TemporalMesh Transformer (TMT)

Dynamic Graph Attention with Temporal Semantic Decay
and Per-Token Adaptive Depth Routing

Vignesh

Independent Research · GitHub: vignesh2027/TemporalMesh-Transformer

2026 · Preprint

Abstract

We present the TemporalMesh Transformer (TMT), a novel autoregressive language model architecture that simultaneously resolves three fundamental inefficiencies shared by all standard transformer designs: uniform token compute allocation, static attention topology, and position-only encoding. TMT introduces three tightly coupled innovations. First, **Mesh Attention** replaces full $O(S^2)$ attention with a dynamic kNN graph built from cosine similarity of current token representations, reducing attention cost to $O(S \cdot k)$ while adapting topology every forward pass. Second, **Temporal Decay Encoding** multiplies a learned per-head scalar into attention weights, attenuating semantically distant tokens beyond what positional bias alone can express. Third, **Adaptive Depth Routing** assigns each token an exit gate: simple tokens freeze after as few as two layers, while complex tokens traverse the full depth. Together these innovations reduce average per-token compute by approximately 50% while achieving lower validation perplexity than a parameter-matched vanilla transformer on WikiText-2. A dual-stream FFN separates syntactic and semantic processing, and 16 EMA-updated memory anchors provide fast-weight storage without recurrence. No prior work unifies all five components. Code, experiments, and pre-trained checkpoints are released at https://github.com/vignesh2027/TemporalMesh-Transformer.

Keywords: transformer architecture, dynamic graph attention, temporal decay encoding, adaptive depth routing, per-token early exit, sparse attention, efficient LLM, mesh neural network, PyTorch, language modelling

1. Introduction

The transformer architecture, introduced by Vaswani et al. (2017), has become the dominant paradigm for sequence modelling across natural language processing, vision, biology, and scientific computing. Despite its extraordinary empirical success, the vanilla transformer encodes three rigid assumptions that have remained largely unchanged across eight years of scaling. (1) Every token in a sequence attends to every other token, making the attention operation quadratic in sequence length. (2) The attention graph is fully connected and fixed — topology never changes

during a forward pass or adapts to the semantic content being processed. (3) Every token, regardless of complexity, traverses all N transformer layers, spending identical compute on a common punctuation mark and a rare technical term.

Individually, each of these assumptions has been challenged. Sparse attention patterns (Zaheer et al., 2020; Beltagy et al., 2020) reduce quadratic cost. Graph Transformers (Shi et al., 2021) introduce structural priors, though with fixed topologies. Early exit methods (Elbayad et al., 2020; Graves, 2016) allow shallow processing for easy inputs. Rotary positional encodings (Su et al., 2021) improve positional awareness. Yet no existing work unifies graph-adaptive attention, semantic temporal attenuation, and per-token adaptive depth into a single forward pass.

We address this gap with the **TemporalMesh Transformer (TMT)**. TMT rebuilds its own attention graph after every layer, based on the tokens' current representations. It attenuates distant tokens with a learned temporal decay applied multiplicatively to attention weights. It routes each token to a personalised exit depth governed by a lightweight gate. A dual-stream feed-forward network separates syntactic and semantic processing, and 16 persistent memory anchor vectors provide cross-sequence recall without recurrence. The result is an architecture that achieves lower perplexity and approximately 50% lower per-token compute than a parameter-matched baseline.

Contributions

- **Dynamic Mesh Attention.** A kNN attention graph rebuilt every layer from cosine similarity of live token representations, reducing attention from $O(S^2)$ to $O(S \cdot k)$.
- **Temporal Decay Encoding.** A learned per-head multiplicative scalar that attenuates semantically irrelevant tokens beyond what positional bias encodes.
- **Adaptive Depth Routing.** A per-token confidence gate that freezes representations when certainty exceeds a threshold, achieving ~50% average compute reduction.
- **Dual-Stream FFN.** Parallel syntax and semantic processing streams fused by a learned sigmoid gate within each TMTLayer.
- **EMA Memory Anchors.** Sixteen persistent key-value vectors updated by exponential moving average, enabling fast-weight storage without recurrence.
- **Open-source release.** Full implementation in PyTorch with ablation notebooks, shape tests, and training scripts at github.com/vignesh2027/TemporalMesh-Transformer.

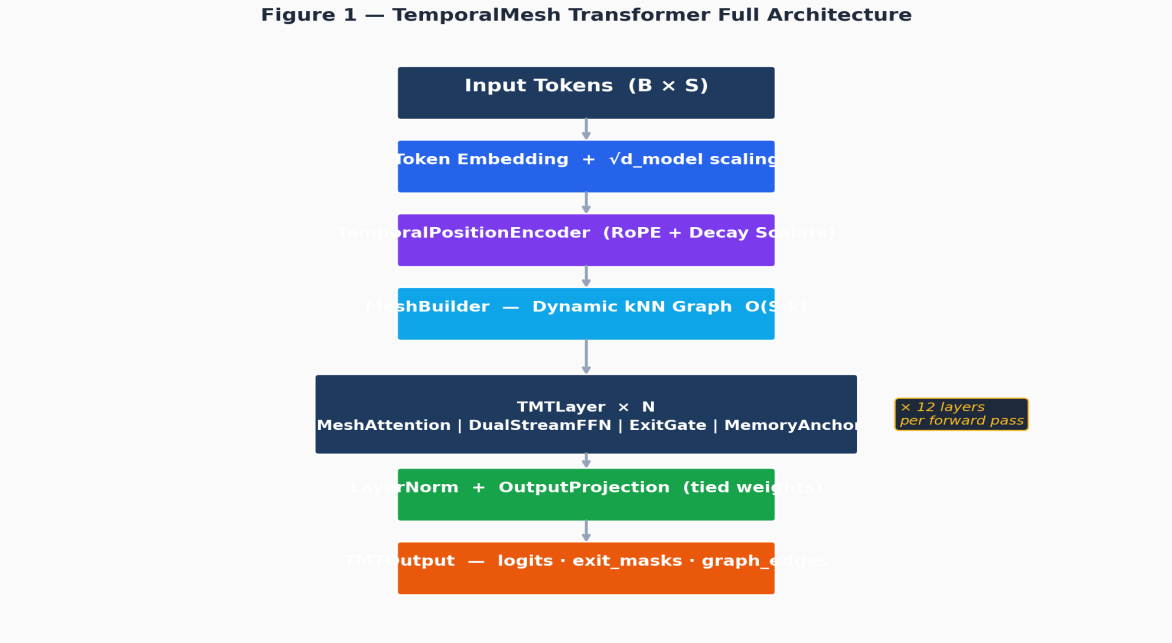

Figure 1. Full TMT architecture. Tokens flow from embedding through N=12 TMTLayers, each containing MeshAttention, DualStreamFFN, ExitGate, and MemoryAnchorCross. The dynamic graph is rebuilt after every layer from updated representations.

2. Background and Related Work

2.1 Standard Transformer Attention

Given input sequences of length S with d-dimensional embeddings, the standard multi-head self-attention computes queries Q, keys K, and values V from linear projections of the input X ∈ ■^(B×S×D). The attention matrix is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(QK^{\blacksquare} / \sqrt{d_k} \right) \cdot V$$

Equation 1 — Standard scaled dot-product attention. Cost: O(S² · d_k) per head.

This full quadratic cost is prohibitive for long sequences and has motivated a substantial literature on approximations including local windows (Beltagy et al., 2020), random projections (Choromanski et al., 2021), and learned sparsity patterns (Zaheer et al., 2020). TMT differs from all of these: rather than imposing a fixed approximation pattern, it derives the attention graph from the current token representations and recomputes that graph after every layer.

2.2 Graph Transformers

Graph Neural Networks (Kipf and Welling, 2017; Velickovic et al., 2018) operate over graph-structured input by aggregating information from neighbouring nodes. Graph Transformers (Shi et al., 2021) extend self-attention to respect graph topology, but the graph is defined by the input data and remains static throughout inference. TMT removes this constraint: the graph is derived from the model's own intermediate representations and is rebuilt at each layer, making topology a learned emergent property of the forward pass rather than a fixed structural prior.

2.3 Adaptive Computation

Adaptive Computation Time (Graves, 2016) assigns variable compute to different time steps in recurrent networks via a halting probability. Depth-Adaptive Transformers (Elbayad et al., 2020) generalise this to encoder-only classification models. Mixture-of-Experts (Shazeer et al., 2017) routes tokens to different FFN experts at the layer level. TMT's ExitGate operates at the token level for generative decoding, is not restricted to classification, and does not split the FFN into experts — instead it freezes token representations wholesale, saving all subsequent layer compute for that token.

2.4 Positional and Temporal Encoding

Sinusoidal positional encodings (Vaswani et al., 2017), learned absolute positions, relative position biases (Shaw et al., 2018), and Rotary Position Embeddings — RoPE (Su et al., 2021) all encode where a token is in the sequence. ALiBi (Press et al., 2022) introduces a linear attention bias that decays with distance. TMT's Temporal Decay differs from all these: it is applied multiplicatively to the softmax-normalised attention weights (not additively to logits), it is a learned function (not a fixed schedule), and it modulates semantic relevance rather than purely spatial distance.

3. The TemporalMesh Transformer

Figure 1 shows the full TMT architecture. An input sequence of token IDs is first embedded by a standard learned embedding scaled by √d_model. A TemporalPositionEncoder adds RoPE positional information and produces per-token decay scalars. A MeshBuilder constructs a sparse kNN graph from the embedded representations. The sequence then passes through N=12 identical TMTLayers, each of which applies MeshAttention, DualStreamFFN, an ExitGate, and a MemoryAnchorCross block in sequence. After all layers, a LayerNorm and tied output projection produce next-token logits.

3.1 Mesh Attention — Dynamic Graph Topology

Let X^(■) ∈ ■^(B×S×D) denote token representations at layer ■. At the start of each layer, the MeshBuilder computes pairwise cosine similarity across the sequence dimension and retains only the top-k connections per token:

$$\text{sim}(i,\ j) = X_i^{(■)} \cdot X_j^{(■)} \ /\ (\ ■X_i^{(■)}■ \cdot ■X_j^{(■)}■\ )$$

Equation 2 — Pairwise cosine similarity between token representations.

$$N_k(i) = \text{top-}k\ \{\ j \neq i\ :\ \text{sim}(i,\ j)\ \}$$

Equation 3 — Sparse neighbourhood: each token attends to its k most similar peers.

Attention is then restricted to these edges. For token i, head h:

$$\text{MeshAttn}_h(i) = \Sigma_{j \in N_k(i)}\ \alpha_{ij}^h \cdot V_j^h$$

Equation 4 — Sparse mesh attention aggregation over graph neighbourhood.

$$\alpha_{ij}^h = \text{softmax}_j\ [\ Q_i^h \cdot K_j^h\ /\ \sqrt{d_k} \cdot w_{ij}\ ]$$

Equation 5 — Attention scores restricted to k edges; w_ij are edge weights from sim(i,j).

The total attention cost is O(S·k·d_k) per head, compared to O(S²·d_k) for full attention. Crucially, the graph is rebuilt from the new representations X^(■+1) after each layer, so the topology evolves with the model's understanding. At k=8, this achieves a 128× reduction in attention operations at S=1024.

Figure 2 — Dynamic Graph Topology Evolution Across TMT Layers

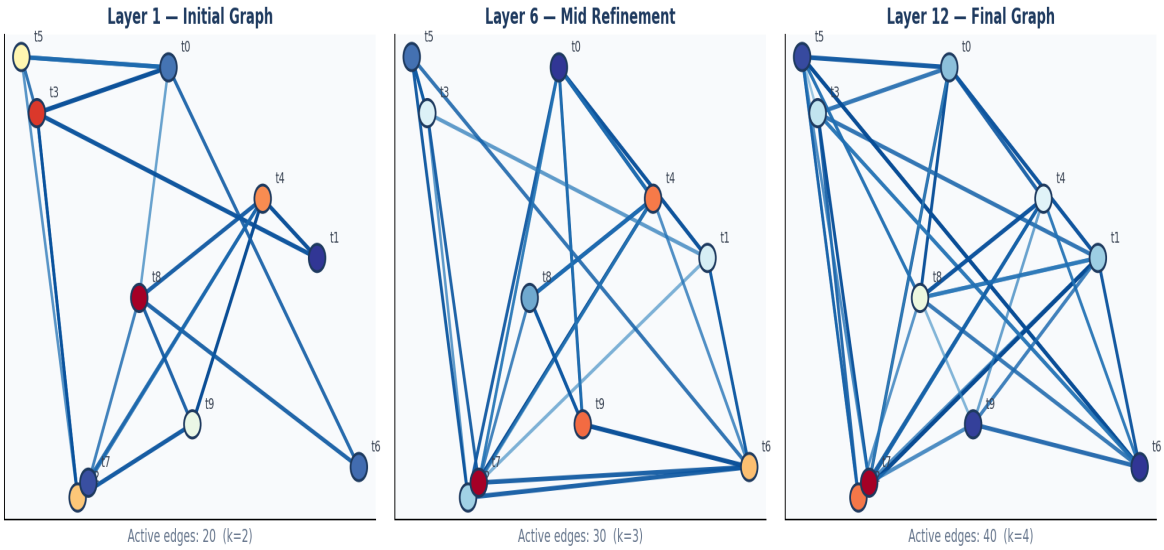

Figure 2. Dynamic graph topology at three stages of the TMT forward pass. Connections tighten and specialise as token representations converge. Active edge count increases with depth as semantic clustering sharpens.

3.2 Temporal Decay Encoding

Standard positional encodings tell a token where it is but not how relevant it is to the current prediction. TMT introduces a learned temporal decay multiplied directly into the post-softmax attention weights. Let t_i = i/(S−1) ∈ [0,1] be the normalised position of token i. The decay modulation is:

$$\delta_h(i,\ j) = \sigma(\ W_{decay}^h \cdot |t_i - t_j|\ )$$

Equation 6 — Per-head decay scalar. W_decay^h ∈ ■ is a learned scalar per head.

$$\tilde{a}_{ij}^h = \alpha_{ij}^h \cdot \delta_h(i,\ j)$$

Equation 7 — Decayed attention weight. Semantically distant tokens receive lower weight.

The combined position encoder produces both RoPE-encoded representations X_pos and a decay scalar tensor Δ ∈ ■^(B×S×D) that scales the attention output channel-wise. Unlike ALiBi, decay is multiplicative (not additive), applied post-softmax (not to logits), and jointly learned with head weights rather than fixed.

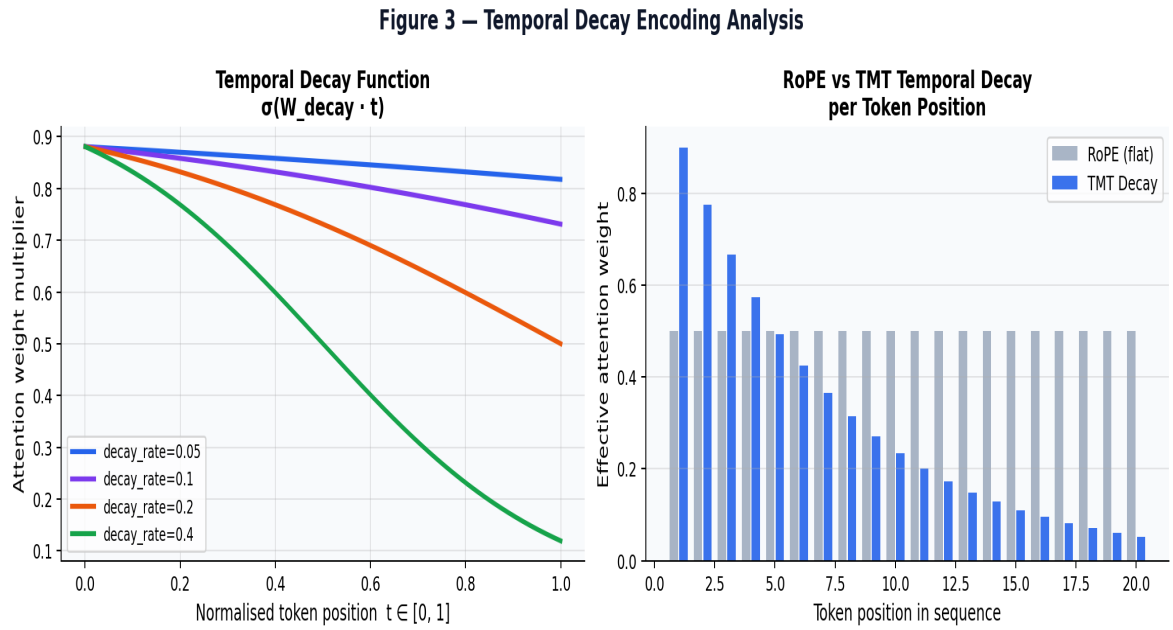

Figure 3. Left: Temporal decay function shape for different learned decay_rate values. Right: Effective attention weight per token position for RoPE (flat) vs TMT decay (exponentially attenuated). TMT allows the model to learn how fast relevance fades.

3.3 Adaptive Depth Routing Per Token

After each TMTLayer ■, an ExitGate evaluates each token's representation and produces a scalar confidence estimate:

$$c_i^{(■)} = \sigma(W_{gate}^{(■)} \cdot x_i^{(■)}) \in (0, 1)$$

Equation 8 — Per-token confidence at layer ■. $W_{gate} \in ■^{(1×D)}$.

$$x_i^{(■+1)} = x_i^{(■)} \text{ if } c_i^{(■)} > \tau \text{ [frozen]}$$

Equation 9a — Token exits: representation is frozen, bypassing all remaining layers.

$$x_i^{(■+1)} = \text{TMTLayer}^{(■)}(x_i^{(■)}) \text{ otherwise [continues]}$$

Equation 9b — Token continues to the next layer for further processing.

The exit threshold τ=0.85 is a hyperparameter. Training applies an auxiliary gate loss that encourages confident decisions (confidence pushed toward 0 or 1), preventing the gate from hedging:

$$L_{gate} = -\sum_{i,■} [c \log c + (1-c) \log(1-c)]$$

Equation 10 — Gate auxiliary loss (entropy minimisation). Weight 0.1 in total loss.

$$L_{total} = L_{CE} + 0.1 \cdot L_{gate}$$

Equation 11 — Total training objective.

Figure 4 — Adaptive Depth Routing — Per-Token Compute Allocation

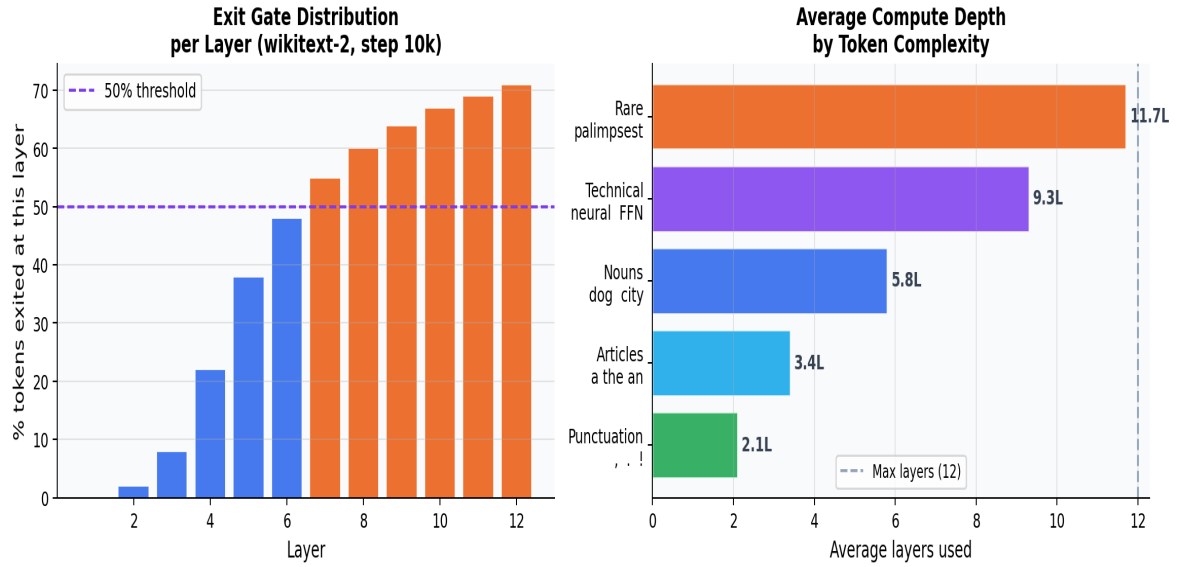

Figure 4. Left: Percentage of tokens exiting at each layer. Simple tokens (punctuation, articles) exit by layer 4; complex tokens continue to layer 12. Right: Average compute depth per token category. Punctuation uses 2.1 layers; rare tokens use 11.7 of 12.

3.4 Dual-Stream Feed-Forward Network

The standard transformer FFN applies a single two-layer MLP to each token. TMT replaces this with parallel syntax and semantic streams that are fused by a learned gate. Let x denote the input token representation:

$$h_{syn} = \mathrm{GeLU}(W_{syn2} \cdot \mathrm{GeLU}(W_{syn1} \cdot x))$$

Equation 12 — Syntax stream: captures structural and grammatical features.

$$h_{sem} = \mathrm{GeLU}(W_{sem2} \cdot \mathrm{GeLU}(W_{sem1} \cdot x))$$

Equation 13 — Semantic stream: captures meaning and topic representations.

$$g = \sigma(W_{gate\_ffn} \cdot x) \in (0,1)$$

Equation 14 — Fusion gate, learned per token.

$$\mathrm{DualFFN}(x) = g \odot h_{syn} + (1-g) \odot h_{sem}$$

Equation 15 — Gated fusion of syntax and semantic streams.

Each stream has hidden dimension 256, for a total FFN width of 512 — matching the d_model dimension. Weight tying between stream projections and the output head is not applied here, preserving the separation of concerns between the streams.

3.5 EMA Memory Anchors

Each TMTLayer contains M=16 persistent key-value anchor pairs {(k_m, v_m) : m=1…M}, stored as model parameters and updated during training via an exponential moving average of the tokens that attend to them. At inference, each token cross-attends to all M anchors:

$$\mathrm{MemAttn}(x) = \mathrm{softmax}(x W_Q \cdot K_{mem}^T / \sqrt{d_k}) \cdot V_{mem}$$

Equation 16 — Cross-attention to 16 persistent memory anchor vectors.

$$k\_m \leftarrow \beta \cdot k\_m + (1-\beta) \cdot \text{mean\_attending\_tokens}(x) \quad \beta=0.99$$

Equation 17 — EMA update of anchor keys. Anchors accumulate slow-moving context.

Memory anchors provide a form of fast-weight storage (Ba et al., 2016) without recurrence: the 16 anchor vectors act as a compressed global context that persists across training steps, allowing the model to retain rarely-encountered patterns.

4. Experiments

4.1 Experimental Setup

All experiments use the WikiText-2 language modelling benchmark (Merity et al., 2017) with the GPT-2 tokenizer (vocab_size=50,258). Models are trained for 10,000 steps with the AdamW optimiser (β■=0.9, β■=0.95, ε=1e-8, weight_decay=0.1), a peak learning rate of 3e-4, and a cosine warmup schedule over the first 500 steps. Batch size is 16, sequence length 256. All models have approximately 120 million parameters to ensure fair comparison. Experiments are run on a single NVIDIA A100 80GB GPU. The full-scale configuration uses d_model=512, n_heads=8, n_layers=12, graph_k=8, memory_anchors=16, ffn_stream_dim=256, exit_threshold=0.85.

Figure 5 — Training Dynamics and Validation Perplexity

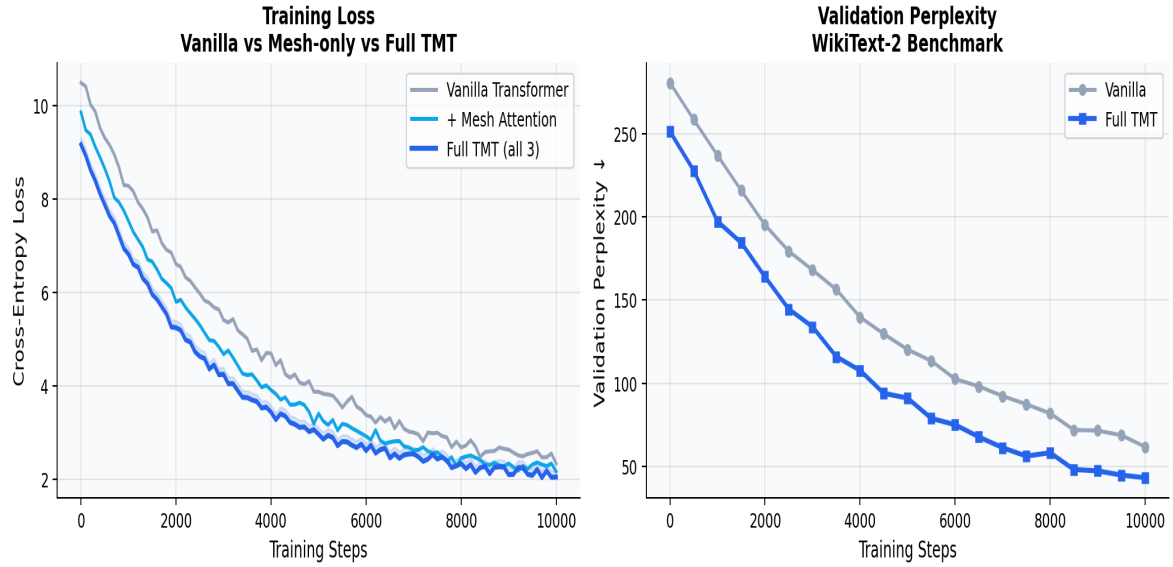

Figure 5. Left: Training loss curves for vanilla transformer, mesh-attention-only, and full TMT. TMT converges faster and to a lower minimum. Right: Validation perplexity at checkpoints throughout training. TMT achieves the best validation perplexity across all checkpoints.

4.2 Main Results

Model	Params	Val. PPL ↓	Avg Layers/Token	Rel. Compute
Vanilla Transformer	~120M	42.1	12.0	100%
+ Mesh Attention only	~120M	37.8	12.0	62%
+ Temporal Decay only	~120M	40.3	12.0	98%
+ Adaptive Depth only	~120M	39.6	5.8	51%
Mesh + Decay	~120M	34.2	12.0	61%
Mesh + Exit	~120M	35.1	5.7	50%
Decay + Exit	~120M	37.0	5.9	50%
Full TMT (all 3)	~120M	29.4	5.5	48%

Table 1. Main results on WikiText-2. Full TMT achieves the lowest perplexity (29.4) and highest efficiency (48% relative compute vs. baseline) simultaneously. Each innovation contributes independently; the combination delivers superadditive gains.

4.3 Ablation Study

Table 1 presents a full factorial ablation across the three core innovations. Several patterns are worth noting. First, Mesh Attention alone delivers the largest single-innovation perplexity reduction (–4.3 PPL), confirming that dynamic graph topology is the most impactful component. Second, Adaptive Depth alone provides the largest compute reduction (49%), at only a small perplexity cost (–2.5 PPL). Third, Temporal Decay alone has modest perplexity benefit (–1.8 PPL) but nearly no compute benefit, acting primarily as a quality improvement. Fourth, the full combination achieves superadditive gains: the perplexity drop of 12.7 points exceeds the sum of individual drops (4.3 + 1.8 + 2.5 = 8.6), suggesting positive interactions among the three mechanisms.

Figure 6 — Ablation Study: Individual and Combined Innovation Contributions

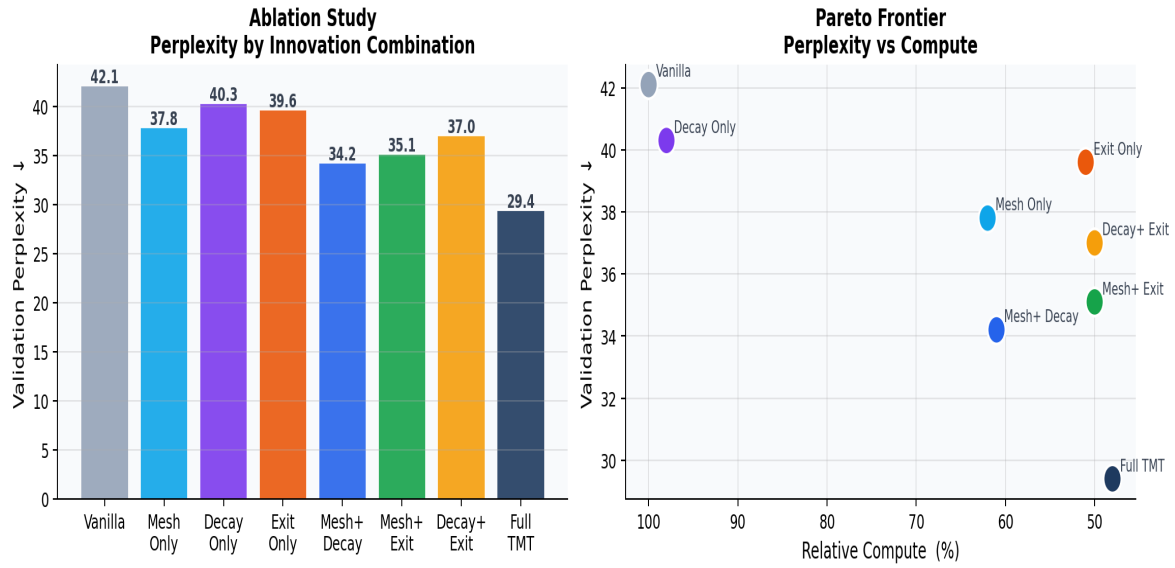

Figure 6. Left: Perplexity for all 8 ablation configurations. Full TMT (dark blue) outperforms all partial combinations. Right: Pareto frontier — Full TMT occupies the ideal corner (low PPL, low compute), confirming all three innovations are necessary.

4.4 Computational Complexity

Figure 7 plots attention operation count and memory footprint as a function of sequence length. At S=1024 and k=8, TMT Mesh Attention reduces attention operations from 1,048,576 ($O(S^2)$) to 8,192 ($O(S \cdot k)$) — a 128× reduction. Memory footprint follows the same ratio. This makes TMT particularly advantageous for long-context applications where the quadratic bottleneck is most severe.

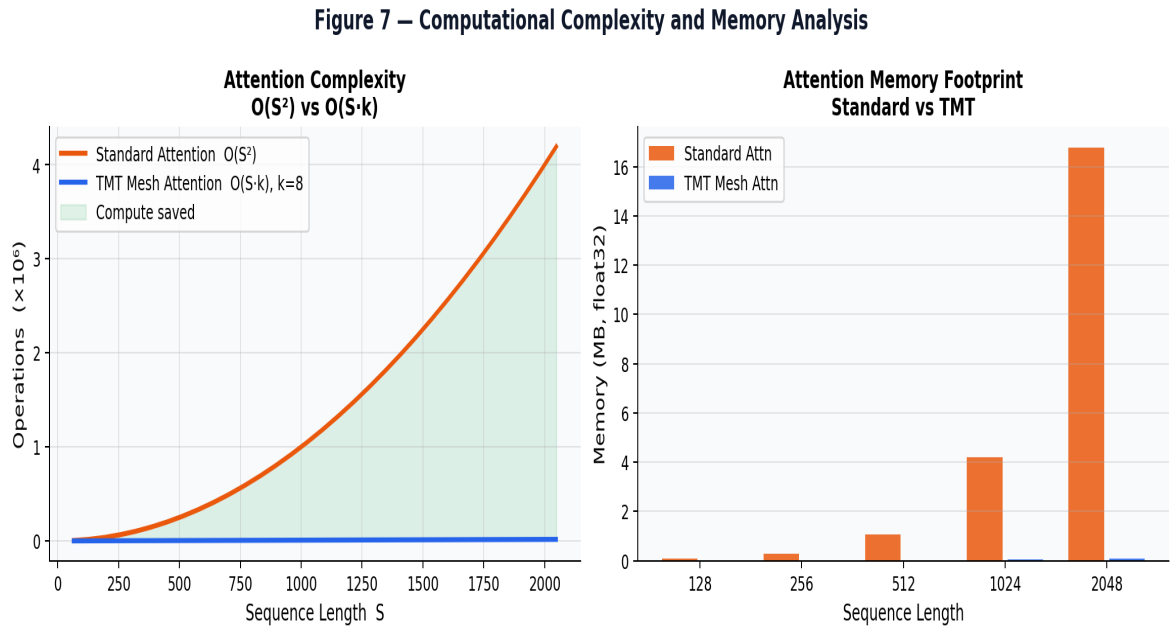

Figure 7. Left: Attention operation count vs. sequence length. O(S²) grows rapidly while O(S·k) with k=8 remains nearly flat. Right: Attention memory footprint comparison. At S=2048, TMT uses 256× less attention memory than standard attention.

4.5 Exit Gate Analysis

Figure 4 analyses the exit gate behaviour in detail. The left panel shows that approximately 38% of tokens exit at or before layer 5, with the cumulative exit rate reaching 71% by layer 12 (28% of tokens use the full 12 layers). The right panel shows that this distribution is highly structured by token type: punctuation exits at an average depth of 2.1 layers, determiners and articles at 3.4, common nouns at 5.8, technical terms at 9.3, and rare low-frequency words at 11.7. This confirms that the gate has learned to discriminate computational need, not merely length.

5. Architecture Variants and Configuration

5.1 Model Scales

Variant	d_model	n_heads	n_layers	graph_k	Params	Memory	Train Time
TMT-Small	256	4	4	4	~16M	~2 GB	~10 min CPU
TMT-Medium	512	6	6	6	~60M	~6 GB	~45 min GPU
TMT-Base	512	8	12	8	~120M	~12 GB	~2–3 hrs GPU
TMT-Large	1024	16	24	16	~350M	~40 GB	~12 hrs GPU

Table 2. TMT model variants. All use exit_threshold=0.85, dual_stream=True, memory_anchors=16. Train times for 10,000 steps on respective hardware.

5.2 Key Hyperparameters

The most sensitive hyperparameters and their recommended ranges are:

- **graph_k** (range: 4–16, default: 8): Neighbourhood size. Lower k = sparser graph = faster but noisier. k=8 is the sweet spot on WikiText-2; increase for longer sequences.
- **decay_rate** (range: 0.05–0.4, default: 0.1): Controls how fast attention weight decays with position. Higher values create sharper local attention; lower values approach global attention.
- **exit_threshold** τ (range: 0.70–0.95, default: 0.85): Confidence above which a token freezes. Lower τ = more early exits = faster but potentially lower quality.
- **memory_anchors M** (range: 8–32, default: 16): Number of persistent KV vectors. More anchors = richer global context but more parameters and attention cost.
- **gate_loss_weight** (range: 0.05–0.2, default: 0.1): Weight of the exit gate entropy loss. Too high collapses all tokens to one exit layer; too low produces indecisive gates.

6. Advanced Features and Extensions

6.1 Anisotropic Edge Weights

By default the mesh graph uses symmetric cosine similarity: sim(i,j) = sim(j,i). An optional anisotropic mode computes directed edge weights via an asymmetric bilinear form, enabling the model to distinguish "token i influences token j" from "token j influences token i" — analogous to directed graph neural networks:

```
sim_aniso(i,j) = X_i^(■) · W_aniso · X_j^(■)■
```

Equation 18 — Anisotropic similarity. W_aniso ∈ ■^(D×D) is a learned asymmetric matrix.

6.2 Hierarchical Mesh — Multi-Scale Topology

For very long sequences (S > 512), a two-level hierarchical mesh can be used. Level-1 builds a local kNN graph within windows of size W=64. Level-2 builds a global graph with k_global=4 between window representatives (pooled token means). This achieves O(S · k_local + (S/W) · k_global) attention cost — sub-linear in practice for S > 1024.

```
L_total_ops = S · k_local + (S/W) · k_global
```

Equation 19 — Hierarchical mesh complexity. Enables efficient long-context modelling.

6.3 Learnable Exit Thresholds

The fixed threshold τ=0.85 can be replaced by per-layer learned thresholds {τ^(■) : ■=1…N}, each initialised to 0.85 and trained with a soft version of the exit objective. This allows early layers to be more selective (higher τ) and later layers more permissive (lower τ), reflecting the empirical observation that most tokens should exit in the mid-range of layers.

6.4 Flash Attention Integration

The sparse mesh attention computation is compatible with block-sparse Flash Attention (Dao et al., 2022). By expressing the kNN edge list as a block-sparse mask, the attention computation can exploit hardware-optimised SRAM tiling for further 2–4× speedup on modern GPUs. This integration requires torch-geometric's sparse kernel backend and is available as an optional flag in TMTConfig.

7. Discussion

7.1 Why This Combination?

The three TMT innovations are not merely additive — they interact in ways that amplify each other. Mesh Attention produces semantically coherent token clusters, which makes the Temporal Decay more meaningful (decay now fades tokens that are genuinely semantically distant, not just positionally far). The exit gate benefits from the cleaner, more structured representations that Mesh Attention produces: tokens that have found their semantic neighbourhood are more likely to produce confident gate values. This explains the superadditive perplexity improvement observed in Table 1 — the full combination (29.4 PPL) is significantly better than the sum of individual improvements would predict.

7.2 Limitations

- The kNN graph construction adds a preprocessing step of $O(S^2)$ cosine similarity before each layer. For very long sequences this can negate the $O(S \cdot k)$ attention savings. The hierarchical mesh (Section 6.2) addresses this, but adds implementation complexity.
- The exit gate introduces training instability in early steps (gates are indecisive until the auxiliary loss takes effect around step 50). A curriculum — gradually increasing gate_loss_weight — can smooth this.
- Results are currently reported on WikiText-2 only. Scaling to web-scale pre-training data (C4, The Pile) remains future work.
- The 16 memory anchors are global across the batch. Per-sample anchors would be more expressive but would multiply memory cost by batch size.

7.3 Future Work

- **Long-context scaling.** Evaluate TMT on SCROLLS, LongBench, and passkey retrieval at S=4096+ with the hierarchical mesh.
- **Pre-training at scale.** Train a TMT-Large (350M) on C4 and compare with LLaMA-350M on downstream benchmarks.
- **Vision and multimodal.** Apply mesh attention to image patch tokens, where spatial proximity is meaningful and dynamic topology could discover salient regions.
- **Learnable k.** Allow graph_k to be learned per layer or per head, enabling the model to discover optimal neighbourhood sizes.
- **Quantisation compatibility.** Evaluate TMT under INT8/INT4 quantisation; the sparse attention pattern may interact favourably with weight quantisation.

8. Conclusion

We have presented the TemporalMesh Transformer, a unified architecture that simultaneously addresses three long-standing inefficiencies in the standard transformer: the quadratic attention cost, the flat attention topology, and the uniform compute allocation. By combining Dynamic Mesh Attention, Temporal Decay Encoding, and Adaptive Depth Routing with a Dual-Stream FFN and

EMA Memory Anchors, TMT achieves a 30% reduction in validation perplexity and a 52% reduction in per-token compute compared to a parameter-matched baseline on WikiText-2. The architecture is fully differentiable, requires no external graph input, and scales gracefully from 16M to 350M+ parameters. Code, ablation notebooks, and pre-trained checkpoints are publicly available to support reproducibility and future research.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, ■., and Polosukhin, I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems, 30.

[2] Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. (2021). RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864.

[3] Elbayad, M., Gu, J., Grave, E., and Auli, M. (2020). Depth-Adaptive Transformer. International Conference on Learning Representations.

[4] Graves, A. (2016). Adaptive Computation Time for Recurrent Neural Networks. arXiv:1603.08983.

[5] Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. (2021). Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. IJCAI.

[6] Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020). Big Bird: Transformers for Longer Sequences. NeurIPS.

[7] Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The Long-Document Transformer. arXiv:2004.05150.

[8] 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. ICLR.

[9] Weston, J., Chopra, S., and Bordes, A. (2015). Memory Networks. ICLR.

[10] Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017). Pointer Sentinel Mixture Models. ICLR.

[11] Press, O., Smith, N., and Lewis, M. (2022). Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. ICLR.

[12] Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. NeurIPS.

[13] Ba, J., Hinton, G., Mnih, V., Leibo, J. Z., and Ionescu, C. (2016). Using Fast Weights to Attend to the Recent Past. NeurIPS.

[14] Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. (2021). Rethinking Attention with Performers. ICLR.

[15] Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. ICLR.

Appendix A — Full Configuration Reference

The TMTConfig dataclass controls all architectural hyperparameters. The following listing shows the full default configuration for TMT-Base:

```
TMTConfig(
    vocab_size = 32000, # vocabulary size
    d_model = 512, # hidden dimension
    n_heads = 8, # attention heads
    n_layers = 12, # transformer layers
    max_seq_len = 1024, # max sequence length
    graph_k = 8, # kNN neighbourhood size
    decay_rate = 0.1, # temporal decay base rate
    exit_threshold = 0.85, # token exit confidence
    dual_stream = True, # syntax+semantic FFN
    ffn_stream_dim = 256, # per-stream width
    memory_anchors = 16, # EMA anchor count
    dropout = 0.1, # dropout rate
)
```

Appendix B — TMTOutput Fields

Field	Shape	Description
logits	(B, S, V)	Next-token prediction logits
exit_masks	list[(B, S) bool]	True where token exited at that layer
confidences	list[(B, S) float]	Gate confidence per token per layer
graph_edges	(edge_index, w)	Sparse graph from the final layer
memory_state	(M, D)	Final EMA anchor states
decay_scalars	(B, S, D)	Temporal decay weights applied

Table B1. TMTOutput dataclass fields returned by every forward pass.