Neural Networks 205 (2027) 109495

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

Neuronal attention circuit (NAC) for representation learning

Waleed Razzaq [a], Yun-Bo Zhao [a,b,*]

[a] Department of Automation, University of Science & Technology of China, Hefei, China
[b] Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, China

ARTICLE INFO

Keywords:
Continuous-time attention
Liquid neural networks (LNNs)
Transformers
Time-series modeling

ABSTRACT

Attention improves representation learning over RNNs, but its discrete nature limits continuous-time (CT) modeling. We introduce Neuronal Attention Circuit (NAC), a novel, biologically inspired CT-attention mechanism that reformulates attention logit computation as the solution to a linear first-order ODE with nonlinear interlinked gates derived from repurposing the wiring of *C. elegans* Neuronal Circuit Policies (NCPs). NAC replaces dense projections with sparse sensory gates for query-key projections and introduces a sparse backbone network with two heads for computing *content-target* and *learnable time-constant* gates, enabling efficient adaptive dynamics. To improve efficiency and memory consumption, we implement an adaptable, sparse, subquadratic Top-K pairwise concatenation mechanism that selectively curates query-key interactions. We provide rigorous theoretical guarantees, including state stability and bounded approximation errors. Empirically, we implement NAC in diverse domains, including irregular time-series, long-range forecasting, lane-keeping for autonomous vehicles, and industrial prognostics. We observe that NAC matches or outperforms in accuracy against several CT state-of-the-art baselines, while being interpretable at the neuron cell level.

1. Introduction

Learning representations of sequential data in temporal or spatiotemporal domains is essential for capturing patterns and enabling accurate forecasting. Discrete-time Recurrent neural networks (DT-RNNs), such as RNN (Jordan, 1997; Rumelhart et al., 1985), Long-short term memory (LSTM) (Hochreiter & Schmidhuber, 1997), and Gated Recurrent Unit (GRU) (Cho et al., 2014) model sequential dependencies by iteratively updating hidden states to represent or predict future elements in a sequence. While effective for regularly sampled sequences, DT-RNNs face challenges with irregularly sampled data because they assume uniform time intervals. In addition, vanishing gradients can make it difficult to capture long-term dependencies (Hochreiter, 1998).

Continuous-time RNNs (CT-RNNs) (Rubanova et al., 2019) model hidden states as ordinary differential equations (ODEs), allowing them to process inputs that arrive at arbitrary or irregular time intervals. Mixed-memory RNNs (mmRNNs) (Lechner & Hasani, 2022) build on this idea by separating memory compartments from time-continuous states, helping maintain stable error propagation while capturing continuous-time dynamics. Liquid neural networks (LNNs) (Hasani et al., 2022, 2021) take a biologically inspired approach by assigning variable time-constants to hidden states, improving adaptability and robustness, although vanishing gradients can still pose challenges during training.

The scaled-dot-product-attention (SDPA) mechanism (Vaswani et al., 2017) mitigate this limitation by treating all time steps equally and allowing models to focus on the most relevant observations. It computes the similarity between queries (q) and keys (k), scaling by the key dimension to keep the gradients stable. Multihead Attention (MHA) (Vaswani et al., 2017) extends this by allowing the model to attend to different representation subspaces in parallel. Variants such as Sparse Attention (Zhang et al., 2021), BigBird (Zaheer et al., 2020), and Longformer (Beltagy et al., 2020) modify the attention pattern to reduce computational cost, particularly for long sequences, by attending only to selected positions rather than all pairs. Even with these improvements, the computation paths in these methods are discrete in nature, limiting their ability to model continuous trajectories often captured by their CT counterparts.

Recent work has explored bridging this gap through the Neural-ODE (Chen et al., 2018) formulation. mTAN (Shukla & Marlin, 2021) learns CT embeddings and uses time-based attention to interpolate irregular observations into a fixed-length representation for downstream encoder-decoder modeling. Continuous-time Attention (CTA) (Chien & Chen, 2021) embeds a CT-attention mechanism within a Neural ODE, allowing attention weights and hidden states to evolve jointly over time. Nevertheless, it remains computationally intensive and sensitive to the accuracy of the ODE solver. ODEFormer (d'Ascoli et al., 2023) trains a sequence-to-sequence transformer on synthetic trajectories to infer symbolic ODE systems directly from noisy, irregular data, although it

* Corresponding author.
E-mail addresses: waleedrazzaq@mail.ustc.edu.cn (W. Razzaq), ybzhao@ustc.edu.cn (Y.-B. Zhao).

https://doi.org/10.1016/j.neunet.2026.109495
Received 5 May 2026; Received in revised form 19 July 2026; Accepted 9 August 2026
Available online 12 August 2026

struggles with chaotic systems and generalization beyond observed conditions. ContiFormer (Chen et al., 2023) builds a CT-Transformer by pairing ODE-defined latent trajectories with a time-aware attention mechanism to model dynamic relationships in data.

Despite these advancements, a persistent and underexplored gap remains in developing a biologically inspired attention mechanism that seamlessly integrates CT dynamics with the abstraction of the brain's connectome. Building on this, we propose a novel attention mechanism called the *Neuronal Attention Circuit* (NAC), in which attention logits are computed as the solution to a first-order ODE modulated by nonlinear, interlinked gates derived from repurposing Neuronal Circuit Policies (NCPs) from the nervous system of *C.elegans* nematode (refer to Section A.1 for more information). Unlike SDPA, which projects query, key, value vectors through a linear layer, NAC employs a sensory gate to transform input features and a backbone network to model nonlinear interactions, with multiple heads producing outputs structured for attention logit computation. Based on the solutions to ODE, we define three computation modes: (i) Exact, using the closed-form ODE solution; (ii) Euler, approximating the solution via *explicit Euler* integration; and (iii) Steady, using only the steady-state solution, analogous to SDPA scores. To reduce computational complexity, we implement a sparse subquadratic Top-K pairwise concatenation algorithm that selectively curates query-key inputs. We evaluate NAC across multiple domains, including irregularly sampled time-series, long-range forecasting, autonomous vehicle lane-keeping, and Industry 4.0, and compare it to state-of-the-art baselines. NAC consistently matches or outperforms these models, while runtime and peak memory benchmarks place it between CT-RNNs in terms of speed and CT-attentions in terms of memory requirements. We also highlight the interpretability of NAC at the cell level.

Our contributions are summarized as:

- We proposed Neuronal Attention Circuit (NAC), a novel biologically-inspired attention mechanism that reformulates attention logits through an input-dependent linear ODE modulated with nonlinear interlinked gates.
- We repurposed *C.elegans* nematode's Neuronal Circuit Policies (NCPs) wiring mechanism to derive the nonlinear gating mechanism to introduce sparsity and cell-level interpretability into the attention mechanism.
- We evaluate NAC in a broad set of tasks including (i) irregular time-series; (ii) long-range dependencies; (iii) autonomous-vehicle lane-keeping control; and (iv) Industry 4.0.

The rest of the paper is organized as follows. Section 2 reviews the related literature and highlights the contributions. Section 3 presents the comprehensive architectural details of proposed algorithm. Section 4 comprehensively evaluate the proposed algorithm in a broad set of learning tasks. Section 5 discuss the limitations and possible future directions. Section 6 conclude the paper.

2. Related works

Bio-inspired models: LNNs (Hasani et al., 2022, 2021) introduce input-dependent hidden states dynamics by allowing the integration time constants of recurrent hidden states to vary through nonlinear gating mechanisms. These models are often implemented within the NCPs (Lechner et al., 2020, 2018) framework, leveraging the sparse connectivity of the *C.elegans* connectome to produce neural dynamics. NAC draws on these input-dependent mechanisms but applies them to attention score computation rather than hidden state evolution.

CT Attention: Prior work incorporates attention mechanisms into Neural ODE frameworks, allowing hidden states and attention weights to evolve jointly over time (Chien & Chen, 2021), couples latent dynamics with time-aware attention (Chen et al., 2023), trains transformers to learn underlying CT sequence representations (d'Ascoli et al., 2023), formulates entire transformer blocks as a single ODE with optimal-transport regularization (Kan et al., 2025), or evolves the full attention matrix over pseudo-time via PDEs (Zhang & Zhou, 2025). However, the underlying attention score computation path in all these approaches remains inherently discrete. NAC rethinks attention score computation as the solution to a first-order ODE with an input-dependent nonlinear gate, turning the discrete attention itself into an adaptive continuous-depth process.

3. Neuronal attention circuit (NAC)

We propose a simple alternative formulation of the attention logits a (refer to Section A.2 for more information), interpreting them as the solution to a first-order linear ODE modulated by nonlinear, interlinked gates:

$$\frac{da_t}{dt} = -\underbrace{f_{\omega_\tau}([\mathbf{q};\mathbf{k}], t, \theta_{\omega_\tau})}_{\omega_\tau(\mathbf{u})} a_t + \underbrace{f_{\phi}([\mathbf{q};\mathbf{k}], t, \theta_{\phi})}_{\phi(\mathbf{u})}, \tag{1}$$

where $\mathbf{u} = [\mathbf{q};\mathbf{k}]$ denotes the sparse Top-K concatenated query–key input. ω_τ represents a *learnable time-constant* gate head, and ϕ denotes a nonlinear *content-target* head. Both gates are parameterized by a backbone network derived from repurposing NCPs. We refer to this formulation as the Neuronal Attention Circuit (NAC). It enables the logit a_t to evolve dynamically with input-dependent, variable time constants, mirroring the adaptive temporal dynamics found in *C.elegans* nervous systems while improving computational efficiency and expressiveness. Moreover, it introduces continuous depth into the attention mechanism, bridging discrete-layer computation with dynamic temporal evolution.

Motivation behind this formulation: The proposed formulation is loosely motivated by the input-dependent time-constant mechanism of LNNs, a class of CT-RNNs inspired by biological nervous systems and synaptic transmission. In LNNs, the dynamics of nonspiking neurons are described by a linear ODE with nonlinear, interlinked gates: $\frac{d\mathbf{x_t}}{dt} = -\frac{\mathbf{x_t}}{\tau} + \mathbf{S_t}$, where $\mathbf{x_t}$ denotes the hidden state and $\mathbf{S_t} \in \mathbb{R}^M$ represents nonlinear synapse defined as $f(\mathbf{x_t}, \mathbf{u}, t, \theta)(A - \mathbf{x_t})$. Here, A and θ are learnable parameters. Plugging $\mathbf{S_t}$ yields $\frac{d\mathbf{x_t}}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x_t}, \mathbf{u}, t, \theta)\right]\mathbf{x_t} + f(\mathbf{x_t}, \mathbf{u}, t, \theta)A$. LNNs are known for their strong expressivity, stability, and performance in irregularly sampled time-series modeling (Hasani et al., 2022, 2021).

NAC's forward-pass update using ODE solver: The state of NAC at time t can be computed using a numerical ODE solver that simulates the dynamics from an initial state a_0 to a_t. The solver discretizes the continuous interval $[0, T]$ into steps $[t_0, t_1, t_2, \ldots, t_n]$, with each step updating the state from t_i to t_{i+1}. For our purposes, we use the *explicit Euler* solver, which is simple, efficient, and easy to implement. Although methods such as Runge-Kutta may offer higher accuracy, their computational overhead makes them less suitable for large-scale neural simulations that require numerous updates, especially since the logits are normalized, and exact precision is not necessary. Let the step size be Δt, with discrete times $t_n = n\Delta t$ and logit states $a_n = a(t_n)$. Using the *explicit Euler* method, the update is

$$a_{n+1} = a_n + \Delta t\,(-\omega_\tau a_n + \phi). \tag{2}$$

Closed-form (Exact) Computation of NAC: We derive the analytical solution for Eq. (1). The gates ϕ and ω_τ are time-varying and depend on the input sequence at each time step. However, to enable an efficient closed-form forward-pass update, we apply a piecewise constant assumption (locally frozen-coefficient approximation John, 1952) over a small interval. Under this assumption, with initial condition a_0, the analytical closed-form solution is as follows:

$$a_t = \underbrace{a^*}_{\text{steady-state}} + \underbrace{(a_0 - a^*)\, e^{-\omega_\tau t}}_{\text{transient}} \tag{3}$$

Here, $a^* = \phi/\omega_\tau$ is the steady-state solution. The detailed derivation and analysis are provided in Section B.1.

3.1. Stability analysis of NAC

We now investigate the stability bounds of NAC under both the ODE-based and the Closed-Form formulations.

3.1.1. State stability

We analyze state stability in both single-connection and multi-connection settings. This analysis establishes the boundness of the attention logit state trajectory, ensuring that, under positive decay rates, the dynamics remain well-behaved without divergence or overshoot.

Theorem 1 (State Stability). *Let $a_t^{(i)}$ denote the state of the ith attention logit governed by $da_t^{(i)}/dt = -\omega_\tau a_t^{(i)} + \phi$. Assume that ϕ and ω_τ decompose across M incoming connections as $\phi = \sum_{j=1}^{M} f_\phi([\mathbf{q}_i;\mathbf{k}_j])$, and $\omega_\tau = \sum_{j=1}^{M} f_{\omega_\tau}([\mathbf{q}_i;\mathbf{k}_j])$, with $f_{\omega_\tau} > 0$. Define the per-connection equilibrium $A_{i,j} = f_\phi([\mathbf{q}_i;\mathbf{k}_j])/f_{\omega_\tau}([\mathbf{q}_i;\mathbf{k}_j])$, and let $A_i^{\min} = \min_j A_{i,j}$ and $A_i^{\max} = \max_j A_{i,j}$. Then for any finite horizon $t \in [0,T]$, the state trajectory satisfies*

$$\min(0, A_i^{\min}) \;\le\; a_t^{(i)} \;\le\; \max(0, A_i^{\max}), \tag{4}$$

provided the initial condition $a_i(0)$ lies within this range. In the special case of a single connection ($M = 1$), the bounds collapse to

$$\min(0, A_i) \;\le\; a_t^{(i)} \;\le\; \max(0, A_i), \tag{5}$$

where $A_i = f_\phi/\omega_\tau$ is exactly the steady-state solution from Eq. (3). The proof is provided in the Section B.2.

3.1.2. Closed-form error & exponential bounds

We now examine the asymptotic stability, error characterization, and exponential boundedness of the closed-form formulation. We begin by quantifying the deviation of the trajectory from its steady-state solution. Define the instantaneous error

$$\varepsilon_t = a_t - a^*, \tag{6}$$

which measures the distance of the system state to equilibrium at time t. From Eq. (3), the error admits the exact representation

$$\varepsilon_t = (a_0 - a^*)e^{-\omega_\tau t} \tag{7}$$

In particular, the pointwise absolute error is given by

$$|\varepsilon_t| = |a_0 - a^*|\, e^{-\omega_\tau t} \tag{8}$$

This reveals that convergence is not merely asymptotic but follows an exact exponential law, controlled by the rate parameter ω_τ. This yields the following finite-time guarantee.

Corollary 1 (Exponential decay bound). *If $\omega_\tau > 0$, then for all $t \ge 0$,*

$$|a_t - a^*| \le |a_0 - a^*| e^{-\omega_\tau t}. \tag{9}$$

Remark: If $\omega_\tau > 0$ then $\lim_{t\to\infty} e^{-\omega_\tau t} = 0$, therefore $a_t \to a^*$. The convergence is exponential with rate ω_τ. If $\omega_\tau < 0$ then $e^{-\omega_\tau t} = e^{|\omega_\tau| t}$ diverges so that a_t grows exponentially away from a^* in magnitude (unless initial offset $a_0 - a^* = 0$, a measure-zero case). If $\omega_\tau = 0$ the ODE is $\dot{a} = \phi$ and the solution is linear in t (unless $\phi = 0$). For bounded dynamics that converge to an interpretable steady-state, it is required that $\omega_\tau > 0$.

Corollary 2 (Uniform initialization). *If the initialization is known only to belong to a bounded set, i.e., $|a_0 - a^*| \le M$ for some $M > 0$, then the error admits the uniform bound*

$$|a_t - a^*| \le M e^{-\omega_\tau t}. \tag{10}$$

Remark: This bound highlights that exponential convergence holds uniformly across all admissible initial conditions, with the constant M capturing the worst-case deviation.

Corollary 3 (Convergence time to δ-accuracy). *A natural operational question is the time required to achieve a target tolerance $\delta > 0$. Solving*

$$|a_0 - a^*| e^{-\omega_\tau t} \le \delta, \tag{11}$$

We obtain the threshold

$$t \ge \frac{1}{\omega_\tau} \ln \frac{|a_0 - a^*|}{\delta}. \tag{12}$$

Remark: The convergence time rate is inversely proportional to ω_τ, and the required time scales only logarithmically at the accuracy level $1/\delta$. Intuitively, a larger ω_τ accelerates contraction toward equilibrium, yielding faster attainment of any prescribed tolerance.

Algorithm 1 Repurposed NCPCell.

Require: Wiring $\mathcal{W}$ with $(A_{\text{in}}, A_{\text{rec}})$, groups (N_s, N_i, N_c, N_m), activation α, input group G_{input}, output group G_{output}, disabled groups $\mathcal{D}$
Ensure: Output $y_t \in \mathbb{R}^{B\times d_{\text{out}}}$, state $\mathbf{x}_t \in \mathbb{R}^{B\times d_h}$
Binary mask: $M_{\text{rec}} \leftarrow |A_{\text{rec}}|, \quad M_{\text{in}} \leftarrow |A_{\text{in}}|$
Initialize parameters : $W_{\text{in}}, W_{\text{rec}}, b, w_{\text{in}}, b_{\text{in}}, w_{\text{out}}, b_{\text{out}}$
Input neurons: $I_{\text{in}} \leftarrow G_{\text{input}}$
Output neurons: $I_{\text{out}} \leftarrow G_{\text{output}}$
Define activation mask: $\text{mask}_{\text{act},i} = 0$ if $i \in \mathcal{D}$ else 1
Input Projections: $\tilde{u}_t \leftarrow u_t \odot w_{\text{in}} + b_{\text{in}}$
Recurrent computation: $r_t \leftarrow \mathbf{x}_{t-1}(W_{\text{rec}} \odot M_{\text{rec}})$
Sparse computation: $s_t \leftarrow \tilde{u}_t(W_{\text{in}} \odot M_{\text{in}})$
Neuron update: $\mathbf{x}_t \leftarrow \alpha(r_t + s_t + b) \odot \text{mask}_{\text{act}}$
Output mapping: $y_t \leftarrow (\mathbf{x}_t[I_{\text{out}}] \odot w_{\text{out}}) + b_{\text{out}}$
return $(y_t, \mathbf{x}_t)$

Algorithm 2 Sparse Top-K Pairwise Concatenation.

Require: Queries $Q \in \mathbb{R}^{B\times H\times T_q\times D}$, Keys $K \in \mathbb{R}^{B\times H\times T_k\times D}$, Top-$K$ value K
Ensure: Concatenated tensor $U \in \mathbb{R}^{B\times H\times T_q\times K_{\text{eff}}\times 2D}$
Block Size: $B_s \leftarrow \lfloor\sqrt{T_k}\rfloor$
Partition K into N_b blocks: $K_{\text{blocks}} \in \mathbb{R}^{B\times H\times N_b\times B_s\times D}$
Block Centroids: $\bar{K} \leftarrow \text{mean}(K_{\text{blocks}})$
Coarse Scores: $S_{\text{coarse}} \leftarrow Q \cdot \bar{K}^\top$
Select Top-M Blocks: $I_{\text{blocks}} \leftarrow \text{top\_k}(S_{\text{coarse}}, \lceil K/B_s\rceil)$
Gather Candidates: $K_{\text{cand}} \leftarrow \text{gather}(K_{\text{blocks}}, I_{\text{blocks}})$
Fine Scores: $S_{\text{fine}} \leftarrow \text{reduce\_sum}(Q \odot K_{\text{cand}})$
Effective Top-K: $K_{\text{eff}} \leftarrow \min(K, \text{size}(K_{\text{cand}}))$
Indices: $I_{\text{topk}} \leftarrow \text{top\_k}(S_{\text{fine}}, K_{\text{eff}})$
Selected Keys: $K_{\text{selected}} \leftarrow \text{gather}(K_{\text{cand}}, I_{\text{topk}})$
Concatenate: $U_{\text{topk}} \leftarrow [\,Q;\; K_{\text{selected}}\,] \in \mathbb{R}^{B\times H\times T_q\times K_{\text{eff}}\times 2D}$
return U_{topk}

3.2. Designing NAC as neural network layer

We now outline the design of NAC as neural network layer guided by the preceding analysis. The process involves five steps: (i) repurposing NCPs; (ii) input curation; (iii) construction of the time vector (t); (iv) computing attention logits and weights; and (v) generating the attention output. Fig. 1 provides a graphical overview of NAC.

Repurposing NCPs: We repurpose the NCPs framework by converting its fixed, biologically derived wiring (see Fig. 2(a)) into a flexible recurrent architecture that allows configurable input–output mappings. Instead of enforcing a static connectome, our approach exposes adjacency matrices as modifiable structures defining sparse input and recurrent connections. This enables selective information routing across neuron groups while retaining the original circuit topology. Decoupling wiring specifications from model instantiation allows dynamic connectivity adjustments to accommodate different input modalities without full retraining. Algorithm 1 summarizes the steps for repurposing the NCPs wiring mechanism. Key features include groupwise masking for neuron isolation, adaptive remapping of inputs and outputs for task-specific adaptation, and tunable sparsity s to balance expressiveness and efficiency.

In our implementation, the sensory neuron gate ($\mathcal{NN}_{\text{sensory}}$) projects the q, k, and v representations (see Fig. 2(b)). This enables sensory neurons to maintain structured, context-aware representations rather than

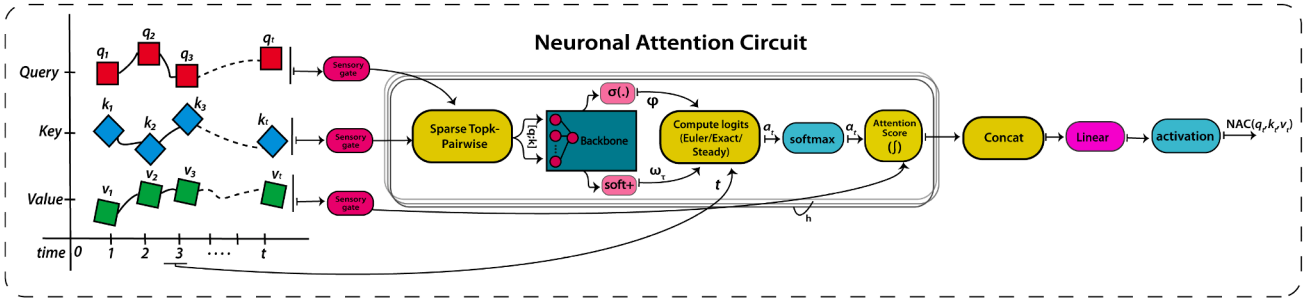

Fig. 1. Illustration of the architecture of Neuronal Attention Circuit (NAC) mechanism.

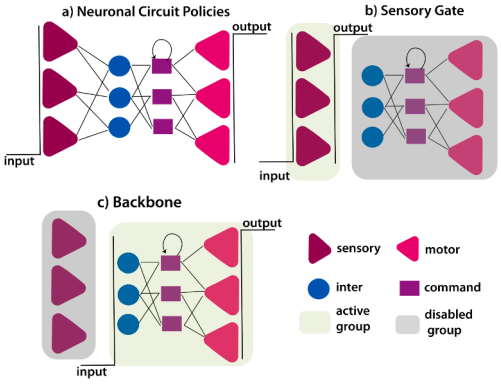

Fig. 2. Illustration of **(A)** NCPs with predetermined wiring; **(B)** Sensory gate, where sensory neurons are active, and the remaining neurons are disabled for the q, k, and v projections; **(C)** Backbone, showing inter-motor projections with sensory neurons disabled in extended heads for computing ϕ and ω_τ.

collapsing inputs into fully connected layers. As a result, the network preserves locality and modularity, which improves information routing. Intuitively, it acts like biological receptor neurons, converting external stimulus inputs into organized signals that feed into downstream circuits.

$$\mathcal{NN}_{\text{sensory}} = \text{NCPCell}(G_{\text{input}} = [N_s],\ G_{\text{output}} = [N_s],\ D = [N_i, N_c, N_m],\ s) \tag{13}$$

The inter-to-motor pathways form a backbone network ($\mathcal{NN}_{\text{backbone}}$) with branches that compute ϕ and ω_τ (see Fig. 2(c)). Intuitively, this integrates neuronal activity over time and produces outputs that regulate gating signals. Instead of learning ϕ and ω_τ independently, this backbone allows the model to learn shared representations, enabling multiple benefits: (i) accelerates convergence during training; (ii) separate head layers enable the system to capture temporal and structural dependencies independently.

$$\mathcal{NN}_{\text{backbone}} = \text{NCPCell}(G_{\text{input}} = [N_i],\ G_{\text{output}} = [N_m],\ D = [N_s],\ s) \tag{14}$$

The output heads are defined as:

$$\phi = \sigma(\mathcal{NN}_{\text{backbone}}(\mathbf{u})) \tag{15}$$

$$\omega_\tau = \text{softplus}(\mathcal{NN}_{\text{backbone}}(\mathbf{u})) + \varepsilon, \quad \varepsilon > 0 \tag{16}$$

Here, ϕ serves as a *content-target* gate head, where the sigmoid function $\sigma(\cdot)$ determines the target signal strength. In contrast, ω_τ is a strictly positive *time-constant* gate head that controls the rate of convergence and the steady-state amplitude. Conceptually, this parallels recurrent gating: ϕ regulates *what* content to emphasize, while ω_τ governs *how quickly* and *to what extent* it is expressed.

Input Curation: Initially, full pairwise concatenation was used to form a joint tensor $U \in \mathbb{R}^{B\times H\times T_q\times T_k\times 2D}$ from $Q \in \mathbb{R}^{B\times H\times T_q\times D}$ and $K \in \mathbb{R}^{B\times H\times T_k\times D}$. However, this approach becomes computationally intractable as the sequence length increases, leading to $\mathcal{O}(n^2)$ memory growth, severe memory fragmentation, and frequent hardware memory exhaustion. A naive sparse Top-K pruning strategy was then implemented to reduce the load on subsequent layers. While this reduced downstream computation, it failed to address the fundamental quadratic bottleneck in the initial scoring and storage stages, rendering the optimization ineffective for long sequences. To mitigate this issue, we then implemented a subquadratic Top-K selection strategy based on square-root partitioning (Child et al., 2019). By dynamically segmenting the key sequence into blocks and scoring queries against block centroids, the method identifies high-salience candidate regions without ever materializing the full $T_q \times T_k$ matrix. This hierarchical approach reduces the scoring complexity to $\mathcal{O}(n\sqrt{n})$, producing a sparse tensor U_{topk} that enables linear memory scaling in subsequent backbone layers. The mechanism is self-adaptive and eliminates the need for manual block-size tuning (Algorithm 2).

Construction of Time Vector (t): NAC builds on continuous-depth models as in Hasani et al. (2022), which adapt their temporal dynamics to the task. It constructs an internally normalized per-head time vector $t = \sigma(-t_a t_{\text{sample}} + t_b)$, where t_a and t_b are learnable affine parameters and t_{sample} is derived from the input sample. When the task contains meaningful temporal structure, t_{sample} introduces input-dependent modulation of the dynamics. When temporal information is not meaningful, t_{sample} is set to 1, and t reduces to a learned vector through $\sigma(-t_a + t_b)$. In both cases, the σ ensures $t \in [0, 1]$, providing a smooth, bounded representation of t for modulating the network's dynamics.

Attention logits and weights: Starting from Eq. (3), consider the trajectory of a query–key pair with initial condition $a_0 = 0$:

$$a_t = \frac{\phi}{\omega_\tau}\left(1 - e^{-\omega_\tau t}\right), \tag{17}$$

For finite t_{sample}, the exponential factor $(1 - e^{-\omega_\tau t})$ regulates the buildup of attention, giving ω_τ a temporal gating role. Normalizing across all keys via *softmax* yields attention weights $\alpha_t = \text{softmax}(a_t)$, defining a valid probability distribution where ϕ amplifies or suppresses content alignments, and ω_τ shapes both the speed and saturation of these preferences.

As $t_{\text{sample}} \to \infty$, the trajectory converges to the steady state

$$a_t^* = \frac{\phi}{\omega_\tau}, \tag{18}$$

which we introduce as a computation mode. Conceptually, it is analogous to SDPA: it aggregates information across inputs in a similar form, even though ϕ cannot reproduce exact dot products.

Attention output: Finally, the attention output is computed by integrating the α_t with the v_t matrix over internally constructed time-window t:

$$\text{NAC}(q, k, v) = \int_T \alpha_t v_t dt \tag{19}$$

Intuitively, this integral acts as a continuous aggregator of the v_t, where each contribution is weighted by its instantaneous relevance. In practice, we approximate it via a time-modulated sum: $\sum_i \alpha_t^{(i)} v_t^{(i)} w^{(i)}$ where $w^{(i)} \in [0, 1]$ is an adaptive input-dependent quadrature weight vector derived from the normalized time vector t. Unlike a fixed step size Δt, $w^{(i)}$ is optimized end-to-end and allows the model to modulate temporal contributions independently of semantic alignment $\alpha_t^{(i)}$, while remaining bounded and differentiable.

3.2.1. Extension to multihead

To scale this mechanism to multihead setting, we project the input sequence into H independent subspaces (heads) of dimension d_{model}/H, yielding query, key, and value tensors $(q^{(h)}, k^{(h)}, v^{(h)})$ for $h \in \{1, \ldots, H\}$. For each head, separate concatenated pairs and logits are computed according to Eqs. (2), (3) or (18), followed by *Softmax* normalization to calculate attention weights. The resulting attention weights $\alpha_t^{(h)}$ are then integrated with the value vector $v^{(h)}$, to produce head-specific attention outputs. Finally, these outputs are concatenated and linearly projected back into the model dimension. This formulation ensures that each head learns distinct dynamic compatibilities governed by its own parameterization of ϕ and ω_τ, aggregation across heads preserves the expressive capacity of the standard multihead attention mechanism.

4. Evaluation

We evaluate NAC against a range of baselines, including CT-RNNs, (DT & CT) Attentions, and multiple ablation configurations. Experiments are conducted across diverse domains, including irregular time-series, long-range dependencies, lane keeping of autonomous vehicles, and Industry 4.0 prognostics. All the results are obtained using 5-fold cross-validation, where the models are trained using BPTT (see Section C.2) on each fold and evaluated across all the folds. We report the mean (μ) and standard deviation (σ) to capture variability and quantify uncertainty in the predictions. To ensure best possible hyperparameters we utilized Bayesian optimization (Snoek et al., 2012) to tune NAC (see Section D.1). Table 1 provides the results for all the experiments, and the details of the data, preprocessing, and neural network architectures for all the experiments are provided in Section D.2.

4.1. Baselines

The brief descriptions of baselines are divided into three subcategories.

CT-RNNs: CT-RNN (Rubanova et al., 2019) models temporal dynamics using differential equations, enabling hidden states to evolve continuously over time in response to inputs, which is particularly useful for irregularly sampled time-series data. PhasedLSTM (Neil et al., 2016) introduces a time gate that updates hidden states according to a rhythmic schedule, enabling efficient modeling of asynchronous or irregularly sampled time-series. GRU-ODE (De Brouwer et al., 2019) extends the GRU to continuous time, evolving hidden states via ODEs to handle sequences with nonuniform time intervals. mmRNN (Lechner & Hasani, 2022) combines short-term and long-term memory units to capture both fast-changing and slowly evolving patterns in sequential data. LTC (Hasani et al., 2021) uses neurons with learnable, input-dependent time constants to adapt the speed of dynamics and capture complex temporal patterns in continuous-time data: LTC-FC used FullyConnected layer and LTC-NCP uses NCPs framework. CfC (Hasani et al., 2022) approximate LTC dynamics analytically, providing efficient continuous-time modeling without relying on numerical ODE solvers: CfC-FC used FullyConnected layer and CfC-NCP uses NCPs framework. DeepState (Rangapuram et al., 2018) combines RNNs with linear state-space models (SSMs), allowing the RNN to produce time-varying SSM parameters to model CT dynamics. S4 (Gu et al., 2021) parametrizes SSMs with a structured low-rank correction to the state matrix, enabling efficient convolution-based computation and effective modeling of long-range dependencies.

DT-Attentions: SDPA (Vaswani et al., 2017) computes attention weights by measuring the similarity between queries and keys, scaling the results, and applying Softmax to weigh time-step contributions. Linear Attention (Zhang et al., 2021) replaces the softmax activation with a *ReLU* function to induce sparsity and enable heads to selectively switch off for certain queries. Performer (Choromanski et al., 2020) approximates Softmax attention with linear-time FAVOR+, enabling efficient and scalable Attention without losing theoretical guarantees.

CT-Attentions: mTAN (Shukla & Marlin, 2021) learns continuous-time embeddings and uses time-based attention to interpolate irregular observations into a fixed-length representation for downstream encoder-decoder modeling. CTA (Chien & Chen, 2021) generalizes discrete-time attention to continuous-time by representing hidden states, context vectors, and attention scores as functions whose dynamics are modeled via neural networks and integrated using ODE solvers for irregular sequences. ODEFormer (d'Ascoli et al., 2023) trains a sequence-to-sequence transformer on synthetic trajectories to directly output a symbolic ODE system from noisy, irregular time-series data. ContiFormer (Chen et al., 2023) builds a CT-Transformer by pairing ODE-defined latent trajectories with a time-aware attention mechanism to model dynamic relationships in irregular time-series data. OT-Transformer (Kan et al., 2025) modifies the transformers by interpreting the composition of blocks as an ODE and regularizes training with an optimal transport cost that penalizes the squared norm of the dynamics to improve stability. PDE-Attention (Zhang & Zhou, 2025) proposes to evolve the attention matrix over a pseudo-time dimension using partial differential equations, smoothing local noise, and enabling polynomial decay of long-range dependencies to improve performance.

4.2. Ablations details

The brief descriptions of variants and ablations are divided into five subcategories:

Top-K Ablations: *NAC-2k* uses Top-$K = 2$ to compute the logits and *NAC-32k* uses Top-$K = 32$. *NAC-PW* uses fully pairwise (no sparsity) concatenation. All variants use the exact computation mode with 50% sparsity.

NAC-Sparsity Ablations: *NAC-02s* uses 20% sparsity to compute the logits, and *NAC-09s* uses 90%. All variants use the exact computation mode with Top-$K = 8$.

NAC with isolated NCPs Ablations: To isolate the effect of the NCP gating mechanism, we ablate it using MLP layers with matched sparsity. *NAC-FC* replaces the gating module with a standard fully connected layer. *NAC-05s* applies 50% sparsity to fully connected layer. *NAC-RW* replaces the gated module with Random Wiring with 50% sparsity. All variants use the exact computation mode with Top-$K = 8$.

NAC with unconstrained ϕ: Applying a sigmoidal nonlinearity to ϕ yields the logit to only positive values. To allow negative logits, we tested two variants: (i) *NAC-ϕ_{linear}* with no activation, and (ii) *NAC-ϕ_{tanh}* with a tanh activation.

Modes Ablations: *NAC-Euler* computes attention logits using the explicit Euler integration method. *NAC-Steady* derives attention logits from the steady-state solution of the exact formulation. *NAC (main)* computes attention logits using the closed-form exact solution. It also overlaps with other ablations, so we combined it into a single one. All modes use Top-$K = 8$ and 50% sparsity.

4.3. Experiments

4.3.1. Irregular time-series

We evaluate NAC on two irregular time-series datasets: (i) Event-based MNIST; and (ii) Person Activity Recognition (PAR).

Table 1
Model performance across all categories and datasets.

Model	Irregular Time-Series		Multivariate LRD		Lane-Keeping of AVs		Industry 4.0		
	E-MNIST (↑)	PAR (↑)	ETTm1 (↓)	J.Climate (↓)	CarRacing (↑)	Udacity (↓)	PRONOSTIA (↓)	XJTU-SY (↓)	HUST (↓)
CT-RNN	95.18 ± 0.20	88.71 ± 0.87	0.0315 ± 0.0010	**0.0153 ± 0.0002**	80.21 ± 0.27	0.0206 ± 0.0013	44.32 ± 8.69	26.01 ± 8.74	39.99 ± 6.33
GRU-ODE	96.04 ± 0.13	89.01 ± 1.55	0.0361 ± 0.0013	0.0286 ± 0.0017	80.29 ± 0.72	0.0188 ± 0.0016	45.11 ± 3.19	31.20 ± 8.69	43.91 ± 7.10
PhasedLSTM	95.79 ± 0.14	88.93 ± 1.08	0.0371 ± 0.0011	0.0700 ± 0.0004	80.35 ± 0.38	0.0186 ± 0.0015	44.15 ± 4.80	35.49 ± 5.54	38.66 ± 5.55
mmRNN	95.74 ± 0.27	88.48 ± 0.46	0.0377 ± 0.0024	0.0523 ± 0.0026	80.13 ± 0.54	0.0205 ± 0.0027	48.50 ± 4.60	27.84 ± 4.05	40.11 ± 9.56
LTC-FC	95.25 ± 0.00	88.12 ± 0.68	0.0310 ± 0.0004	0.0161 ± 0.0001	76.37 ± 3.01	0.0245 ± 0.0024	48.14 ± 5.01	36.83 ± 8.57	61.82 ± 15.64
CfC-FC	94.16 ± 0.49	88.60 ± 0.34	0.5884 ± 0.3023	0.0169 ± 0.0001	80.59 ± 0.33	0.0198 ± 0.0022	47.78 ± 3.54	35.51 ± 3.94	54.09 ± 10.13
LTC-NCP	94.77 ± 0.26	88.79 ± 0.15	0.0294 ± 0.0007	0.2508 ± 0.1211	79.19 ± 1.55	0.0228 ± 0.0022	52.96 ± 6.04	55.34 ± 16.03	78.99 ± 16.96
CfC-NCP	95.63 ± 1.47	87.24 ± 0.73	0.0355 ± 0.0063	0.8972 ± 0.2534	80.73 ± 0.22	0.0212 ± 0.0014	39.13 ± 4.96	27.08 ± 1.34	94.35 ± 15.22
DeepState	96.01 ± 0.07	86.25 ± 0.62	0.0325 ± 0.0007	0.0160 ± 0.0003	80.25 ± 0.20	0.0175 ± 0.0007	40.86 ± 4.45	30.42 ± 5.24	40.88 ± 2.11
S4	95.21 ± 0.07	89.68 ± 0.84	0.2676 ± 0.0017	0.0205 ± 0.0001	80.32 ± 0.68	0.0183 ± 0.0010	42.88 ± 8.88	29.49 ± 3.94	36.20 ± 9.27
SDPA	95.94 ± 0.15	88.36 ± 1.06	0.0353 ± 0.0008	0.0172 ± 0.0003	79.99 ± 0.49	0.0185 ± 0.0017	45.36 ± 5.16	37.31 ± 12.20	41.40 ± 7.72
Linear Attention	95.87 ± 0.14	88.75 ± 0.85	0.0407 ± 0.0008	0.0353 ± 0.0003	80.16 ± 0.30	0.0196 ± 0.0015	58.35 ± 8.67	87.80 ± 19.52	99.38 ± 37.47
Performer	96.01 ± 0.18	88.71 ± 1.06	0.0349 ± 0.0007	0.0253 ± 0.0003	80.11 ± 0.39	0.0180 ± 0.0007	45.37 ± 4.51	38.05 ± 5.75	39.06 ± 7.23
mTAN	95.97 ± 0.25	88.08 ± 0.94	0.4394 ± 0.0066	0.0312 ± 0.0003	80.56 ± 0.22	0.0178 ± 0.0005	44.41 ± 7.15	41.34 ± 3.72	66.29 ± 4.25
CTA	95.86 ± 0.14	88.10 ± 1.10	**0.0266 ± 0.0006**	0.0174 ± 0.0001	80.54 ± 0.40	0.0197 ± 0.0016	39.16 ± 3.54	**25.86** ± 1.47	38.41 ± 4.51
ODEFormer	95.62 ± 0.20	88.25 ± 0.66	0.0278 ± 0.0008	0.0249 ± 0.0002	80.54 ± 0.40	0.0190 ± 0.0012	42.42 ± 6.98	35.63 ± 9.24	40.60 ± 6.83
ContiFormer	96.04 ± 0.23	81.28 ± 0.85	0.0279 ± 0.0008	0.0184 ± 0.0002	80.47 ± 0.50	**0.0174 ± 0.0013**	37.82 ± 7.09	34.71 ± 4.98	43.81 ± 10.18
OT-Transformer	**96.30 ± 0.19**	89.49 ± 0.52	0.0347 ± 0.0006	0.0291 ± 0.0002	80.41 ± 0.32	0.0196 ± 0.0013	**37.77 ± 3.45**	31.38 ± 8.73	36.85 ± 4.77
PDE-Attention	95.95 ± 0.22	89.47 ± 0.70	0.0377 ± 0.0008	0.0336 ± 0.0003	71.53 ± 10.76	0.0195 ± 0.0004	40.61 ± 7.69	35.32 ± 8.23	39.08 ± 4.67
NAC-2k	95.73 ± 0.07	89.70 ± 0.98	0.0275 ± 0.0006	0.0151 ± 0.0001	80.59 ± 0.46	0.0208 ± 0.0015	43.78 ± 2.71	37.43 ± 9.28	40.51 ± 6.61
NAC-32k	95.15 ± 0.11	89.44 ± 0.87	0.0272 ± 0.0006	**0.0148 ± 0.0001**	80.38 ± 0.16	**0.0170 ± 0.0007**	49.53 ± 4.89	32.45 ± 10.84	39.17 ± 12.23
NAC-PW	**96.64** ± 0.12	**90.18 ± 1.01**	0.0374 ± 0.0004	0.0218 ± 0.0002	**80.72 ± 0.41**	0.0177 ± 0.0008	**37.77 ± 2.56**	28.01 ± 4.93	**30.14 ± 6.87**
NAC-09s	95.86 ± 0.11	89.76 ± 0.65	0.0308 ± 0.0013	0.0150 ± 0.0002	80.43 ± 0.17	0.0188 ± 0.0013	47.29 ± 5.52	40.40 ± 8.85	44.39 ± 6.82
NAC-02s	95.51 ± 0.06	89.47 ± 1.38	0.0279 ± 0.0007	0.0149 ± 0.0001	80.47 ± 0.27	0.0188 ± 0.0013	39.43 ± 5.94	35.59 ± 3.86	38.90 ± 6.43
NAC-ϕ_{linear}	95.89 ± 0.22	89.73 ± 0.89	**0.0273 ± 0.0006**	0.0154 ± 0.0001	80.58 ± 0.24	0.0184 ± 0.0024	42.64 ± 3.68	29.53 ± 4.49	**35.89 ± 7.08**
NAC-ϕ_{tanh}	95.70 ± 0.32	89.67 ± 1.27	0.0282 ± 0.0006	0.0161 ± 0.0001	80.60 ± 0.06	0.0175 ± 0.0016	45.99 ± 4.55	33.99 ± 3.96	38.75 ± 9.69
NAC-FC	95.31 ± 0.07	88.48 ± 1.24	0.0347 ± 0.0007	0.0294 ± 0.0002	80.49 ± 0.46	0.0183 ± 0.0017	47.89 ± 5.16	41.94 ± 9.57	46.25 ± 6.83
NAC-05sFC	95.13 ± 0.14	88.23 ± 1.44	0.0353 ± 0.0005	0.0272 ± 0.0001	75.99 ± 8.81	0.0263 ± 0.0039	42.58 ± 4.53	32.59 ± 10.80	38.76 ± 9.11
NAC-RW	95.99 ± 0.47	88.90 ± 1.43	0.0314 ± 0.0006	0.0153 ± 0.0001	79.75 ± 0.84	0.0178 ± 0.0010	45.88 ± 8.44	**24.55 ± 8.85**	37.79 ± 9.93
NAC-Euler	95.67 ± 0.26	89.70 ± 1.54	0.0352 ± 0.0009	**0.0148 ± 0.0001**	**80.61** ± 0.28	0.0181 ± 0.0017	42.08 ± 6.14	28.46 ± 8.18	39.32 ± 9.15
NAC-Steady	95.75 ± 0.28	**89.79 ± 1.67**	0.0266 ± 0.0007	0.0150 ± 0.0001	**80.62 ± 0.26**	0.0181 ± 0.0012	40.95 ± 5.77	26.76 ± 7.36	37.12 ± 12.43
NAC (main)	**96.35** ± 0.11	**90.11 ± 0.79**	**0.0258 ± 0.0012**	**0.0149 ± 0.0001**	80.59 ± 1.82	**0.0173 ± 0.0006**	**37.75 ± 4.72**	**22.87 ± 1.75**	**27.82 ± 7.09**

Note: (↑) higher is better; (↓) lower is better. *NAC (main)* uses Exact mode with Top-$K = 8$ and 50% sparsity.

Event-based MNIST: Event-based MNIST is the transformation of the widely recognized MNIST dataset with irregular sampling added originally proposed in Lechner and Hasani (2022). The transformation was performed in two steps: (i) flattening each 28×28 image into a time-series of length 784 and (ii) encoding the binary time-series into an event-based format by collapsing consecutive identical values (e.g., 1,1,1,1 → (1, t = 4)). This representation requires models to handle temporal dependencies effectively. NAC-PW achieves an accuracy of 96.64%, followed by NAC (main) at 96.35%. OT-Transformer ranked third with 96.30%.

Person Activity Recognition (PAR): We employed the Localized Person Activity dataset from UC Irvine (Vidulin et al., 2010). The dataset contains data from five participants, each equipped with inertial measurement sensors sampled every 211 ms. The goal of this experiment is to predict a person's activity from a set of predefined actions, making it a classification task. All models performed well on this task, with NAC-PW achieving 90.18% accuracy and taking first place. NAC (main) ranked second with 90.11% accuracy, while NAC-Steady ranked third with 89.79% mean accuracy.

4.3.2. Long-range dependencies (LRD)

To evaluate NAC's ability to capture LRD, we conduct experiments on two real-world multivariate time series datasets: (i) Electricity Transformer (ETTm1) (Zhou et al., 2021) and (ii) Jena Climate (Biogeochemistry, 2017). Prior to training, both datasets are normalized using `MinMaxScalar`, and forecasting performance is measured using MSE. We restrict the qualitative illustration of results to Transformer-based models, namely SDPA, Contiformer, ODEFormer, OT-Transformer and NAC (main) in Fig. 3.

ETTm1: ETTm1 consists of electricity transformer measurements recorded at 15-min intervals, including load and oil temperature variable that capture system dynamics. We train each model on the first 12 h (48 intervals) of sequence and forecast the subsequent 6 h (24 intervals). NAC (main) took first place with the lowest error at 0.0258, followed by CTA at 0.0266, and NAC-ϕ_{linear} at 0.0273. Fig. 3(A) shows that NAC significantly performs better in the highlighted region while competition diverges.

Jena Climate: Jena Climate consists of atmospheric variables recorded at 10-min intervals, including 14 variables such as temperature, pressure, humidity, and wind measurements. We train each model on the first 8 h (48 intervals) of data and forecast the subsequent 4 h (24 intervals). NAC-Euler and NAC-32k took first place with the lowest error at 0.0148, followed by NAC (main) at 0.0149, and CT-RNN at 0.0153. Fig. 3(B) shows the results for Jena-Climate and it shows that NAC performs considerably well against competition.

4.3.3. Lane keeping of autonomous vehicles

Lane keeping in autonomous vehicles (AVs) is a fundamental problem in robotics and AI. Early works (Park et al., 2021; Tiang et al., 2018) focused primarily on accuracy, often relying on large models. More recent research (Lechner et al., 2020; Razzaq & Hongwei, 2023) has shifted toward designing compact architectures suitable for resource-constrained devices. The goal of this experiment is to exploit NAC as

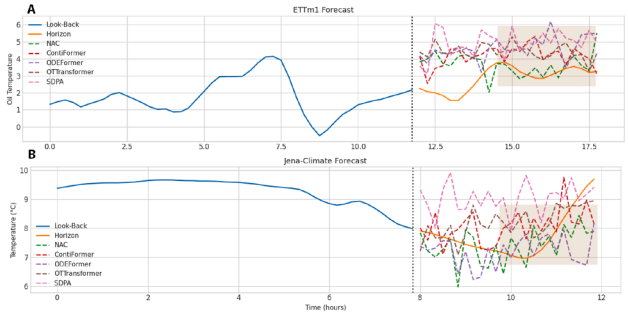

Fig. 3. Illustration of forecasting **(A)** ETTm1 (oil temperature); **(B)** jena-climate (atmosphere temperature).

a step-wise control module for AVs that creates a temporal structure between the road's horizon and the corresponding steering commands. For evaluation, we used two widely adopted simulation environments: (i) OpenAI CarRacing (Brockman et al., 2016); and (ii) the Udacity Self-Driving Car Simulator (URL). In OpenAI CarRacing, the task is to classify steering actions from a predefined action set. In contrast, the Udacity Simulator requires the prediction of a continuous trajectory of steering values. We implemented the AI models proposed by Razzaq and Hongwei (2023), replacing the recurrent layer with NAC and its counterparts. All models achieved approximately 80% accuracy on average in the OpenAI CarRacing benchmark. Notably, the NAC-PW performed the best, reaching the highest accuracy of 80.72%, followed by NAC-Steady, ranked second with 80.62%. NAC-Euler take third position, achieving 80.61% on average. For the Udacity benchmark, NAC-32k performed the best, achieving the lowest MSE of 0.0170. NAC (main) followed by 0.0173, and ContiFormer ranked third with 0.0174.

Closed-loop Test: Beyond prediction accuracy, we performed a closed-loop control experiment on CarRacing to evaluate Transformer models, namely SDPA, ODEFormer, ContiFormer, OT-Transformer, and NAC (main), to better understand their action dynamics. Fig. 4(A) illustrates the driving track. Fig. 4(B) shows the trajectories produced by each model, along with the corresponding position error relative to the centerline of the track. In the highlighted region, NAC maintains a substantially lower error and closely follows the centerline, whereas competition deviates from the track, in some cases leaving the track entirely (Fig. 4(C)). To further interpret model behavior, we apply Grad CAM (Selvaraju et al., 2017) to visualize the spatial regions influencing the decision. NAC and ContiFormer consistently attended to the road's horizon, indicating stable task-relevant focus. In contrast, OT-Transformer exhibits attention drift outside road boundaries while ODE-Former and SDPA show weaker and less consistent activation, suggesting reduced confidence in salient regions (Fig. 4(D). We additionally assess robustness under noise by exploiting the CarRacing built-in stochastic variations, which alter background color and texture, while preserving track geometry. Each model is evaluated over 10 runs of 10 episodes, and we report the mean success rate(Fig. 4(E)). NAC achieves the highest performance (70%), followed by ContiFormer (55%), with all other methods remaining below 50%. We attributed NAC's improved robustness to its dynamical properties, which resemble a low-pass filtering effect and help suppress high-frequency perturbations.

4.3.4. Industry 4.0

Industry 4.0 has revolutionized manufacturing, rendering prognostic health management (PHM) systems indispensable. A key PHM task is estimating the remaining useful life (RUL) of components, particularly rolling element bearings (REB), which account for 40-50% of machine failures (Ding et al., 2021; Zhuang et al., 2021). The objective is to learn degradation features from one operating condition of a dataset and generalize them to unseen conditions within the same dataset. Furthermore, the model must learn long-term temporal dependencies to accurately capture degradation behavior over time. In addition, it should provide accurate RUL estimation on entirely different datasets while maintaining a compact architecture suitable for resource-constrained devices, thereby supporting localized safety.

We utilized three benchmark datasets: (i) PRONOSTIA (Nectoux et al., 2012), (ii) XJTU-SY (Wang et al., 2018), and (iii) HUST (Thuan & Hong, 2023). Training is performed on PRONOSTIA, while XJTU-SY and HUST are used to assess zero-shot validation. We used the Score metric (Nectoux et al., 2012) to assess the performance. We evaluate generalization using *Bearing 1* from the first operating condition of each dataset. Fig. 5 visualizes the expected degradation alongside the outputs of NAC. For the PRONOSTIA dataset, NAC (main) performed the best, with the lowest score of 37.75. NAC-PW and OT-Transformer achieved nearly identical scores, with average values of 37.77 and 37.82, respectively. For the XJTU-SY dataset, NAC (main) has the lowest score of 22.87. NAC-RW ranked second with 24.55, followed by CTA with 25.86. A similar trend was observed on the HUST dataset, where NAC (main) achieved first place with a score of 27.82, NAC-PW ranked second with 30.14, and NAC-ϕ_{linear} ranked third with 35.89. These results demonstrated the strong cross-validation adaptability of NAC.

4.3.5. Interpreting results

The optimal NAC configuration depends on the temporal characteristics of each domain. For E-MNIST and PAR, NAC-PW performs best, as each query-key pair carries highly discriminative information and sparsity could remove valuable signals. For ETTm1 and Jena Climate, Exact mode with higher Top-K captures slow-forming dependencies across diffuse correlations. In autonomous driving, CarRacing relies mostly on steady-state logits since the CNN front-end captures most spatial features, whereas Udacity driving benefits from NAC-32k to retain peripheral road information from side cameras. Industrial prognostics naturally favor Exact mode, given the monotonic bearing degradation that resembles ODE-like convergence. Across all tasks, NAC-Exact with Top-$K = 8$ and 50% sparsity consistently ranks among the top three configurations.

4.4. NAC enhances interpretability

Interpretability refers to the process of providing explanations that are understandable to humans, either through visualization or analysis at the neuron cell level. We analyze NAC's neuron outputs at the cell level on OpenAI CarRacing trained with NAC-Steady, starting from the sensory gates, including the q, k, and v projections, and extending through the backbone inter-command-motor neurons to the computation of the attention logits. Fig. 6 illustrates the cell-level neuron states of NAC. We observe that ω_τ couples its values between 0.6 and 0.85, depending on the input scene. When the car turns left or left-straight, the ω_τ value remains below 0.65. When it turns right or right-straight, ω_τ is maintained above 0.8. In intermediate cases, the model drives straight. This behavior indicates that the evolution of ω_τ changes over time, confirming its input dependency. Similarly, the attention logits reflect this behavior, with a tighter range for left and left-straight actions (below 0.70), higher values for right and right-straight actions (above 0.74), and intermediate values corresponding to straight driving. These cell-level neuron states highlight how activations depend on the input, providing a way to understand and evaluate safety-critical decisions.

4.5. Analyzing scalability of NAC

Efficiency and Complexity: Table 2 summarizes the computational complexity of different sequence models. For sequence prediction over length n with hidden dimension k, CT-RNN and LNN scale $\mathcal{O}(nk \cdot S)$, while SDPA scale quadratically, $\mathcal{O}(n^2 k)$. NAC scale sub quadratically $\mathcal{O}(n\sqrt{n}k)$. For single-time-step prediction, CT-RNNs, LNN require $\mathcal{O}(k \cdot S)$, whereas SDPA and NAC require $\mathcal{O}(nk)$ when recomputing attention over the full sequence. This places NAC at intermediate

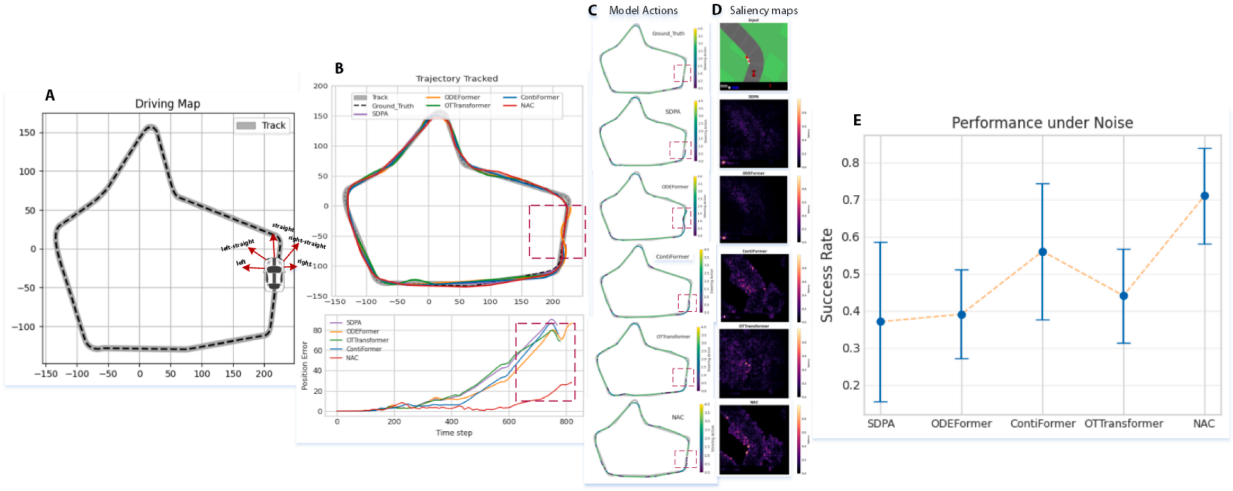

Fig. 4. Closed-loop test results visualization. **(A)** driving map; **(B)** Trajectory tracked and relative position error; **(C)** Model actions/behavior; **(D)** Saliency maps; **(E)** Noise test.

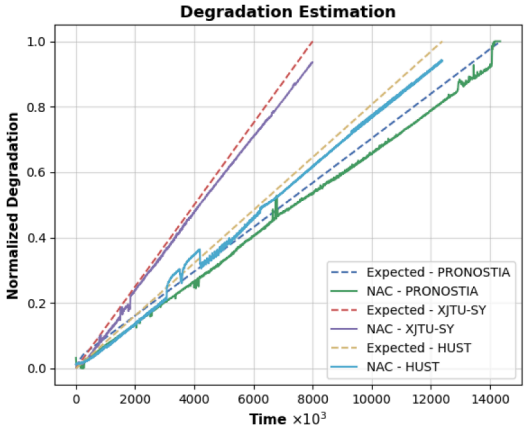

Fig. 5. Degradation estimation results.

Table 2
Sequence and time-step prediction complexity. n is the sequence and k is the hidden/model dimension. S is number of ODE solver steps.

Model	Sequence	Time-step
CT-RNN	$\mathcal{O}(nk \cdot S)$	$\mathcal{O}(k \cdot S)$
LNN (ODEsolve)	$\mathcal{O}(nk \cdot S)$	$\mathcal{O}(k \cdot S)$
SDPA	$\mathcal{O}(n^2 k)$	$\mathcal{O}(nk)$
NAC (main)	$\mathcal{O}(n\sqrt{nk})$	$\mathcal{O}(nk)$

position.

Empirical Analysis: To analyze the scalability of NAC, we conduct run-time and memory test on fixed-length sequences of 1024 steps, 64-dimensional features, 4 heads, and a batch size of 1. Each model is subjected to 10 forward passes on Google Colab T4-GPU, and we report the mean runtime with standard deviation, throughput, and peak memory usage in Table 3. NAC occupies an intermediate position in runtime relative to several CT-RNN models, including GRU-ODE, CfCs, and LTCs variants. In terms of memory consumption, NAC uses significantly less memory than mTAN does, with NAC-2k being the least memory-consuming among the NAC ablations. Increasing the NAC sparsity from 20% to 90% has a minimal effect on memory, decreasing usage slightly from 151.54 MB to 150.85 MB. In contrast, decreasing the Top-K selection from $k = 32$ to $k = 2$ drastically reduces the memory consumption from 549 MB to 44.75 MB, demonstrating the flexibility of NAC. Compared to LTC-NCP and CfC-NCP architectures, NAC reduces computational cost, achieving approximately a 3× reduction in forward-pass time.

Table 3
Run-time and memory benchmark results.

Model	Run-Time (s)	Throughput (seq/s)	Peak Memory (MB)
CT-RNN	7.1097 ± 0.3048	0.141	0.67
GRU-ODE	12.2498 ± 0.0525	0.082	0.64
PhasedLSTM	4.9812 ± 0.272	0.201	0.80
mmRNN	7.5852 ± 0.2785	0.132	0.96
LTC-FC	14.643 ± 0.2445	0.068	0.99
CfC-FC	6.0988 ± 0.2135	0.164	0.76
LTC-NCP	21.182 ± 0.2383	0.05	0.56
CfC-NCP	21.239 ± 0.3327	0.05	0.57
mTAN	0.2721 ± 0.0054	36.76	790.16
ODEFormer	0.3171 ± 0.0016	31.55	167.71
CTA	8.5275 ± 0.2355	0.117	1.43
ContiFormer	0.662 ± 0.0075	15.15	67.71
PDE-Attention	0.1299 ± 0.0026	0.50	163.85
OT-Transformer	0.768 ± 0.0043	0.13	66.67
NAC-2k	7.3071 ± 0.1547	0.137	44.75
NAC-32k	7.2313 ± 0.219	0.138	549.86
NAC-09s	7.222 ± 0.176	0.139	150.85
NAC-02s	7.252 ± 0.2018	0.138	151.54
NAC-Euler	7.3367 ± 0.1719	0.136	152.22
NAC-Steady	7.2942 ± 0.1451	0.137	150.86
NAC (main)	7.4101 ± 0.1586	0.135	151.50

4.6. Analyzing impact of key hyperparameters

We now examine how key hyperparameters of NAC including attention heads (1–32), sparsity (0.1-0.9), Top-K (2–32) selection relative to sequence lengths (100–50000) affect NAC performance on irregular spiral regression, implemented based on instruction in Chen et al. (2023). All experiments are conducted in Exact mode with a fixed model dimension $d_{\text{model}} = 64$. We report both MSE and MAE for interpolation and extrapolation, averaging results over five random initializations and including the corresponding standard deviations. Increasing heads from 1 (no multihead) to 2 slightly reduces performance. Increasing to 4 heads improves accuracy around 37%, and remains stable (Fig. 7(A)). Changing sparsity has non-monotonc effect, with the lowest error at

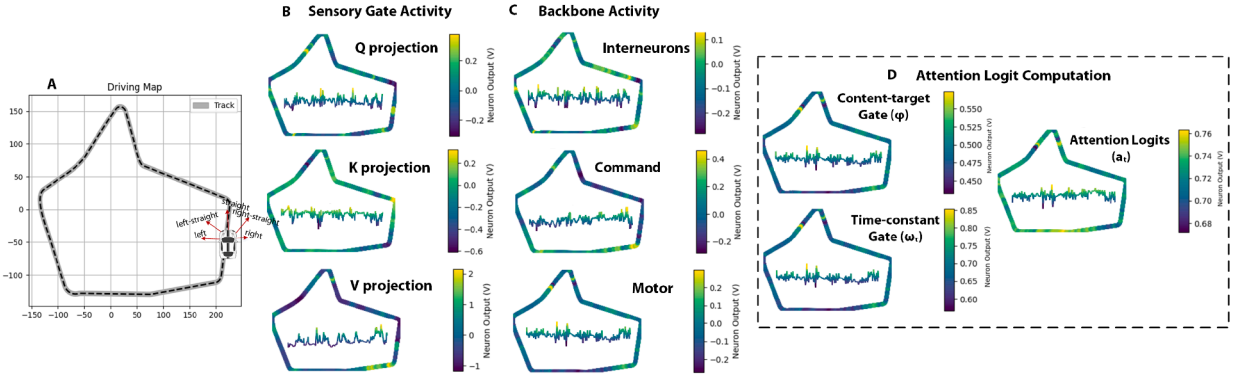

Fig. 6. Intuitive visualization of NAC cell activity during driving. **(A)** NAC-controlled car on the driving map. **(B)** Sensory gate neurons output (centroid) for the q, k, and v projections. **(C)** Backbone neuron activity (centroid), including interneurons, command neurons, and motor neurons. **(D)** ω_τ and ϕ and a_t computations. Internal plots show actual neuron activities.

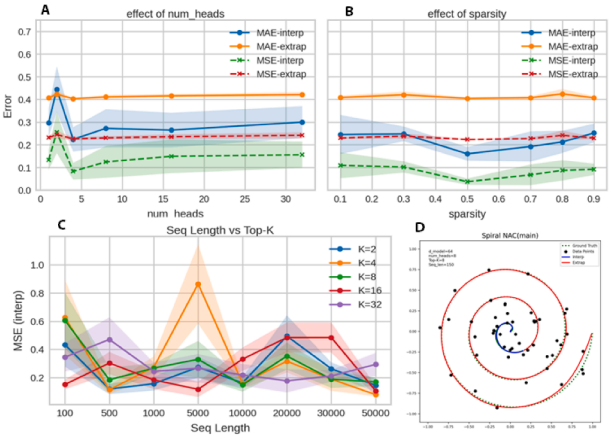

Fig. 7. Effect of key hyperparameters on interpolation and extrapolation performance (loss curves) for spiral trajectories: **(A)** heads; **(B)** sparsity; **(C)** Seq_length vs. Top-K; **(D)** Optimized Spiral.

50% (Fig. 7(B)). For Top-K selection across sequence length, results are mixed, but $8 <$ Top-$K < 16$ gives the most consistent performance, though the best value may depends on the task (Fig. 7(C)). Based on these results and Table 1, we recommend Top-$K = 8$, sparsity $= 50\%$, and Exact mode as the default main NAC configuration. Fig. 7(D) shows interpolation and extrapolation trajectories for this default setup, illustrating accurate and stable reconstructions of the spiral sequences.

5. Discussions

This paper is part of ongoing research on biologically plausible attention mechanisms and represents a pioneering step, with limitations to be addressed in future work.

Architectural improvement: Currently, NAC uses predetermined wiring (AutoNCP) requiring three inputs: the number of units (interneuron + motor), motor neurons, and sparsity. To integrate with the attention mechanism while preserving wiring, sensory units for $\mathcal{NN}_{\text{sensory}}$ are set as $\text{units}_{\text{sensory}} = \left\lceil \frac{d_{\text{model}} - 0.5}{0.6} \right\rceil$ and backbone units as $\text{units}_{backbone} = d_{\text{model}} + \left\lfloor \frac{d_{\text{model}}}{0.6} \right\rfloor$, where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denote the ceiling and floor functions, respectively. This results in a larger overall architectural size. Future work will support user-defined NCPs configurations to enable more efficient architectures.

Neuronal Stochastic Attention Circuit (NSAC): By reformulating the attention mechanism as a continuous-time evolution process, NAC enables the direct application of Neural ODE frameworks to attention. Extending this formulation by replacing the deterministic ODE with a stochastic differential equation (SDE) is a natural progression, equipping the model with principled uncertainty estimates intrinsic to the attention dynamics.

6. Conclusion

In this paper, we introduce the Neuronal Attention Circuit (NAC), a biologically inspired attention mechanism that reformulates attention logits as the solution to a first-order ODE modulated by nonlinear, interlinked gates derived from repurposing *C.elegans* nematode NCPs. NAC bridges discrete attention with CT dynamics, enabling adaptive temporal processing without the limitations inherent to traditional scaled dot-product attention. Based on the solution to ODE, we introduce three computational modes: (i) Euler, based on *explicit Euler* integration; (ii) Exact, providing closed-form solutions; and (iii) Steady, approximating equilibrium states. In addition, an adaptable sparse Top-K pairwise concatenation mechanism is implemented to mitigate the quadratic memory footprint. Theoretically, we establish NAC's logit-state stability and exponential and frozen approximation error bounds, thereby providing rigorous guarantees of convergence and expressiveness. Empirically, we demonstrated that NAC achieves state-of-the-art performance across diverse tasks, including irregularly time-series, long-range forecasting, autonomous vehicle lane-keeping, and industrial prognostics, while being interpretable at neuron cell-level.

CRediT authorship contribution statement

Waleed Razzaq: Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Data curation; **Yun-Bo Zhao:** Supervision, Resources, Project administration, Conceptualization.

Reproducibility Statement

The code is available at:
Tensorflow: https://github.com/itxwaleedrazzaq/keras-nac
PyTorch: https://github.com/itxwaleedrazzaq/torch-nac.

Ethics approval

This study was conducted in accordance with ethical standards.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was supported by the CAS-ANSO Scholarship. We acknowledge the intellectual and material contributions of the University of Science and Technology of China (USTC) and the Alliance of International Science Organizations (ANSO). AI/LLM tools were used to polish the writing of the manuscript under strict human supervision.

Appendix A. Preliminaries

A.1. Neuronal circuit policies (NCPs)

Neuronal Circuit Policies (NCPs) provide a biologically inspired approach for creating interpretable neural control agents, derived from the tap-withdrawal (TW) neural circuit of the nematode *C. elegans* (Lechner et al., 2018). In contrast to conventional spiking neural networks, most neurons in this circuit exhibit electronic dynamics, characterized by the passive propagation of electrical charges and resulting in graded membrane potentials. The NCP architecture is organized into a four-layer hierarchy: sensory neurons (N_s), touch interneurons (N_i), command interneurons (N_c), and motor neurons (N_m). The sensory neurons (N_s) detect external environmental states and initiate signal processing. For each input state variable x, an N_s includes positive (S_p) and negative (S_n) subneurons, with their membrane voltages modulated based on the sign and magnitude of x: S_p depolarizes for $x > 0$ (remaining at rest for $x \leq 0$), while S_n depolarizes for $x < 0$ (remaining at rest for $x \geq 0$). This input x is nonlinearly mapped to membrane potentials in the range $[-70\,\text{mV}, -20\,\text{mV}]$, aligning with biological neuron properties where $-70\,\text{mV}$ represents the resting potential and $-20\,\text{mV}$ approximates the peak activation level. Similarly, each motor neuron (N_m) comprises positive (M_p) and negative (M_n) subneurons, whose membrane potentials are inversely mapped to produce the output action y (as $y = y_p + y_n$), ensuring outputs remain within task-specific bounds while preserving biological voltage scales. The NCP wiring emulates the sparse and modular nature of biological neural circuits, featuring feedforward connections from N_s to N_i, recurrent interactions primarily among N_c (enabling competitive dynamics for action selection), and targeted projections from N_c to N_m (Lechner et al., 2018). Fig. 2(a) depicts the NCP connectome.

A.2. SDPA mechanism

SDPA mechanisms have become a cornerstone in modern neural architectures, enabling models to dynamically focus on relevant parts of the input. The concept was first introduced in the context of neural machine translation, where it allows the decoder to weight encoder outputs according to their importance for generating each target token. Given a query vector $q \in \mathbb{R}^d$, key vectors $K = [k_1, k_2, \ldots, k_n] \in \mathbb{R}^{n\times d}$, and value vectors $V = [v_1, v_2, \ldots, v_n] \in \mathbb{R}^{n\times d}$, the attention mechanism can be expressed in two steps:

1. Compute the scaled dot attention logits:

$$a_i = \frac{q^T k_i}{\sqrt{d}} \tag{20}$$

2. Normalize the logits to obtain attention weights and compute the output:

$$\alpha_i = \text{softmax}(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{n} e^{a_j}} \tag{21}$$

$$\text{Attention}(q, k, v) = \sum_{i=1}^{n} \alpha_i v_i \tag{22}$$

Here, a_i is the raw attention logit between the query and each key, and the scaling factor $\sqrt{d}$ prevents large dot products from destabilizing the Softmax (Vaswani et al., 2017).

Appendix B. Proofs

In this section, we provide all the proofs.

B.1. Analyzing closed-form solution

B.1.1. Deriving closed-form (Exact) solution

Although ϕ and ω_τ are nonlinear functions of the input $\mathbf{u} = [\mathbf{q}; \mathbf{k}]$, we derive a closed-form solution by treating them as locally constant over the small integration interval for each query–key pair based on frozen-coefficient approximation (John, 1952). Rewrite Eq. (1) as

$$\frac{da_t}{dt} + \omega_\tau a_t = \phi. \tag{23}$$

The integrating factor is

$$\mu = e^{\left(\int \omega_\tau \, dt\right)} = e^{\omega_\tau t}. \tag{24}$$

Multiply both sides by $\mu(t)$:

$$e^{\omega_\tau t}\frac{da_t}{dt} + \omega_\tau e^{\omega_\tau t} a_t = \phi e^{\omega_\tau t}. \tag{25}$$

Recognizing the left-hand side as the derivative of $e^{\omega_\tau t} a_t$:

$$\frac{d}{dt}\left(e^{\omega_\tau t} a_t\right) = \phi e^{\omega_\tau t}. \tag{26}$$

Integrate from 0 to T:

$$e^{\omega_\tau t} a_t - e^0 a_0 = \phi \int_0^t e^{\omega_\tau s}\, ds. \tag{27}$$

Compute the integral:

$$\int_0^t e^{\omega_\tau s}\, ds = \frac{1}{\omega_\tau}\left(e^{\omega_\tau t} - 1\right). \tag{28}$$

Substitute back:

$$e^{\omega_\tau t} a_t - a_0 = \phi \cdot \frac{e^{\omega_\tau t} - 1}{\omega_\tau}. \tag{29}$$

Rearrange:

$$e^{\omega_\tau t} a_t = a_0 + \frac{\phi}{\omega_\tau}\left(e^{\omega_\tau t} - 1\right). \tag{30}$$

Divide both sides by $e^{\omega_\tau t}$:

$$a_t = a_0 e^{-\omega_\tau t} + \frac{\phi}{\omega_\tau}\left(1 - e^{-\omega_\tau t}\right). \tag{31}$$

Set $a^* := \frac{\phi}{\omega_\tau}$. Then $a_t = a^* + (a_0 - a^*)e^{-\omega_\tau t}$, is proved.

B.1.2. Frozen coefficient error analysis

In this subsection, we analyze the resulting error due to the frozen approximation with respect to the time-varying behavior of ϕ and ω_τ, both mathematically and empirically.

Theorem 2 (Frozen-Coefficient Approximation Error). *Consider the attention logit ODE*

$$\frac{da(t)}{dt} = -\omega_\tau(t)a(t) + \phi(t), \tag{32}$$

with time-varying coefficients $\phi(t)$ and $\omega_\tau(t) > 0$. Let $a_{varying}(t)$ denote the solution with varying coefficients (numerically integrated) and $a_{frozen}(t)$ denote the exact closed-form solution obtained by freezing the coefficients at their initial values $\phi_0 = \phi(0)$, $\omega_{\tau 0} = \omega_\tau(0)$ over an integration window [0, T]. Assume that $\phi(t)$ and $\omega_\tau(t)$ are Lipschitz continuous in t with constants L_ϕ and L_ω (i.e., $|\phi(t) - \phi(s)| \leq L_\phi|t - s|$, $|\omega_\tau(t) - \omega_\tau(s)| \leq L_\omega|t - s|$). Define $C = |a_0| + |\phi_0|/\omega_{\tau 0}$. Then, for any $t \in [0, T]$, the error $\varepsilon(t) = a_{varying}(t) - a_{frozen}(t)$ satisfies

$$|\varepsilon(t)| \leq \frac{L_\phi + CL_\omega}{L_\omega}\left(e^{L_\omega t^2/2} - 1\right), \tag{33}$$

For small $L_\omega t^2 \ll 1$, this simplifies to $\mathcal{O}(t^2)$.

Proof. The frozen-coefficient solution satisfies

$$\frac{da_{\text{frozen}}}{dt} = -\omega_{\tau 0} a_{\text{frozen}} + \phi_0, \quad a_{\text{frozen}}(0) = a(0), \tag{34}$$

Subtracting Eq. (34) from Eq. (32) yields the error dynamics

$$\frac{d\varepsilon}{dt} = -\omega_{\tau 0}\varepsilon + \delta\phi(t) + \delta\omega_\tau(t) a_{\text{varying}}(t), \tag{35}$$

where $\delta\phi(t) = \phi(t) - \phi_0$ and $\delta\omega_\tau(t) = \omega_{\tau 0} - \omega_\tau(t)$. By Lipschitz, $|\delta\phi(t)| \le L_\phi t$ and $|\delta\omega_\tau(t)| \le L_\omega t$. Using the integrating factor $e^{\omega_{\tau 0} t}$,

$$\frac{d}{dt}\left(e^{\omega_{\tau 0} t}\varepsilon(t)\right) = e^{\omega_{\tau 0} t}\left(\delta\phi(t) + \delta\omega_\tau(t) a_{\text{varying}}(t)\right), \tag{36}$$

Integrating from 0 to T and noting that $\varepsilon(0) = 0$,

$$\varepsilon(t) = \int_0^t e^{-\omega_{\tau 0}(t-s)}\left(\delta\phi(s) + \delta\omega_\tau(s) a_{\text{varying}}(s)\right) ds \tag{37}$$

The frozen solution from Eq. (31) is

$$a_{\text{frozen}}(s) = a_0 e^{-\omega_{\tau 0} s} + \frac{\phi_0}{\omega_{\tau 0}}(1 - e^{-\omega_{\tau 0} s}), \tag{38}$$

so that $|a_{\text{frozen}}(s)| \le C$. Hence $|a_{\text{varying}}(s)| \le C + |\varepsilon(s)|$. Taking absolute values in Eq. (37) and using Lipschitz bounds,

$$|\varepsilon(t)| \le \int_0^t e^{-\omega_{\tau 0}(t-s)}\left(L_\phi s + L_\omega s(C + |\varepsilon(s)|)\right) ds, \tag{39}$$

Since $e^{-\omega_{\tau 0}(t-s)} \le 1$,

$$|\varepsilon(t)| \le \frac{1}{2}(L_\phi + C L_\omega) t^2 + L_\omega \int_0^t s|\varepsilon(s)| ds \tag{40}$$

Applying Gronwall's inequality to $|\varepsilon(t)|$ yields

$$|\varepsilon(t)| \le \frac{L_\phi + C L_\omega}{L_\omega}\left(e^{L_\omega t^2/2} - 1\right), \tag{41}$$

which is Eq. (33). Expanding for $L_\omega t^2/2 \ll 1$ gives the leading term $\frac{1}{2}(L_\phi + C L_\omega) t^2 = \mathcal{O}(t^2)$. □

Empirical Analysis: We now empirically analyze the error introduced by freezing $\phi(t)$ and $\omega_\tau(t)$. We evaluate the closed-form frozen solution against continuously varying coefficients. Empirical analysis across four regimes: (i) slow variation (gradual linear change); (ii) fast variation (exponential growth); (iii) step changes (abrupt discontinuities); and (iv) oscillatory behavior (periodic modulation) shows that the frozen approximation maintains bounded error even under substantial parameter variation. Table B.1 summarizes the empirical results. For typical slow-varying irregular sequences, with ϕ varying by 0.5% and ω_τ varying by 0.26%, the frozen solution exhibits minimal error, with a mean of 1.87%, and a maximum of 3.05%. Even under more challenging regimes, the error remains predictable and bounded, reaching 6.41% for fast variation, 6.53% for step changes, and 7.81% for oscillatory behavior. An illustration of oscillatory evolution is shown in Fig. B.1.

Interpretation: Given that the variation rates in real-world irregular sequences rarely exceed 10%, these results confirm the adequacy of the frozen approximation for practical deployment. This conclusion is further supported by the real-data mode ablation results in Table 1, where the *Exact* mode and the numerically integrated Euler variant achieve nearly identical performance on the Jena Climate benchmark (MSE: 0.0149 vs. 0.0148) and comparable results across the remaining datasets. The absence of a consistent advantage for Euler integration indicates that the approximation error introduced by coefficient freezing is negligible in practice. We therefore recommend Exact as the default computation mode for NAC while preserving adaptive Euler integration for edge cases with exceptionally rapid dynamics.

B.2. Proof of Theorem 1

Following the footprints of Hasani et al. (2021), we divide the proof into two parts: (i) the single-connection case $M = 1$, and (ii) the general multi-connection case $M > 1$. The main technique is to evaluate the ODE at boundary values of the proposed invariant interval and show that the derivative points inward, ensuring that trajectories cannot escape.

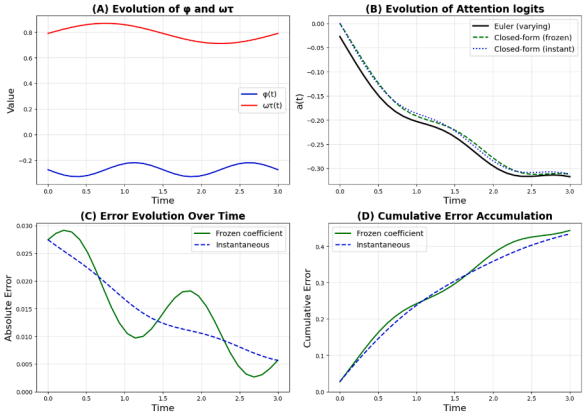

Fig. B.1. Visualization of oscillatory evolution of NAC.

Table B.1
Empirical error analysis of frozen-coefficient approximation across variation regimes.

Variation	mean (%)	max (%)	ϕ (%)	ω_τ (%)
Slow	1.87	3.05	0.50	0.26
Fast	6.41	6.50	2.63	1.58
Step	6.53	7.20	3.51	2.11
Oscillatory	7.81	6.90	8.28	2.10
Average	**5.65**	**5.18**	**3.73**	**1.51**

Case 1: single connection ($M = 1$)

The ODE reduces to

$$\frac{da}{dt} = -\omega_\tau a + \phi = -\omega_\tau(a - A), \quad A = \frac{\phi}{\omega_\tau}. \tag{42}$$

Here A is the unique equilibrium. We now check both bounds.

- *Upper bound:* Let $M = \max(0, A)$. At $a = M$,

$$\left.\frac{da}{dt}\right|_{a=M} = -\omega_\tau(M - A). \tag{43}$$

If $A \ge 0$, then $M = A$ and $\frac{da}{dt} = 0$. If $A < 0$, then $M = 0$, so $\frac{da}{dt} = -\omega_\tau(0 - A) = \omega_\tau A \le 0$ since $A < 0$. In both cases, $\frac{da}{dt} \le 0$. Thus, trajectories cannot cross above M.

- *Lower bound:* Let $m = \min(0, A)$. At $a = m$,

$$\left.\frac{da}{dt}\right|_{a=m} = -\omega_\tau(m - A). \tag{44}$$

If $A \le 0$, then $m = A$ and $\frac{da}{dt} = 0$. If $A > 0$, then $m = 0$, so $\frac{da}{dt} = -\omega_\tau(0 - A) = \omega_\tau A \ge 0$. In both cases, $\frac{da}{dt} \ge 0$. Thus trajectories cannot cross below m. Therefore, the interval $[m, M]$ is forward-invariant.

To see this explicitly under Euler discretization with step size $\Delta t > 0$,

$$a(t + \Delta t) = a_t + \Delta t \cdot \frac{da}{dt}. \tag{45}$$

At $a = M$, $\frac{da}{dt} \le 0 \implies a(t + \Delta t) \le M$. At $a = m$, $\frac{da}{dt} \ge 0 \implies a(t + \Delta t) \ge m$. By induction over steps, $a_t \in [m, M]$ for all $t \in [0, T]$.

Case 2: multiple connections ($M > 1$)

The ODE is

$$\frac{da}{dt} = -\left(\sum_{j=1}^{M} f_j\right) a + \sum_{j=1}^{M} f_j A_j, \tag{46}$$

with per-connection equilibria $A_j = \phi_j / f_j$. The effective equilibrium is

$$A = \frac{\sum_{j=1}^{M} f_j A_j}{\sum_{j=1}^{M} f_j}. \tag{47}$$

Since the weights $\frac{f_j}{\sum f_j}$ are positive and sum to 1, A is a convex combination of $\{A_j\}$. Therefore,

$$A \in [A^{\min}, A^{\max}]. \tag{48}$$

- *Upper bound:* Let $M = \max(0, A^{\max})$. Then

$$\frac{da}{dt}\Big|_{a=M} = \sum_{j=1}^{M} f_j(A_j - M). \tag{49}$$

Since $A_j \le A^{\max} \le M$, each term $(A_j - M) \le 0$, and thus $\sum f_j(A_j - M) \le 0$. Hence $\frac{da}{dt} \le 0$, proving trajectories cannot exceed M.

- *Lower bound:* Let $m = \min(0, A^{\min})$. Then

$$\frac{da}{dt}\Big|_{a=m} = \sum_{j=1}^{M} f_j(A_j - m). \tag{50}$$

Since $A_j \ge A^{\min} \ge m$, each $(A_j - m) \ge 0$, so $\sum f_j(A_j - m) \ge 0$. Hence $\frac{da}{dt} \ge 0$, proving trajectories cannot fall below m. Thus, the interval $[m, M]$ is forward-invariant.

Remark 1. This result guarantees that the continuous-time attention state converges within a well-defined interval dictated by the per-connection equilibria. In particular, for the single-connection case ($M = 1$), the state trajectory converges monotonically toward the closed-form equilibrium solution (Eq. (18)) without overshoot.

Appendix C. Training, gradients and complexity

C.1. Gradient characterization

We analyze the sensitivity of the dynamics with respect to the underlying learnable parameters. Specifically, we compute closed-form derivatives of both the steady state and the full trajectory a_t with respect to the parameters ϕ and ω_τ. These expressions illuminate how gradients flow through the system, and provide guidance for selecting parameterizations that avoid vanishing or exploding gradients.

C.1.1. Trajectory sensitivities for closed-form formulation

The trajectory is given by

$$a_t = a^* + (a_0 - a^*)e^{-\omega_\tau t}, \tag{51}$$

which depends on (ϕ, ω_τ) both through the equilibrium a^* and the exponential term.

Derivative with respect to ϕ: We obtain

$$\frac{\partial a_t}{\partial \phi} = \frac{1 - e^{-\omega_\tau t}}{\omega_\tau} \tag{52}$$

Interpretation: For large ω_τ, the gradient with respect to ϕ saturates quickly but shrinks to scale $\mathcal{O}(1/\omega_\tau)$, potentially slowing learning of ϕ. Conversely, very small ω_τ leads to large steady-state gradients, which may destabilize optimization.

Derivative with respect to ω_τ: Here, both the equilibrium and the decay rate depend on ω_τ, yielding

$$\frac{\partial a_t}{\partial \omega_\tau} = -\frac{\phi}{\omega_\tau^2}\left(1 - e^{-\omega_\tau t}\right) - (a_0 - a^*)\, t\, e^{-\omega_\tau t}. \tag{53}$$

Interpretation: The gradient with respect to ω_τ contains a transient term proportional to $te^{-\omega_\tau t}$, which dominates at intermediate times, and a steady-state contribution proportional to $-\phi/\omega_\tau^2$, which persists asymptotically. Thus, sensitivity to ω_τ is time-dependent, peaking before vanishing exponentially in the transient component.

C.2. Gradient-based training

Like Neural ODEs (Chen et al., 2018) and CT-RNNs (Rubanova et al., 2019), NAC produces differentiable computational graphs and can be trained using gradient-based optimization, such as the adjoint sensitivity method (Cao et al., 2003) or backpropagation through time (BPTT) (LeCun et al., 1988). In this work, we use BPTT exclusively, as the adjoint sensitivity method can introduce numerical errors (Zhuang et al., 2020).

Appendix D. Evaluation

D.1. Hyperparameter tuning

To reduce manual tuning and ensure well-chosen hyperparameters, we employ Bayesian optimization to tune all the hyperparameters. We optimize all key hyperparameters of the neural network, including the number of Conv1D layers, filters, kernel sizes, activation functions, NAC's d_{model}, num_heads, Top-K, sparsity. We also tune the Dense layer widths, optimizer choice, training batch size, and learning rate. The optimization is performed over 200 trials using *MSE* as the training objective. The best performing hyperparameter for all experiments are summarized in Table D.1. The brief overview of Bayesian Optimization is provided below.

D.1.1. Overview of Bayesian optimization

Bayesian Optimization (BO) is a sample-efficient strategy for hyperparameter tuning of neural networks, particularly useful when the objective function $f(\mathbf{x})$ is expensive to evaluate and gradients are unavailable. BO builds a probabilistic surrogate model using a Gaussian Process (GP), which provides a posterior mean $\mu(\mathbf{x})$ and uncertainty $\sigma(\mathbf{x})$ over the objective function values. To select promising hyperparameter configurations $\mathbf{x}$, BO maximizes an acquisition function, such as Expected Improvement (EI), defined by

$$\text{EI}(\mathbf{x}) = \big(f(\mathbf{x}^+) - \mu(\mathbf{x}) - \xi\big)\Phi(Z) + \sigma(\mathbf{x})\phi(Z), \tag{54}$$

where $f(\mathbf{x}^+)$ is the best observed function value, $\xi \ge 0$ controls the exploration-exploitation trade-off,

$$Z = \frac{f(\mathbf{x}^+) - \mu(\mathbf{x}) - \xi}{\sigma(\mathbf{x})}, \tag{55}$$

and Φ and ϕ are the cumulative distribution function (CDF) and probability density function (PDF) of the standard normal distribution, respectively. Iteratively, BO proposes new hyperparameters by maximizing EI, evaluates these with the true objective, and updates the surrogate model until a predefined stopping criterion is met, such as iterations (N) (Snoek et al., 2012). The full algorithm is provided in Algorithm 3.

Algorithm 3 Optimization for Hyperparameter Tuning.

Require: Objective function $f(\mathbf{x})$, Initial dataset $X_0 = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $y_i = f(\mathbf{x}_i)$, Number of iteration N, Exploration parameter $\xi \ge 0$
for $t = 1$ to N **do**
  Fit GP to X_{t-1}, get $\mu(\mathbf{x}), \sigma(\mathbf{x})$
  Acquisition function: $\text{EI}(\mathbf{x}) = (f(\mathbf{x}^+) - \mu(\mathbf{x}) - \xi)\Phi(Z) + \sigma(\mathbf{x})\phi(Z)$ where
  $Z = \frac{f(\mathbf{x}^+) - \mu(\mathbf{x}) - \xi}{\sigma(\mathbf{x})}, \quad f(\mathbf{x}^+) = \min_{\mathbf{x}_i \in X_{t-1}} y_i$
  Select $\mathbf{x}_t = \arg\max_{\mathbf{x}} \text{EI}(\mathbf{x})$
  Evaluate $y_t = f(\mathbf{x}_t)$
  Update $X_t = X_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$
end for
return $\mathbf{x}^+ = \arg\min_{\mathbf{x}_i \in X_N} y_i$

D.2. Experimental details

D.2.1. Event-based MNIST

Dataset Explanation and Curation: The MNIST dataset, introduced by Deng (2012), is a widely used benchmark for computer vision and image classification tasks. It consists of 70,000 grayscale images of handwritten digits (0-9), each of size 28×28 pixels, split into 60,000 training and 10,000 testing samples.

Preprocessing: We follow the preprocessing pipeline described in Lechner and Hasani (2022), which proceeds as follows. First, a threshold is

Table D.1
Summary of key hyperparameters of all experiments.

Param.	E-MNIST	PAR	LRD	CarRacing	Udacity	RUL EST.
Conv layers	2×**1D**(64@5)	**1D**(64@5, 64@3)	–	3×TD-**2D**(10–30@3–5)	5×**2D**(24–64@5–3, ELU)	2×**1D**(32@3, 16@2)
NAC	64-d, 8h	32-d, 4h	64-d, 16h	64-d, 16h	100-d, 20h	16-d, 8h
Seq_len	256	1	96	1	1	1000
Dense	32–10(SM)	32–11(SM)	–	64–5(SM)	64–1(Lin)	1(Lin)
Dropout	–	–	–	0.2	0.5	–
Opt.	AdamW	AdamW	AdamW	Adam	AdamW	AdamW
LR	0.001	0.001 0.001	0.0001	0.001	0.001	
Loss	SCE	SCE	MSE	SCE	MSE	MSE
Metric	Acc	Acc	MSE	Acc	MSE	Score
Batch	32	20	64	32	40	32
Epochs	150	100	50	100	10	150

Note: SCE = Sparse Categorical Crossentropy; Acc = Accuracy; MAE = Mean Absolute Error; MSE = Mean Squared Error; SM = softmax; Lin = Linear; TD = TimeDistributed; Conv1D/2D = Conv1D/2D; d = model dimension; h = attention heads.

applied to convert the 8-bit pixel values into binary values, with 128 as the threshold on a scale from 0 (minimum intensity) to 255 (maximum intensity). Second, each 28×28 image is reshaped into a one-dimensional time-series of length 784. Third, the binary time-series is encoded in an event-based format, eliminating consecutive occurrences of the same value; for example, the sequence $[1, 1, 1, 1]$ is transformed into $(1, t = 4)$. This encoding introduces a temporal dimension and compresses the sequences from 784 to an average of 53 time steps. Finally, to facilitate efficient batching and training, each sequence is padded to a fixed length of 256, and the time dimension is normalized such that each symbol corresponds to one unit of time. The resulting dataset defines a per-sequence classification problem on irregularly sampled time-series.
Neural Network Architecture: We develop an end-to-end hybrid neural network by combining compact convolutional layers with NAC or counterparts baselines for fair comparison. Detailed hyperparameters and architectural specifications are provided in Table D.1.

D.2.2. Person activity recognition (PAR)

Dataset Explanation and Curation: We used the Localized Person Activity Recognition dataset provided by UC Irvine (Vidulin et al., 2010). The dataset comprises 25 recordings of human participants performing different physical activities. The eleven possible activities are "walking," "falling," "lying down," "lying," "sitting down," "sitting," "standing up from lying," "on all fours," "sitting on the ground," "standing up from sitting," and "standing up from sitting on the ground." The objective of this experiment is to recognize the participant's activity from inertial sensors, formulating the task as a per-time-step classification problem. The input data consist of sensor readings from four inertial measurement units placed on participants' arms and feet. While the sensors are sampled at a fixed interval of 211 ms, recordings exhibit different phase shifts and are thus treated as irregularly sampled time-series.
Preprocessing: We first separated each participant's recordings based on sequence identity and calculated elapsed time in seconds using the sampling period. To mitigate class imbalance, we removed excess samples from overrepresented classes to match the size of the smallest class. Subsequently, the data were normalized using a standard scaler.
Neural Network Architecture: Following the approach in Section D.2.1, we developed an end-to-end neural network combining convolutional heads with NAC or other baselines. Hyperparameter details are summarized in Table D.1.

D.2.3. Multivariate long-range dependencies (LRDs)

Dataset Explanation and Curation: We utilized two well-known multivariate long-range datasets to evaluate NAC's capabilities in modeling long-range dependencies: (i) Electrical Transformer Temperature (ETTm1) and (ii) Jena Climate. ETTm1 contains 7 features and is used for long-term electricity transformer load forecasting. It provides data at 15-min intervals, capturing real-world temporal patterns for time-series analysis. Jena Climate contains 14 features capturing various atmospheric and environmental measurements. It provides data at 10-min intervals for modeling and forecasting climate-related time series.
Preprocessing: We used 12 h of data from ETTm1 to train NAC and the baseline models, and the following 6 h were forecasted. Similarly, we used 8 h of data from Jena Climate for training and the subsequent 4 h for forecasting. All features were normalized using a standard scaler before training.
Neural Network Architecture: The neural network consists of a single NAC layer with $d_{\text{model}} = 64$, 16 heads, Top-$K = 32$ and 50% sparsity. Detailed hyperparameters are provided in Table D.1.

D.2.4. Autonomous vehicle

Dataset Explanation and Curation: We followed the data collection methodology described in Razzaq and Hongwei (2023). For OpenAI-CarRacing, a Proximal Policy Optimization (PPO) trained agent (5M timesteps) was used to record 20 episodes, yielding approximately 48,174 RGB images of size $92 \times 92 \times 3$ with corresponding action labels across five discrete actions (no-act, move left, forward, move right, stop). For the Udacity simulator, we manually controlled the vehicle for 50 min, producing 15,647 RGB images of size $320 \times 160 \times 3$, captured from three camera streams (left, center, right) along with their corresponding continuous steering values.
Preprocessing: No preprocessing was applied to the OpenAI-CarRacing dataset. For the Udacity simulator, we followed the preprocessing steps in Shibuya (2017). Each image was first cropped to remove irrelevant regions and resized to $66 \times 120 \times 3$. Images were then converted from RGB to YUV color space to match the network input. To improve robustness, data augmentation techniques, including random flips, translations, shadow overlays, and brightness variations, were applied to simulate lateral shifts and diverse lighting conditions.
Neural Network Architecture: For OpenAI-CarRacing, we modified the neural network architecture proposed in Razzaq and Hongwei (2023), which combines compact CNN layers for spatial feature extraction with LNNs to capture temporal dynamics. In our implementation, the LNN layers were replaced with NAC and its comparable alternatives for fair evaluation. Full hyperparameter configurations are provided in Table D.1. For the Udacity simulator, we modified the network proposed in Bojarski et al. (2016) by replacing three latent MLP layers with NAC and its counterparts. Full hyperparameters are summarized in Table D.1.

D.2.5. Industry 4.0

Dataset Explanation and Curation: *PRONOSTIA dataset* is a widely recognized benchmark dataset in the field of condition monitoring and degradation estimation of rolling-element bearings. Nectoux et al. (2012) developed this dataset as part of the PRONOSTIA experimental platform. The dataset comprises 16 complete run-to-failure experiments performed under accelerated wear conditions across three different operating settings: 1800 rpm with 4 kN radial load, 1650 rpm

with a 4.2 kN load, and 1500 rpm with a 5 kN load, all at a frequency of 100 Hz. Vibration data were recorded using accelerometers placed along the horizontal and vertical axes, which were sampled at 25.6 kHz. Additionally, temperature readings were collected at a sampling rate of 10 Hz.

XJTU-SY Dataset is another widely recognized benchmark dataset developed through collaboration between Xi'an Jiaotong University and Changxing Sumyoung Technology (Wang et al., 2018). The dataset comprises 15 complete run-to-failure experiments performed under accelerated degradation conditions with three distinct operational settings: 1200 rpm (35 Hz) with a 12 kN radial load, 2250 rpm (37.5 Hz) with an 11 kN radial load, and 2400 rpm (40 Hz) with a 10 kN radial load. Vibrational signals were recorded using an accelerometer mounted on the horizontal and vertical axes and sampled at 25.6 kHz. This dataset is only used for the cross-validation test.

HUST Dataset is a practical dataset developed by Hanoi University of Science and Technology to support research on ball bearing fault diagnosis (Hong & Thuan, 2023). The dataset includes vibration data collected from five bearing types (6204, 6205, 6206, 6207, and 6208) under three different load conditions: 0 W, 200 W, and 400 W. Six fault categories were introduced, consisting of single faults (inner race, outer race, and ball) and compound faults (inner-outer, inner-ball, and outer-ball). Faults were created as early-stage defects in the form of 0.2 mm micro-cracks, simulating real degradation scenarios. The vibration signals were sampled at 51.2 kHz with approximately 10-second recordings for each case. This dataset is only used for the cross-validation test.

Preprocessing: Condition monitoring data comprises 1D non-stationary vibrational signals collected from multiple sensors. To extract meaningful information, these signals must be transformed into features that possess meaningful physical interpretability. We utilized the preprocessing proposed in Razzaq and Zhao (2025a), and physically plausible labels are generated according to Razzaq and Zhao (2025b). Initially, the signal is segmented into small, rectangularized vectors using a windowing technique, enabling better localization of transient characteristics. The continuous wavelet transform (CWT) with the Morlet wavelet as the mother wavelet is then applied to obtain a time-frequency representation (TFR). The CWT is defined as $\Gamma(a,b) = \int_{-\infty}^{\infty} x_w(t) \frac{1}{\sqrt{a}} \psi^* \left(\frac{t-b}{a} \right) dt$, where a and b denote the scale and translation parameters, respectively, and ψ is the Morlet wavelet function. From the TFR, a compact set of statistical and domain-specific features is extracted to characterize the operational condition of the bearing.

Neural Network Architecture: The objective of this problem is to design a compact neural network that can effectively model degradation dynamics while remaining feasible for deployment on resource-constrained devices, enabling localized and personalized prognostics for individual machines. To achieve this, we combine a compact convolutional network with NAC. The CNN component extracts spatial degradation features from the training data, while NAC performs temporal filtering to emphasize informative features. This architecture maintains a small model size without sacrificing representational capacity. Full hyperparameter configurations are reported in Table D.1.

Evaluation Metric: Score is a metric specifically designed for RUL estimation in the IEEE PHM (Nectoux et al., 2012) to score the estimates. The scoring function is asymmetric, penalizing overestimations more heavily than early predictions. This reflects practical considerations, as late maintenance prediction can lead to unexpected failures with more severe consequences than early intervention can.

$$Score = \sum_{i:\hat{y}_i < y_i} \left(e^{-\frac{\hat{y}_i - y_i}{13}} - 1 \right) + \sum_{i:\hat{y}_i \geq y_i} \left(e^{\frac{\hat{y}_i - y_i}{10}} - 1 \right) \tag{56}$$

References

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

Biogeochemistry, M. (2017). Jena climate time series dataset (2009–2017). https://www.kaggle.com/datasets/samehraouf/jena-climate-time-series-2009-2017-dataset.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J. et al. (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. arXiv preprint arXiv:1606.01540.

Cao, Y., Li, S., Petzold, L., & Serban, R. (2003). Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. *SIAM Journal on Scientific Computing*, *24*(3), 1076–1089.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in neural information processing systems*, *vol. 31*.

Chen, Y., Ren, K., Wang, Y., Fang, Y., Sun, W., & Li, D. (2023). Contiformer: Continuous-time transformer for irregular time series modeling. *Advances in Neural Information Processing Systems*, *36*, 47143–47175.

Chien, J.-T., & Chen, Y.-H. (2021). Continuous-time attention for sequential learning. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 7116–7124). (*vol. 35*).

Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

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.

d'Ascoli, S., Becker, S., Mathis, A., Schwaller, P., & Kilbertus, N. (2023). Odeformer: Symbolic regression of dynamical systems with transformers. arXiv preprint arXiv:2310.05573.

De Brouwer, E., Simm, J., Arany, A., & Moreau, Y. (2019). Gru-ode-bayes: Continuous modeling of sporadically-observed time series. In *Advances in neural information processing systems*, *vol. 32*.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, *29*(6), 141–142.

Ding, Y., Jia, M., Miao, Q., & Huang, P. (2021). Remaining useful life estimation using deep metric transfer learning for kernel regression. *Reliability Engineering & System Safety*, *212*, 107583.

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

Hasani, R., Lechner, M., Amini, A., Liebenwein, L., Ray, A., Tschaikowski, M., Teschl, G., & Rus, D. (2022). Closed-form continuous-time neural networks. *Nature Machine Intelligence*, *4*(11), 992–1003.

Hasani, R., Lechner, M., Amini, A., Rus, D., & Grosu, R. (2021). Liquid time-constant networks. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 7657–7666). (*vol. 35*).

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107–116.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hong, H. S., & Thuan, N. D. (2023). Hust bearing: A practical dataset for ball bearing fault diagnosis. *BMC Research Notes*, *16*(1), 138.

Introduction to self-driving cars. https://www.udacity.com/course/intro-to-self-driving-cars--nd113.

John, F. (1952). On integration of parabolic equations by difference methods: I. linear and quasi-linear equations for the infinite interval. *Communications on Pure and Applied Mathematics*, *5*(2), 155–211.

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. In *Advances in psychology* (pp. 471–495). Elsevier (*vol. 121*).

Kan, K., Li, X., & Osher, S. (2025). Ot-transformer: A continuous-time transformer architecture with optimal transport regularization. arXiv preprint arXiv:2501.18793.

Lechner, M., & Hasani, R. (2022). Learning Long-Term Dependencies in Irregularly-Sampled Time Series. https://arxiv.org/abs/2006.04418

Lechner, M., Hasani, R., Amini, A., Henzinger, T. A., Rus, D., & Grosu, R. (2020). Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, *2*(10), 642–652.

Lechner, M., Hasani, R. M., & Grosu, R. (2018). Neuronal circuit policies. arXiv preprint arXiv:1803.08554.

LeCun, Y., Touresky, D., Hinton, G., & Sejnowski, T. (1988). A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school* (pp. 21–28). (*vol. 1*).

Nectoux, P., Gouriveau, R., Medjaher, K., Ramasso, E., Chebel-Morello, B., Zerhouni, N., & Varnier, C. (2012). Pronostia: An experimental platform for bearings accelerated degradation tests. In *Ieee international conference on prognostics and health management, phm'12.* (pp. 1–8). IEEE Catalog Number: CPF12PHM-CDR.

Neil, D., Pfeiffer, M., & Liu, S.-C. (2016). Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in neural information processing systems*, *vol. 29*.

Park, M., Kim, H., & Park, S. (2021). A convolutional neural network-based end-to-end self-driving using liDAR and camera fusion: Analysis perspectives in a real-world environment. *Electronics*, *10*(21), 2608.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in neural information processing systems*, *vol. 31*.

Razzaq, W., & Hongwei, M. (2023). Neural circuit policies imposing visual perceptual autonomy. *Neural Processing Letters*, *55*(7), 9101–9116.

Razzaq, W., & Zhao, Y.-B. (2025a). Carle: A hybrid deep-shallow learning framework for robust and explainable rul estimation of rolling element bearings. *Soft Computing*, *29*(23), 6269–6292.

Razzaq, W., & Zhao, Y.-B. (2025b). Developing distance-aware uncertainty quantification methods in physics-guided neural networks for reliable bearing health prediction. https://arxiv.org/abs/2512.08499.

Rubanova, Y., Chen, R. T. Q., & Duvenaud, D. K. (2019). Latent ordinary differential equations for irregularly-sampled time series. In *Advances in neural information processing systems*, *vol. 32*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagatig error. *Nature*, *323*(6088), 533–536

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618–626).

Shibuya, N. (2017). Car behavioral cloning. Accessed: 2025-10-05 https://github.com/naokishibuya/car-behavioral-cloning.

Shukla, S. N., & Marlin, B. M. (2021). Multi-time attention networks for irregularly sampled time series. arXiv preprint arXiv:2101.10318.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, *vol. 25*.

Thuan, N. D., & Hong, H. S. (2023). Hust bearing: A practical dataset for ball bearing fault diagnosis. *BMC Research Notes*, *16*(1), 138.

Tiang, Y., Gelernter, J. R., Wang, X., Chen, W., Gao, J., Zhang, Y., & Li, X. (2018). Lane marking detection via deep convolutional neural network. *Neurocomputing*, *280*, 46–55.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, *vol. 30*.

Vidulin, V., Lustrek, M., Kaluza, B., Piltaver, R., & Krivec, J. (2010). Localization data for person activity. UCI machine learning repository. https://doi.org/{{10.24432/C57G8X}}.

Wang, B., Lei, Y., Han, T. et al. (2019). XJTU-SY rolling element bearing accelerated life test datasets: A tutorial. *Journal of Mechanical Engineering*, *55*(16), 1–6.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L. et al. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, *33*, 17283–17297.

Zhang, B., Titov, I., & Sennrich, R. (2021). Sparse attention with linear units. arXiv preprint arXiv:2104.07012.

Zhang, Y., & Zhou, X. (2025). Continuous-time attention: PDE-guided mechanisms for long-sequence transformers. In *Proceedings of the 2025 Conference on empirical methods in natural language processing* (pp. 21654–21674).

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., & Xiong, H. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the thirty-fifth AAAI conference on artificial intelligence (AAAI)*.

Zhuang, J., Dvornek, N., Li, X., Tatikonda, S., Papademetris, X., & Duncan, J. (2020). Adaptive checkpoint adjoint method for gradient estimation in neural ode. In *International conference on machine learning* (pp. 11639–11649). PMLR.

Zhuang, J., Jia, M., Ding, Y., & Ding, P. (2021). Temporal convolution-based transferable cross-domain adaptation approach for remaining useful life estimation under variable failure behaviors. *Reliability Engineering & System Safety*, *216*, 107946.