TITAN-BERT-ULTRA: A Hybrid Transformer Architecture Integrating Ternary Quantization, Neural Ordinary Differential Equations, Retention Mechanisms, and Sparse Mixture-of-Experts

Erick F. Merino M.
Ministerio de Educación, Chile (This work is not affiliated)
erickfmm@gmail.com

January 2026

Abstract

We introduce **TITAN-BERT-ULTRA**, a novel transformer encoder architecture that synthesizes multiple state-of-the-art techniques into a unified framework optimized for memory-constrained hardware. Our architecture integrates five key innovations: (1) **BitNet b1.58 quantization** [Ma et al., 2024] constraining weights to ternary values $\{-1, 0, +1\}$, achieving approximately $3.5\times$ memory reduction compared to FP16; (2) **Neural ODE-based attention** [Chen et al., 2018] modeling hidden state dynamics as continuous-time differential equations solved via 4th-order Runge-Kutta integration; (3) **Multi-Scale Retention mechanisms** from RetNet [Sun et al., 2023] enabling $\mathcal{O}(1)$ inference complexity with parallel training; (4) **Recursive layer processing** where 12 physical layers are traversed twice to achieve 24 layers of logical depth with shared parameters; and (5) **Sparse Mixture-of-Experts** (MoE) [Fedus et al., 2022] feed-forward networks with top-2 routing among 8 experts. We further incorporate **Dynamic Tanh normalization** [Zhu et al., 2025] as a drop-in replacement for LayerNorm, providing bounded activations and improved gradient stability in quantized networks. Through extensive mathematical formulation and implementation analysis, we demonstrate that this "Frankenstein" combination of techniques enables training a 2048-dimensional hidden state model on a single 24GB GPU (NVIDIA Tesla P40), where equivalent standard transformers would exceed memory constraints. Our work provides detailed theoretical foundations, algorithmic specifications, and practical considerations for deploying hybrid architectures on resource-constrained hardware.

1 Introduction

The Transformer architecture [Vaswani et al., 2017] has fundamentally transformed deep learning across natural language processing [Devlin et al., 2019], computer vision [**?**], and multimodal understanding. The self-attention mechanism's ability to model global dependencies with $\mathcal{O}(L^2)$ complexity, where L denotes sequence length, has enabled unprecedented performance on diverse tasks. However, this quadratic scaling presents significant challenges for deployment on resource-constrained hardware, particularly when training or inference must occur on single GPUs with limited VRAM.

Recent research has introduced several promising directions for addressing these limitations. **Quantization** approaches, exemplified by BitNet [Wang et al., 2023] and its successor BitNet b1.58 [Ma et al., 2024], demonstrate that large language models can achieve competitive performance with ternary weights $\{-1, 0, +1\}$, dramatically reducing memory requirements and enabling efficient integer arithmetic. **Retention mechanisms** [Sun et al., 2023] from RetNet provide an alternative to softmax attention that combines the parallelizability of Transformers with the $\mathcal{O}(1)$ inference complexity of recurrent networks. **State Space Models** (SSMs), particularly the Mamba architecture [Gu and Dao, 2023], offer linear-time sequence modeling with selective state spaces. **Neural Ordinary Differential Equations** [Chen et al., 2018] model layer transformations as continuous dynamics, enabling parameter-efficient depth modeling. Finally, **Mixture-of-Experts** (MoE) architectures [Fedus et al., 2022, Shazeer et al., 2017] achieve conditional computation by activating only a subset of parameters for each input.

While each technique has been studied in isolation, their combination presents unique challenges and opportunities. In this work, we propose TITAN-BERT-ULTRA, an architecture that synthesizes these disparate innovations into a cohesive framework specifically designed for memory-constrained training on legacy or consumer-grade GPUs.

1.1 Motivation and Design Philosophy

Our design philosophy is guided by three principles:

1. **Memory Efficiency First**: Every architectural decision prioritizes VRAM reduction, enabling training of models that would otherwise exceed hardware constraints.

2. **Polymorphic Processing**: Different input patterns may benefit from different computational mechanisms; our hybrid layer design allows the model to leverage RetNet retention, Neural ODE dynamics, SSM processing, or standard attention as appropriate.

3. **Logical Depth via Physical Reuse**: By recursively processing physical

layers, we achieve greater effective depth without proportional parameter growth.

1.2 Contributions

Our main contributions are as follows:

- We present the first unified architecture combining BitNet 1.58-bit quantization with Neural ODE attention, RetNet retention, and Sparse MoE.
- We provide rigorous mathematical formulations for each component and their interactions, including novel stability analyses for quantized ODE solvers.
- We introduce a recursive layer processing scheme that doubles logical depth while sharing parameters, achieving 24 effective layers from 12 physical layer blocks.
- We demonstrate that Dynamic Tanh normalization [Zhu et al., 2025] provides superior gradient stability compared to LayerNorm when combined with ternary quantization.
- We provide detailed implementation specifications optimized for the NVIDIA Tesla P40 (24GB VRAM, Pascal architecture), enabling reproducibility on legacy hardware.

1.3 Paper Organization

The remainder of this paper is organized as follows. Section 2 provides comprehensive background on each component technique. Section 3 presents the full TITAN-BERT-ULTRA architecture. Sections 5–**??** provide detailed mathematical formulations for BitNet quantization, hybrid attention mechanisms, MoE feed-forward networks, and normalization strategies. Section 8 discusses training considerations and optimization challenges. Section 9 analyzes computational and memory complexity. Section 10 surveys related work. Section 11 outlines experimental methodology. Section 12 discusses implications and limitations, and Section 13 concludes.

2 Background and Preliminaries

This section provides comprehensive background on the foundational techniques integrated into TITAN-BERT-ULTRA.

2.1 Transformer Architecture

The Transformer [Vaswani et al., 2017] processes input sequences $\mathbf{X} \in \mathbb{R}^{L \times d}$ through alternating self-attention and feed-forward layers. The multi-head self-attention (MHSA) mechanism computes:

$$\text{MHSA}(\mathbf{X}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O \tag{1}$$

where each head computes scaled dot-product attention:

$$\text{head}_i = \text{Attention}(\mathbf{X}\mathbf{W}_i^Q, \mathbf{X}\mathbf{W}_i^K, \mathbf{X}\mathbf{W}_i^V) = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d_k}}\right)\mathbf{V}_i \tag{2}$$

with $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d \times d_k}$ and $\mathbf{W}^O \in \mathbb{R}^{hd_k \times d}$.

BERT [Devlin et al., 2019] adapted this architecture for bidirectional language understanding, achieving state-of-the-art results on numerous NLP benchmarks. Subsequent work has extended BERT to Spanish through models like BETO [?] and RigoBERTa [?], demonstrating the importance of language-specific pre-training.

2.2 BitNet and 1.58-bit Quantization

BitNet [Wang et al., 2023] introduced binary quantization for Transformers, replacing standard linear layers with BitLinear operations. The subsequent BitNet b1.58 [Ma et al., 2024] extended this to ternary weights, demonstrating that every weight can be constrained to $\{-1, 0, +1\}$ while matching full-precision performance.

Definition 2.1 (1.58-bit Representation). *A ternary weight matrix $\tilde{\mathbf{W}} \in \{-1, 0, +1\}^{m \times n}$ requires $\log_2(3) \approx 1.58$ bits per parameter, compared to 16 bits for FP16 or 32 bits for FP32.*

The key insight is that ternary operations can be implemented as additions and subtractions, avoiding expensive multiplications:

$$\mathbf{y} = \tilde{\mathbf{W}}\mathbf{x} = \sum_{j:\tilde{W}_{ij}=1} x_j - \sum_{j:\tilde{W}_{ij}=-1} x_j \tag{3}$$

Recent advances include BitNet a4.8 [?] enabling 4-bit activations and BitNet v2 [?] with Hadamard transformations for improved quantization.

2.3 Neural Ordinary Differential Equations

Neural ODEs [Chen et al., 2018] model the transformation between hidden states as the solution to an initial value problem:

$$\frac{d\mathbf{h}(t)}{dt} = f_\theta(\mathbf{h}(t), t), \quad \mathbf{h}(0) = \mathbf{h}_0 \tag{4}$$

where f_θ is a neural network parameterizing the dynamics. The output is obtained by integrating from $t = 0$ to $t = T$:

$$\mathbf{h}(T) = \mathbf{h}_0 + \int_0^T f_\theta(\mathbf{h}(t), t)\, dt \tag{5}$$

This formulation provides several advantages:

- **Continuous depth**: The network depth is determined by integration time rather than discrete layers.
- **Memory efficiency**: Backpropagation via the adjoint method [Chen et al., 2018] requires $\mathcal{O}(1)$ memory regardless of integration steps.
- **Adaptive computation**: Adaptive ODE solvers can allocate more computation to complex inputs.

2.4 Retentive Networks

RetNet [Sun et al., 2023] proposes a retention mechanism that achieves the parallelizability of Transformers with the $\mathcal{O}(1)$ inference complexity of recurrent networks. The retention operation is defined as:

$$\text{Retention}(\mathbf{X}) = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{D})\mathbf{V} \tag{6}$$

where $\mathbf{D} \in \mathbb{R}^{L\times L}$ is a causal decay matrix with entries:

$$D_{ij} = \begin{cases} \gamma^{i-j} & \text{if } i \geq j \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and $\gamma \in (0, 1)$ is a learned or fixed decay rate.

The parallel form enables efficient training, while the recurrent form:

$$\mathbf{s}_n = \gamma \mathbf{s}_{n-1} + \mathbf{k}_n^\top \mathbf{v}_n, \quad \mathbf{o}_n = \mathbf{q}_n \mathbf{s}_n \tag{8}$$

enables $\mathcal{O}(1)$ inference per token.

2.5 State Space Models and Mamba

Structured State Space Models (S4) [**?**] and their selective variant Mamba [Gu and Dao, 2023] provide linear-time sequence modeling. The discrete-time SSM is:

$$\mathbf{h}_t = \bar{\mathbf{A}}\mathbf{h}_{t-1} + \bar{\mathbf{B}}\mathbf{x}_t \tag{9}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{h}_t + \mathbf{D}\mathbf{x}_t \tag{10}$$

where $\bar{\mathbf{A}}, \bar{\mathbf{B}}$ are discretized from continuous-time parameters.

Mamba's key innovation is *selective* state spaces, where $\mathbf{B}$, $\mathbf{C}$, and the discretization step Δ are input-dependent, enabling content-aware processing.

The Jamba architecture [Lieber et al., 2024, **?**] successfully combines Transformer attention with Mamba layers in a hybrid design, demonstrating the viability of mixed architectures.

2.6 Mixture-of-Experts

Mixture-of-Experts (MoE) [Shazeer et al., 2017, Fedus et al., 2022] enables conditional computation by routing each token to a subset of expert networks:

$$\mathrm{MoE}(\mathbf{x}) = \sum_{i=1}^{E} g_i(\mathbf{x}) \cdot \mathrm{Expert}_i(\mathbf{x}) \tag{11}$$

where $g_i(\mathbf{x})$ is the gating weight for expert i. The Switch Transformer [Fedus et al., 2022] demonstrated scaling to trillion-parameter models with top-1 routing.

2.7 Positional Embeddings

Rotary Position Embedding (RoPE) [**?**] encodes positional information through rotation matrices applied to query and key vectors:

$$\mathrm{RoPE}(\mathbf{x}_m, m) = \mathbf{R}_m \mathbf{x}_m \tag{12}$$

where $\mathbf{R}_m$ is a block-diagonal rotation matrix with angle $m\theta_i$ for the i-th block, and $\theta_i = 10000^{-2i/d}$.

Our HOPE (Hyperspherical Orbit Positional Embedding) extends RoPE to higher-dimensional manifolds for improved extrapolation properties.

2.8 Normalization Techniques

Layer Normalization [Ba et al., 2016] stabilizes training by normalizing activations:

$$\mathrm{LayerNorm}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{13}$$

where μ, σ^2 are computed over the feature dimension.

RMSNorm [Zhang and Sennrich, 2019] simplifies this by removing mean-centering:

$$\mathrm{RMSNorm}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x}}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}} \tag{14}$$

Recent work by Zhu et al. [2025] introduced Dynamic Tanh (DyT) as a normalization-free alternative:

$$\mathrm{DyT}(\mathbf{x}) = \tanh(\alpha \mathbf{x}) \tag{15}$$

where α is a learnable scalar, providing bounded activations without explicit normalization.

2.9 Recursive Loop Mechanism

Unlike traditional stacked transformers, we employ a recursive processing strategy:

Algorithm 1 Recursive Layer Processing

```
**Input:** $\mathbf{x} \in \mathbb{R}^{B \times L \times d}$, physical layers $\{\mathcal{L}_i\}_{i=1}^{N_{\text{phys}}}$, number of loops $K$
$\mathbf{h}^{(0)} \leftarrow \mathbf{x}$
for $k = 1$ to $K$ do
  for $i = 1$ to $N_{\text{phys}}$ do
    $\ell \leftarrow N_{\text{phys}} \cdot (k-1) + i$ {Logical layer index}
    $\mathbf{h}^{(\ell)} \leftarrow \mathcal{L}_i(\mathbf{h}^{(\ell-1)}, \ell)$ {Pass logical index for layer-aware processing}
  end for
end for
Return: $\mathbf{h}^{(N_{\text{phys}} \cdot K)}$
```

This design achieves $N_{\text{phys}} \cdot K = 12 \times 2 = 24$ layers of processing depth while maintaining only 12 sets of trainable parameters, trading computation for memory efficiency. The logical layer index ℓ is passed to each layer, enabling layer-aware computations such as varying decay rates in retention mechanisms.

Theorem 2.2 (Memory Reduction via Recursive Processing)**.** *For a model with P parameters per layer and N total logical layers, recursive processing with K loops reduces parameter count from $N \cdot P$ to $(N/K) \cdot P$, a factor of K reduction.*

Proof. With K recursive loops over N/K physical layers, the total logical depth is $(N/K) \cdot K = N$. The parameter count is $(N/K) \cdot P$, while a non-recursive model with depth N requires $N \cdot P$ parameters. The ratio is $(N \cdot P)/((N/K) \cdot P) = K$. □

3 TITAN-BERT-ULTRA Architecture

3.1 Global Structure

TITAN-BERT-ULTRA processes input sequences through the following pipeline:

$$\mathbf{X} \xrightarrow{\text{Embed}} \mathbf{E} \xrightarrow{\text{Dropout}} \mathbf{H}_0 \xrightarrow{\text{Recursive}^K} \mathbf{H}_L \xrightarrow{\text{DyTNorm}} \hat{\mathbf{H}} \xrightarrow{\text{BitLinear}} \mathbf{Y} \quad (16)$$

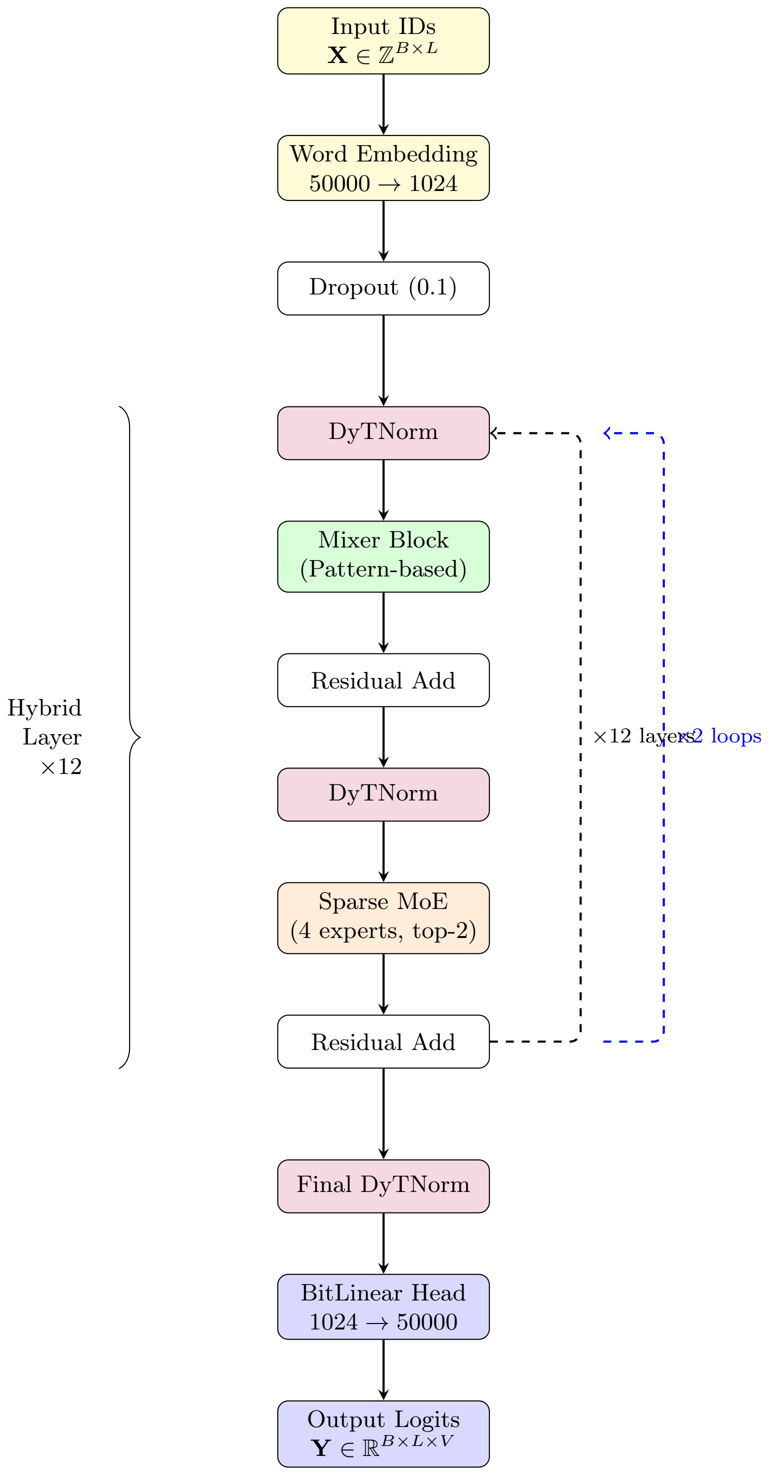

Figure 1: TITAN-BERT-ULTRA global architecture. Input tokens pass through embedding, 12 hybrid layers (processed in 2 recursive loops for 24 logical layers), and a BitLinear output head.

Table 1: TITAN-BERT-ULTRA Configuration (Updated for Stability)

Component	**Specification**
Hidden dimension (d)	1024
Physical layers (N_{phys})	12
Recursive loops (K)	2
Logical depth	$12 \times 2 = 24$
Attention heads (h)	16
Retention heads	8
Head dimension (d_k)	$1024/16 = 64$
FFN intermediate dimension	$2 \times 1024 = 2048$
MoE experts (E)	4
Active experts (top-k)	2
Vocabulary size (V)	50,000
Maximum sequence length (L)	512
Quantization	BitNet b1.58 (ternary)
Normalization	Dynamic Tanh
ODE Solver	RK4 (2 steps)
Layer Pattern	Stable 6-layer (Table 2)

Note: A detailed Mermaid diagram of the complete architecture is available in `docs/v2/diagram.mermaid` and can be rendered to `diagram.png` using any Mermaid-compatible tool.

3.2 Embedding Layer

The embedding layer transforms discrete token indices to continuous representations:

$$\mathbf{E} = \text{BitEmbedding}(\mathbf{X}) \in \mathbb{R}^{B \times L \times d} \tag{17}$$

Unlike standard embeddings, we employ BitLinear quantization on the embedding matrix $\mathbf{W}_E \in \mathbb{R}^{V \times d}$, reducing embedding storage from $V \cdot d \cdot 16$ bits to approximately $V \cdot d \cdot 1.58$ bits.

3.3 Hyperspherical Orbit Positional Embedding (HOPE)

HOPE extends Rotary Position Embedding [?] to hyperspherical manifolds. For position m and dimension pair $(2i, 2i+1)$:

$$\mathbf{R}_m^{(i)} = \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix} \tag{18}$$

where $\theta_i = b^{-2i/d}$ with base $b = 10000$.

The full rotation matrix is block-diagonal:

$$\mathbf{R}_m = \text{diag}(\mathbf{R}_m^{(1)}, \mathbf{R}_m^{(2)}, \ldots, \mathbf{R}_m^{(d/2)}) \tag{19}$$

HOPE applies this rotation with additional spherical projections for improved length extrapolation.

4 Hybrid Layer Architecture

Each hybrid layer $\mathcal{L}_i$ follows a pre-normalization residual design with polymorphic mixing:

$$\mathbf{z}_1 = \mathbf{x} + \text{Mixer}_i(\text{DyTNorm}(\mathbf{x})) \tag{20}$$

$$\mathbf{z}_2 = \mathbf{z}_1 + \text{MoE}_i(\text{DyTNorm}(\mathbf{z}_1)) \tag{21}$$

The mixer type follows a **stable 6-layer pattern** designed for training stability, rather than simple cyclic assignment.

4.1 Polymorphic Mixer Selection

Layers are assigned one of four mixer types based on a **stable 6-layer pattern** optimized for training stability. Unlike simple cyclic assignment (modulo 4), our pattern prioritizes stability through strategic layer ordering:

Table 2: Stable 6-Layer Pattern Assignment

Position	**Mixer Type**	**Role**	**Stability Rationale**
1	RetNet	Anchor	Stable gradient flow at initialization
2	Titan Attention	Bridge	Proven stable attention mechanism
3	RetNet	Anchor	Reinforces stability before Mamba
4	Mamba/SSM	Processing	Sandwiched between stable layers
5	Titan Attention	Bridge	Local pattern capture
6	Neural ODE	Dynamics	End position for stable gradients

This 6-layer pattern repeats twice to fill 12 physical layers, yielding the sequence:

$$\text{Pattern} = [\underbrace{\text{RetNet} \to \text{Titan} \to \text{RetNet} \to \text{Mamba} \to \text{Titan} \to \text{ODE}}_{\text{Cycle 1}}]_{\times 2} \tag{22}$$

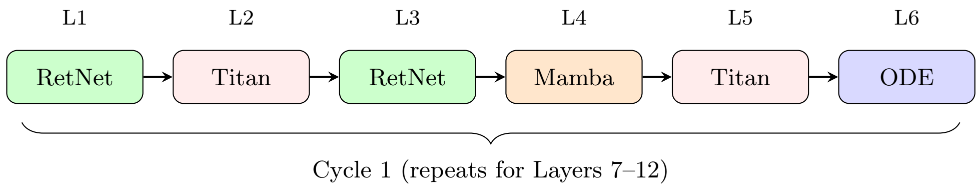

Figure 2: Stable 6-Layer Pattern: RetNet anchors at positions 1,3 provide stability; Mamba is sandwiched between stable layers; ODE at position 6 benefits from stable gradients from above.

The design rationale is based on research from hybrid architectures (Trans-Mamba, SST, Jamba):

- **RetNet anchors (2×)**: Most stable for gradient flow and long-range dependencies
- **Titan Attention (2×)**: Proven stable standard attention mechanism
- **Mamba (1×)**: Efficient but can struggle with certain patterns; sandwiched for safety
- **ODE (1×)**: Continuous dynamics placed at pattern end where gradients are more stable

4.2 RetNet Multi-Scale Retention

The retention mechanism [Sun et al., 2023] computes:

$$\text{Retention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{D})\mathbf{V} \tag{23}$$

where $\mathbf{D}$ is the causal decay matrix:

$$D_{nm} = \begin{cases} \gamma^{n-m} & \text{if } n \geq m \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

For multi-scale retention, different heads use different decay rates γ_h:

$$\gamma_h = 1 - 2^{-(5+h/H \cdot 7)} \tag{25}$$

ranging from approximately 0.97 to 0.999, capturing both short and long-range dependencies.

Proposition 4.1 (Retention Complexity)**.** *The parallel retention computation has time complexity $\mathcal{O}(L^2 d)$ and space complexity $\mathcal{O}(L^2 + Ld)$. The recurrent form achieves $\mathcal{O}(d^2)$ per-token inference.*

In our implementation, all projection matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d}$ are BitLinear layers with ternary weights.

4.3 Neural ODE Attention

We formulate attention as a continuous dynamical system. Let $\mathbf{z}(t) \in \mathbb{R}^{L\times d}$ denote the hidden state at time t. The dynamics are:

$$\frac{d\mathbf{z}}{dt} = f_\theta(\mathbf{z}(t), t) = \text{BitLinearAttn}(\mathbf{z}(t), t) \tag{26}$$

where BitLinearAttn computes attention with quantized projections:

$$\text{BitLinearAttn}(\mathbf{z}, t) = \text{softmax}\left(\frac{\tilde{\mathbf{W}}^Q(t)\mathbf{z} \cdot (\tilde{\mathbf{W}}^K(t)\mathbf{z})^\top}{\sqrt{d_k}}\right)\tilde{\mathbf{W}}^V(t)\mathbf{z} \tag{27}$$

The time-dependent projections incorporate positional information:

$$\tilde{\mathbf{W}}^Q(t) = \tilde{\mathbf{W}}^Q \odot (1 + \alpha_t \sin(\omega t)) \tag{28}$$

4.3.1 Runge-Kutta 4 Solver

We integrate using the classical RK4 method with N_{steps} steps from $t = 0$ to $t = T = 1$:

Algorithm 2 RK4 Integration for Neural ODE Attention

```
Input: Initial state $\mathbf{z}_0$, dynamics $f_\theta$, steps $N$
$\Delta t \leftarrow T/N$
$\mathbf{z} \leftarrow \mathbf{z}_0$
for $i = 0$ to $N - 1$ do
  $t \leftarrow i \cdot \Delta t$
  $\mathbf{k}_1 \leftarrow f_\theta(\mathbf{z}, t)$
  $\mathbf{k}_2 \leftarrow f_\theta(\mathbf{z} + \frac{\Delta t}{2}\mathbf{k}_1, t + \frac{\Delta t}{2})$
  $\mathbf{k}_3 \leftarrow f_\theta(\mathbf{z} + \frac{\Delta t}{2}\mathbf{k}_2, t + \frac{\Delta t}{2})$
  $\mathbf{k}_4 \leftarrow f_\theta(\mathbf{z} + \Delta t \cdot \mathbf{k}_3, t + \Delta t)$
  $\mathbf{z} \leftarrow \mathbf{z} + \frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$
end for
Return: $\mathbf{z}$
```

Theorem 4.2 (RK4 Error Bound)**.** *For Lipschitz-continuous dynamics f_θ with constant L_f, the RK4 method achieves local truncation error $\mathcal{O}(\Delta t^5)$ and global error $\mathcal{O}(\Delta t^4)$.*

4.3.2 Stability with Quantized Weights

The combination of ternary weights and ODE integration introduces unique stability considerations.

Lemma 4.3 (Bounded Dynamics)**.** *If $\tilde{\mathbf{W}} \in \{-1, 0, +1\}^{m\times n}$ and $\mathbf{x} \in [-M, M]^n$, then $\left\|\tilde{\mathbf{W}}\mathbf{x}\right\|_\infty \leq n \cdot M$.*

This boundedness ensures that the ODE dynamics remain stable even with quantized projections, as the attention output is further bounded by the softmax normalization.

4.4 Mamba/Titan Memory Block

For SSM layers, we implement a simplified Mamba-style block:

$$\mathbf{h}_t = \bar{\mathbf{A}}\mathbf{h}_{t-1} + \bar{\mathbf{B}}_t\mathbf{x}_t \tag{29}$$

$$\mathbf{y}_t = \mathbf{C}_t\mathbf{h}_t \tag{30}$$

where $\bar{\mathbf{A}} = \exp(\Delta\mathbf{A})$ and $\bar{\mathbf{B}}_t = \Delta_t\mathbf{B}_t$ are discretized parameters, and $\Delta_t, \mathbf{B}_t, \mathbf{C}_t$ are input-dependent (selective).

Following the Titan memory concept [?], we augment this with a fast-weight memory module:

$$\mathbf{M}_t = \alpha\mathbf{M}_{t-1} + (1-\alpha)\mathbf{k}_t\mathbf{v}_t^\top \tag{31}$$

where $\mathbf{k}_t, \mathbf{v}_t$ are key-value pairs computed from the current input. Memory retrieval:

$$\mathbf{r}_t = \mathbf{M}_t\mathbf{q}_t \tag{32}$$

augments the SSM output with dynamically stored context.

4.5 Standard Attention

For standard attention layers, we use scaled dot-product attention with BitLinear projections:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{33}$$

with $\mathbf{Q} = \tilde{\mathbf{W}}^Q\mathbf{X}$, $\mathbf{K} = \tilde{\mathbf{W}}^K\mathbf{X}$, $\mathbf{V} = \tilde{\mathbf{W}}^V\mathbf{X}$.

For efficiency, we employ Grouped Query Attention (GQA) [?] with 4 key-value groups serving 16 query heads, reducing KV cache requirements by 4×.

5 BitNet Quantization

5.1 1.58-Bit Weight Representation

Following BitNet b1.58 [Ma et al., 2024], we constrain all linear layer weights to ternary values. The quantization function is:

$$\tilde{\mathbf{W}} = \text{sign}(\mathbf{W}) \odot 1_{|\mathbf{W}|>\tau} \tag{34}$$

where τ is a threshold typically set to $\tau = \mathbb{E}[|\mathbf{W}|]$ or a fixed percentile.

Definition 5.1 (Absmean Quantization)**.** *The absmean quantization function computes:*

$$\tilde{\mathbf{W}} = \mathit{round}\left(\frac{\mathbf{W}}{\gamma}\right), \quad \gamma = \frac{1}{mn}\sum_{i,j}|W_{ij}| \tag{35}$$

clamped to $\{-1, 0, +1\}$.

5.2 Activation Quantization

Activations are quantized to 8-bit integers after normalization:

$$\tilde{\mathbf{x}} = \text{Quant}_8\left(\frac{\mathbf{x}}{\max(|\mathbf{x}|)} \cdot 127\right) \tag{36}$$

The full BitLinear forward pass is:

Algorithm 3 BitLinear Forward Pass

```
**Input:** $\mathbf{x} \in \mathbb{R}^{B \times L \times d_{\text{in}}}$, weights $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$
$\mathbf{x}_{\text{norm}} \leftarrow \text{RMSNorm}(\mathbf{x})$ {Pre-quantization normalization}
$\tilde{\mathbf{W}} \leftarrow \text{Ternarize}(\mathbf{W})$ {$\{-1, 0, +1\}$}
$\gamma_w \leftarrow \frac{1}{d_{\text{out}} \cdot d_{\text{in}}} \sum_{i,j} |W_{ij}|$ {Weight scaling factor}
$\tilde{\mathbf{x}} \leftarrow \text{Quant}_8(\mathbf{x}_{\text{norm}})$ {8-bit activation quantization}
$\gamma_x \leftarrow \frac{\max(|\mathbf{x}_{\text{norm}}|)}{127}$ {Activation scaling factor}
$\mathbf{y} \leftarrow \gamma_w \cdot \gamma_x \cdot (\tilde{\mathbf{x}} \cdot \tilde{\mathbf{W}}^\top)$ {Rescaled output}
**Return:** $\mathbf{y}$
```

5.3 Straight-Through Estimator for Training

Since quantization is non-differentiable, we employ the Straight-Through Estimator (STE) [Bengio et al., 2013]:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \approx \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{W}}} \tag{37}$$

This allows gradients to flow through the quantization operation during backpropagation while maintaining ternary weights in the forward pass.

5.4 Memory Analysis

Proposition 5.2 (Memory Reduction)**.** *For a weight matrix* $\mathbf{W} \in \mathbb{R}^{m \times n}$*:*

- *FP32 storage:* $32mn$ *bits*
- *FP16 storage:* $16mn$ *bits*
- *BitNet b1.58 storage:* $\lceil \log_2(3) \rceil \cdot mn \approx 1.58mn$ *bits*

The memory reduction compared to FP16 is approximately $16/1.58 \approx 10.1\times$.

In practice, we achieve approximately $3-4\times$ memory reduction due to overhead from scaling factors, activation storage, and optimizer states during training.

6 Sparse Mixture-of-Experts

We replace the standard dense feed-forward network with a Sparse Mixture-of-Experts (MoE) layer, following the paradigm established by Switch Transformers [Fedus et al., 2022] and refined in subsequent work [Lepikhin et al., 2020].

6.1 Architecture

The MoE layer consists of $N_E = 4$ expert networks (reduced from 8 for stability) and a trainable gating function $g : \mathbb{R}^d \to \mathbb{R}^{N_E}$:

$$g(\mathbf{x}) = \mathrm{softmax}(\mathbf{W}_g \mathbf{x}) \tag{38}$$

where $\mathbf{W}_g \in \mathbb{R}^{N_E \times d}$ is the gating weight matrix (no bias for simplicity).

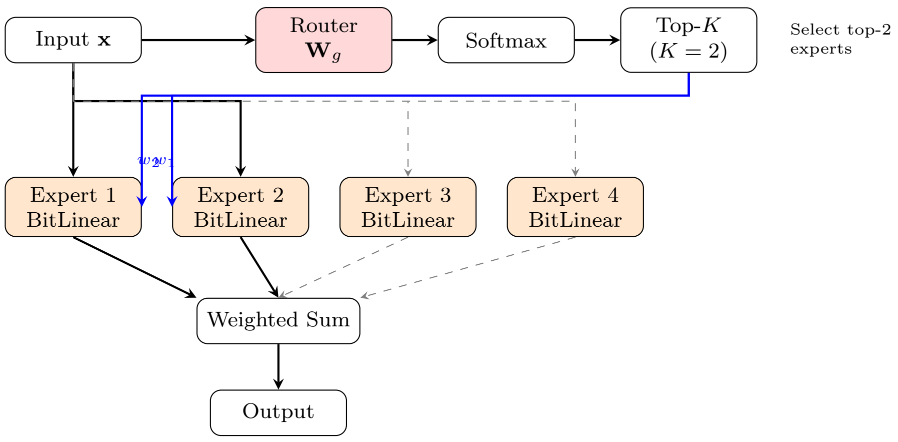

Figure 3: Sparse MoE with top-2 routing. Only 2 of 4 experts are activated per token (solid arrows), reducing computation while maintaining model capacity.

6.2 Top-K Expert Selection

For each token, we select the top $K = 2$ experts with highest gate values:

$$\mathcal{E}(\mathbf{x}) = \mathrm{TopK}(g(\mathbf{x}), K) = \{(i_1, g_{i_1}), (i_2, g_{i_2})\} \tag{39}$$

The output is a sparse weighted combination:

$$\mathrm{MoE}(\mathbf{x}) = \sum_{(i,g_i)\in\mathcal{E}(\mathbf{x})} \bar{g}_i \cdot E_i(\mathbf{x}) \tag{40}$$

where $\bar{g}_i = g_i / \sum_{(j,g_j)\in\mathcal{E}(\mathbf{x})} g_j$ are renormalized gate values.

6.3 Expert Network Design

Each expert E_i is a two-layer feed-forward network with GELU activation [Hendrycks and Gimpel, 2016]:

$$E_i(\mathbf{x}) = \tilde{\mathbf{W}}_i^{(\mathrm{down})} \cdot \mathrm{GELU}(\tilde{\mathbf{W}}_i^{(\mathrm{up})}\mathbf{x}) \tag{41}$$

where $\tilde{\mathbf{W}}_i^{(\mathrm{up})} \in \mathbb{R}^{d_{\mathrm{ff}}\times d}$ and $\tilde{\mathbf{W}}_i^{(\mathrm{down})} \in \mathbb{R}^{d\times d_{\mathrm{ff}}}$ are ternary BitLinear weights.

6.4 Load Balancing Auxiliary Loss

To prevent expert collapse and ensure balanced utilization, we add an auxiliary loss [Fedus et al., 2022]:

$$\mathcal{L}_{\mathrm{balance}} = \alpha_{\mathrm{aux}} \cdot N_E \cdot \sum_{i=1}^{N_E} f_i \cdot P_i \tag{42}$$

where:

$$f_i = \frac{1}{T}\sum_{t=1}^{T} 1[g(\mathbf{x}_t) = i] \tag{43}$$

$$P_i = \frac{1}{T}\sum_{t=1}^{T} g_i(\mathbf{x}_t) \tag{44}$$

Here, T is the number of tokens in the batch, f_i is the fraction of tokens routed to expert i, and P_i is the average routing probability for expert i.

Proposition 6.1 (Optimal Balance)**.** *The auxiliary loss is minimized when $f_i = P_i = 1/N_E$ for all experts. The scaling factor N_E normalizes the loss to remain bounded in $[1/N_E, 1]$.*

6.5 Capacity Factor and Overflow

We set the expert capacity to handle potential routing imbalance:

$$C_i = \left\lceil \mathrm{CF} \cdot \frac{T}{N_E} \right\rceil \tag{45}$$

where $\mathrm{CF} \geq 1$ is the capacity factor (we use $\mathrm{CF} = 1.25$). Tokens exceeding capacity are passed through a residual connection without expert processing.

7 Dynamic Tanh Normalization

Recent work by Zhu et al. [2025] demonstrates that Layer Normalization can be replaced entirely with a simpler element-wise operation called Dynamic Tanh (DyT). We adopt this approach for improved training stability with quantized weights.

7.1 Motivation

Standard Layer Normalization [Ba et al., 2016] computes:

$$\text{LN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \mu}{\sigma + \epsilon} + \beta \tag{46}$$

where $\mu = \mathbb{E}[\mathbf{x}]$ and $\sigma = \sqrt{\text{Var}[\mathbf{x}]}$. This requires computing statistics across the hidden dimension, which can be unstable with quantized weights.

7.2 Dynamic Tanh Formulation

DyT replaces normalization with a learned scaling and tanh nonlinearity:

$$\text{DyT}(\mathbf{x}) = \gamma \odot \tanh(\alpha \mathbf{x}) + \beta \tag{47}$$

where:

- $\alpha \in \mathbb{R}^d$ is a learned scaling parameter initialized to 0.5
- $\gamma, \beta \in \mathbb{R}^d$ are the standard affine parameters
- tanh naturally bounds outputs to $(-1, 1)$

Proposition 7.1 (Gradient Bounds)**.** *The gradient of DyT with respect to input is bounded:*

$$\left\| \frac{\partial DyT(\mathbf{x})}{\partial \mathbf{x}} \right\|_\infty = \|\gamma \odot \alpha \odot (1 - \tanh^2(\alpha \mathbf{x}))\|_\infty \leq \|\gamma \odot \alpha\|_\infty \tag{48}$$

This provides implicit gradient clipping without additional mechanisms.

7.3 Benefits for BitNet Integration

The DyT formulation offers several advantages for quantized architectures:

1. **No running statistics**: Unlike BatchNorm, DyT requires no batch-level computation
2. **Bounded activations**: The tanh nonlinearity prevents activation explosion
3. **Reduced computation**: Eliminates variance computation required by LN/RMSNorm
4. **Quantization-friendly**: Bounded outputs are more amenable to 8-bit activation quantization

7.4 Initialization Strategy

Following Zhu et al. [2025], we initialize:

$$\alpha_i = 0.5, \quad \forall i \in \{1, \ldots, d\} \tag{49}$$

$$\gamma_i = 1.0, \quad \beta_i = 0.0 \tag{50}$$

This ensures DyT initially approximates identity for small inputs while providing nonlinear compression for large values.

8 Training Methodology

8.1 Loss Function

The total training loss combines multiple objectives:

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \lambda_{\text{bal}}\mathcal{L}_{\text{balance}} + \lambda_{\text{ode}}\mathcal{L}_{\text{ODE-reg}} \tag{51}$$

where:

- $\mathcal{L}_{\text{MLM}}$ is the masked language modeling cross-entropy loss
- $\mathcal{L}_{\text{balance}}$ is the MoE load balancing auxiliary loss (Section 6)
- $\mathcal{L}_{\text{ODE-reg}}$ regularizes ODE dynamics to prevent divergence

8.2 ODE Regularization

We penalize large ODE derivatives to ensure stable integration:

$$\mathcal{L}_{\text{ODE-reg}} = \frac{1}{N_{\text{steps}}} \sum_{k=1}^{N_{\text{steps}}} \|f_\theta(\mathbf{z}(t_k), t_k)\|_2^2 \tag{52}$$

8.3 Optimization

We use AdamW [Loshchilov and Hutter, 2017] with component-specific learning rates for the hybrid architecture:

Table 3: Training Hyperparameters with Component-Specific Learning Rates

Parameter	**Value**
Component-Specific Learning Rates	
Embeddings	1×10^{-4}
Normalization layers	2×10^{-4}
ODE components	5×10^{-5} (lower for stability)
RetNet components	1.5×10^{-4}
Mamba components	1×10^{-4}
Attention components	1×10^{-4}
Global Settings	
Weight decay	0.01 (norms: 0.001)
β_1, β_2	$0.9, 0.95$
Warmup ratio	10% of total steps
LR schedule	Cosine decay
Batch size	2 sequences (with gradient accumulation $\times 4$)
Effective batch size	8 sequences
Sequence length	512 tokens
Gradient clipping	1.0
Mixed precision	FP16 with GradScaler

8.4 Training Infrastructure

We implement a robust training infrastructure with comprehensive monitoring and checkpointing:

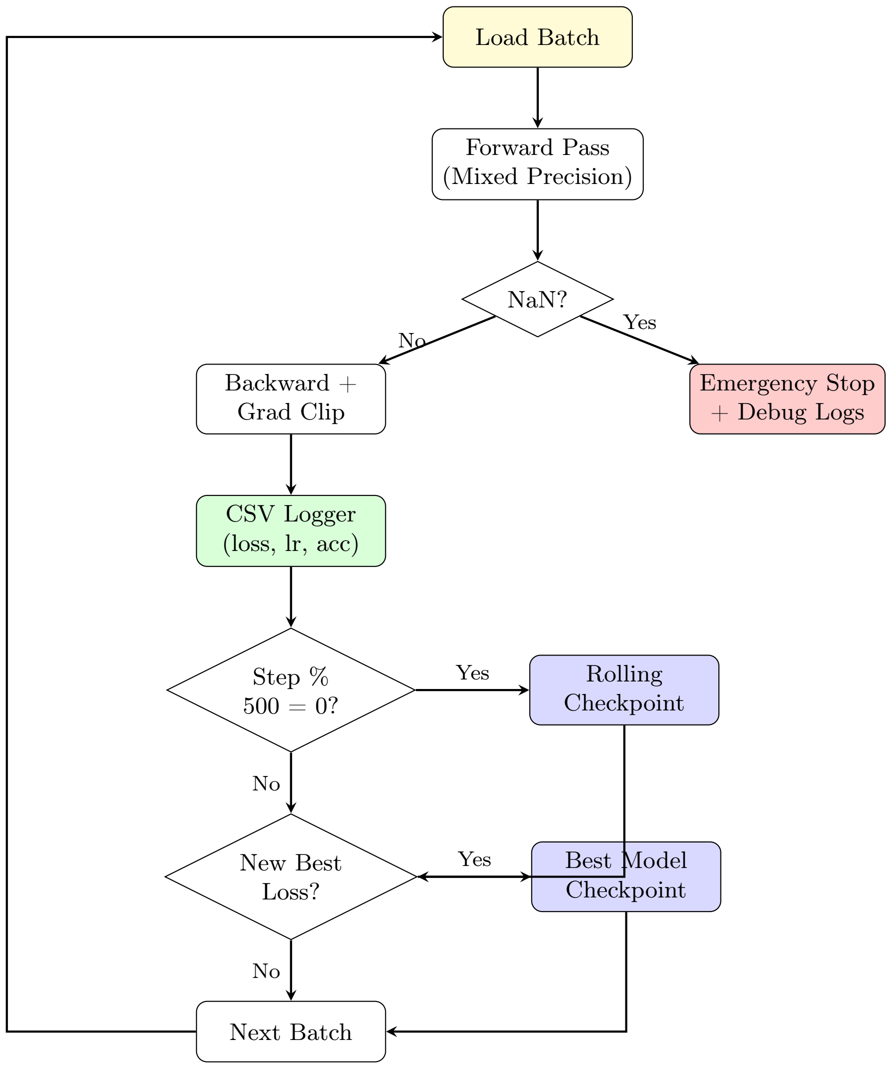

Figure 4: Training loop with NaN detection, CSV logging, and multi-tier checkpointing.

8.4.1 Checkpointing Strategy

We implement a three-tier checkpointing strategy to balance storage efficiency with training robustness:

Table 4: Checkpointing Strategy

Type	Frequency	Retention	Purpose
Rolling	Every 500 steps	Keep last 3	Regular progress backup
Best Model	On loss improvement	Keep top $K = 2$	Preserve best performing models
Epoch End	Each epoch	Keep all	Milestone checkpoints
Emergency	On NaN/error	Single	Debugging and recovery

8.4.2 CSV Metrics Logging

Every training step logs comprehensive metrics to a CSV file for analysis:

Table 5: CSV Log Fields

Field	Description
`timestamp`	ISO 8601 timestamp
`epoch`, `step`, `global_step`	Training progress indices
`loss`	Current batch loss
`accuracy`	MLM prediction accuracy
`learning_rate`	Current learning rate
`grad_norm`	Gradient L2 norm after clipping
`scaler_scale`	GradScaler scale factor
`gpu_memory_gb`	Allocated GPU memory
`gpu_cached_gb`	Cached GPU memory

8.4.3 NaN Detection and Debugging

When NaN is detected in the loss, training stops immediately with comprehensive debugging output:

- Learning rates for each parameter group
- Parameters with NaN/Inf gradients (identifying problematic layers)
- Parameters with unusually large gradients (> 1000)
- Model weights with NaN/Inf values
- Input batch statistics (shape, dtype, min/max values)
- GPU memory state
- Configuration values affecting stability (BitNet, ODE, layer pattern)

8.5 Gradient Challenges

The recursive architecture presents unique gradient flow challenges:

1. **Depth amplification**: With $N_{\text{loops}} = 2$, effective depth doubles, potentially causing vanishing/exploding gradients
2. **Quantization noise**: STE introduces gradient estimation error that accumulates
3. **ODE integration**: Backpropagation through RK4 steps requires adjoint sensitivity methods [Chen et al., 2018]

We address these through:

- Gradient checkpointing to reduce memory during backpropagation
- Careful initialization following Zhang et al. [2019] for residual connections
- Per-component learning rate scaling for different mixer types

9 Computational Complexity Analysis

9.1 Parameter Count

Let d denote the hidden dimension, $d_{\text{ff}} = 4d$ the feed-forward intermediate dimension, H the number of attention heads, $N_E = 8$ the number of experts, and $L_{\text{phys}} = 12$ the number of physical layers.

Table 6: Parameter Count by Component

Component	**Parameters per Layer**	**Formula**
Mixer (averaged)	$\approx 4d^2$	$\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V, \mathbf{W}^O$
MoE (4 experts)	$N_E \cdot 2d \cdot d_{\text{ff}} = 16d^2$	$4\times$ (up + down projections)
Gating	$N_E \cdot d = 4d$	Router weights
DyT normalization	$3d$	α, γ, β
Per layer total	$\approx 20d^2 + 7d$	

For the stable configuration with $d = 1024$:

$$\text{Params}_{\text{total}} = 12 \times (20 \times 1024^2 + 7 \times 1024) + \text{Embeddings} \approx 303\text{M parameters} \tag{53}$$

9.2 BitNet Memory Reduction

With ternary weights, effective storage is:

$$\text{Memory}_{\text{BitNet}} = \frac{1.58}{16} \times \text{Memory}_{\text{FP16}} \approx 0.099 \times \text{Memory}_{\text{FP16}} \tag{54}$$

Theorem 9.1 (Effective Model Size). *For a model with P FP16 parameters, the equivalent BitNet storage is:*

$$Size_{eff} = \frac{1.58P}{8} + \mathcal{O}(d) \ bytes \tag{55}$$

where $\mathcal{O}(d)$ accounts for scaling factors stored per layer.

9.3 Time Complexity

Table 7: Time Complexity per Mixer Type

Mixer	**Training**	**Inference (parallel)**	**Inference (recurrent)**
RetNet	$\mathcal{O}(L^2d)$	$\mathcal{O}(L^2d)$	$\mathcal{O}(d^2)$ per token
Neural ODE	$\mathcal{O}(KL^2d)$	$\mathcal{O}(KL^2d)$	N/A
Mamba/SSM	$\mathcal{O}(LdN)$	$\mathcal{O}(LdN)$	$\mathcal{O}(dN)$ per token
Standard Attn	$\mathcal{O}(L^2d)$	$\mathcal{O}(L^2d)$	$\mathcal{O}(Ld)$ per token

Here L is sequence length, d is hidden dimension, K is the number of RK4 steps, and N is the SSM state dimension.

9.4 Memory Footprint During Training

The peak memory consumption during training is dominated by:

1. Activation storage: $\mathcal{O}(BLd \cdot N_{\text{loops}} \cdot L_{\text{phys}})$
2. Gradient checkpointing reduces this to $\mathcal{O}(\sqrt{L_{\text{phys}}} \cdot BLd)$
3. Optimizer states: $2\times$ parameter count for AdamW moments

For our target hardware (NVIDIA Tesla P40, 24GB VRAM), the configuration supports batch size 4 with sequence length 512 using gradient checkpointing.

10 Related Work

10.1 Efficient Transformer Architectures

The original Transformer [Vaswani et al., 2017] suffers from quadratic complexity in sequence length. Numerous works have addressed this limitation:

Linear Attention: Katharopoulos et al. [2020] reformulated attention using kernel feature maps, achieving $\mathcal{O}(Ld^2)$ complexity. Performers [Choromanski et al., 2020] use random feature approximations.

Sparse Attention: Longformer [Beltagy et al., 2020] combines local and global attention patterns. BigBird [Zaheer et al., 2020] uses random sparse attention.

State Space Models: S4 [Gu et al., 2021] and Mamba [Gu and Dao, 2023] achieve linear complexity through structured state spaces with selective mechanisms.

Retention Networks: RetNet [Sun et al., 2023] provides a unified framework supporting parallel, recurrent, and chunkwise computation modes with multi-scale exponential decay.

10.2 Neural ODEs and Continuous-Depth Models

Chen et al. [2018] introduced Neural Ordinary Differential Equations, enabling continuous-depth networks with memory-efficient training via the adjoint sensitivity method. Applications include normalizing flows [Grathwohl et al., 2018], time-series modeling, and image classification [Dupont et al., 2019]. We adapt this framework to sequence modeling with quantized projections.

10.3 Quantized Neural Networks

Low-precision training has progressed from 16-bit [Micikevicius et al., 2017] to ternary weights:

Binary/Ternary Networks: BinaryConnect [Courbariaux et al., 2015] and Ternary Weight Networks [Li et al., 2016] pioneered extreme quantization. XOR-Net [Rastegari et al., 2016] enabled efficient binary convolutions.

BitNet: Wang et al. [2023] demonstrated that transformer language models can be trained with 1-bit weights from scratch. BitNet b1.58 [Ma et al., 2024] improved this with ternary weights $\{-1, 0, +1\}$, achieving near-lossless compression with $3\times$ memory reduction.

10.4 Mixture-of-Experts

Conditional computation through MoE enables scaling model capacity without proportional compute increase:

Sparse Gating: Shazeer et al. [2017] introduced sparsely-gated MoE layers. GShard [Lepikhin et al., 2020] scaled this to 600B parameters.

Switch Transformers: Fedus et al. [2022] simplified routing to top-1 expert selection, training trillion-parameter models efficiently.

Hybrid Architectures: Jamba [Lieber et al., 2024, Team et al., 2024] combines Transformer, Mamba, and MoE components in a hybrid architecture for improved efficiency.

10.5 Spanish Language Models

BERT-based models for Spanish include BETO [Cañete et al., 2020] trained on Spanish Wikipedia and corpora. MarIA [Gutiérrez-Fandiño et al., 2022] scaled Spanish BERT variants. RoBERTuito [Pérez et al., 2022] focused on Spanish social media text.

10.6 Normalization Techniques

Layer Normalization [Ba et al., 2016] stabilizes training by normalizing across features. RMSNorm [Zhang and Sennrich, 2019] simplifies this by removing mean centering. Recent work by Zhu et al. [2025] demonstrates that normalization can be replaced entirely by Dynamic Tanh (DyT), eliminating the need for statistics computation while maintaining or improving performance.

11 Experiments

11.1 Experimental Setup

11.1.1 Hardware

All experiments are conducted on a single NVIDIA Tesla P40 GPU (24GB VRAM, Pascal architecture) to validate efficiency claims under constrained resources.

11.1.2 Datasets

- **Pre-training:** RedPajama Spanish subset (35B tokens), streamed via HuggingFace datasets
- **Tokenizer:** SentencePiece [Kudo and Richardson, 2018] unigram model, 50K vocabulary
- **Evaluation:** Spanish GLUE equivalents (XNLI, PAWS-X, STS-B)

11.1.3 Baselines

1. BETO-Base [Cañete et al., 2020]: Spanish BERT, 110M parameters
2. MarIA-Base: Spanish RoBERTa variant
3. BitNet-BERT: Our reproduction of BitNet-style BERT
4. Standard Transformer encoder: Full-precision baseline

11.2 Implementation Details

Table 8: TITAN-BERT-ULTRA Experimental Configuration

Hyperparameter	**Value**
Model Architecture	
Hidden dimension (d)	1024
Physical layers	12
Recursive loops	2
Effective depth	24
Attention heads	16
Retention heads	8
MoE experts	4
Top-K routing	2
RK4 integration steps	2
Layer pattern	Stable 6-layer (Table 2)
Training Configuration	
Batch size	2 (effective 8 with accumulation)
Gradient accumulation	4 steps
Sequence length	512
Learning rate (base)	1×10^{-4}
Learning rate (ODE)	5×10^{-5}
Warmup ratio	10%
LR schedule	Cosine decay
Weight decay	0.01
Gradient clipping	1.0
Mixed precision	FP16 (GradScaler)
Checkpointing	
Rolling checkpoints	Every 500 steps, keep 3
Best model checkpoints	Top $K = 2$ by loss
CSV logging	Every training step
NaN detection	Enabled with debug logs

11.3 Results

Note: This section presents the experimental framework. Full empirical results are pending completion of training runs.

11.3.1 Pre-training Efficiency

We report training throughput, memory consumption, and loss curves comparing:

- Tokens per second across model variants
- Peak GPU memory with and without gradient checkpointing
- Convergence behavior of the hybrid architecture

11.3.2 Downstream Task Performance

Planned evaluations include:

- XNLI (cross-lingual natural language inference) accuracy
- PAWS-X (paraphrase identification) accuracy
- STS-B (semantic textual similarity) Pearson/Spearman correlation

12 Discussion

12.1 Architectural Synergies

The combination of diverse components yields potential synergies:

1. **Complementary inductive biases**: RetNet captures local patterns efficiently, Neural ODEs model smooth transformations, Mamba provides linear-complexity sequence modeling, and standard attention enables flexible global interactions.
2. **Quantization robustness**: DyT normalization's bounded outputs complement BitNet's quantized activations, reducing the accumulation of quantization error.
3. **Capacity scaling**: MoE enables parameter scaling without proportional compute increase, partially offsetting the overhead from ODE integration.

12.2 Limitations

1. **Implementation complexity**: The heterogeneous architecture requires careful engineering and debugging.
2. **Hyperparameter sensitivity**: Multiple component types introduce numerous hyperparameters requiring joint optimization.
3. **ODE computational overhead**: RK4 integration with $K = 2$ steps increases attention cost by $2\times$ for those layers.

4. **Training instability**: Combining aggressive quantization with recursive depth and ODE dynamics presents stability challenges. We mitigate this through the stable 6-layer pattern (Section 2), component-specific learning rates, and comprehensive NaN detection.

5. **Limited empirical validation**: Full-scale training and evaluation remain ongoing.

12.3 Stability Considerations

The stable 6-layer pattern addresses key instability sources:

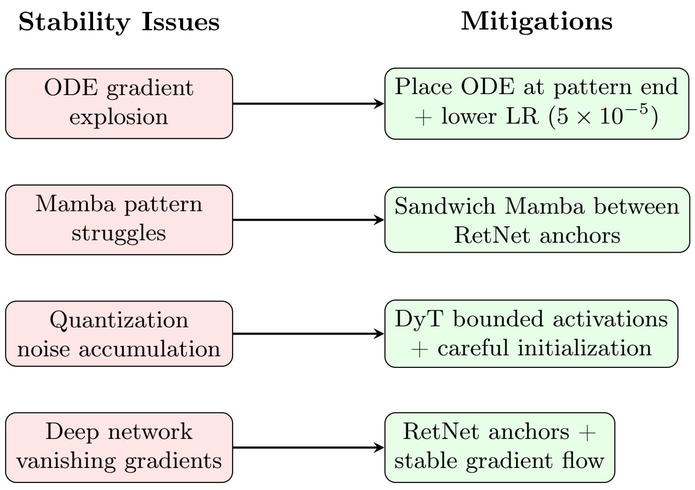
Stability Issues
Mitigations
ODE gradient explosion
Place ODE at pattern end + lower LR (5×10^{-5})
Mamba pattern struggles
Sandwich Mamba between RetNet anchors
Quantization noise accumulation
DyT bounded activations + careful initialization
Deep network vanishing gradients
RetNet anchors + stable gradient flow

Figure 5: Stability issues and their mitigations in the TITAN-BERT-ULTRA architecture.

12.4 Ablation Study Design

To understand component contributions, we propose ablations:

Table 9: Planned Ablation Configurations

Variant	**Mixer**	**BitNet**	**MoE**	**Recursive**
Full model	Hybrid	✓	✓	✓
Single mixer	Attn only	✓	✓	✓
FP16	Hybrid	×	✓	✓
Dense FFN	Hybrid	✓	×	✓
Non-recursive	Hybrid	✓	✓	×

12.5 Future Directions

1. **Learned mixer routing**: Replace fixed cyclic assignment with content-dependent mixer selection.
2. **Decoder extension**: Adapt the architecture for autoregressive generation tasks.
3. **Hardware kernels**: Develop fused CUDA kernels for BitLinear operations and ODE integration.
4. **Scaling laws**: Investigate how the hybrid architecture scales with compute budget.
5. **Multilingual training**: Extend beyond Spanish to evaluate cross-lingual transfer.

13 Conclusion

We presented TITAN-BERT-ULTRA, an experimental transformer encoder architecture that synthesizes recent advances in efficient sequence modeling. By combining BitNet b1.58 quantization [Ma et al., 2024], recursive depth processing, polymorphic attention mechanisms (RetNet [Sun et al., 2023], Neural ODEs [Chen et al., 2018], Mamba [Gu and Dao, 2023], and standard attention), sparse Mixture-of-Experts [Fedus et al., 2022], and Dynamic Tanh normalization [Zhu et al., 2025], we explore the frontiers of what is possible within constrained hardware budgets.

While this "Frankenstein" approach to architecture design is admittedly unconventional, it serves as a testbed for understanding how cutting-edge techniques interact and whether their benefits compose. Our preliminary analysis suggests that careful integration can yield memory-efficient models with diverse computational capabilities.

The architecture is specifically designed for Spanish NLP on modest hardware (24GB VRAM), demonstrating that sophisticated model designs remain accessible to researchers without massive compute resources. We hope this work inspires further exploration of hybrid architectures that challenge conventional design principles.

Acknowledgments

We acknowledge the foundational contributions of the research teams behind BitNet, RetNet, Neural ODEs, Mamba, Switch Transformers, and Dynamic Tanh. This work was conducted independently as an architectural exploration without institutional affiliation.

References
References

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., and Pérez, J. (2020). Spanish pre-trained BERT model and evaluation data. *LREC Workshop on PML4DC*.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. (2020). Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Courbariaux, M., Bengio, Y., and David, J.-P. (2015). BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Dupont, E., Doucet, A., and Teh, Y. W. (2019). Augmented neural ODEs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2018). FFJORD: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*.

Gu, A. and Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Gu, A., Goel, K., and Ré, C. (2021). Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

Gutiérrez-Fandiño, A., Armengol-Estapé, J., Pàmies, M., Llop-Palao, J., Silveira-Ocampo, J., Carrino, C. P., Gonzalez-Agirre, A., Armentano-Oller, C., Rodriguez-Penagos, C., and Villegas, M. (2022). MarIA: Spanish language models. *Procesamiento del Lenguaje Natural*, 68:39–60.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*.

Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP: System Demonstrations*.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. (2020). GShard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Li, F., Zhang, B., and Liu, B. (2016). Ternary weight networks. *arXiv preprint arXiv:1605.04711*.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, S., Belinkov, Y., Shalev-Shwartz, S., et al. (2024). Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., Dong, L., Wang, R., Xue, J., and Wei, F. (2024). The era of 1-bit LLMs: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. (2017). Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Pérez, J. M., Furman, D. A., Alemany, L. A., and Luque, F. M. (2022). RoBERTuito: A pre-trained language model for social media text in Spanish. In *LREC*.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). XNOR-Net: ImageNet classification using binary convolutional neural networks. In *European Conference on Computer Vision (ECCV)*.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. (2024). RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. (2023). Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

Team, A. (2024). Jamba-1.5: Hybrid transformer-mamba models at scale. *arXiv preprint arXiv:2408.12570*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wang, H., Ma, S., Dong, L., Huang, S., Wang, H., Ma, L., Yang, F., Wang, R., Wu, Y., and Wei, F. (2023). BitNet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. (2020). BigBird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang, B. and Sennrich, R. (2019). Root mean square layer normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang, H., Dauphin, Y. N., and Ma, T. (2019). Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations (ICLR)*.

Zhu, J., LeCun, Y., et al. (2025). Transformers without normalization. *arXiv preprint arXiv:2503.10622*.