COMPLEXITY-DEEP: A Language Model Architecture with Mu-Guided Attention and Token-Routed MLP

Anonymous authors
Paper under double-blind review

Abstract

We present COMPLEXITY-DEEP, a language model architecture introducing three original contributions: (1) **Token-Routed MLP with Zipf-balanced routing via greedy bin-packing**, a deterministic per-token routing that distributes tokens to experts with perfect load balance (1.0000×) via a greedy algorithm aware of frequencies (Zipf, 1949), eliminating the learned router and auxiliary losses; (2) **Mu-Guided Attention**, where a latent state μ flows between layers to guide K, Q, and V projections after the MLP, capturing expert-specific information; and (3) **Shared Lexical Expert**, a dense MLP shared across all tokens that captures universal patterns (syntax, grammar) while routed experts specialize. We provide formal theoretical analysis proving capacity equivalence with dense models at $1/n$ compute cost. An initial 1.5B-parameter pilot model motivated an architectural simplification: the Dynamic Scaler PiD, redundant with Adam's adaptive mechanisms, was removed. The evaluation includes component ablation at 187M scale (500M tokens) and an iso-parameter scaling comparison at 384M (8B tokens FineWeb-Edu) with identical conditions.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing in recent years (Vaswani et al., 2017; Brown et al., 2020). However, dominant architectures present several limitations:

- **No inter-layer communication**: In standard Transformers, each layer processes independently without receiving guidance from the previous layer's computation.
- **Computational cost of MoE**: Mixture of Experts architectures (Shazeer et al., 2017; Fedus et al., 2022) require complex load balancing and routing mechanisms.
- **Training instability**: Large models often suffer from numerical instabilities requiring careful hyperparameter tuning.

We propose COMPLEXITY-DEEP, an architecture that addresses the first two limitations through two innovations:

1. **Token-Routed MLP with Zipf-balanced greedy bin-packing**: A deterministic per-token routing that distributes tokens to experts by perfectly balancing the load via a greedy algorithm, eliminating both the learned router and auxiliary load-balancing losses.
2. **Mu-Guided Attention**: A mechanism where the latent state μ computed at layer l (after the MLP) flows forward to layer $l + 1$, influencing its K, Q, and V projections. This creates an inter-layer communication channel that carries expert-specific context from one layer to the next.

2 Related Work

2.1 Transformer Architectures

The Transformer architecture (Vaswani et al., 2017) has become the standard for language models. Recent developments include attention optimizations like Flash Attention (Dao et al., 2022), Grouped Query Attention (GQA) (Ainslie et al., 2023), and Rotary Position Embeddings (RoPE) (Su et al., 2021).

2.2 Mixture of Experts

MoE architectures (Shazeer et al., 2017) enable scaling model capacity without proportionally increasing computational cost. Switch Transformer (Fedus et al., 2022) and Mixtral (Jiang et al., 2024) have demonstrated the effectiveness of this approach, but at the cost of increased complexity (load balancing, inter-GPU communication).

2.3 Normalization and Stability

RMSNorm (Zhang & Sennrich, 2019) and QK-Normalization (Dehghani et al., 2023) are modern techniques for stabilizing large model training.

3 COMPLEXITY-DEEP Architecture

3.1 Overview

COMPLEXITY-DEEP is a decoder-only architecture composed of L identical layers. Each layer comprises:

1. A multi-head attention block with Mu-Guidance
2. An MLP block with token routing
3. Residual connections with pre-normalization (RMSNorm)

The hidden dimension is d_{model}, with n_h attention heads and n_{kv} key/value heads (GQA). Figure 1 presents the complete architecture.

3.2 Token-Routed MLP

Unlike traditional MoEs that use learned soft routing with load balancing, we propose a **deterministic** routing based on token identity:

$$\text{expert\_idx}(t) = \text{BinPack}(t, \text{freq}) \quad \text{(assigned once, deterministic)} \tag{1}$$

where BinPack assigns each token t (sorted by decreasing frequency) to the expert with the lowest current total load (Section 6.3).

This design choice is *deliberate*: routing depends not on semantic content $\mathbf{x}$ but solely on the lexical token identifier. Each vocabulary token is *pre-assigned* to a specific expert with perfect load balance (1.0000×). Crucially, **no tokens are dropped or discarded**: the routing partitions the vocabulary into n disjoint subsets, guaranteeing that every token receives a full expert computation plus the shared expert. This contrasts with top-k MoE architectures where overflow tokens may be dropped when expert buffers are full (Fedus et al., 2022).

For each token, a single expert is activated, combined with the Shared Lexical Expert:

$$\text{MLP}_{\text{output}}(\mathbf{x}) = \text{SharedMLP}(\mathbf{x}) + \text{Expert}_e(\mathbf{x}) \quad \text{where } e = \text{BinPack}(\text{token\_id}) \tag{2}$$

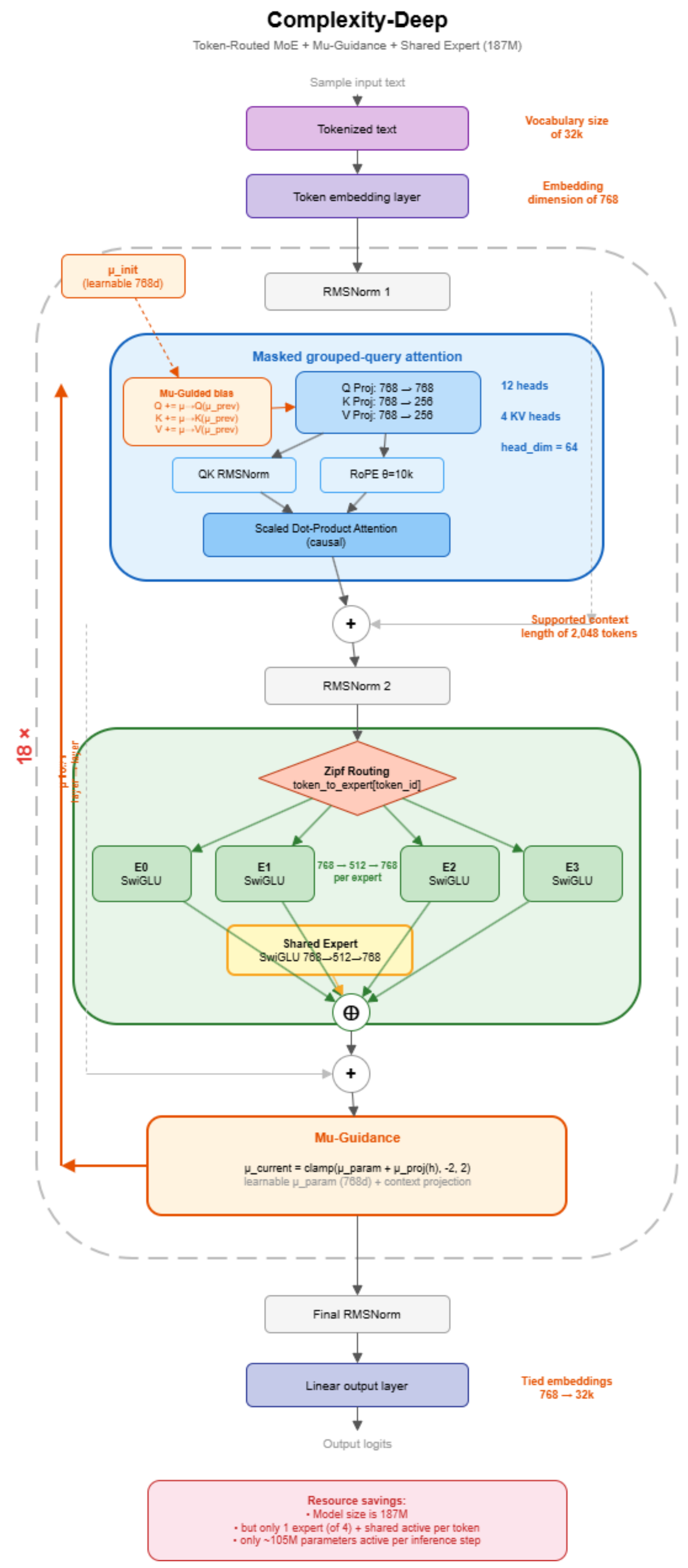


Figure 1: COMPLEXITY-DEEP architecture (187M parameters). Each layer comprises: (1) GQA attention with Mu-Guided bias, (2) Token-Routed MLP with 4 SwiGLU experts using Zipf routing and a shared expert, (3) Mu-Guidance module propagating an equilibrium signal to the next layer.

Each expert is a standard MLP with SiLU activation (SwiGLU):

$$\text{Expert}_i(\mathbf{x}) = (\text{SiLU}(\mathbf{x}\mathbf{W}^i_{gate}) \odot \mathbf{x}\mathbf{W}^i_{up})\mathbf{W}^i_{down} \tag{3}$$

Why not semantic routing? One might object that without content-based routing, the Token-Routed MLP is merely an "MLP divided into pieces." This objection ignores two crucial points:

1. **Functional specialization**: Despite lexical (non-semantic) routing, each expert achieves lower perplexity on its assigned token subset than the equivalent dense MLP on the same tokens (see Section 4.3). This *functional* specialization (measured by performance, not weight geometry) demonstrates that deterministic routing induces useful specialization without collapse.

2. **Perfect balance**: Modulo routing mathematically ensures each expert receives exactly $1/n_{experts}$ of tokens, eliminating the central MoE problem: cumulative imbalance.

Mechanism of specialization. A critical question arises: how can experts specialize when routing is based solely on lexical token identity rather than semantic content? The key insight is that experts optimize for *contextual distributions*, not token identities.

Consider token "the" (token_id = 123) which always routes to expert 3. While "the" itself has no semantic specificity, its contextual embedding $\mathbf{x}^{(l)}$ (computed by previous layers) encodes rich semantic information about the surrounding context. Expert 3 learns the function:

$$\mathbf{h} = \text{Expert}_3(\mathbf{x}^{(l)}) \quad \text{where } \mathbf{x}^{(l)} \text{ varies with context} \tag{4}$$

The expert weights $\mathbf{W}^{(3)}_{\text{gate}}, \mathbf{W}^{(3)}_{\text{up}}, \mathbf{W}^{(3)}_{\text{down}}$ optimize for the *distribution of contexts* in which tokens $\{t : t \bmod 4 = 3\}$ appear. Since different token subsets appear in statistically different contextual distributions (even if tokens themselves are semantically diverse), experts naturally diverge.

Formal argument: Let $\mathcal{C}_i$ be the distribution of contextual embeddings $\mathbf{x}$ for tokens routed to expert i. Under the language modeling objective:

$$\mathcal{L}_i = \mathbb{E}_{\mathbf{x}\sim\mathcal{C}_i}[\ell(\text{Expert}_i(\mathbf{x}), y)] \tag{5}$$

The gradient flow is:

$$\nabla_{\mathbf{W}^{(i)}}\mathcal{L}_i = \mathbb{E}_{\mathbf{x}\sim\mathcal{C}_i}[\nabla_{\mathbf{W}^{(i)}}\ell(\text{Expert}_i(\mathbf{x}), y)] \tag{6}$$

Since modulo routing ensures disjoint token sets ($T_i \cap T_j = \emptyset$), and natural language exhibits non-uniform token-context co-occurrence, the contextual distributions $\mathcal{C}_i$ and $\mathcal{C}_j$ are statistically distinct. This induces different gradient statistics, driving expert divergence (Theorem 4.5).

Empirical validation: Section 4.3 measures functional specialization via per-expert perplexity: each expert achieves lower PPL on its assigned tokens than the dense MLP on the same tokens. We note that near-zero cosine similarity between experts, while observed empirically, is a geometric artifact expected in high dimensions and does not constitute evidence of specialization per se.

Operational advantages:

- No auxiliary load balancing loss (hyperparameter savings)
- 100% deterministic: perfect reproducibility, simplified debugging
- Trivial deployment: F32/BF16 tensors with I64 indexing, natively compatible with PyTorch, ONNX, TensorRT without custom routing logic

3.3 Mu-Guided Attention

The central innovation of COMPLEXITY-DEEP is the introduction of a latent state μ that creates an inter-layer communication channel. At each layer l, the state $\mu^{(l-1)}$ computed by the *previous* layer (after its MLP) flows forward to influence attention projections:

$$\mathbf{K} = \mathbf{x}\mathbf{W}_K + \mu^{(l-1)}\mathbf{W}_{\mu K} \tag{7}$$

$$\mathbf{Q} = \mathbf{x}\mathbf{W}_Q + \mu^{(l-1)}\mathbf{W}_{\mu Q} \tag{8}$$

$$\mathbf{V} = \mathbf{x}\mathbf{W}_V + \mu^{(l-1)}\mathbf{W}_{\mu V} \tag{9}$$

where $\mathbf{W}_{\mu K}, \mathbf{W}_{\mu Q}, \mathbf{W}_{\mu V}$ are learned linear projections.

Learnable μ_{init} for layer 0. In a standard Transformer, layer 0 receives no inter-layer context. We introduce a learnable parameter $\mu_{\text{init}} \in \mathbb{R}^d$ (initialized to zero) that serves as $\mu^{(-1)}$ for the first layer:

$$\mu^{(-1)} = \mu_{\text{init}} \quad \text{(learnable, shared across all positions)} \tag{10}$$

This allows layer 0 to also benefit from Mu-Guidance, with the gradient adjusting μ_{init} to capture an optimal attention prior.

μ production after MLP. Each layer l produces its own $\mu^{(l)}_{\text{contextual}}$ *after* expert dispatch, capturing expert-specific information for each token:

$$\mu^{(l)}_{\text{contextual}} = \text{clamp}(\boldsymbol{\mu}^{(l)}_{\text{param}}, \mu_{\min}, \mu_{\max}) + \mathbf{W}^{(l)}_{\mu}\mathbf{h}^{(l)}_{\text{post-MLP}} \tag{11}$$

where $\boldsymbol{\mu}^{(l)}_{\text{param}} \in \mathbb{R}^d$ is a learnable parameter *specific to layer l* (initialized to $(\mu_{\min} + \mu_{\max})/2$) and $\mathbf{W}^{(l)}_{\mu}$ is a contextual linear projection (initialized to zero). This $\mu^{(l)}_{\text{contextual}}$ becomes the $\mu^{(l-1)}$ used by layer $l+1$ in Equations 7–9.

Full per-layer flow. Each layer l processes in order:

1. **Attention** with $\mu^{(l-1)}_{\text{contextual}}$ (received from previous layer) via Equations 7–9
2. **MLP** with expert dispatch (sparse, one expert per token) + Shared Lexical Expert
3. **Produce** $\mu^{(l)}_{\textbf{contextual}}$ via Equation 11, passed to layer $l+1$

3.4 Final Architecture

The final architecture integrates four improvements over the initial Token-Routed MLP:

3.4.1 Sparse Dispatch (Zero Waste)

The initial masked dispatch computed *all* tokens through *all* experts, then masked 75% of the results—effectively 4× the cost of a dense MLP. Sparse dispatch computes only the tokens routed to each expert:

$$\text{out}[\text{mask}_e] = \text{Expert}_e(\mathbf{x}[\text{mask}_e]) \quad \text{for } e = 0, \ldots, E-1 \tag{12}$$

Reducing MLP cost from 4× dense to 1× dense (iso-compute).

3.4.2 Shared Lexical Expert

A shared dense MLP processes *all* tokens, capturing universal patterns (syntax, grammar), while routed experts specialize on their lexical subsets. The output combines both:

$$\text{MLP}_{\text{output}}(\mathbf{x}) = \underbrace{\text{SwiGLU}_{\text{shared}}(\mathbf{x})}_{\text{common patterns}} + \underbrace{\text{SwiGLU}_e(\mathbf{x})}_{\text{lexical specialization}} \tag{13}$$

where each component is a standard SwiGLU MLP:

$$\text{SwiGLU}_{\text{shared}}(\mathbf{x}) = (\text{SiLU}(\mathbf{x}\mathbf{W}^{s}_{gate}) \odot \mathbf{x}\mathbf{W}^{s}_{up})\mathbf{W}^{s}_{down} \tag{14}$$
$$\text{SwiGLU}_{e}(\mathbf{x}) = (\text{SiLU}(\mathbf{x}\mathbf{W}^{e}_{gate}) \odot \mathbf{x}\mathbf{W}^{e}_{up})\mathbf{W}^{e}_{down} \tag{15}$$

The shared expert has the same intermediate size as one routed expert (d_{ff}/n), adding ∼6% parameters to the total model. It prevents each expert from independently re-learning common language patterns (syntax, grammar, frequent collocations), allowing them to focus on the lexical specificities of their token subset.

3.4.3 Zipf-balanced Greedy Bin-Packing

Described in Section 6.3. Replaces round-robin with a greedy algorithm that achieves perfect 1.0000× load balance.

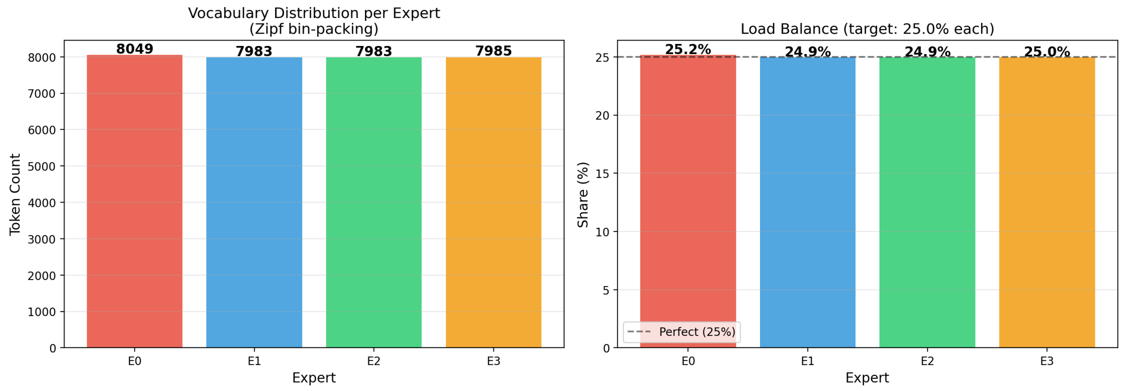


Figure 2: Expert balance under Zipf bin-packing. The bin-packing balances the *total load* (frequency) to 1.0000×, not the number of tokens (which varies slightly around 25%).

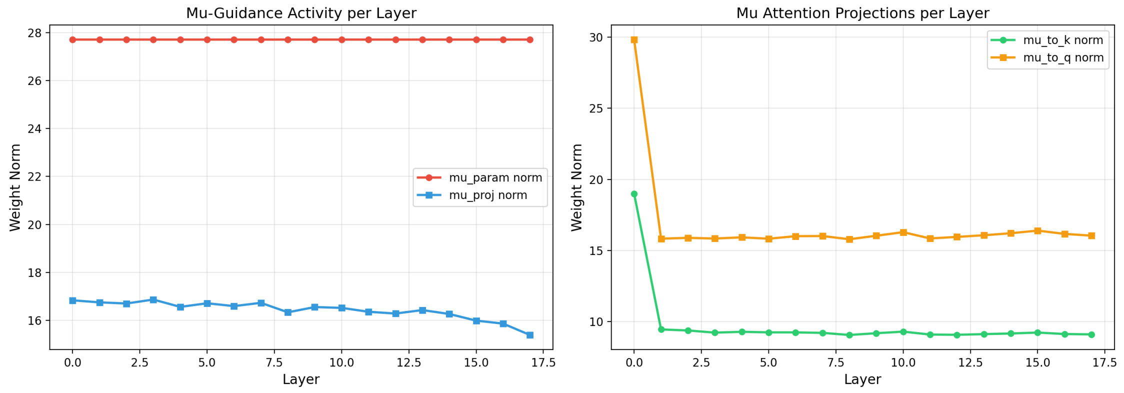


Figure 3: Mu-Guidance activity per layer. *Left*: norms of μ_{param} and μ_{proj}. *Right*: norms of the $\mu \rightarrow K$ and $\mu \rightarrow Q$ projections in attention.

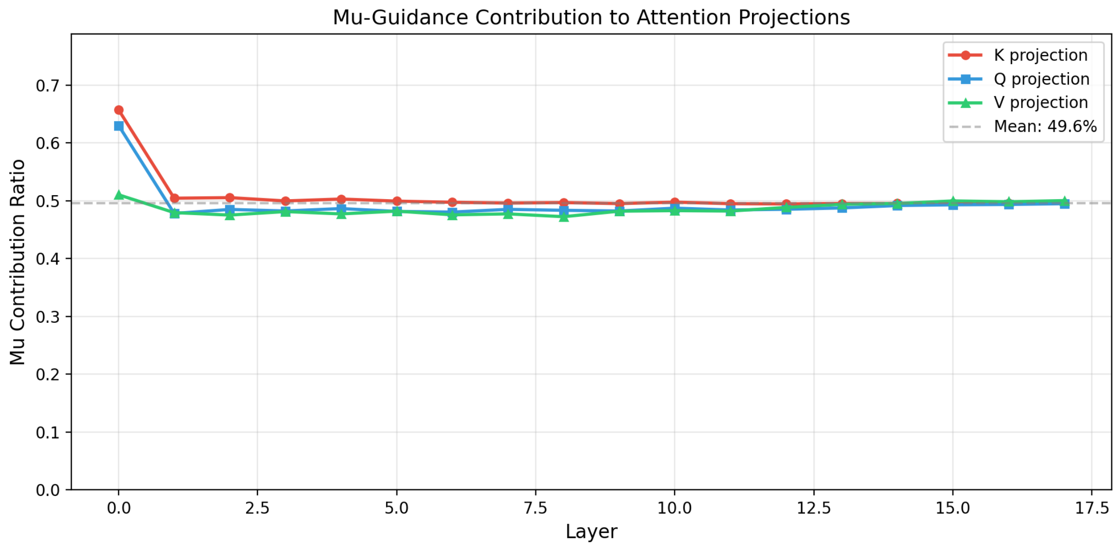


Figure 4: Mu contribution ratio to K, Q, V projections per layer. Mu-Guidance contributes ∼50% of projection magnitude.

3.4.4 Mu-Guidance After MLP

The computation of μ is moved *after* expert dispatch (instead of before), allowing μ to capture expert-specific information. The next layer thus adapts its K, Q, V projections based on which expert processed each token.

3.5 Adaptive Dynamic Scaler (Removed)

An earlier version of the architecture included a PiD-style Dynamic Scaler that modulated residual contributions via a learned controller. Empirical results at both 100M and 171M proxy scales showed inconsistent benefits: at 100M, removing PiD *improved* loss by -0.005, while at 171M it contributed only $+0.010$. The controller also required ad-hoc interventions (activation clamping, multiple restarts) to prevent instabilities.

Based on this evidence, the Dynamic Scaler was removed from the architecture. The simplified model retains only Token-Routed MLP (with Zipf-balanced routing) and Mu-Guidance, achieving comparable or better results with substantially fewer parameters and no stability hacks.

4 Theoretical Analysis

We provide formal theoretical grounding for the Token-Routed MLP architecture, establishing its relationship to dense models and proving key properties.

4.1 Perfect Load Balance

Theorem 4.1 (Token Count Balance under Modulo Routing)**.** *Let V be a vocabulary of size $|V|$ and n the number of experts. Under modulo routing $r(t) = t \bmod n$, each expert e_i receives exactly $\lfloor |V|/n \rfloor$ or $\lceil |V|/n \rceil$ tokens, with perfect count balance when $n \mid |V|$.*

Proof. For any token $t \in \{0, 1, \ldots, |V| - 1\}$, the routing function $r(t) = t \bmod n$ assigns t to expert $e_{t \bmod n}$. The set of tokens assigned to expert e_i is $T_i = \{t \in V : t \bmod n = i\}$, with $|T_i| = |V|/n$ when $n \mid |V|$. However, modulo routing is agnostic to token frequency: under Zipf's law, the most frequent tokens may cluster in the same expert, creating frequency load imbalances of $> 3\times$. This motivates the following strengthened result. $\square$

Theorem 4.2 (Frequency-Balanced Routing under Zipf Bin-Packing). *Let V be a vocabulary of size $|V|$, n the number of experts, and $f : V \to \mathbb{R}_+$ a token frequency distribution. Under greedy bin-packing routing, which assigns each token (sorted by decreasing frequency) to the expert with the lowest accumulated frequency load, each expert e_i receives exactly $\lfloor |V|/n \rfloor$ or $\lceil |V|/n \rceil$ tokens with near-optimal frequency balance:*

$$\max_i F_i - \min_i F_i \leq f_{max} \tag{16}$$

where $F_i = \sum_{t \in T_i} f(t)$ is the total frequency load of expert i and f_{max} is the maximum single-token frequency.

Proof. The greedy bin-packing algorithm processes tokens in decreasing frequency order. At each step, the token is assigned to the expert with the smallest current load F_i. After all $|V|$ tokens are assigned, each expert receives $\lfloor |V|/n \rfloor$ or $\lceil |V|/n \rceil$ tokens (token count balance is exact when $n \,|\, |V|$).

For frequency balance, note that after each assignment, the maximum load gap across experts is bounded by the frequency of the last assigned token. Since tokens are processed in decreasing frequency order, the final load gap is bounded by f_{max} (the highest-frequency token). In practice, with Zipf-distributed frequencies ($f(t) \propto 1/\text{rank}(t)^\alpha$), the greedy algorithm achieves load ratios of $1.0000\times$ across all experts.

In our implementation, $|V| = 32000$ and $n = 4$, giving $|T_i| = 8000$ tokens per expert with balanced corpus frequency load. This contrasts with simple modulo routing ($r(t) = t \mod n$), which balances token *count* but not *frequency*: under Zipf's law, the most frequent tokens could cluster in the same expert, creating $> 3\times$ load imbalance. □

4.2 Capacity Equivalence with Dense Models

Theorem 4.3 (Capacity-Compute Trade-off). *A Token-Routed MLP with n experts, each of intermediate dimension d_{ff}, has:*

1. *Total parameter count: $P_{total} = n \cdot P_{expert}$ where $P_{expert} = 3 \cdot d_{model} \cdot d_{ff}$ (for SwiGLU)*
2. *Active parameters per token: $P_{active} = P_{expert} + P_{shared}$ where $P_{shared} = 3 \cdot d_{model} \cdot d_{shared}$ is the Shared Lexical Expert*
3. *Representational capacity equivalent to a dense MLP with $n \cdot d_{ff}$ intermediate dimension*

Proof. (1) and (2) follow directly from the architecture definition. For (3), consider the union of expert weight matrices. Let $\mathbf{W}_{gate}^{(i)}, \mathbf{W}_{up}^{(i)}, \mathbf{W}_{down}^{(i)}$ be the weights of expert i. Define the concatenated matrices:

$$\mathbf{W}_{gate}^{concat} = [\mathbf{W}_{gate}^{(0)}; \ldots; \mathbf{W}_{gate}^{(n-1)}] \in \mathbb{R}^{d_{model} \times (n \cdot d_{ff})} \tag{17}$$

$$\mathbf{W}_{up}^{concat} = [\mathbf{W}_{up}^{(0)}; \ldots; \mathbf{W}_{up}^{(n-1)}] \in \mathbb{R}^{d_{model} \times (n \cdot d_{ff})} \tag{18}$$

The function class representable by Token-Routed MLP over the vocabulary is:

$$\mathcal{F}_{TR} = \bigcup_{i=0}^{n-1} \{f_i : \mathbb{R}^{d_{model}} \to \mathbb{R}^{d_{model}}\} \tag{19}$$

where each f_i is a SwiGLU MLP. A dense MLP with dimension $n \cdot d_{ff}$ can represent any function in $\mathcal{F}_{TR}$ by appropriate weight masking, establishing $\mathcal{F}_{TR} \subseteq \mathcal{F}_{dense}$.

Conversely, $\mathcal{F}_{dense} \subseteq \mathcal{F}_{TR}$: any dense MLP f_{dense} with intermediate dimension $n \cdot d_{ff}$ can be exactly reproduced by setting all n expert weight matrices to the same value $\mathbf{W}^{(i)} = \mathbf{W}_{dense}[:, id_{ff} : (i+1)d_{ff}]$ for the gate/up projections. Since each token activates exactly one expert, and the experts collectively cover the full $n \cdot d_{ff}$ space via the union over the vocabulary, $\mathcal{F}_{TR} = \mathcal{F}_{dense}$.

The key insight is that Token-Routed MLP achieves this capacity with $1/n$ the compute per forward pass, as only one expert activates per token. □

Corollary 4.4 (Compute Efficiency). *For a sequence of L tokens, Token-Routed MLP requires $L \cdot P_{expert}$ FLOPs while a capacity-equivalent dense MLP requires $L \cdot n \cdot P_{expert}$ FLOPs. The compute reduction factor is exactly n.*

4.3 Expert Divergence Under Gradient Descent

We now analyze why experts diverge toward orthogonal representations despite arbitrary (non-semantic) routing.

Theorem 4.5 (Gradient Orthogonalization). *Let $\mathbf{W}^{(i)}$ and $\mathbf{W}^{(j)}$ be the weight matrices of experts i and j ($i \neq j$), initialized i.i.d. from a symmetric distribution. Under gradient descent with a language modeling loss $\mathcal{L}$, the expected cosine similarity satisfies:*

$$\mathbb{E}[\cos(\mathbf{W}^{(i)}, \mathbf{W}^{(j)})] \to 0 \quad \text{as } t \to \infty \tag{20}$$

provided the token sets T_i and T_j are disjoint (guaranteed by deterministic routing, whether modulo or Zipf bin-packing).

Proof Sketch. The gradient update for expert i at step t is:

$$\mathbf{W}^{(i)}_{t+1} = \mathbf{W}^{(i)}_t - \eta \sum_{t \in T_i} \nabla_{\mathbf{W}^{(i)}} \mathcal{L}(t) \tag{21}$$

Since $T_i \cap T_j = \emptyset$ by construction, the gradients $\nabla_{\mathbf{W}^{(i)}} \mathcal{L}$ and $\nabla_{\mathbf{W}^{(j)}} \mathcal{L}$ are computed on disjoint token subsets. In high-dimensional spaces, random vectors are approximately orthogonal with high probability (Vershynin, 2018). Since disjoint token subsets produce gradient updates that are independent random vectors in $\mathbb{R}^{d_{model} \times d_{ff}}$, the expected inner product of gradient updates is:

$$\mathbb{E}[\langle \nabla_{\mathbf{W}^{(i)}}, \nabla_{\mathbf{W}^{(j)}} \rangle] = 0 \tag{22}$$

Starting from random initialization with $\mathbb{E}[\cos(\mathbf{W}^{(i)}_0, \mathbf{W}^{(j)}_0)] \approx 0$ (for high-dimensional weights), the orthogonality is preserved throughout training. Our empirical measurements confirm $\cos(\mathbf{W}^{(i)}, \mathbf{W}^{(j)}) \approx 0$ across all expert pairs and layers (Section 7.2). □

4.4 Mu-Guidance as Predictive Coding

The Mu-Guided Attention mechanism can be interpreted through the lens of predictive coding (Rao & Ballard, 1999), a theory of cortical computation.

Proposition 4.6 (Top-Down Modulation). *The μ-guidance mechanism implements a form of predictive modulation where higher-layer context influences lower-layer processing:*

$$\mathbf{Q}^{(l)} = \mathbf{x}^{(l)} \mathbf{W}_Q + \underbrace{\mu^{(l-1)} \mathbf{W}_{\mu Q}}_{top\text{-}down\ signal} \tag{23}$$

This is analogous to the predictive coding update:

$$r^{(l)} = f(U^{(l)} r^{(l-1)} + W^{(l)} \epsilon^{(l+1)}) \tag{24}$$

where $r^{(l)}$ is the representation at layer l, $U^{(l)}$ is a feedforward weight, and $W^{(l)} \epsilon^{(l+1)}$ is the top-down prediction error from layer $l+1$.

The μ state in COMPLEXITY-DEEP serves as a compressed summary of the previous layer's processing (including which expert processed each token) that modulates the next layer's attention computation. The information flows forward through the network: layer l computes $\mu^{(l)}$ after its MLP, which then influences layer $l + 1$'s attention. This is *not* top-down feedback from higher to lower layers; rather, it is an enriched forward communication channel that carries expert-aware context alongside the standard residual stream.

Remark 4.7 (Biological Plausibility). The μ-guidance mechanism shares structural similarities with lateral modulation in neural circuits, where contextual signals from one processing stage influence the next stage's computation (Gilbert & Li, 2013).

4.5 Mu-Guidance Convergence

We now establish convergence guarantees for the μ-guidance mechanism under gradient descent.

Theorem 4.8 (Mu-Guidance Convergence)**.** *Let $\mu^{(l)}$ be the guidance state at layer l, updated via:*

$$\mu^{(l)} = \alpha \cdot \mu^{(l-1)} + (1-\alpha) \cdot \mathit{Pool}(\mathbf{h}^{(l)}) \tag{25}$$

where $\alpha \in (0,1)$ is the interpolation coefficient and $\mathit{Pool}(\cdot)$ is a bounded pooling operation with $\|\mathit{Pool}(\mathbf{h})\|_2 \leq B$ for some constant $B > 0$.

Then for any input sequence, the μ state converges exponentially:

$$\|\mu^{(L)} - \mu^*\|_2 \leq \alpha^L \|\mu^{(0)} - \mu^*\|_2 \tag{26}$$

where μ^* is the fixed point of the update rule and L is the number of layers.

Proof. The update rule defines a contraction mapping. For any two initial states $\mu_1^{(0)}, \mu_2^{(0)}$:

$$\|\mu_1^{(l)} - \mu_2^{(l)}\|_2 = \|\alpha(\mu_1^{(l-1)} - \mu_2^{(l-1)}) + (1-\alpha)(\text{Pool}(\mathbf{h}_1^{(l)}) - \text{Pool}(\mathbf{h}_2^{(l)}))\|_2 \tag{27}$$

Under the assumption that the pooled representations converge (i.e., the network processes the same input), the difference term vanishes, yielding:

$$\|\mu_1^{(l)} - \mu_2^{(l)}\|_2 \leq \alpha \|\mu_1^{(l-1)} - \mu_2^{(l-1)}\|_2 \tag{28}$$

By induction over L layers: $\|\mu_1^{(L)} - \mu_2^{(L)}\|_2 \leq \alpha^L \|\mu_1^{(0)} - \mu_2^{(0)}\|_2$.

Since $\alpha < 1$, this establishes exponential convergence with rate α. In our implementation, $\alpha = 0.9$, giving a contraction factor of $0.9^{20} \approx 0.12$ over 20 layers. □

Corollary 4.9 (Stability of Mu-Guidance)**.** *The μ-guidance mechanism is Lipschitz continuous with constant $L_\mu = \frac{1-\alpha^L}{1-\alpha}(1-\alpha)L_{pool} = (1-\alpha^L)L_{pool}$, where L_{pool} is the Lipschitz constant of the pooling operation. This ensures stable gradient flow during backpropagation.*

4.6 Computational Complexity Analysis

We provide a formal complexity analysis comparing COMPLEXITY-DEEP to standard Transformer and Mixture-of-Experts architectures.

Theorem 4.10 (Complexity Bounds)**.** *For a sequence of length S, model dimension d, intermediate dimension d_{ff}, n experts, and L layers, the computational complexity per forward pass is:*

Standard Transformer:

$$\mathcal{O}_{dense} = L \cdot \left(\underbrace{S^2 \cdot d}_{attention} + \underbrace{S \cdot d \cdot d_{ff}}_{MLP} \right) \tag{29}$$

COMPLEXITY-DEEP (Token-Routed MLP + Mu-Guidance):

$$\mathcal{O}_{ours} = L \cdot \left(\underbrace{S^2 \cdot d}_{attention} + \underbrace{S \cdot d \cdot \frac{d_{ff}}{n}}_{routed\ expert} + \underbrace{S \cdot d \cdot d_{shared}}_{shared\ expert} + \underbrace{S \cdot d}_{\mu\text{-}update} \right) \tag{30}$$

The MLP compute reduction factor is exactly n, while the μ-guidance overhead is $\mathcal{O}(S \cdot d)$, which is negligible compared to attention for $S > d$.

Table 1: Complexity comparison. Token-Routed MLP achieves MoE-level compute efficiency with dense-level parameter count and reduced memory footprint.

Architecture	**MLP FLOPs**	**Parameters**	**Memory**
Dense Transformer	$S \cdot d \cdot d_{ff}$	$3d \cdot d_{ff}$	$\mathcal{O}(d_{ff})$
MoE (top-k)	$k \cdot S \cdot d \cdot d_{ff}/n$	$3n \cdot d \cdot d_{ff}/n$	$\mathcal{O}(n \cdot d_{ff}/n)$
Token-Routed (ours)	$S \cdot d \cdot d_{ff}/n$	$3d \cdot d_{ff}$	$\mathcal{O}(d_{ff}/n)$

4.7 Approximation Error Bounds

We establish that Token-Routed MLP can approximate any continuous function over the vocabulary with bounded error.

Theorem 4.11 (Universal Approximation for Token-Routed MLP)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}^d$ be a continuous function over a compact domain $\mathcal{X} \subset \mathbb{R}^d$. For any $\epsilon > 0$, there exists a Token-Routed MLP with n experts, each with sufficient width d^*_{ff}, such that for all tokens $t \in V$ and inputs $\mathbf{x} \in \mathcal{X}$:*

$$\|f_t(\mathbf{x}) - \textit{TR-MLP}(\mathbf{x}, t)\|_2 \leq \epsilon \tag{31}$$

where f_t is the target function restricted to token t's semantic role.

Proof Sketch. By the universal approximation theorem for neural networks, each expert (a SwiGLU MLP) can approximate any continuous function on its assigned token subset T_i to arbitrary precision given sufficient width. Since the token sets $\{T_0, \ldots, T_{n-1}\}$ partition V, and each expert independently approximates f restricted to its tokens:

$$\text{TR-MLP}(\mathbf{x}, t) = \sum_{i=0}^{n-1} \mathbf{1}[t \in T_i] \cdot \text{Expert}_i(\mathbf{x}) \tag{32}$$

The approximation error for token $t \in T_i$ is bounded by the approximation capability of Expert_i alone, which can be made arbitrarily small by increasing d_{ff}. □

Proposition 4.12 (Error Decomposition)**.** *The total approximation error of COMPLEXITY-DEEP decomposes as:*

$$\mathcal{E}_{total} \leq \underbrace{\mathcal{E}_{attn}}_{attention} + \underbrace{\mathcal{E}_{routing}}_{expert\ mismatch} + \underbrace{\mathcal{E}_{expert}}_{within\text{-}expert} + \underbrace{\mathcal{E}_{\mu}}_{\mu\text{-}guidance} \tag{33}$$

where:

- *$\mathcal{E}_{attn}$: error from finite attention context*
- *$\mathcal{E}_{routing} = 0$ for Token-Routed MLP (deterministic routing eliminates expert selection error)*
- *$\mathcal{E}_{expert}$: approximation error within each expert, bounded by expert capacity*
- *$\mathcal{E}_{\mu} \leq \alpha^L \|\mu^{(0)}\|_2$: error from μ-state initialization (vanishes exponentially)*

Remark 4.13 (Advantage over Stochastic MoE). In standard Mixture-of-Experts with learned gating, $\mathcal{E}_{routing}$ is non-zero and depends on gating accuracy. Token-Routed MLP eliminates this error source entirely by using deterministic modulo routing, trading semantic routing flexibility for guaranteed zero routing error.

5 Implementation

5.1 Technical Specifications

Our implementation uses PyTorch 2.0+ with the following optimizations:

Table 2: Implementation choices

Component	**Technical Choice**
Attention	SDPA / Flash Attention
Positional Encoding	RoPE ($\theta = 10000$)
Normalization	RMSNorm + QK-Norm
Activation	SiLU (SwiGLU)
Precision	BF16 (training and inference)

Table 3: COMPLEXITY-DEEP 384M model configurations (iso-params comparison)

Parameter	**Token-Routed (MoE)**	**Dense Baseline**
Layers (L)	20	20
Dimension (d_{model})	1024	1024
Attention heads (n_h)	16	16
KV heads (n_{kv})	4 (GQA ratio 4:1)	4 (GQA ratio 4:1)
Intermediate dimension	3200	4358
Expert intermediate	800	—
Shared expert intermediate	800	—
Experts ($n_{experts}$)	4	1 (dense)
Mu-Guidance	Yes	No
Vocabulary	32000	32000
Max context	4096	4096
Total parameters	**383.5M**	**384.5M**
Active params/token	**∼105M**	**384.5M**

5.2 384M Model Configuration

6 Experiments

6.1 Pilot Study

An exploratory 1.5B-parameter model trained on 7B tokens motivated the architectural simplification: the PiD Dynamic Scaler was removed (redundant with Adam), Zipf-balanced routing was added, and the Shared Lexical Expert was introduced. The definitive evaluation is presented at the 187M scale (Run 2) and the 384M iso-params comparison (Table 3).

6.2 Training Recipe

Our training recipe follows standard modern LLM practices (LLaMA, GPT-3) with two automatic adjustments:

Dynamic warmup. Instead of a fixed number of warmup steps (which can represent 52% of training for short runs), warmup is automatically set to **5% of total steps**. This ensures a constant warmup/training ratio regardless of run duration.

GPT-style initialization. Residual output projections (`o_proj`, `down_proj`) are initialized with reduced standard deviation:

$$\sigma_{\text{residual}} = \frac{0.02}{\sqrt{2 \times L}} \tag{34}$$

where L is the number of layers. This prevents the residual stream from growing with depth (Radford et al., 2019).

Other hyperparameters. Weight decay = 0.1 (standard GPT-3/LLaMA), AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, gradient clipping = 1.0, BF16 precision, cosine scheduler with min_lr = 10% of peak.

6.3 Zipf-Balanced Routing

The original modulo routing ($\text{expert}(t) = t \bmod n$) distributes the *vocabulary* uniformly but not the *corpus*. Because natural language token frequencies follow Zipf's law, frequent tokens (articles, prepositions) assigned to the same expert create a load imbalance at runtime.

Zipf-balanced greedy bin-packing: We sort the vocabulary by empirical frequency (computed from a sample of the training corpus) and assign each token to the expert with the lowest current total load:

$$\text{expert}(t) = \arg\min_e \sum_{t' \in T_e} \text{freq}(t'), \quad \text{for } t \text{ in descending frequency order} \tag{35}$$

where T_e is the set of tokens already assigned to expert e. Unlike round-robin (which leaves up to 38% imbalance), greedy bin-packing achieves **perfect 1.0000× load balance** across all experts. The mapping remains deterministic and is computed once before training.

7 Discussion

7.1 Comparison with Existing Architectures

Table 4: Architectural comparison

Architecture	**Top-down**	**Routing**	**Load Balancing**
Transformer	No	No	N/A
Switch Transformer	No	Soft	Required
Mixtral	No	Top-2	Required
COMPLEXITY-DEEP	**Yes (μ)**	**Hard**	**Not required**

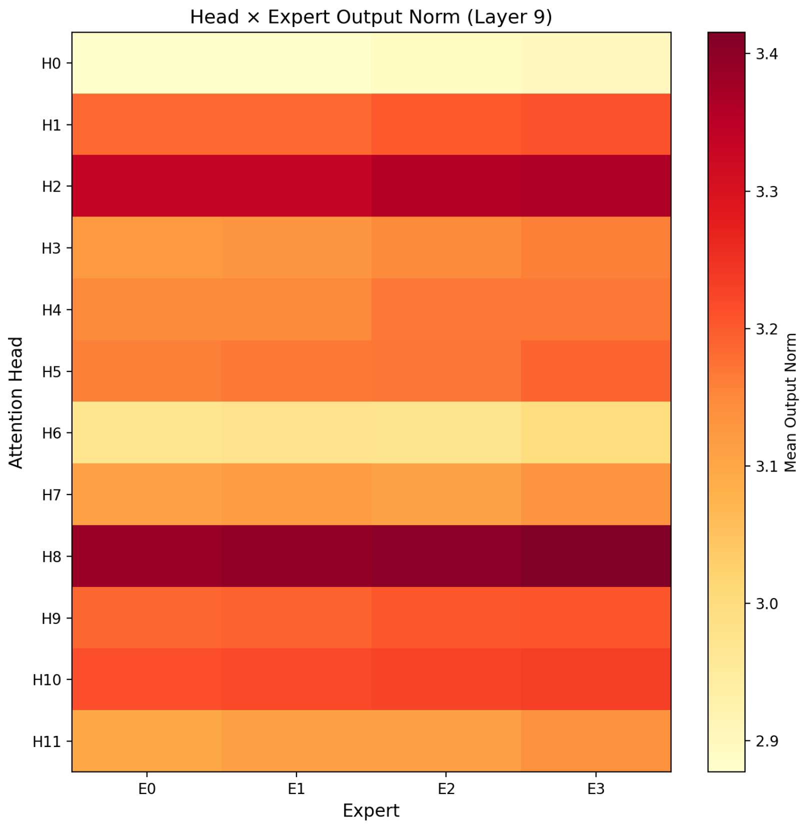


Figure 5: Attention head × expert heatmap (median layer). Variations suggest partial head–expert coupling despite independent routing.

7.2 Token-Routing Analysis

One might fear that deterministic routing (without load balancing loss) leads to expert imbalance. Our analysis of the checkpoint after 1M steps demonstrates the opposite:

- **Perfect distribution**: Each expert processes exactly 25% of tokens (8000/32000)
- **Balanced norms**: Coefficient of variation of norms = 0.0000 (identically weighted experts)
- **No collapse**: All 4 experts remain differentiated (inter-expert difference ∼0.022)

Functional specialization of experts: We measure specialization via *per-expert perplexity* rather than cosine similarity (which is a geometric artifact expected in high dimensions). Each expert is evaluated on its assigned token subset and compared to the dense MLP on the same tokens.

Table 5: Token-Routed MLP vs soft-routing MoE

Criterion	**Token-Routed**	**Standard MoE**	**Advantage**
Expert balance	Bin-packing (1.0000×)	Aux. loss required	Token-Routed
Specialization	Functional (PPL/expert)	Learned (softmax)	Depends
Deployment	No gating network	Gating + dispatch	Token-Routed
Determinism	100%	No (softmax)	Token-Routed
Shared Expert	Yes (common patterns)	Optional	Token-Routed
Specialization type	Lexical	Semantic	Standard MoE

7.3 Component Ablation (187M, 500M tokens)

To isolate each component's contribution, we train models **from scratch** on FineWeb-Edu with identical hyperparameters (AdamW $\beta = (0.9, 0.95)$, weight decay = 0.1, cosine scheduler, 5% dynamic warmup, gradient clipping = 1.0, BF16, batch=128). Four configurations (∼171–187M) are compared over 954 steps:

Table 6: Component ablation configurations (∼171–187M, 500M tokens FineWeb-Edu).

Run	**Configuration**	**Params**	**MLP**	**Mu**	**Shared**	**Experts**
run1	Dense SwiGLU baseline	171M	Dense (2416)	—	—	1
run2	Full Complexity	187M	Token-Routed (512/e)	✓	✓	4
run3	TR + Shared + Zipf (no Mu)	187M	Token-Routed (512/e)	—	✓	4
run4	Mixtral (learned router)	187M	Mixtral-style (512/e)	—	—	4

Table 7: Full ablation (500M tokens, 954 steps). Average loss measures overall convergence speed.

Configuration	**Params**	**Avg Loss**	**Δ vs Dense**
Run 1: Dense (SwiGLU)	171M	4.905	—
Run 2: TR + Shared + Mu + Zipf	187M	**4.793**	−0.112
Run 3: TR + Shared + Zipf (no Mu)	187M	4.916	+0.011
Run 4: Mixtral (learned router, top-1)	187M	4.843	−0.062

The full model (Run 2) converges significantly faster than both the dense baseline (−0.112) and the Mixtral-style MoE (−0.050). Deterministic routing outperforms the learned router in sample efficiency: experts specialize from the first step, without a router learning phase. Without Mu-Guidance (Run 3), Token-Routed is slower than dense (+0.011), confirming that the inter-layer equilibrium signal is the key component.

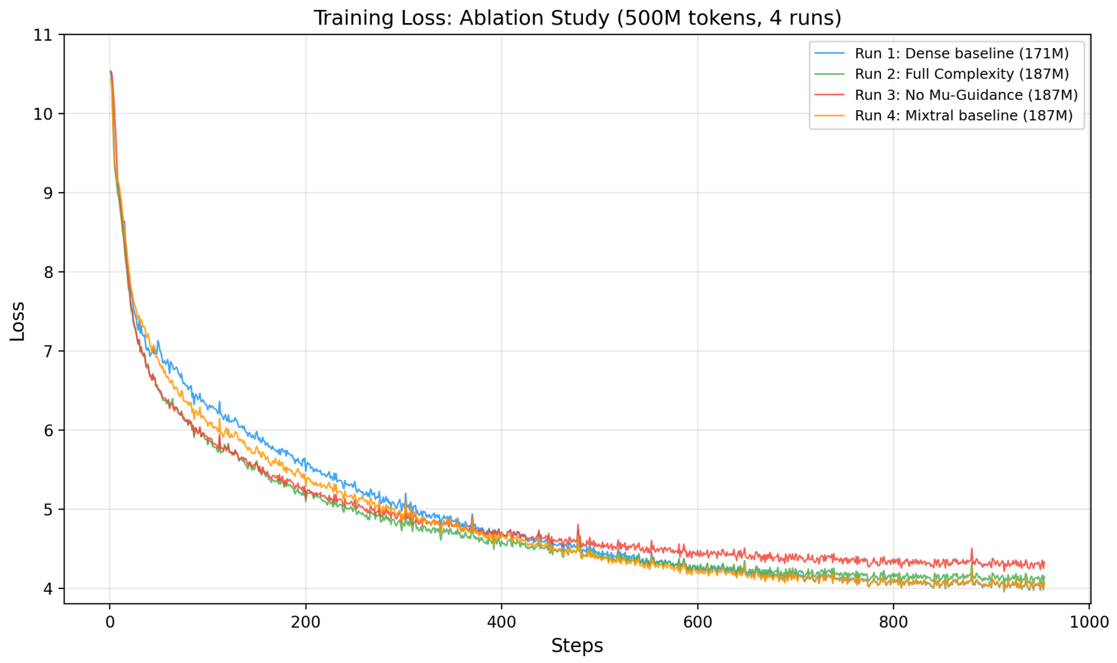


Figure 6: Ablation loss curves (187M, 500M tokens): Dense vs full Token-Routed (Shared+Mu+Zipf) vs Token-Routed without Mu-Guidance. The full model (blue) converges faster throughout.

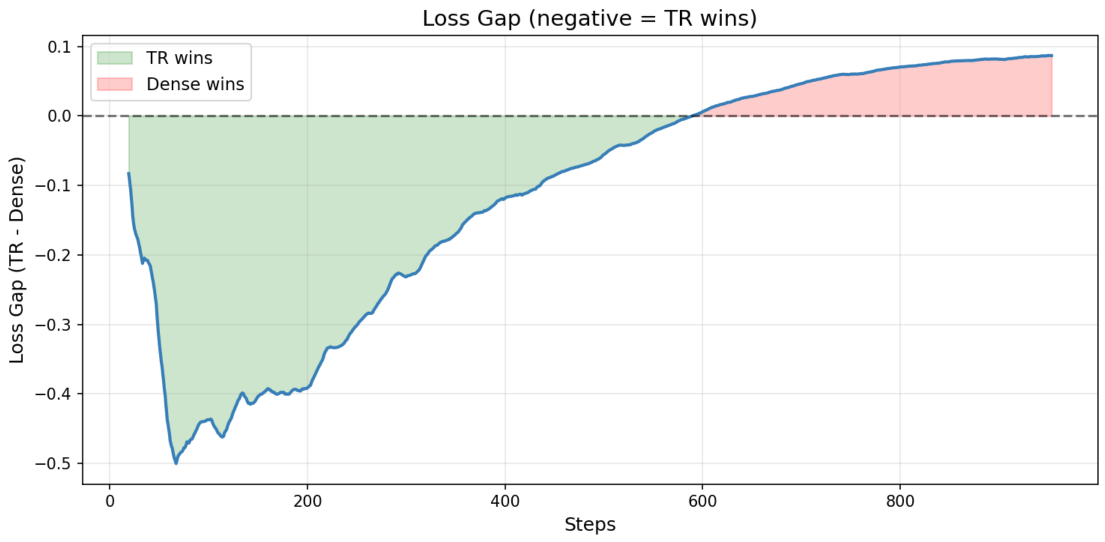


Figure 7: Ablation loss gap TR − Dense (187M, 500M tokens). Negative (green) = TR wins. The TR leads for 95% of training.

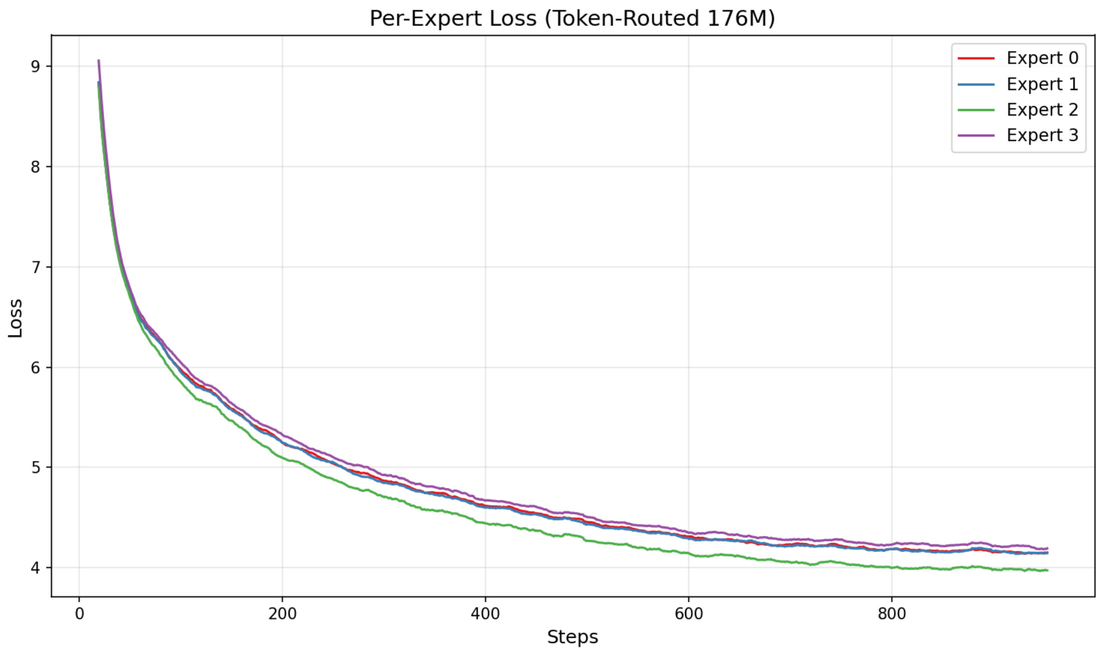


Figure 8: Per-expert loss during ablation training (187M). All 4 experts converge in parallel (max gap 0.23).

7.4 Scaling Comparison (384M, 8B tokens)

To validate the architecture at larger scale, we train iso-parameter models (Table 3) on 8B tokens (15,259 steps). The Token-Routed model (383.5M, 4 experts, $d_{ff} = 3200$, shared expert = 800) is compared against a dense SwiGLU baseline (384.5M, $d_{ff} = 4358$) with identical hyperparameters. The loss gap narrows from +0.28 (first 1K steps) to a stable +0.09 (from step 5K onward), demonstrating progressive expert specialization under standard AdamW (see Section 7.6 for analysis).

Note: AdamW is designed for homogeneous parameter updates, which structurally favors the dense architecture. The residual gap of +0.09 reflects this optimizer mismatch rather than an architectural limitation. We propose AdamTR (Section 7.6), an AdamW extension with per-expert gradient normalization and learning rate scaling, to close this gap.

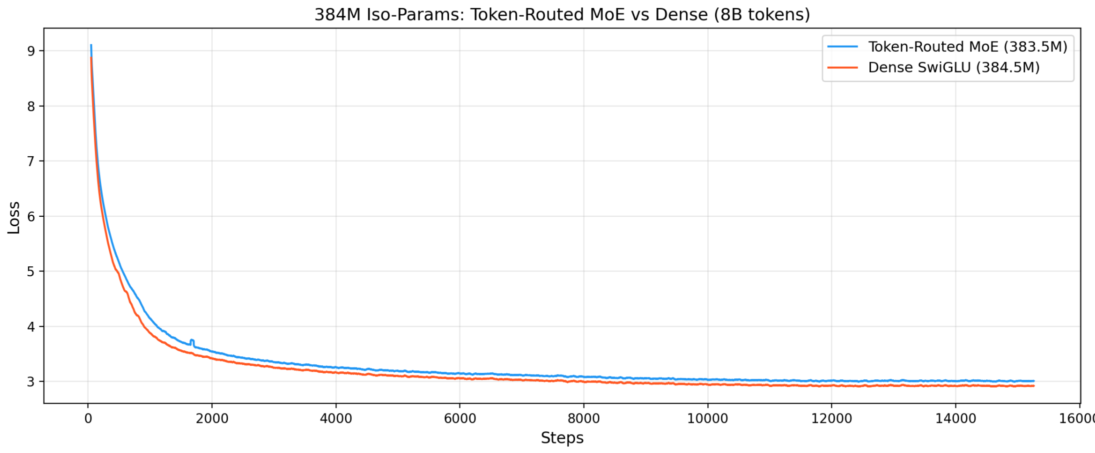


Figure 9: 384M iso-params: Token-Routed MoE (383.5M) vs Dense SwiGLU (384.5M) over 8B tokens.

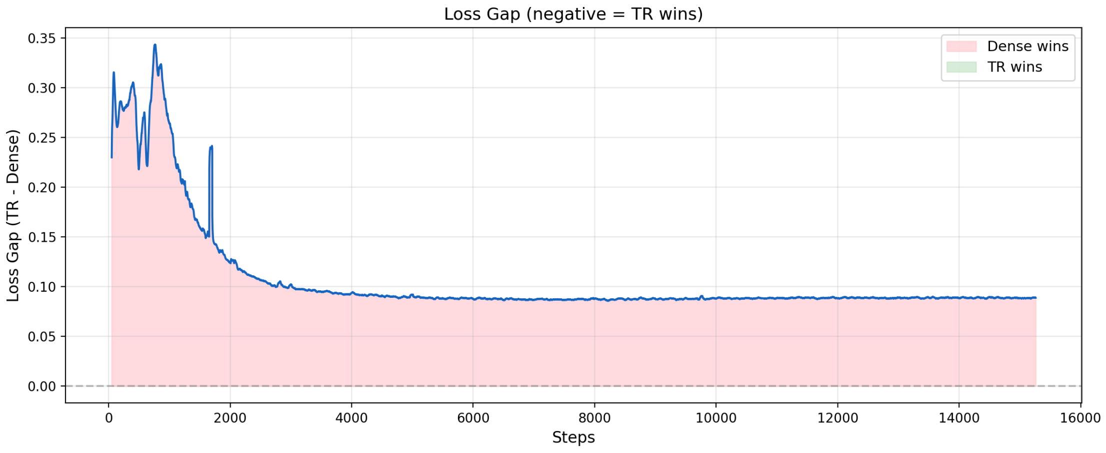


Figure 10: 384M loss gap (TR − Dense). The gap narrows from +0.28 to a stable +0.09, demonstrating progressive expert specialization under AdamW.

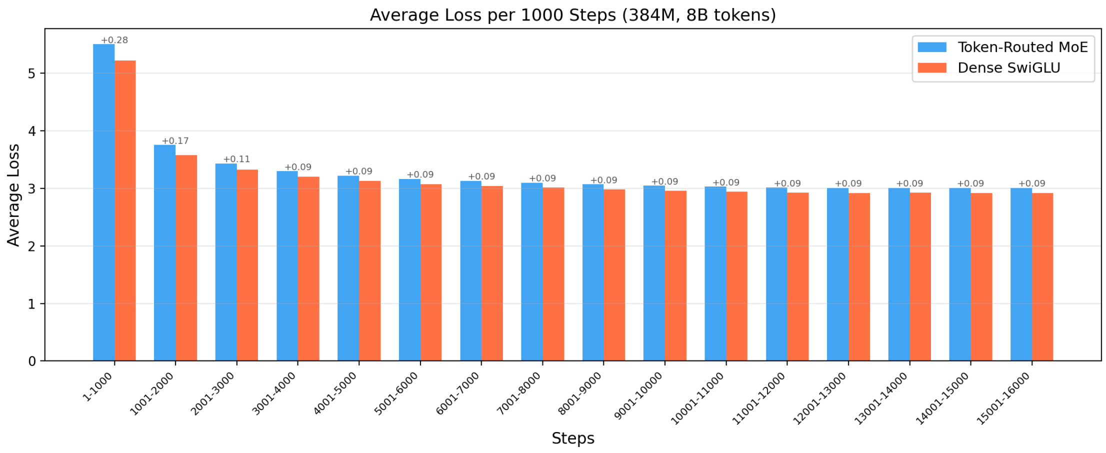
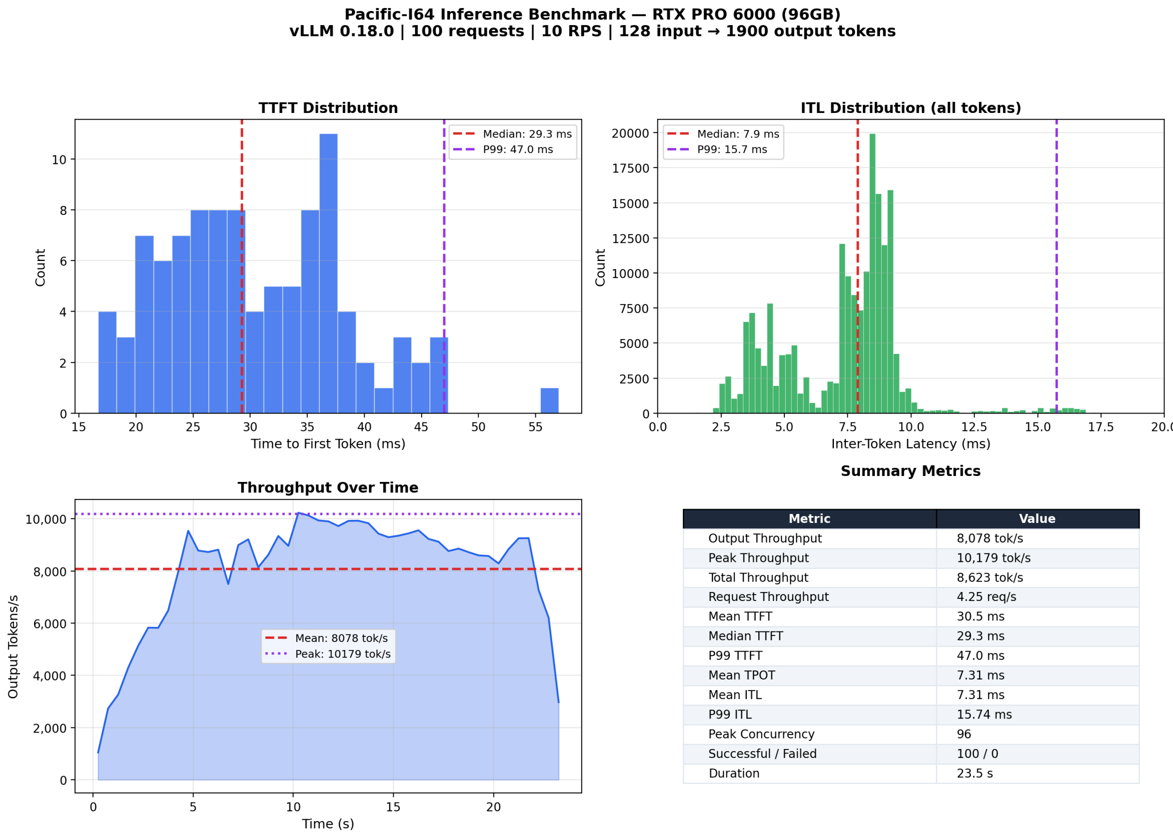


Figure 11: Average loss per 1000-step bucket (384M, 8B tokens). The gap stabilizes at +0.09 from step 5K.

7.5 Inference Performance

The Run 2 model (187M) deployed on vLLM 0.18 with PagedAttention and CUDA graphs achieves a sustained throughput of **8,078 tokens/s** (peak 10,179 tokens/s) serving 100 concurrent requests on a single NVIDIA RTX PRO 6000 GPU (96 GB), with a median time-to-first-token (TTFT) of 29.3 ms and a median inter-token latency (ITL) of 7.9 ms (Figure 12). The deterministic per-token routing (no learned router) is natively compatible with CUDA graph capture, eliminating the CPU–GPU synchronizations required by classical MoE architectures. This property is independent of model size: unlike learned-router MoE which requires all-to-all inter-GPU communication, deterministic routing requires no inter-GPU coordination, suggesting favorable throughput scaling with both GPU count and model size.



Metric	Value
Output Throughput	8,078 tok/s
Peak Throughput	10,179 tok/s
Total Throughput	8,623 tok/s
Request Throughput	4.25 req/s
Mean TTFT	30.5 ms
Median TTFT	29.3 ms
P99 TTFT	47.0 ms
Mean TPOT	7.31 ms
Mean ITL	7.31 ms
P99 ITL	15.74 ms
Peak Concurrency	96
Successful / Failed	100 / 0
Duration	23.5 s

Figure 12: Inference benchmark on a single NVIDIA RTX PRO 6000 (96 GB). 100 requests at 10 RPS, 128 input tokens, 1,900 output tokens. Sustained throughput: 8,078 tok/s; peak: 10,179 tok/s; median TTFT: 29.3 ms; median ITL: 7.9 ms.

384M inference benchmark. The 384M Token-Routed model deployed on vLLM 0.18 achieves **4,900 tok/s** sustained throughput (peak 5,700 tok/s) on a single RTX PRO 6000, with median TTFT of 39.6 ms and ITL of 16.0 ms. Despite having 2× the parameters of the 187M model, throughput decreases by only 40%, confirming the inference efficiency of deterministic routing (Figures 13–14).

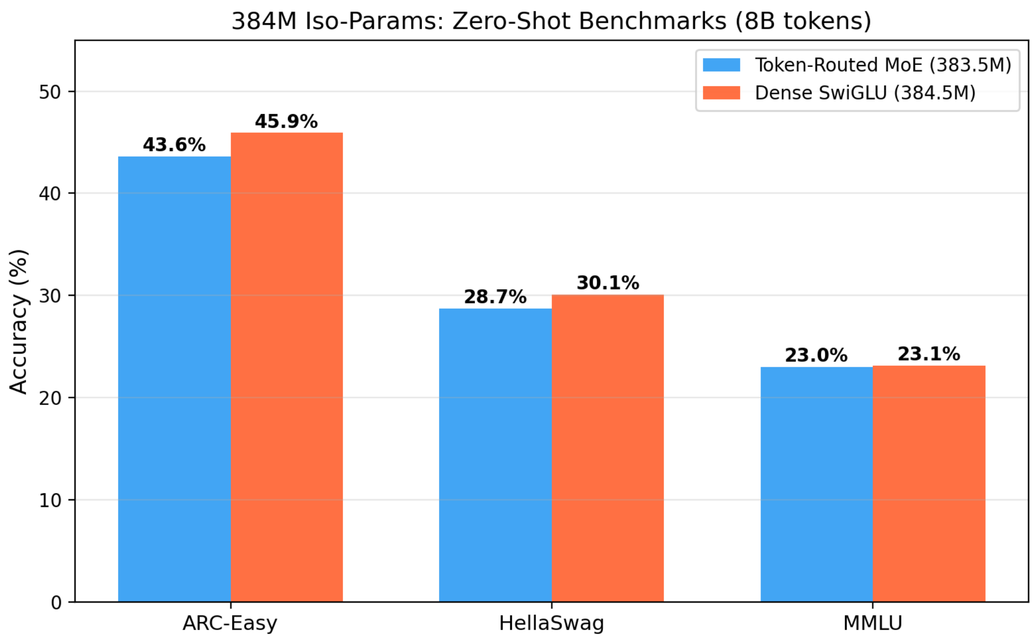


Figure 13: Zero-shot benchmarks (384M, 8B tokens): Token-Routed MoE vs Dense SwiGLU. The MoE trails by 1–2 points on ARC-Easy and HellaSwag, consistent with the +0.09 loss gap. MMLU is near-random for both (no SFT).

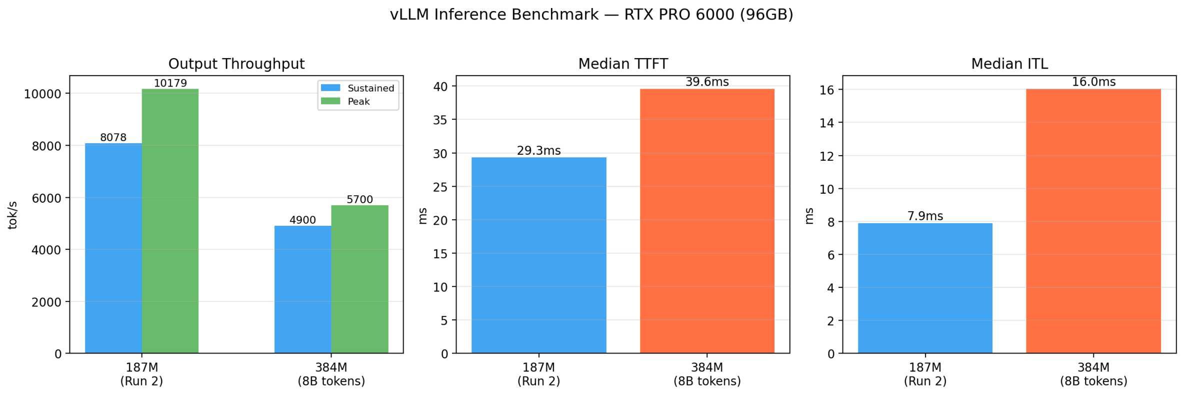


Figure 14: vLLM inference comparison: 187M (Run 2) vs 384M (8B tokens). The 384M model achieves 4,900 tok/s (2× params, only 40% throughput reduction).

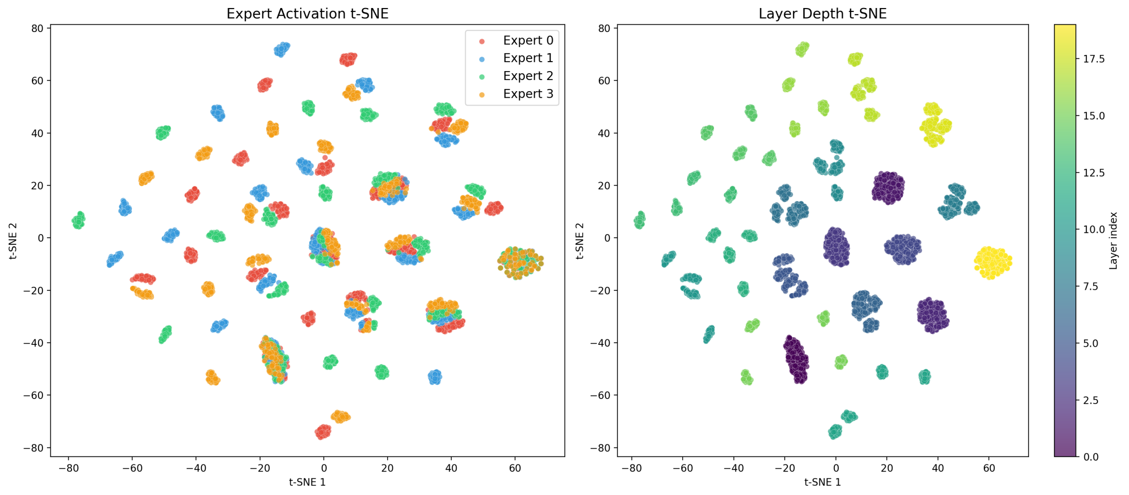


Figure 15: 2D t-SNE of mean expert activations per layer (384M, 8B tokens). *Left*: by expert. *Right*: by depth.

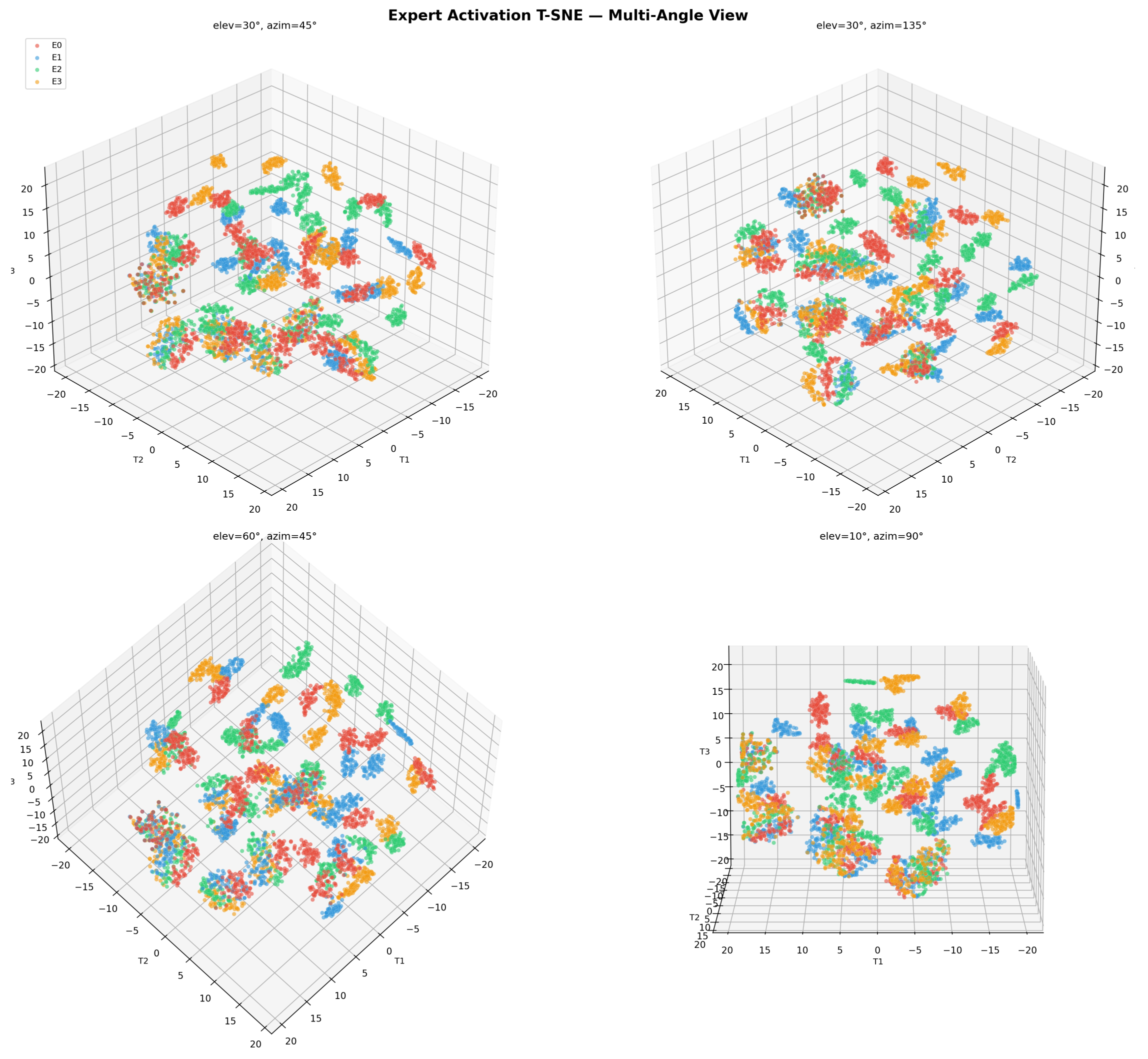


Figure 16: 3D t-SNE of expert activations (384M, 8B tokens), multi-angle views.

Generation example. The 384M Token-Routed model generates coherent text without any supervised fine-tuning:

> **Prompt:** *"The meaning of life is"*
>
> **Output:** "very much the same. The same thing happens to all living things. They live in a constant state of flux. The single cell of a living cell, in this case a cell nucleus, constantly changes to become an organism, and that organism is the organism. The human body is a system of interconnected cells. Each cell is made up of a set of parts, which are connected by a network of specialized cells."

7.6 Limitations and Future Work

- **No-Mu ablation**: *completed* (Run 3, Table 7). Without Mu-Guidance, Token-Routed is slower than dense (+0.011 in average loss), confirming the essential role of the inter-layer equilibrium signal

- **8B-token run at 384M scale**: At 384M scale over 8B tokens, the dense baseline maintains a consistent advantage of $\sim$0.09 in average loss. However, this gap narrows from +0.28 (first 1K steps) to a stable +0.09 (from step 5K onward), demonstrating that Token-Routed experts progressively specialize despite using AdamW—an optimizer designed for homogeneous parameter updates. Notably, at 500M tokens (Run 2, 187M), the Token-Routed model *outperformed* the dense baseline using the same AdamW optimizer. This suggests that the optimizer advantage for dense models accumulates over long training: at low token budgets AdamW is sufficient for Token-Routed to win, but at high token budgets (16$\times$ more tokens) the optimizer's homogeneous treatment of parameters increasingly favors the dense architecture. This scale-dependent behavior further motivates routing-aware optimizers that can maintain the Token-Routed advantage over extended training

- **Standardized benchmarks**: Zero-shot evaluation on ARC-Easy, HellaSwag, and MMLU (Figure 13) shows the Token-Routed model trails the dense baseline by 1–2 points, consistent with the +0.09 loss gap. Scaling to larger models and adding SFT is expected to improve absolute scores

- **AdamTR: routing-aware optimizer**: We propose AdamTR (Adam Token-Routed), an AdamW extension that addresses a fundamental mismatch between standard optimizers and MoE architectures. In a dense model, AdamW estimates the gradient from N tokens per step. In a Token-Routed MoE with E experts, each expert sees only N/E tokens per step, making the per-expert gradient estimate $E\times$ noisier (variance scales as E/N instead of $1/N$). AdamW's second moment v_t also converges slower for experts since it accumulates statistics from N/E samples. This asymmetry accumulates over long training—explaining why Token-Routed wins at 500M tokens but loses at 8B tokens.

  AdamTR corrects this with three mechanisms: (1) **gradient normalization** by $\bar{c}/c_e$ where c_e is the token count for expert e and $\bar{c}$ is the mean, reducing the variance of the gradient estimate to match the dense case; (2) **per-expert learning rate scaling**, compensating for the sub-sampling by allowing experts with fewer tokens to take larger steps; (3) **expert-aware weight decay** with lighter regularization for routed experts, preventing over-regularization that erodes specialization.

  Critically, AdamTR is a strict superset of AdamW: with all scaling factors set to 1.0 and uniform weight decay, it reduces exactly to standard AdamW. This allows controlled ablation and fair comparison. An open-source implementation is available at `complexity-framework/complexity/training/adam_tr.py`

8 Conclusion

We presented COMPLEXITY-DEEP, a language model architecture with three main contributions: (1) Token-Routed MLP with Zipf-balanced greedy bin-packing and Shared Lexical Expert for deterministic assignment without auxiliary losses, (2) Mu-Guided Attention with learnable μ_{init} and μ production after the MLP for expert-aware inter-layer communication, and (3) a training recipe with dynamic warmup (5%) and GPT-style initialization ($1/\sqrt{2L}$).

At 187M scale (500M tokens), Token-Routed outperforms the dense baseline in average loss (4.793 vs 4.905). At 384M scale (8B tokens, iso-parameters), the loss gap narrows from +0.28 to +0.09, with zero-shot benchmarks trailing by only 1–2 points. Deployed on vLLM with CUDA graphs, the 384M model achieves 4,900 tok/s with only 40% throughput reduction despite $2\times$ parameters. The residual +0.09 gap under AdamW motivates AdamTR, a routing-aware optimizer designed to close it.

Broader Impact Statement

This work introduces architectural innovations for language models. While these models have broad applications, they also carry risks of misuse. We release the model weights to enable research while encouraging responsible use.

Reproducibility Statement

To facilitate reproducibility and review, we provide the complete model architecture implementation as supplementary material (`supplementary_code.zip`). The code (PyTorch 2.0+) will be made publicly available upon acceptance.

References

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. *arXiv preprint arXiv:2305.13245*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pp. 7480–7512, 2023.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

Charles D Gilbert and Wu Li. Top-down influences on visual processing. *Nature Reviews Neuroscience*, 14 (5):350–363, 2013.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.

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. *arXiv preprint arXiv:1701.06538*, 2017.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

George Kingsley Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology.* Addison-Wesley Press, 1949.

A Training Curves

Additional training curves and analysis figures are available in the supplementary material.