEE AI

Curt Larson
Scott Zembsch
Gabriel Santos
Ben Langer
Sam Fraser
Eric Lau
Mariam Jabara
James Weir
Davis Stone
Dante Simon
Molly Niland
Zachary Kirkendall
Mohit Khullar

DATOLOGY AI

TECHNICAL STAFF

Amro Abbas
Rishabh Adiga
Cody Blakeney
Paul Burstein
Aldo Carranza
Spandan Das
Alvin Deng
Vineeth Dorna
Parth Doshi
Alex Fang
Tony Jiang
Siddharth Joshi
Brett Larsen
Jason Lee
Pratyush Maini
Kaleigh Mentzer
Luke Merrick
Haakon Mongstad
Ricardo Monti
Fan Pan
David Schwab
Darren Teh
Jason Telanoff
Jack Urbanek
Zhengping Wang
Josh Wills
Haoli Yin

LEADERSHIP

Bogdan Gaza
Matthew Leavitt
Ari Morcos

References

Kwangjun Ahn, Byron Xu, Natalie Abreu, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint: 2504.05295*, 2025.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023. URL `https://arxiv.org/abs/2305.13245`.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Anomaly. Opencode: The open source ai coding agent. `https://opencode.ai/`, 2026. Accessed: 2026-01-27.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL `https://arxiv.org/abs/2004.05150`.

Cody Blakeney, Mansheej Paul, Brett W Larsen, Sean Owen, and Jonathan Frankle. Does your data spark joy? performance gains from domain upsampling at the end of training. *First Conference on Language Modeling (CoLM)*, 2024.

William Brown. Verifiers: Environments for llm reinforcement learning. `https://github.com/PrimeIntellect-ai/verifiers`, 2025. Commit fd5c7b73faac97673420ff82ade43015a0841aa1, accessed 2026-01-27.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL `https://arxiv.org/abs/1904.10509`.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. URL `https://arxiv.org/abs/2210.11416`.

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, 2018. URL `https://arxiv.org/abs/1803.05457`.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL `https://arxiv.org/abs/2401.06066`.

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, 2025a. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025b. URL https://arxiv.org/abs/2512.02556.

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, 2021. URL https://arxiv.org/abs/2105.13290.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively), 2025. URL https://arxiv.org/abs/2410.02660.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024a. URL `https://arxiv.org/abs/2403.08295`.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024b. URL `https://arxiv.org/abs/2408.00118`.

GLM-4.5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin Niu, Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang, Zilin Zhu, Bohan Zhang, Bosi Wen, Bowen Wu, Bowen Xu, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li, Congfeng Yin, Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan, Guo Wang, Hailong Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai, Haoke Zhang, Haoran Wang, Haoyu Yang, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan Liu, Huilong Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi Wang, Jiayi Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan, Jingxuan Li, Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang, Liang Xu, Lin Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan Xu, Mingming Zhao, Mingshu Zhai, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu, Shangtong Yang, Shaoyou Lu, Shijie Li, Shuang Li, Shuang-Li, Shuxun Yang, Sibo Yi, Tianshu Yu, Wei Tian, Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao Wang, Xiaohan Jia, Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan Zhang, Xinze Zhang, Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong Wang, Yilin Zhou, Yiming Pan, Ying Zhang, Yingli Wang, Yingru Li, Yinpei Su, Yipeng Geng, Yitong Zhu, Yongkun Yang, Yuhang Li, Yuhao Wu, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yuxuan Zhang, Zezhen Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao, Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zihan Wang, Zijun Yao, Zikang Wang, Ziqiang Liu, Ziwei Chai, Zixuan Li, Zuodong Zhao, Wenguang Chen, Jidong Zhai, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models, 2025. URL `https://arxiv.org/abs/2508.06471`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL `http://dx.doi.org/10.1038/s41586-025-09422-z`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL `https://arxiv.org/abs/2009.03300`.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers, 2020. URL `https://arxiv.org/abs/2010.04245`.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024. URL `https://arxiv.org/abs/2404.06654`.

Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, Yanning Chen, and Zhipeng Wang. Liger-kernel: Efficient triton kernels for LLM training. In *Championing Open-source Development in ML Workshop @ ICML25*, 2025. URL `https://openreview.net/forum?id=36SjAIT42G`.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Miniccpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Hugging Face. Transformers. GitHub repository, November 2018. URL `https://github.com/huggingface/transformers`. First release 2018-11-17. Accessed 2026-01-26.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Keller Jordan et al. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024a. Accessed: 2026-01-25.

Keller Jordan et al. Muon (github repository): An optimizer for hidden layers in neural networks, 2024b. URL https://github.com/KellerJordan/Muon. GitHub repository, master branch.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

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, 2020. URL https://arxiv.org/abs/2001.08361.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers, 2023. URL https://arxiv.org/abs/2305.19466.

Jeonghoon Kim, Byeongchan Lee, Cheonbok Park, Yeontaek Oh, Beomjun Kim, Taehwan Yoo, Seongjin Shin, Dongyoon Han, Jinwoo Shin, and Kang Min Yoo. Peri-ln: Revisiting normalization layer in the transformer architecture, 2025. URL https://arxiv.org/abs/2502.02732.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2025a. URL https://arxiv.org/abs/2507.20534.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025b. URL https://arxiv.org/abs/2510.26692.

Hynek Kydlíček, Guilherme Penedo, and Leandro Von Werra. Finepdfs: Liberating 3t of the finest tokens from pdfs, 2025.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, Sanket Purandare, Gokul Nadathur, and Stratos Idreos. Torchtitan: One-stop pytorch native solution for production ready LLM pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=SFN6Wm7YBI`.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023. URL `https://arxiv.org/abs/2305.01210`.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025. URL `https://arxiv.org/abs/2502.16982`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL `https://arxiv.org/abs/1711.05101`.

Pratyush Maini, Vineeth Dorna, Parth Doshi, Aldo Carranza, Fan Pan, Jack Urbanek, Paul Burstein, Alex Fang, Alvin Deng, Amro Abbas, et al. Beyondweb: Lessons from scaling synthetic data for trillion-scale pretraining. *arXiv preprint arXiv:2508.10975*, 2025.

Marin Community. Marin 8b retrospective. Marin Documentation (Read the Docs), May 2025. URL `https://marin.readthedocs.io/en/latest/reports/marin-8b-retro/`. Accessed 2026-01-26. Publicly linked from the Marin announcement dated 2025-05-19.

MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, Ning Ding, Shengda Fan, Yewei Fang, Zixuan Fu, Wenyu Guan, Yitong Guan, Junshao Guo, Yufeng Han, Bingxiang He, Yuxiang Huang, Baoxi Ji, Cunliang Kong, Qiuzuo Li, Siyuan Li, Wenhao Li, Xin Li, Yanghao Li, Yishan Li, Zhen Li, Dan Liu, Biyuan Lin, Yankai Lin, Xiang Long, Quanyu Lu, Yaxi Lu, Peiyan Luo, Hongya Lyu, Litu Ou, Yinxu Pan, Lushi Pu, Zekai Qu, Qundong Shi, Zijun Song, Jiayuan Su, Zhou Su, Ao Sun, Xianghui Sun, Peijun Tang, Fangzheng Wang, Feng Wang, Shuo Wang, Yudong Wang, Zheng Wang, Yesai Wu, Zhenyu Xiao, Jie Xie, Zihao Xie, Xiaoyue Xu, Yukun Yan, Jiarui Yuan, Jinqian Zhang, Kaihuo Zhang, Lei Zhang, Linyue Zhang, Xueren Zhang, Yudi Zhang, Hengyu Zhao, Weilin Zhao, Weilun Zhao, Yuanqian Zhao, Zhi Zheng, Chuyue Zhou, Ge Zhou, Jie Zhou, Wei Zhou, Yanghao Zhou, Zihan Zhou, Zixuan Zhou, Zhiyuan Liu, Guoyang Zeng, Chao Jia, Dahai Li, and Maosong Sun. Minicpm4: Ultra-efficient llms on end devices, 2025. URL `https://arxiv.org/abs/2506.07900`.

MiniMax, Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, Enwei Jiao, Gengxin Li, Guojun Zhang, Haohai Sun, Houze Dong, Jiadai Zhu, Jiaqi Zhuang, Jiayuan Song, Jin Zhu, Jingtao Han, Jingyang Li, Junbin Xie, Junhao Xu, Junjie Yan, Kaishun Zhang, Kecheng Xiao, Kexi Kang, Le Han, Leyang Wang, Lianfei Yu, Liheng Feng, Lin Zheng, Linbo Chai, Long Xing, Meizhi Ju, Mingyuan Chi, Mozhi Zhang, Peikai Huang, Pengcheng Niu, Pengfei Li, Pengyu Zhao, Qi Yang, Qidi Xu, Qiexiang Wang, Qin Wang, Qiuhui Li, Ruitao Leng, Shengmin Shi, Shuqi Yu, Sichen Li, Songquan Zhu, Tao Huang, Tianrun Liang, Weigao Sun, Weixuan Sun, Weiyu Cheng, Wenkai Li, Xiangjun Song, Xiao Su, Xiaodong Han, Xinjie Zhang, Xinzhu Hou, Xu Min, Xun Zou, Xuyang Shen, Yan Gong, Yingjie Zhu, Yipeng Zhou, Yiran Zhong, Yongyi Hu, Yuanxiang Fan, Yue Yu, Yufeng Yang, Yuhao Li, Yunan Huang, Yunji Li, Yunpeng Huang, Yunzhi Xu, Yuxin Mao, Zehan Li, Zekang Li, Zewei Tao, Zewen Ying, Zhaoyang Cong, Zhen Qin, Zhenhua Fan, Zhihang Yu, Zhuo Jiang, and Zijia Wu. Minimax-01: Scaling foundation models with lightning attention, 2025. URL `https://arxiv.org/abs/2501.08313`.

NVIDIA, Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, Alex Kondratenko, Alex Shaposhnikov, Alexander Bukharin, Ali Taghibakhshi, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amy Shen, Andrew Tao, Ann Guan, Anna Shors, Anubhav Mandarwal, Arham Mehta, Arun Venkatesan, Ashton Sharabiani, Ashwath Aithal, Ashwin Poojary, Ayush Dattagupta, Balaram Buddharaju, Banghua Zhu, Barnaby Simkin, Bilal Kartal, Bita Darvish Rouhani, Bobby Chen, Boris Ginsburg, Brandon Norick, Brian Yu, Bryan Catanzaro, Charles Wang, Charlie Truong, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Christian Munley, Christopher Parisien, Dan Su, Daniel Afrimi, Daniel Korzekwa, Daniel Rohrer, Daria Gitman, David Mosallanezhad, Deepak Narayanan, Dima Rekesh, Dina Yared, Dmytro Pykhtar, Dong Ahn, Duncan Riach, Eileen Long, Elliott Ning, Eric Chung, Erick Galinkin, Evelina Bakhturina, Gargi Prasad, Gerald Shen, Haifeng Qian, Haim Elisha, Harsh Sharma, Hayley Ross, Helen Ngo, Herman Sahota, Hexin Wang, Hoo Chang Shin, Hua Huang, Iain Cunningham, Igor Gitman, Ivan Moshkov, Jaehun Jung, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jian Zhang, Jiaqi Zeng, Jimmy Zhang, Jinze Xue, Jocelyn Huang, Joey Conway, John Kamalu, Jonathan Cohen, Joseph Jennings, Julien Veron Vialard, Junkeun Yi, Jupinder Parmar, Kari Briski, Katherine Cheung, Katherine Luna, Keith Wyss, Keshav Santhanam, Kezhi Kong, Krzysztof Pawelec, Kumar Anik, Kunlun Li, Kushan Ahmadian, Lawrence McAfee, Laya Sleiman, Leon Derczynski, Luis Vega, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Marcin Chochowski, Mark Cai, Markus Kliegl, Marta Stepniewska-Dziubinska, Matvei Novikov, Mehrzad Samadi, Meredith Price, Meriem Boubdir, Michael Boone, Michael Evans, Michal Bien, Michal Zawalski, Miguel Martinez, Mike Chrzanowski, Mohammad Shoeybi, Mostofa Patwary, Namit Dhameja, Nave Assaf, Negar Habibi, Nidhi Bhatia, Nikki Pope, Nima Tajbakhsh, Nirmal Kumar Juluru, Oleg Rybakov, Oleksii Hrinchuk, Oleksii Kuchaiev, Oluwatobi Olabiyi, Pablo Ribalta, Padmavathy Subramanian, Parth Chadha, Pavlo Molchanov, Peter Dykas, Peter Jin, Piotr Bialecki, Piotr Januszewski, Pradeep Thalasta, Prashant Gaikwad, Prasoon Varshney, Pritam Gundecha, Przemek Tredak, Rabeeh Karimi Mahabadi, Rajen Patel, Ran El-Yaniv, Ranjit Rajan, Ria Cheruvu, Rima Shahbazyan, Ritika Borkar, Ritu Gala, Roger Waleffe, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Sahil Jain, Samuel Kriman, Sanjeev Satheesh, Saori Kaji, Sarah Yurick, Saurav Muralidharan, Sean Narenthiran, Seonmyeong Bak, Sepehr Sameni, Seungju Han, Shanmugam Ramasamy, Shaona Ghosh, Sharath Turuvekere Sreenivas, Shelby Thomas, Shizhe Diao, Shreya Gopal, Shrimai Prabhumoye, Shubham Toshniwal, Shuoyang Ding, Siddharth Singh, Siddhartha Jain, Somshubra Majumdar, Soumye Singhal, Stefania Alborghetti, Syeda Nahida Akter, Terry Kong, Tim Moon, Tomasz Hliwiak, Tomer Asida, Tony Wang, Tugrul Konuk, Twinkle Vashishth, Tyler Poon, Udi Karpas, Vahid Noroozi, Venkat Srinivasan, Vijay Korthikanti, Vikram Fugro, Vineeth Kalluru, Vitaly Kurin, Vitaly Lavrukhin, Wasi Uddin Ahmad, Wei Du, Wonmin Byeon, Ximing Lu, Xin Dong, Yashaswi Karnati, Yejin Choi, Yian Zhang, Ying Lin, Yonggan Fu, Yoshi Suhara, Zhen Dong, Zhiyu Li, Zhongbo Zhu, and Zijia Chen. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025a. URL `https://arxiv.org/abs/2508.14444`.

NVIDIA, Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, Aleksandr Shaposhnikov, Alex Kondratenko, Alexander Bukharin, Alexandre Milesi, Ali Taghibakhshi, Alisa Liu, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amir Klein, Amit Zuker, Amnon Geifman, Amy Shen, Anahita Bhiwandiwalla, Andrew Tao, Ann Guan, Anubhav Mandarwal, Arham Mehta, Ashwath Aithal, Ashwin Poojary, Asif Ahamed, Asma Kuriparambil Thekkumpate, Ayush Dattagupta, Banghua Zhu, Bardiya Sadeghi, Barnaby Simkin, Ben Lanir, Benedikt Schifferer, Besmira Nushi, Bilal Kartal, Bita Darvish Rouhani, Boris Ginsburg, Brandon Norick, Brandon Soubasis, Branislav Kisacanin, Brian Yu, Bryan Catanzaro, Carlo del Mundo, Chantal Hwang, Charles Wang, Cheng-Ping Hsieh, Chenghao Zhang, Chenhan Yu, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Christopher Parisien, Collin Neale, Damon Mosk-Aoyama, Dan Su, Dane Corneil, Daniel Afrimi, Daniel Rohrer, Daniel Serebrenik, Daria Gitman, Daria Levy, Darko Stosic, David Mosallanezhad, Deepak Narayanan, Dhruv Nathawani, Dima Rekesh, Dina Yared, Divyanshu Kakwani, Dong Ahn, Duncan Riach, Dusan Stosic, Edgar Minasyan, Edward Lin, Eileen Long, Eileen Peters Long, Elena Lantz, Ellie Evans, Elliott Ning, Eric Chung, Eric Harper, Eric Tramel, Erick Galinkin, Erik Pounds, Evan Briones, Evelina Bakhturina, Faisal Ladhak, Fay Wang, Fei Jia, Felipe Soares, Feng Chen, Ferenc Galko, Frankie Siino, Gal Hubara Agam, Ganesh Ajjanagadde, Gantavya Bhatt, Gargi Prasad, George Armstrong, Gerald Shen, Gorkem Batmaz, Grigor Nalbandyan, Haifeng Qian, Harsh Sharma, Hayley Ross, Helen Ngo, Herman Sahota, Hexin Wang, Himanshu Soni, Hiren Upadhyay, Huizi Mao, Huy C Nguyen, Huy Q Nguyen, Iain Cunningham, Ido Shahaf, Igor Gitman, Ilya Loshchilov, Ivan Moshkov, Izzy Putterman, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jatin Mitra, Jeffrey Glick, Jenny Chen, Jesse Oliver, Jian Zhang, Jiaqi Zeng, Jie Lou, Jimmy Zhang, Jining Huang, Joey Conway, Joey Guman, John Kamalu, Johnny Greco, Jonathan Cohen, Joseph Jennings, Joyjit Daw, Julien Veron Vialard, Junkeun Yi, Jupinder Parmar, Kai Xu, Kan Zhu, Kari Briski, Katherine Cheung, Katherine Luna, Keshav Santhanam, Kevin Shih, Kezhi Kong, Khushi Bhardwaj, Krishna C. Puvvada, Krzysztof Pawelec, Kumar Anik, Lawrence McAfee, Laya Sleiman, Leon Derczynski, Li Ding, Lucas Liebenwein, Luis Vega, Maanu Grover, Maarten Van Segbroeck, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Manoj Kilaru, Maor Ashkenazi, Marc Romeijn, Mark Cai, Markus Kliegl, Maryam Moosaei, Matvei Novikov, Mehrzad Samadi, Melissa Corpuz, Mengru Wang, Meredith Price, Michael Boone, Michael Evans, Miguel Martinez, Mike Chrzanowski, Mohammad Shoeybi, Mostofa Patwary, Nabin Mulepati, Natalie Hereth, Nave Assaf, Negar Habibi, Neta Zmora, Netanel Haber, Nicola Sessions, Nidhi Bhatia, Nikhil Jukar, Nikki Pope, Nikolai Ludwig, Nima Tajbakhsh, Nirmal Juluru, Oleksii Hrinchuk, Oleksii Kuchaiev, Olivier Delalleau, Oluwatobi Olabiyi, Omer Ullman Argov, Ouye Xie, Parth Chadha, Pasha Shamis, Pavlo Molchanov, Pawel Morkisz, Peter Dykas, Peter Jin, Pinky Xu, Piotr Januszewski, Pranav Prashant Thombre, Prasoon Varshney, Pritam

Gundecha, Qing Miao, Rabeeh Karimi Mahabadi, Ran El-Yaniv, Ran Zilberstein, Rasoul Shafipour, Rich Harang, Rick Izzo, Rima Shahbazyan, Rishabh Garg, Ritika Borkar, Ritu Gala, Riyad Islam, Roger Waleffe, Rohit Watve, Roi Koren, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Ryan Timbrook, Sadegh Mahdavi, Sahil Modi, Samuel Kriman, Sanjay Kariyappa, Sanjeev Satheesh, Saori Kaji, Satish Pasumarthi, Sean Narentharen, Sean Narenthiran, Seonmyeong Bak, Sergey Kashirsky, Seth Poulos, Shahar Mor, Shanmugam Ramasamy, Shantanu Acharya, Shaona Ghosh, Sharath Turuvekere Sreenivas, Shelby Thomas, Shiqing Fan, Shreya Gopal, Shrimai Prabhumoye, Shubham Pachori, Shubham Toshniwal, Shuoyang Ding, Siddharth Singh, Simeng Sun, Smita Ithape, Somshubra Majumdar, Soumye Singhal, Stefania Alborghetti, Stephen Ge, Sugam Dipak Devare, Sumeet Kumar Barua, Suseella Panguluri, Suyog Gupta, Sweta Priyadarshi, Syeda Nahida Akter, Tan Bui, Teodor-Dumitru Ene, Terry Kong, Thanh Do, Tijmen Blankevoort, Tom Balough, Tomer Asida, Tomer Bar Natan, Tugrul Konuk, Twinkle Vashishth, Udi Karpas, Ushnish De, Vahid Noorozi, Vahid Noroozi, Venkat Srinivasan, Venmugil Elango, Vijay Korthikanti, Vitaly Kurin, Vitaly Lavrukhin, Wanli Jiang, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wenfei Zhou, Will Jennings, William Zhang, Wojciech Prazuch, Xiaowei Ren, Yashaswi Karnati, Yejin Choi, Yev Meyer, Yi-Fu Wu, Yian Zhang, Ying Lin, Yonatan Geifman, Yonggan Fu, Yoshi Subara, Yoshi Suhara, Yubo Gao, Zach Moshe, Zhen Dong, Zihan Liu, Zijia Chen, and Zijie Yan. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2025b. URL `https://arxiv.org/abs/2512.20848`.

Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL `https://arxiv.org/abs/2512.13961`.

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. URL `https://arxiv.org/abs/2412.16720`.

Jake Poznanski, Aman Rangapur, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Aman Rangapur, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. olmocr: Unlocking trillions of tokens in pdfs with vision language models, 2025. URL `https://arxiv.org/abs/2502.18443`.

Prime Intellect Team, Mika Senghaas, Fares Obeid, Sami Jaghouar, William Brown, Jack Min Ong, Daniel Auras, Matej Sirovatka, Jannik Straube, Andrew Baker, Sebastian Müller, Justus Mattern, Manveer Basra, Aiman Ismail, Dominik Scherm, Cooper Miller, Ameen Patel, Simon Kirsten, Mario Sieg, Christian Reetz, Kemal Erdem, Vincent Weisser, and Johannes Hagemann. Intellect-3: Technical report, 2025. URL `https://arxiv.org/abs/2512.16144`.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025. URL `https://arxiv.org/abs/2505.06708`.

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 qa benchmark, 2023. URL `https://arxiv.org/abs/2311.12022`.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL `https://arxiv.org/abs/1907.10641`.

Noam Shazeer. Glu variants improve transformer, 2020. URL `https://arxiv.org/abs/2002.05202`.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL `https://arxiv.org/abs/1701.06538`.

Yusuke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, and Takeshi Shinohara. Byte pair encoding: A text compression scheme that accelerates pattern matching. 09 1999.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL `https://arxiv.org/abs/2104.09864`.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, 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, 2022. URL `https://arxiv.org/abs/2210.09261`.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the pre-training of large language models, 2025. URL `https://arxiv.org/abs/2312.16903`.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2025. URL `https://arxiv.org/abs/2501.00656`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL `https://arxiv.org/abs/1706.03762`.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2024a. URL `https://arxiv.org/abs/2408.15664`.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024b. URL https://arxiv.org/abs/2406.01574.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models, 2024. URL https://arxiv.org/abs/2411.04368.

Erik Wijmans, Brody Huval, Alexander Hertzberg, Vladlen Koltun, and Philipp Krähenbühl. Cut your losses in large-vocabulary language models. In *International Conference on Learning Representations*, 2025.

Mitchell Wortsman, Peter J. Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities, 2023. URL https://arxiv.org/abs/2309.14322.

xAI. Grok-1 (xai-org/grok-1). Hugging Face model repository, March 2024. URL https://huggingface.co/xai-org/grok-1. Model weights repository on Hugging Face. Accessed 2026-01-26.

xAI. Grok 2 (xai-org/grok-2). Hugging Face model repository, August 2025. URL https://huggingface.co/xai-org/grok-2. Model weights repository on Hugging Face. Accessed 2026-01-26.

Xiaomi LLM-Core Team, Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, Gang Xie, Hailin Zhang, Hanglong Lv, Hanyu Li, Heyu Chen, Hongshen Xu, Houbin Zhang, Huaqiu Liu, Jiangshan Duo, Jianyu Wei, Jiebao Xiao, Jinhao Dong, Jun Shi, Junhao Hu, Kainan Bao, Kang Zhou, Lei Li, Liang Zhao, Linghao Zhang, Peidian Li, Qianli Chen, Shaohui Liu, Shihua Yu, Shijie Cao, Shimao Chen, Shouqiu Yu, Shuo Liu, Tianling Zhou, Weijiang Su, Weikun Wang, Wenhan Ma, Xiangwei Deng, Bohan Mao, Bowen Ye, Can Cai, Chenghua Wang, Chengxuan Zhu, Chong Ma, Chun Chen, Chunan Li, Dawei Zhu, Deshan Xiao, Dong Zhang, Duo Zhang, Fangyue Liu, Feiyu Yang, Fengyuan Shi, Guoan Wang, Hao Tian, Hao Wu, Heng Qu, Hongfei Yi, Hongxu An, Hongyi Guan, Xing Zhang, Yifan Song, Yihan Yan, Yihao Zhao, Yingchun Lai, Yizhao Gao, Yu Cheng, Yuanyuan Tian, Yudong Wang, Zhen Tang, Zhengju Tang, Zhengtao Wen, Zhichao Song, Zhixian Zheng, Zihan Jiang, Jian Wen, Jiarui Sun, Jiawei Li, Jinlong Xue, Jun Xia, Kai Fang, Menghang Zhu, Nuo Chen, Qian Tu, Qihao Zhang, Qiying Wang, Rang Li, Rui Ma, Shaolei Zhang, Shengfan Wang, Shicheng Li, Shuhao Gu, Shuhuai Ren, Sirui Deng, Tao Guo, Tianyang Lu, Weiji Zhuang, Weikang Zhang, Weimin Xiong, Wenshan Huang, Wenyu Yang, Xin Zhang, Xing Yong, Xu Wang, Xueyang Xie, Yilin Jiang, Yixin Yang, Yongzhe He, Yu Tu, Yuanliang Dong, Yuchen Liu, Yue Ma, Yue Yu, Yuxing Xiang, Zhaojun Huang, Zhenru Lin, Zhipeng Xu, Zhiyang Chen, Zhonghua Deng, Zihan Zhang, and Zihao Yue. Mimo-v2-flash technical report, 2026. URL https://arxiv.org/abs/2601.02780.

Bowen Yang, Bharat Venkitesh, Dwarak Talupuru, Hangyu Lin, David Cairuz, Phil Blunsom, and Acyr Locatelli. Rope to nope and back again: A new hybrid attention strategy, 2025. URL https://arxiv.org/abs/2501.18795.

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, 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, and Zhicheng Liu. Pangu ultra: Pushing the limits of dense large language models on ascend npus, 2025. URL https://arxiv.org/abs/2504.07866.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Chen Zhang, Yang Bai, Jiahuan Li, Anchun Gui, Keheng Wang, Feifan Liu, Guanyu Wu, Yuwei Jiang, Defei Bu, Li Wei, Haihang Jing, Hongyin Tang, Xin Chen, Xiangzhou Huang, Fengcun Li, Rongxiang Weng, Yulei Qian, Yifan Lu, Yerui Sun, Jingang Wang, Yuchen Xie, and Xunliang Cai. Efficient context scaling with longcat zigzag attention, 2026. URL https://arxiv.org/abs/2512.23966.

Yifan Zhang, Yifan Luo, Yang Yuan, and Andrew C Yao. Automathtext: Autonomous data selection with zero-shot generative classifiers for mathematical texts, 2025. URL `https://arxiv.org/abs/2402.07625`.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023. URL `https://arxiv.org/abs/2304.11277`.

Jingwei Zuo, Maksim Velikanov, Ilyas Chahed, Younes Belkada, Dhia Eddine Rhayem, Guillaume Kunsch, Hakim Hacid, Hamza Yous, Brahim Farhat, Ibrahim Khadraoui, Mugariya Farooq, Giulia Campesan, Ruxandra Cojocaru, Yasser Djilali, Shi Hu, Iheb Chaabane, Puneesh Khanna, Mohamed El Amine Seddik, Ngoc Dung Huynh, Phuc Le Khac, Leen AlQadi, Billel Mokeddem, Mohamed Chami, Abdalgader Abubaker, Mikhail Lubinets, Kacper Piskorski, and Slim Frikha. Falcon-h1: A family of hybrid-head language models redefining efficiency and performance, 2025. URL `https://arxiv.org/abs/2507.22448`.