DynaTime Attention for Long-Sequence Time Series Forecasting

Jevon Twitty, Vinh Pham

Advisor: Shihao Yang, PhD Student: Jiecheng Lu

Georgia Tech College of Engineering
H. Milton Stewart School of Industrial and Systems Engineering

Introduction & Motivation

Time series forecasting fundamentally requires comparing the present to the past while accounting for *the time that passed in between*. Standard/Linear attention computes dot-products between static query and key endpoints, ignoring intermediate temporal events.

This structural limitation makes it difficult for models to capture shifting temporal dynamics, localized phase changes, and non-stationary trends that occur between observations.

We introduce **DynaTime Attention (DynaTime)** [1] to solve this. Instead of a standard retrieval mechanism, DynaTime allocates a dedicated neural pathway to model the **Temporal Interval** itself. By actively filtering scores based on events between observations, DynaTime natively maps complex, shifting dependencies in long-horizon multivariate forecasting across backbones (iTransformer, PatchTST, Autoformer) [2–5].

The Three Causal Regions

For a score between a current position t and a context position $i \leq t$, the boundaries induce three causally meaningful regions:

1. **Current Prefix ($X_{1:t}$):** Everything visible at the read endpoint.
2. **Historical Prefix ($X_{1:i}$):** Everything visible when the candidate memory was written.
3. **Temporal Interval ($X_{i+1:t}$):** The intervening events between writing and reading (e.g., trend changes, phase shifts, and shocks).

The Limits of Bilinear Scores

Existing linear variants rely on endpoint bilinear features, which cannot easily represent fixed functions of the between-boundary interval. A score class that exposes all three views can properly allocate neural capacity to each part of the temporal pair.

DynaTime: Tri-Region Scoring

DynaTime captures all three temporal regions by utilizing a nested fast-weight parameterization. The resulting autoregressive causal score takes the compact form:

$$A^{\text{DynaTime}}_{t,i,h} = F^{q}_{t,h}(x_t; X_{1:t}) F^{k}_{i,h}(x_i; X_{1:i})^{\top} F^{\Delta}_{t,i,h}(X_{i+1:t}; x_t) \tag{1}$$

- **Context-Aware Endpoints (F^q, F^k):** The query tower processes the full history $X_{1:t}$, while the key tower processes the past reference $X_{1:i}$.
- **Interval Fast MLP (F^Δ):** A dynamic filter capturing localized temporal decay and phase shifts between i and t, while still admitting linear-time recurrence.

Computational Complexity

- **Standard Softmax Attention:** $O(N^2 \cdot d)$ time and memory complexity, which scales prohibitively on long forecasting horizons.
- **DynaTime Recurrence:** $O(N \cdot d^2)$ time and $O(d^2)$ memory footprint via associative state updates, enabling fast, linear-time scaling.

Architecture & Linear-Time Equivalence

Fast-Weight Generation

Historical data enters the first stage through a cumulative historical state $C_u := X_{1:u}^{\top} X_{1:u}$, which becomes the hidden-layer weight of a context-instantiated MLP:

$$F^{q}_{t,h} = H(x_t, C_t) W^{(2)}_{q,h}, \quad F^{k}_{i,h} = H(x_i, C_i) W^{(2)}_{k,h} \tag{2}$$

The Interval Factor

For R tracks, we define time-step increments $d_{u,h} = \lambda \phi_+(x_u W_{\Delta,h})$ and historical potentials $p^{\Delta}_{t,h} = \sum_{u \leq t} d_{u,h}$. The interval branch forms an exponential-basis MLP over the temporal segment:

$$F^{\Delta}_{t,i,h} = \sum_{r=1}^{R} \omega_{t,h,r} \exp\left(- \sum_{u=i+1}^{t} d_{u,h,r} \right) \tag{3}$$

Linear-Time Recurrence

Setting $\gamma_{t,h,r} = \exp(-p^{\Delta}_{t,h,r})$ allows the interval factor to be absorbed into the endpoints, reducing each track to an ordinary causal linear mixer:

$$S_{t,h,r} = S_{t-1,h,r} + \gamma^{-1}_{t,h,r} F^{k}_{t,h}{}^{\top} v^{(2)}_{t,h} \tag{4}$$

This is exactly equivalent to the explicit tri-region score and scales linearly.

Parameter Budget Alignment

To strictly isolate expressivity gains from parameter scale, parameter counts per Transformer block are aligned to $12d^2$ across all models:

- **Standard Block:** $4d^2$ (Attention) + $8d^2$ (FFN/MLP) = $12d^2$.
- **DynaTime Block:** $8d^2$ (DynaTime Attention) + $4d^2$ (Compressed FFN) = $12d^2$.

This guarantees that performance improvements stem entirely from dynamic routing rather than increased model capacity.

Theoretical Support

Theoretical Component	Design Consequence
Prefix-Difference Limit	Endpoint bilinear scores alone underuse causal regions induced by (t, i).
Context-Aware Fast MLPs	Two-branch DynaTime learns rich historical macro-trends; ReLU/GLU towers approximate shifting dependencies over long horizons.
Explicit Interval Branch	Additive prefix potentials yield exact interval dependence, preserving linear-time computation.
Tri-Region Family	Full module provides a neural score-search space dense on compact feasible state domains.

Dense Patches vs. Noisy Steps

Time series forecasting requires models to map both **global contextual shifts** and **local temporal distances**. We discover that the optimal DynaTime architecture depends entirely on the token representation density:

- **Dense Semantic Representation (iTransformer & PatchTST):** When routing highly dense, information-rich tokens, models require **Full-Rank Dynamic Specialization** (e.g., DeltaNet [6]) to map complex inter-token dependencies without structural loss.
- **Noisy Point-wise Representation (Autoformer):** When operating on raw time steps, models require **Double Linear Attention**, utilizing a strict low-rank bottleneck ($r = 16$) with learned interval phase and decay to structurally filter temporal noise.

The Asymmetric Capacity Insight

Our results demonstrate that modeling dense semantic tokens requires dynamic, data-dependent capacity (DeltaNet β-gating), whereas modeling noisy point-wise steps strictly requires structural bottlenecking (Rank-16 Decay).

Experimental Results (MSE Only)

Model		iTransformer		PatchTST		Autoformer	
		DynaTime	Baseline	DynaTime	Baseline	DynaTime	Baseline
Dataset	Len	MSE	MSE	MSE	MSE	MSE	MSE
ETT	96	**0.295**	0.301	0.296	**0.294**	0.397	**0.381**
	192	**0.356**	0.362	0.356	**0.352**	**0.444**	0.456
	336	**0.404**	0.411	0.403	**0.397**	**0.450**	0.488
	720	**0.451**	0.461	0.447	**0.441**	0.501	**0.492**
	Average	**0.377**	0.384	0.376	**0.371**	**0.448**	0.455
Weather	96	**0.171**	0.182	**0.180**	0.183	0.344	**0.302**
	192	**0.220**	0.232	**0.227**	0.229	**0.304**	0.306
	336	**0.277**	0.288	**0.281**	0.283	**0.370**	0.375
	720	**0.355**	0.362	**0.289**	0.359	**0.629**	0.704
	Average	**0.256**	0.266	**0.244**	0.263	**0.412**	0.422
Electricity	96	**0.176**	0.186	**0.188**	0.194	**0.185**	0.210
	192	**0.190**	0.197	**0.194**	0.199	**0.243**	0.252
	336	**0.210**	0.218	**0.210**	0.215	**0.233**	0.328
	720	**0.253**	0.264	**0.252**	0.256	0.291	**0.280**
	Average	**0.207**	0.216	**0.211**	0.216	**0.238**	0.268
PEMS08	12	**0.088**	0.094	**0.109**	0.112	**0.290**	0.641
	24	**0.138**	0.146	**0.180**	0.189	**0.317**	0.767
	48	**0.254**	0.280	**0.348**	0.379	**0.529**	0.904
	96	**0.477**	0.562	**0.676**	0.822	**0.720**	1.085
	Average	**0.239**	0.270	**0.328**	0.376	**0.464**	0.849

References

[1] Lu, J., Yang, S. (2026). FasterWeight-MLP Attention: Rethinking Sequence Modeling from a Neural-Network-Native Perspective. *Unpublished manuscript*.

[2] Liu, S. et al. (2024). iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. *ICLR*.

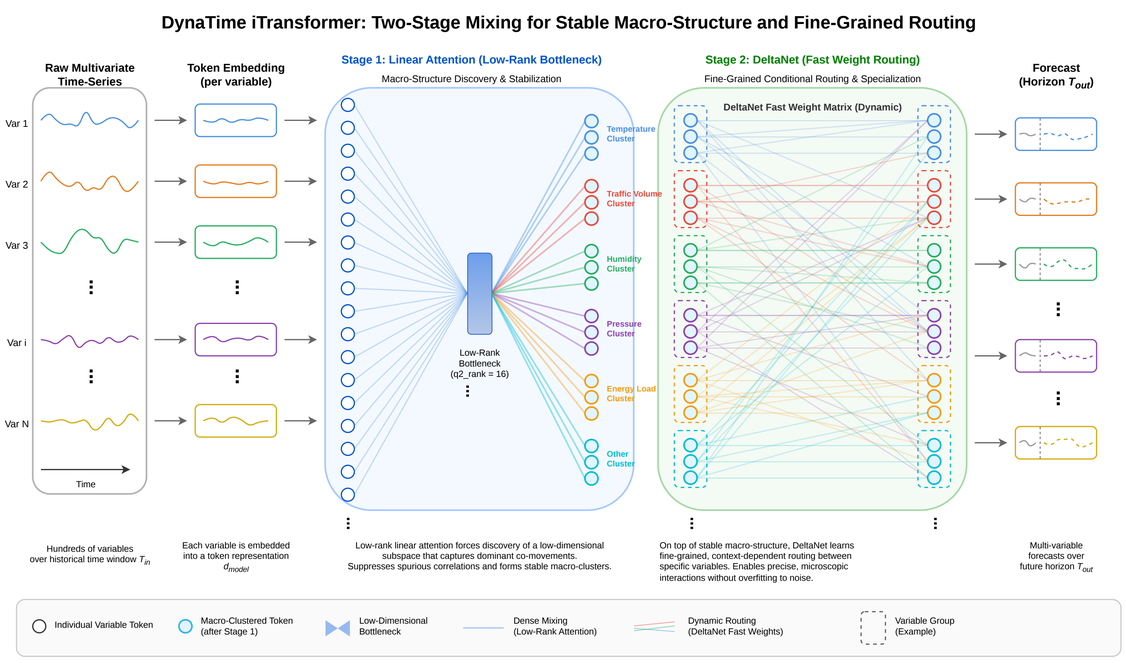
[3] Nie, Y. et al. (2023). A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *ICLR*.

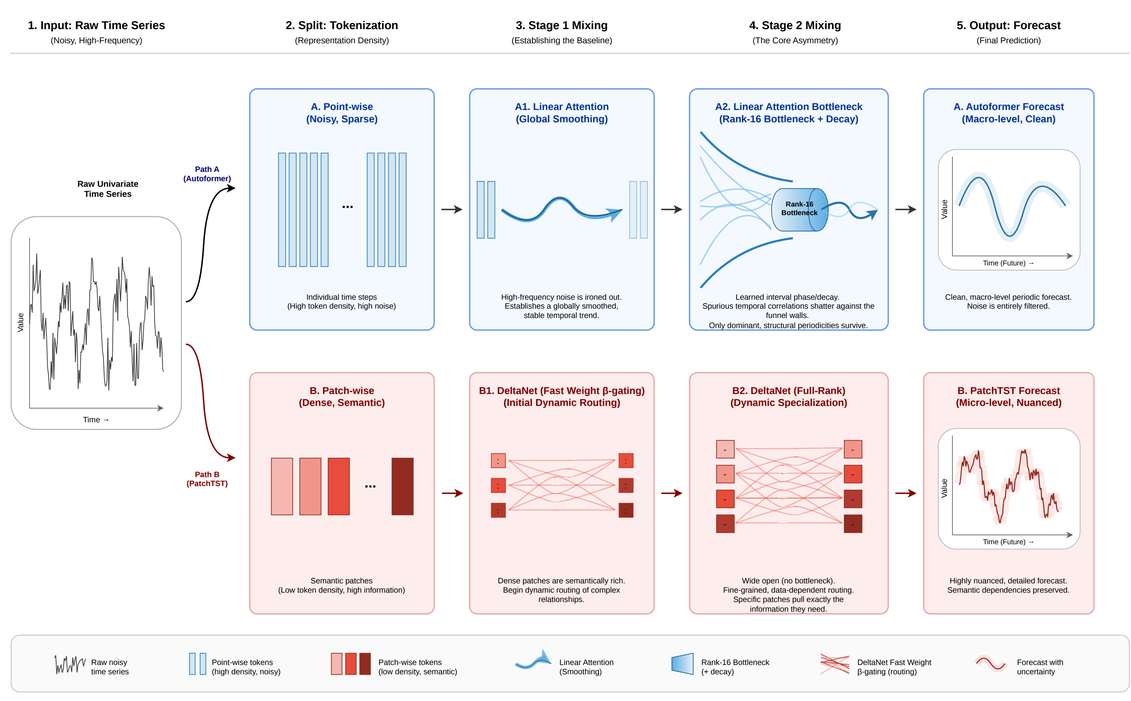
[4] Wu, H. et al. (2021). Autoformer: Decomposition Transformers for Long-Term Series Forecasting. *NeurIPS*.

[5] Wu, H. et al. (2023). Time-Series-Library (TSLib). *GitHub repository*.

[6] Yang, S. et al. (2024). Parallelizing Linear Transformers with the Delta Rule. *NeurIPS*.

Integration Frameworks





jtwitty3@gatech.edu