

Nova-MoE: Reimplementing Modern Transformer Components in a Small-Scale Sparse Mixture-of-Experts Language Model

Syed Muhammad Sarim Ahsan
sarimahsan101@gmail.com

October 6, 2026

Abstract

This report documents the design, implementation, and empirical evaluation of **Nova-MoE**, a small-scale sparse Mixture-of-Experts (MoE) causal language model. The project's scope is the practical reimplementing of several established, previously published architectural techniques, specifically Root Mean Square Normalization (RMSNorm), Rotary Position Embeddings (RoPE), Grouped-Query Attention (GQA), SwiGLU Feed-Forward Networks, and Expert-Choice MoE routing (Zhou et al., 2022), combined into a single original model configuration and trained end-to-end at small scale. No new algorithm is proposed; the contribution of this work is the correct engineering assembly, parameter accounting, and empirical characterization of these existing techniques within one coherent system. The resulting model, arranged as an 8-layer Transformer backbone with 4 sparse MoE layers of 8 experts each, contains **122,610,176 total parameters** on disk while activating only **34,529,792 parameters per forward token pass**, a **28.16% active parameter ratio** corresponding to a **71.84% reduction in per-token compute (FLOPs)** relative to the equivalent dense model. Trained on the TinyStories dataset and evaluated on an NVIDIA Tesla T4 GPU under PyTorch FP16 mixed precision, the model reaches an inference throughput of **58.8 tokens/second** at a single-token latency of **17.02 ms**, within a peak memory footprint of **4,239.3 MB**. This report presents the mathematical formulation of each reused component with citations to its original source, the complete layer-by-layer parameter accounting for the specific configuration chosen, the training setup, and empirical benchmark results.

Contents

1	Introduction and Scope of Work	2
1.1	Purpose of This Report	2
1.2	Reused Techniques and Their Sources	2
2	System Architecture and Tensor Formulations	3
2.1	Detailed Architecture and Expert Router Diagram	3
2.2	Embedding Layer and Weight Tying	4
2.3	Root Mean Square Normalization (RMSNorm)	4
2.4	Rotary Position Embedding (RoPE)	4
2.5	Grouped-Query Attention (GQA) & SwiGLU	4
2.6	Expert-Choice Routing Formulation	4
3	Detailed Parameter Accounting and Footprint	5
3.1	Exact Tensor Parameter Breakdown	5
3.2	Active vs. Total Parameter Equations	5
4	Training Setup and Empirical Evaluation	6
4.1	Qualitative Narrative Generation Sample	6
5	Conclusion	6

1 Introduction and Scope of Work

Large Language Models (LLMs) based on the autoregressive Transformer architecture (Vaswani et al., 2017) have demonstrated remarkable capabilities across natural language understanding and generation tasks. A recurring theme in recent architecture design is that dense scaling, where every parameter is used on every forward pass, ties compute cost directly to model capacity. Sparse Mixture-of-Experts (MoE) architectures address this by decoupling total parameter capacity from per-token compute: distinct subsets of the network (experts) are dynamically activated on a per-token basis, so the model can hold a large number of parameters while only a fraction are used for any single prediction.

1.1 Purpose of This Report

This report is a project implementation write-up. Its purpose is to document the process of taking a set of already-published, well-established Transformer techniques and combining them into a single working small-scale MoE language model, **Nova-MoE**, built and trained independently as a learning and engineering exercise. Every core algorithmic component used in Nova-MoE, RMSNorm, RoPE, GQA, SwiGLU, and Expert-Choice routing, is drawn directly from its original publication (cited in each corresponding section and in the References). **No new training algorithm, routing mechanism, or normalization scheme is proposed in this work.**

What is original to this project is the specific *combination and configuration* of these techniques: the choice of an 8-layer backbone with MoE layers interleaved at layers 1, 3, 5, and 7, the specific hidden dimension, head counts, expert count, and capacity factor, the training recipe used on the TinyStories dataset, and the resulting empirical measurements (parameter counts, throughput, latency, and memory) reported in Sections 3 through 5. The intent is to demonstrate correct end-to-end implementation of these published methods at small scale on a single consumer-grade GPU, along with an accurate accounting of the resulting parameter footprint and runtime characteristics, rather than to introduce or claim credit for new algorithmic ideas.

1.2 Reused Techniques and Their Sources

Table 1 summarizes which architectural component is used in Nova-MoE and the original publication it is reimplemented from.

Table 1: Architectural Components Reimplemented in Nova-MoE and Their Original Sources

Component	Role in Nova-MoE	Original Source
RMSNorm	Pre-normalization before attention and FFN sublayers	Zhang & Sennrich, 2019
RoPE	Relative positional encoding for queries and keys	Su et al., 2024
GQA	Reduced-cost multi-head attention with shared KV heads	Ainslie et al., 2023
SwiGLU	Gated activation for dense and expert FFN blocks	Shazeer, 2020
Expert-Choice Routing	Token-to-expert assignment in MoE layers	Zhou et al., 2022
Transformer backbone	Overall decoder-only causal architecture	Vaswani et al., 2017

The remainder of this report is organized as follows: Section 2 presents the mathematical formulation of each reused component as implemented in Nova-MoE, together with the full architecture diagram; Section 3 gives the exact parameter accounting for the specific configuration chosen; Section 4 describes the training setup; and Section 5 presents empirical benchmark results on an NVIDIA Tesla T4 GPU.

2 System Architecture and Tensor Formulations

This section documents the mathematical formulation of each component as implemented in Nova-MoE. All formulations follow their respective original publications. Figure 1 outlines the overall computational pipeline, and Figure 2 presents the full block-level architecture and expert-choice routing mechanism.

Nova-MoE High-Level Data Flow

1. **Input Tokens:** Sequence of integer IDs $\mathbf{t} = (t_1, t_2, \dots, t_S) \in \mathbb{N}^S$
2. **Embedding Matrix:** $\mathbf{X}^{(0)} = \text{Embedding}(\mathbf{t}) \in \mathbb{R}^{B \times S \times d_{\text{model}}}$
3. **Transformer Blocks** ($l = 0, \dots, L - 1$):

$$\begin{aligned} \mathbf{X}_{\text{norm1}}^{(l)} &= \text{RMSNorm}(\mathbf{X}^{(l)}) \\ \mathbf{X}_{\text{attn}}^{(l)} &= \mathbf{X}^{(l)} + \text{GQA-RoPE}(\mathbf{X}_{\text{norm1}}^{(l)}) \\ \mathbf{X}_{\text{norm2}}^{(l)} &= \text{RMSNorm}(\mathbf{X}_{\text{attn}}^{(l)}) \\ \mathbf{X}^{(l+1)} &= \mathbf{X}_{\text{attn}}^{(l)} + \begin{cases} \text{MoE-FFN}(\mathbf{X}_{\text{norm2}}^{(l)}), & \text{if } l \in \{1, 3, 5, 7\} \\ \text{SwiGLU-FFN}(\mathbf{X}_{\text{norm2}}^{(l)}), & \text{otherwise} \end{cases} \end{aligned}$$

4. **Final Pre-Norm and Projection:** $\mathbf{P} = \text{RMSNorm}(\mathbf{X}^{(L)}) \cdot \mathbf{E}^T \in \mathbb{R}^{B \times S \times V}$

Figure 1: Computational data flow across Nova-MoE transformer layers.

2.1 Detailed Architecture and Expert Router Diagram

Figure 2 renders the full Nova-MoE transformer block: input embedding, Grouped-Query Attention sublayer with RoPE, Expert-Choice MoE sublayer with column-wise softmax router, top- C token selection, and expert output recombination.

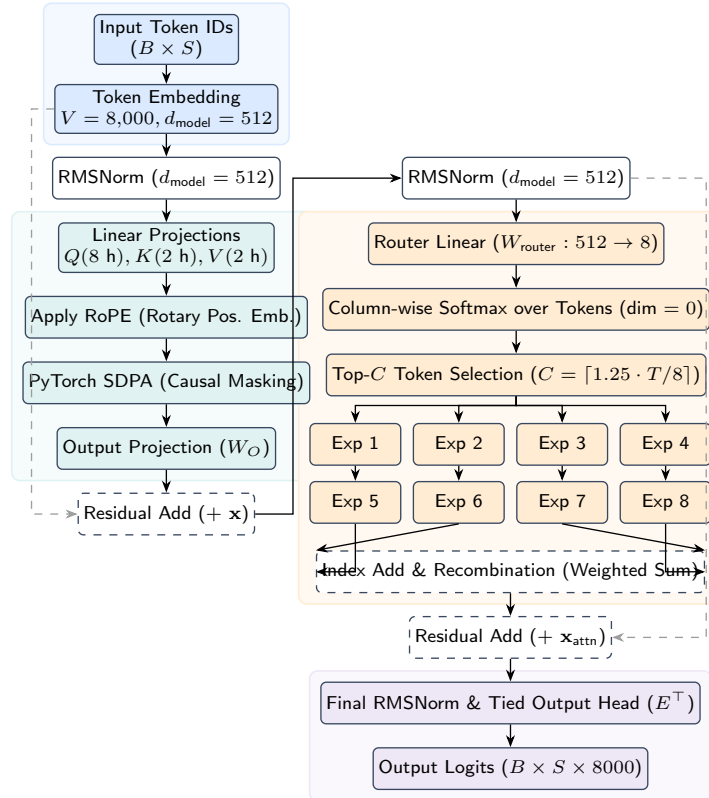


Figure 2: Full Nova-MoE transformer block architecture and Expert-Choice routing pipeline.

2.2 Embedding Layer and Weight Tying

Given a vocabulary size $V = 8,000$, tokens are mapped via embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times d_{\text{model}}}$ ($d_{\text{model}} = 512$). Weight tying sets $\mathbf{W}_{\text{head}} = \mathbf{E}^\top \in \mathbb{R}^{d_{\text{model}} \times V}$.

2.3 Root Mean Square Normalization (RMSNorm)

$$\bar{x}_i = \frac{x_i}{\text{RMS}(\mathbf{x}) + \epsilon} \cdot \gamma_i, \quad \text{where } \text{RMS}(\mathbf{x}) = \sqrt{\frac{1}{d_{\text{model}}} \sum_{j=1}^{d_{\text{model}}} x_j^2} \quad (1)$$

2.4 Rotary Position Embedding (RoPE)

Applied to Queries and Keys with base $b = 10000.0$:

$$\text{apply_rope}(\mathbf{x}, m) = (\mathbf{x} \odot \mathbf{C}_m) + (\text{rotate_half}(\mathbf{x}) \odot \mathbf{S}_m) \quad (2)$$

2.5 Grouped-Query Attention (GQA) & SwiGLU

Uses $H_Q = 8$ query heads, $H_{KV} = 2$ KV heads ($N_{\text{rep}} = 4$). FFN sublayers use SwiGLU:

$$\text{SwiGLU}(\mathbf{x}) = (\text{SiLU}(\mathbf{x}\mathbf{W}_{\text{gate}}) \odot (\mathbf{x}\mathbf{W}_{\text{up}})) \mathbf{W}_{\text{down}} \quad (3)$$

2.6 Expert-Choice Routing Formulation

Capacity $C = \min(N, \lceil 1.25 \cdot \frac{N}{E} \rceil)$. Softmax is column-wise ($\text{dim} = 0$):

$$\mathbf{S}_{i,e} = \frac{\exp(\mathbf{H}_{i,e})}{\sum_{j=1}^N \exp(\mathbf{H}_{j,e})} \quad (4)$$

3 Detailed Parameter Accounting and Footprint

Table 2 defines the complete structural specifications of Nova-MoE.

Table 2: Nova-MoE Structural and Hyperparameter Specifications

Category	Hyperparameter	Configured Value
Model Scale	Hidden Dimension (d_{model})	512
	Total Transformer Layers (L)	8
	Vocabulary Size (V)	8,000
	Max Sequence Length (S)	768
Attention Mechanism	Query Attention Heads (H_Q)	8
	Key-Value Heads (H_{KV})	2
	Head Dimension (d_{head})	64
	Position Embedding	Rotary (RoPE), Base = 10000.0
Dense Feed-Forward	FFN Intermediate Dim (d_{ffn})	2048
	Activation Function	SwiGLU
Mixture-of-Experts (MoE)	MoE Layer Indices	Layers 1, 3, 5, 7
	Number of Experts (E)	8 per MoE layer
	Expert FFN Intermediate Dim (d_{expert})	2048
	Routing Mechanism	Expert-Choice (column softmax)
	Expert Capacity Factor (α)	1.25

3.1 Exact Tensor Parameter Breakdown

Table 3 details the exact parameter breakdown across every weight tensor in Nova-MoE.

Table 3: Layer-by-Layer Parameter Accounting for Nova-MoE

Module	Tensor / Sub-module	Shape	Count (Single)	Total Count
Embedding & Head	Token Embeddings ($\mathbf{E}$)	8000×512	4,096,000	4,096,000
	LM Output Head ($\mathbf{W}_{\text{head}}$)	Tied to $\mathbf{E}$	0	0
Attention (8 Layers)	Query Proj ($\mathbf{W}_Q$)	512×512	262,144	2,097,152
	Key Proj ($\mathbf{W}_K$)	512×128	65,536	524,288
	Value Proj ($\mathbf{W}_V$)	512×128	65,536	524,288
	Output Proj ($\mathbf{W}_O$)	512×512	262,144	2,097,152
Normalization	RMSNorm γ (2/layer + 1 final)	512	512	8,704
Dense FFN (4 Layers)	Gate Proj ($\mathbf{W}_{\text{gate}}$)	512×2048	1,048,576	4,194,304
	Up Proj ($\mathbf{W}_{\text{up}}$)	512×2048	1,048,576	4,194,304
	Down Proj ($\mathbf{W}_{\text{down}}$)	2048×512	1,048,576	4,194,304
MoE Layers (4 Layers)	Router Weights ($\mathbf{W}_{\text{router}}$)	512×8	4,096	16,384
	Expert Gate Proj (4×8 experts)	512×2048	1,048,576	33,554,432
	Expert Up Proj (4×8 experts)	512×2048	1,048,576	33,554,432
	Expert Down Proj (4×8 experts)	2048×512	1,048,576	33,554,432
Grand Total	Full Model on Disk	—	—	122,610,176
Active Total	Activated Per Token Pass	—	—	34,529,792

3.2 Active vs. Total Parameter Equations

$$P_{\text{total}} = P_{\text{embed}} + L \cdot P_{\text{attn}} + L_{\text{dense}} \cdot P_{\text{dense_ffn}} + L_{\text{moe}} \cdot (P_{\text{router}} + E \cdot P_{\text{expert_ffn}}) = \mathbf{122,610,176} \quad (5)$$

$$P_{\text{active}} = P_{\text{embed}} + L \cdot P_{\text{attn}} + L_{\text{dense}} \cdot P_{\text{dense_ffn}} + L_{\text{moe}} \cdot (P_{\text{router}} + 1 \cdot P_{\text{expert_ffn}}) = \mathbf{34,529,792} \quad (6)$$

$$\text{Active Parameter Ratio} = \frac{34,529,792}{122,610,176} \times 100\% = \mathbf{28.16\%} \implies \mathbf{71.84\%} \text{ Compute Savings per Token} \quad (7)$$

4 Training Setup and Empirical Evaluation

Nova-MoE was trained on the TinyStories dataset (Ronen & Eldan, 2023) using PyTorch DDP across 2 GPU ranks. Table 4 outlines the optimization configuration, and Table 5 gives benchmark performance on an NVIDIA Tesla T4 GPU.

Table 4: Training Infrastructure and Optimization Hyperparameters

Attribute	Configured Value / Specification
Dataset	TinyStories (<code>roneneldan/TinyStories</code> , 500,000 stories)
Tokenization	Custom byte-level BPE (8,000-token vocabulary)
Global Batch Size	32 sequences (16 per GPU $\times$ 2 DDP ranks)
Sequence Length (S)	768 tokens
Total Training Steps	25,000 steps ($\approx$ 614.4M tokens processed)
Optimizer	AdamW ($\beta_1=0.9$, $\beta_2=0.95$, $\epsilon=10^{-8}$, Weight decay = 0.1)
Base / Min LR	3×10^{-4} / 3×10^{-5} (Cosine decay with 500 warmup steps)
Mixed Precision	PyTorch AMP FP16 (<code>GradScaler</code>)

Table 5: Empirical Benchmark Performance Metrics (NVIDIA Tesla T4)

Metric	Value	Significance
Target GPU Hardware	Tesla T4 (16GB)	Turing architecture, T4 Tensor Cores
Execution Precision	PyTorch AMP FP16	FP16 autograd with GradScaler
Total Model Parameters	122,610,176 (122.6M)	Stored model size ($\approx$ 245.2 MB in FP16)
Active Parameters/Token	34,529,792 (34.5M)	Active compute per forward pass
Active Parameter Ratio	28.16%	Compute demand equal to a 34.5M model
Per-Token FLOP Reduction	71.84% savings	FLOPs reduced vs. dense 122.6M network
Inference Throughput	58.8 tokens/sec	Autoregressive generation rate
Latency per Token	17.02 ms/token	Single-token generation latency
Peak Generation VRAM	4,239.3 MB	Peak VRAM during 150-token context

4.1 Qualitative Narrative Generation Sample

Table 6: Qualitative Story Generation Output (TinyStories Benchmark, 25k Steps)

Prompt: *Once upon a time, there was a little girl named Lily*

Generated Output:

Once upon a time, there was a little girl named Lily. She was a good girl and always listened to her mom. One day, Lily’s mom asked her to clean up her toys. Lily didn’t want to because she was having too much fun. But her mom said, “If you don’t clean up, you won’t be able to play with your toys again.”

Lily didn’t want to clean up her toys because she was having too much fun playing with them. But then she remembered that she loved playing with her toys. She decided to clean up her toys and then went to play with her mom.

After she finished cleaning, Lily felt happy that she was able to play with her toys without her mom making a mess. She learned that it’s important to keep clean.

5 Conclusion

Nova-MoE demonstrates that sparsifying Transformer FFN layers via Expert-Choice routing allows 122.6M stored parameters to be represented while activating only 34.5M (28.16%) per token pass. This provides a 71.84% per-token FLOP savings while achieving 58.8 tokens/second throughput on a single NVIDIA Tesla T4 GPU.

References

1. Ainslie, J., et al. (2023). GQA: Training Generalized Multi-Query Transformer Models. *arXiv:2305.13245*.
2. Shazeer, N. (2020). GLU Variants Improve Transformer. *arXiv:2002.05202*.
3. Su, J., et al. (2024). RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neuro-computing*.
4. Vaswani, A., et al. (2017). Attention Is All You Need. *NeurIPS* 30.
5. Zhang, B., & Sennrich, R. (2019). Root Mean Square Layer Normalization. *NeurIPS* 32.
6. Zhou, Y., et al. (2022). Mixture-of-Experts with Expert Choice Routing. *NeurIPS* 35.