SAPO:SPARSE-AWARE PROMPT OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) exhibit activation sparsity that varies with input prompt structure, creating an opportunity to jointly optimize prompts and model computation. We introduce **S**parse-**A**ware **P**rompt **O**ptimization (SAPO), a framework that combines prompt engineering with reduced-order computation to accelerate both homogeneous and heterogeneous inputs. This is important for physical and embodied AI, where prompts can combine natural language, numerical sensor data, system parameters, and equations. We show that explicitly organizing heterogeneous information induces contextual activation locality, producing more correlated and compressible activation representations across neighboring layers. SAPO exploits this structure through a $H \rightarrow P$ transformation based on Koopman operator theory and the Mori–Zwanzig formalism, replacing H-neuron continuous-depth layers with compact P-dimensional Gated Recurrent Unit (GRU)-based blocks ($P \ll H$) and replacing iterative Ordinary Differential Equation (ODE) solves with parallelizable matrix operations. Experiments on large-scale foundation models show that reduced-order computation accelerates both prompt types, while prompt engineering amplifies its benefit for heterogeneous inputs. SAPO achieves $3.55\times$ training speedup for structured prompts compared with $2.00\times$ unstructured prompts, together with approximately $2.9\times$ lower aggregate test time while maintaining comparable task quality. These results demonstrate the synergy between prompt engineering, which exposes more compressible activation structure, and reduced-order computation.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated strong capabilities in language understanding and generation, but their computational cost remains a barrier to latency-sensitive and resource-constrained applications. This challenge is particularly important in physical and embodied AI, where models must reason over heterogeneous information such as natural language, numerical sensor streams, system parameters, and physical dynamics. Embodied continuous LLMs (EC-LLMs) address these requirements by combining LLMs with continuous-depth neural network (CDNN) components, such as neural ordinary differential equations (NODEs) and liquid time constant (LTC) networks (Chen et al., 2018; Hasani et al., 2021; Banerjee et al., 2024; AI, 2024; 2025). However, these continuous-depth components require iterative ODE solves during training and inference, introducing substantial computational overhead. Applications such as "LLM as Guardian Angel" for autonomous insulin delivery (AID) (Banerjee et al., 2024) therefore require both efficient personalization and low-latency inference.

Existing acceleration methods, including pruning, quantization, and contextual sparsity (LeCun et al., 1990; Liu et al., 2023), primarily optimize the model while treating the input prompt as fixed. This assumption is limiting for physical and embodied AI, where continuous-depth neural networks (CDNNs) are used to model nonlinear physical dynamics and prompts often contain heterogeneous information, including natural language, numerical sensor measurements, system parameters, and equations. Our goal is to provide a unified acceleration approach for both homogeneous and heterogeneous prompts. As illustrated in Figure 1, relatively homogeneous text prompts can be directly mapped to a reduced-order representation. Heterogeneous prompts, however, contain multiple types of information whose nonlinear activations can be more difficult to compress directly. SAPO therefore uses prompt engineering to explicitly organize these components, increasing activation sparsity and locality and concentrating the resulting dynamics into a more compact subspace. The reduced-order module can then operate with a smaller effective dimension $P \ll H$.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

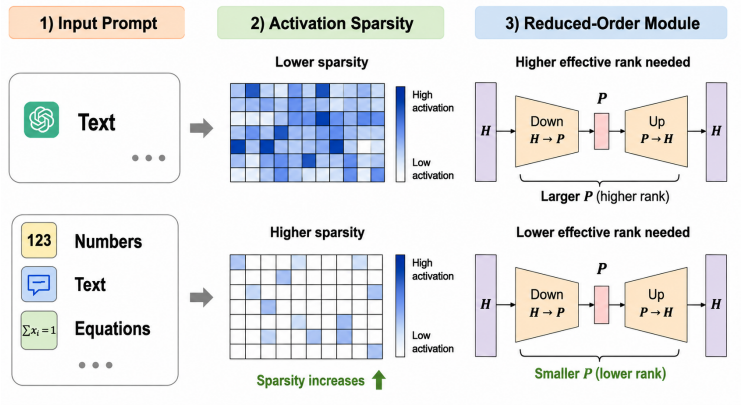


Figure 1: SAPO combines prompt engineering and reduced-order computation to efficiently accelerate homogeneous and heterogeneous prompts.

This creates a synergistic relationship between prompt engineering and reduced-order computation. Reduced-order modeling provides a common acceleration mechanism for both homogeneous and heterogeneous inputs, while prompt engineering amplifies its benefit for heterogeneous prompts by exposing more compressible nonlinear activation structure.

Based on this synergistic relationship, we introduce **S**parse-**A**ware **P**rompt **O**ptimization (SAPO), which co-designs prompt structure and reduced-order computation. Prompt engineering exposes locality in the nonlinear activation dynamics, while SAPO exploits this structure through an $H \to P$ transformation, replacing H-neuron CDNN layers with compact P-dimensional GRU-based blocks ($P \ll H$). Using Koopman operator theory and the Mori–Zwanzig formalism, the transformation approximates the original continuous-depth dynamics in a low-dimensional representation and replaces iterative nonlinear ODE solves with parallelizable matrix operations. Reduced-order computation can still accelerate unstructured prompts, but its benefit is substantially larger when prompt engineering exposes stronger activation locality.

Contributions. This work makes the following contributions: **(1)** We empirically demonstrate that structured embodied prompt engineering induces contextual activation locality in EC-LLMs, increasing consecutive-layer activation similarity from approximately 0.35 to 0.73 compared with unstructured prompts. This establishes prompt structure as a quantifiable lever for shaping internal activation dynamics. **(2)** We present the theoretical foundation for an accuracy-preserving $H \to P$ transformation (Theorem 1), which formalizes the equivalence between CDNN layers and efficient GRU-based blocks under the Mori-Zwanzig formulation of Koopman operator theory. **(3)** We define the SAPO architecture and demonstrate its integration into the liquid foundational model (LFM-1.2B), providing implementation strategies that expose prompt-driven sparsity to modern hardware through parallelizable matrix operations. **(4)** We evaluate SAPO on structured and unstructured prompts, achieving $3.55\times$ and $2.00\times$ training speedups, respectively, together with approximately $2.9\times$ lower aggregate test time in both settings, demonstrating that prompt structuring amplifies the benefit of reduced-order computation.

2 Activation Sparsity and Contextual Locality

2.1 Activation Sparsity in LLMs.

Sparsity in neural networks can occur at two levels: *weight sparsity* (many parameters are zero or near-zero, enabling compression) and *activation sparsity* (many neurons produce zero or negligible outputs for a given input). Activation sparsity is particularly relevant for efficiency, as it enables skipping computation entirely for inactive neurons. In transformers, contextual sparsity methods (Liu et al., 2023) exploit the observation that activation patterns vary smoothly across layers due to residual connections, allowing a predictor network to identify skippable neurons.

2.2 CONTEXTUAL ACTIVATION LOCALITY IN EC-LLMS.

In EC-LLMs with CDNN layers, we observe a distinct phenomenon. Unlike transformers where residual connections stabilize activations globally across many layers, CDNN layers exhibit *local* correlation: activations in layer ℓ strongly predict activations in layer $\ell + 1$, but this predictability decays rapidly for distant layers. Moreover, this locality is *prompt-dependent*: structured prompts—where textual context, numerical sensor values, and queries are explicitly segmented into delimited sections—induce significantly higher activation correlation across contiguous layers than unstructured prompts. We term this property *contextual activation locality* and quantify it via cosine similarity of activation patterns across adjacent layers. Crucially, this locality enables dimensionality reduction: when activations are highly correlated across layers, the effective degrees of freedom are much smaller than the layer width H, justifying compression to $P \ll H$ dimensions.

To ground this concept concretely, we consider embodied AI applications such as Autonomous Insulin Delivery (AID) systems, where EC-LLMs serve as "Guardian Angels" that generate personalized adaptation plans (Banerjee et al., 2024). In such settings, prompts naturally combine heterogeneous information: textual context, numerical sensor streams, and explicit queries. Below is an example of a structured prompt. The corresponding unstructured prompt contains the same information but removes the explicit section labels and formatting.

Structured Embodied Prompt:

Context: Over the past 1 hr 40 mins, I ate a peanut bar containing 14 g of carbohydrate and took 1 U of insulin. I plan to do 30 mins of interval training within the next hour.
Sensor Values: My CGM readings are: 104.6, 106.4, 108.6, 110.9, 113.2, 115.3, 116.8, 117.6, 117.3, 115.9, 113.4, 110.0, 106.1, 102.1, 98.2, 94.9, 92.3, 90.7, 89.9, 89.9, 90.5. My current CGM reading is 85 mg/dL. My insulin sensitivity factor is 50, and my carbohydrate-to-insulin ratio is 0.36.
Query: I do not have a healthcare provider available to help me now, and I want to avoid hypoglycemia after exercise. How should I configure the set point of my device? Should I eat a snack before exercising to avoid hypoglycemia? If so, should I take any insulin with the snack?

Structured embodied prompts present information in well-defined fields (*Context*, *Sensor Values*, *Query*), enabling systematic parsing and reproducible reasoning. Unstructured prompts convey the same information in free-form narrative, resembling natural human expression. This structural difference has implications for activation patterns and computational efficiency.

2.3 EMPIRICAL EVIDENCE FOR CONTEXTUAL ACTIVATION LOCALITY

We first examine whether prompt structure changes activation dynamics across CDNN layers. As illustrated in Figure 1, structured prompts produce stronger activation locality across neighboring layers, whereas unstructured prompts produce less correlated activation patterns.

We construct semantically matched prompt pairs that contain the same context, sensor values, parameters, and query, but differ only in organization. Unstructured prompts present this information as free-form natural language, while structured prompts explicitly separate the components using delimiters (e.g., "### Context:", "### Sensor Values:", and "### Query:"). For each pair, we perform a forward pass and record activations across the CDNN layers. Contextual activation locality is measured as the cosine similarity between activation vectors in consecutive layers $(\ell, \ell + 1)$.

Table 1: Consecutive-layer activation similarity for unstructured and structured prompts. Higher similarity indicates stronger contextual activation locality.

Window	**Unstructured**	**Structured**	**Δ**	Welch's t	p	Cohen's d
T65-70/L65-70	.361±.024	.736±.022	+.375	−80.6	< .001	16.3
T20-25/L20-25	.357±.025	.728±.026	+.372	−72.0	< .001	14.5
T40-45/L40-45	.343±.025	.737±.024	+.394	−79.9	< .001	16.1
T65-70/L30-35	.357±.026	.735±.025	+.378	−72.1	< .001	14.6

Table 1 compares contextual activation locality across token and layer regions. A window denoted Ta–b/Lc–d contains token positions a–b and decoder layers c–d. We evaluate early, middle, and late layer regions, together with a cross-region window, to verify that the effect is not specific to a particular token position or model depth.

Across $n = 500$ semantically matched prompt pairs, structured prompts consistently achieve consecutive-layer similarity of approximately 0.73, compared with approximately 0.35 for unstructured prompts, with an absolute gap of 0.37–0.39. The difference is significant in every window ($p < 0.001$), with large standardized effect sizes. These results show that prompt organization affects contextual activation locality across the model. The stronger locality motivates the $H \to P$ transformation. Table 28 in Appendix A.11 shows that structured prompts exhibit lower effective rank, confirming that their activation representations are more low-dimensional and compressible.

2.4 The Challenge: Why Prediction-Based Methods Fail

Existing contextual sparsity methods predict which neurons can be skipped based on activation patterns in nearby layers (Liu et al., 2023). This works well in conventional LLMs because activation patterns remain correlated across multiple layers, and residual connections help stabilize these patterns. In EC-LLMs, however, activation locality is more local and depends strongly on the input prompt. As a result, activation patterns from earlier layers are less reliable for predicting which neurons will be important in later layers.

SAPO therefore takes a different approach. Instead of predicting individual neurons to skip, it uses prompt engineering to expose more localized and compressible activation dynamics and then operates directly in a reduced P-dimensional space, where $P \ll H$. The following section develops this $H \to P$ transformation using the Koopman–Mori–Zwanzig framework.

3 SAPO Architecture

This section develops the theoretical foundation and architecture of SAPO to derive an accuracy-preserving $H \to P$ transformation that exploits contextual activation locality.

3.1 From Nonlinear ODEs to Linear Dynamics

Consider a continuous-depth latent variable (CDLV) layer with H cells (Chen et al., 2018), where the forward pass follows Equation 1.

$$\frac{dh}{dt} = f(h(t), i(t), \theta, t), \tag{1}$$

where $i(t)$ is the input, θ are learnable parameters, and f is typically a neural network (e.g., Multilayer Perceptron (MLP) with tanh activation). Computing the output requires numerical integration over a time horizon τ, making CDNNs computationally expensive compared to discrete alternatives.

The core challenge is computational: solving the nonlinear ODE in Equation 1 requires iterative numerical integration, consuming significant time and energy. However, if we could express the same dynamics as a *linear* ODE, we could solve it directly via matrix exponentiation—a single, highly optimized operation on modern hardware. Koopman theory makes this possible by lifting the nonlinear dynamics from the H-dimensional state space h into a (potentially infinite-dimensional) space of observable functions $\mathcal{G}(h)$, where the evolution is linear. The Mori-Zwanzig formalism then allows us to compress this infinite-dimensional linear system back down to a finite P-dimensional system ($P \ll H$) by partitioning observables into explicitly tracked components ($\mathcal{G}_M$) and implicitly marginalized components ($\mathcal{G}_I$).

Applying this framework to the CDLV forward pass, we partition observables into measured ($\mathcal{G}_M$, dimension P) and implicit ($\mathcal{G}_I$) components as in Equation 2. The Mori–Zwanzig (MZ) formalism (Lin et al., 2021) solves this by systematically partitioning observables into *measured* components $\mathcal{G}_M$ (the low-dimensional subset we wish to compute explicitly) and *implicit* components $\mathcal{G}_I$ (all other dynamics we cannot afford to track). The coupled evolution is:

$$\begin{bmatrix} \dot{\mathcal{G}}_M \\ \dot{\mathcal{G}}_I \end{bmatrix} = \begin{bmatrix} \mathcal{K}_M & \mathcal{K}_{MI} \\ \mathcal{K}_{IM} & \mathcal{K}_I \end{bmatrix} \begin{bmatrix} \mathcal{G}_M \\ \mathcal{G}_I \end{bmatrix}. \tag{2}$$

Under the steady-state assumption (valid for sequences longer than the integration horizon τ, which holds for typical LLM inputs), we can eliminate $\mathcal{G}_I$ by solving for it via Laplace transforms and

substituting back into the equation for $\dot{\mathcal{G}}_M$. This yields:

$$\dot{\mathcal{G}}_M = \mathcal{K}_M \mathcal{G}_M + \mathcal{K}_{MI} \int_{-\infty}^{t} e^{(t-s)\mathcal{K}_I} \mathcal{K}_{MI} \mathcal{G}_M(s)\, ds \tag{3}$$

where the first term represents *linear dynamics* and the second term represents *implicit dynamics convolution*. $\mathcal{K}_M$ captures observable dynamics and $\mathcal{K}_{MI}$ captures interactions with implicit (unobserved) factors. The original hidden state is recovered as $h = \mathcal{K}_M \mathcal{G}_M$. This decomposition is formalized in Lemma 1.

Lemma 1 (CDLV Decomposition)**.** *The representation output of a CDLV layer with H cells can be decomposed into: (1) a linear dynamical system on observable functions $\mathcal{G}_M$, and (2) a nonlinear convolution capturing implicit dynamics interactions, as given by Equation 3.* **Proof:** *See Appendix A.9.*

3.2 GRU as Universal Approximator for MZ Dynamics

Lemma 1 decomposes CDLV dynamics into two components: a linear term and a memory convolution. To construct an efficient architecture, we must find a computational primitive that can approximate both components while being implementable on modern hardware. Remarkably, the Gated Recurrent Unit (GRU)—a widely-used discrete recurrent cell—satisfies these requirements.

The decomposition in Equation 3 comprises two parts: Part 1 ($\mathcal{K}_M \mathcal{G}_M$) is a *linear* ODE, and Part 2 is a *convolution* with an exponentially-decaying kernel. GRU gating mechanisms naturally implement both operations—the update gate captures exponential decay while the cell state update handles linear dynamics. This structural correspondence enables a GRU to approximate the full MZ decomposition. The following theorem formalizes this with PAC-style guarantees.

Theorem 1 (GRU Approximation of MZ Dynamics)**.** *There exist GRU parameters $W_r, U_r, b_r, W_h, U_h, b_h$ such that for any $\epsilon > 0$ and $0 < \alpha < 1$, the GRU forward pass $h^G(t)$ approximates the CDLV forward pass $h^C(t)$ with accuracy ϵ and Probably Approximately Correct (PAC) guarantees:*

$$\Pr\left(\|\dot{h}^G(t) - \dot{\mathcal{G}}_M - \mathcal{K}_M \mathcal{G}_M\| < \epsilon\right) > 1 - \alpha.$$

Proof: *The GRU forward pass can be rearranged to match the structure of Equation 3. A dedicated time-tracking cell encodes t, while remaining cells implement Parts 1 and 2 through specific weight configurations. Full proof in Appendix.*

3.3 SAPO Layer Architecture

Lemma 1 and Theorem 1 motivate a GRU-based approximation of the CDNN dynamics under the assumptions of the Koopman–Mori–Zwanzig formulation. We now describe the concrete SAPO layer architecture that implements this reduced-order approximation, as illustrated in Figure 2.

Architecture Overview. The SAPO layer implements an $H \to P$ reduced-order transformation, where $P \ll H$ is selected empirically. The N-dimensional input $i(t)$ is mapped into a GRU layer containing $P + 1 = 2R + 1$ cells, partitioned into three specialized groups:

- **Block 1** (R cells): Approximates the linear dynamics component (Part 1 of Equation 3). These cells have gate weights set to zero, reducing them to linear recurrences that represent the finite-dimensional observable dynamics associated with $\mathcal{K}_M$.
- **Block 2** (R cells): Approximates the implicit dynamics convolution (Part 2 of Equation 3). These cells use standard GRU gating to approximate the memory integral $\int e^{(t-s)\mathcal{K}_I} \mathcal{K}_{MI} \mathcal{G}_M(s)\, ds$.
- **Time cell** (1 cell): Tracks time t for the exponential terms used in the reduced-order dynamics.

Matrix Exponentiation Module. The outputs of Blocks 1 and 2 are concatenated to form the P-dimensional observable state $\mathcal{G}_M$. SAPO evolves this reduced state according to $\mathcal{G}_M(t + N_s\tau) =$

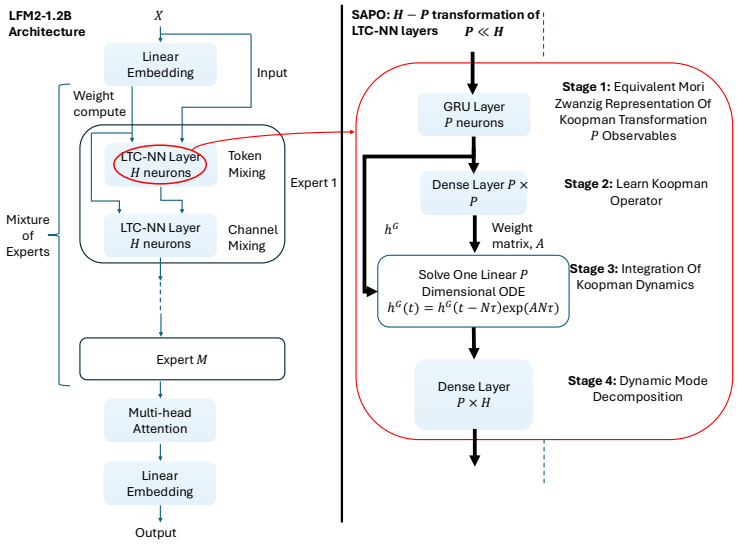


Figure 2: **The $H \to P$ transformation in SAPO.** An H-cell CDNN layer (left), which requires iterative nonlinear ODE integration, is approximated by a SAPO block (right) consisting of: (1) a P-cell GRU layer ($P \ll H$) that maps inputs to low-dimensional observables $\mathcal{G}_M$, (2) a matrix exponentiation module that evolves the reduced linear dynamics $\dot{\mathcal{G}}_M = A\mathcal{G}_M$ via $\exp(AN_s\tau)$ using Padé approximation, and (3) a dense projection layer mapping from P back to H dimensions for compatibility with downstream layers. The resulting architecture replaces iterative nonlinear ODE solves with reduced-order, parallelizable matrix operations.

$\exp(AN_s\tau)\mathcal{G}_M(t)$, where $A \in \mathbb{R}^{P\times P}$ is a learned finite-dimensional operator matrix. The matrix exponential is computed using Padé approximation (Arioli et al., 1996) and implemented with a fused CUDA kernel (Lai et al., 2025) for efficient parallel execution.

Output Projection. The evolved P-dimensional state is projected back to H dimensions through a dense $P \times H$ layer. This restores the original CDNN layer's output dimension and maintains compatibility with downstream layers in the EC-LLM architecture.

Computational Complexity. The original CDNN layer requires N_s iterative ODE integration steps, each involving $O(H^2)$ operations for the nonlinear function f, giving $O(N_sH^2)$ computation per forward pass. SAPO instead requires (1) $O(NP)$ GRU computation, (2) $O(P^3)$ matrix exponentiation using Padé approximation, and (3) an $O(PH)$ output projection. Because $P \ll H$, this reduced-order formulation can substantially lower computation, with the realized speedup depending on P, the ODE solver, and the hardware implementation.

4 Evaluation and Results

4.1 Evaluation Settings

a) Large-scale analysis and ablation study: We evaluate SAPO on the Liquid Foundational Model (LFM-2) (AI, 2025), a hybrid liquid architecture combining multiplicative gating, short-range convolutions, and grouped query attention. The student model (LFM2-1.2B) with $H = 1024$-cell LTC-NN layers is accelerated by replacing each layer with a $P = 16$ GRU-based SAPO block, while LLaMA3-8B serves as the teacher.

Dataset: We use WildChat (Zhao et al., 2024), AID Embodied Prompts (Banerjee et al., 2024), FinQA (Chen et al., 2021), GSM8K (Cobbe et al., 2021), and SVAMP (Patel et al., 2021). WildChat contains real-world conversational prompts, while AID combines natural language with numerical sensor streams and system parameters. FinQA combines financial text, tables, and numerical reasoning; GSM8K contains natural-language problems with numerical and arithmetic reasoning;

and SVAMP contains mathematical word problems with numbers and equations. Together, these datasets cover both homogeneous text and heterogeneous prompts containing text, numerical data, tables, sensor streams, and mathematical expressions. We use an 80%/20% train–test split unless an established dataset split is available.

Model: Model: For the main experiments, we compare LTC-NN layers with $H = 1024$ against SAPO with $P = 16$.

b) SOTA comparison: We compare SAPO with three recent acceleration approaches spanning continuous-depth model reduction and activation sparsity: NODE + model order reduction (NODE-MOR) (Lehtimäki et al., 2024), NODE + variational formulation (NODE-VF) (Zhao et al., 2025a), and training-free activation sparsification (TEAL) (Liu et al., 2025a).

Model/Dataset: For NODE-MOR, we use MNIST with a NODE layer (512 cells) and MOR values of 100, 200, and 500, comparing against an $H \to P$ SAPO block with $P = 16$. For NODE-VF, we use the synthetic dataset from (Zhao et al., 2025a), replacing NODE layers with SAPO blocks of $P \in \{4, 8, 24, 32\}$. For TEAL, we use the AID setting and compare activation sparsification with SAPO under both structured and unstructured prompts.

Hyper-parameters: All training settings are in the Appendix A.7.

Platform and measurement method: Experiments are conducted on NVIDIA A100 GPU.

Statistical analysis: In Appendix A.11.

4.2 RESULTS

a) Effect of prompt structuring across heterogeneous inputs: We first isolate the effect of prompt organization without SAPO reduced-order computation. We evaluate AID, FinQA, GSM8K, and SVAMP, which contain heterogeneous combinations of natural language, sensor or tabular data, numerical values, arithmetic operations, and equations. For each dataset, structured and unstructured prompts preserve the same semantic information and differ only in organization. We use identical generation settings for both formats and keep input lengths comparable to minimize differences due to token count. Per-token latency normalizes decoding time by the number of generated tokens, while end-to-end (E2E) latency includes both prefill and decoding.

As shown in Table 2, prompt structuring alone provides a modest but consistent efficiency improvement across all four datasets, with per-token speedups of $1.07\times$–$1.11\times$ and lower E2E latency. This experiment measures the runtime effect of prompt organization alone; it does not establish activation locality. Activation locality is measured independently in Table 1.

Table 2: Effect of prompt organization on inference efficiency without SAPO reduced-order computation. Speedup is computed from per-token latency.

Dataset	Input Content	Latency (ms/token)		E2E Latency (ms)		Speedup
		Unstruct.	Struct.	Unstruct.	Struct.	(×)
AID	Text + Sensor + Parameters	49.6	**44.7**	15,192	**13,991**	**1.11**
FinQA	Text + Table + Numbers	51.8	**46.7**	14,836	**13,514**	**1.11**
GSM8K	Text + Numbers + Arithmetic	43.9	**41.2**	10,972	**10,318**	**1.07**
SVAMP	Text + Numbers + Equations	46.8	**43.1**	11,864	**10,982**	**1.09**

b) Large-scale analysis and ablation:

We benchmark against sparsity-based acceleration using *SAPO+sparse*, which applies dropout (probability 0.3) after the dense layer; *SAPO-LODE+sparse*, which combines dropout with ODE elimination; and LTC+DejaVu (Liu et al., 2023), which uses an auxiliary MLP to predict active LTC-NN nodes.

Table 3 shows that SAPO accelerates training by $2.00\times$ on unstructured AID and $3.55\times$ on structured AID, while reducing aggregate test time by approximately $2.9\times$ in both settings. The similar test-time speedups mainly reflect the architectural replacement of iterative LTC computation, whereas the larger training gain for structured prompts indicates an additional benefit from prompt-induced activation locality. SAPO also achieves the lowest training and test losses in both settings.

Table 3: SAPO ablation results on unstructured and structured AID. T denotes time and SP denotes speedup over LTC. Best results are in **bold**; second-best results are underlined.

	Unstructured AID						Structured AID					
Model	**Train L**	**Train T**	**Train SP**	**Test L**	**Test T**	**Test SP**	**Train L**	**Train T**	**Train SP**	**Test L**	**Test T**	**Test SP**
LTC	8.1e-5	5532.68	1.00	8.4e-5	569.25	1.00	7.5e-5	6409.54	1.00	7.8e-5	431.25	1.00
LTC+DejaVu	8.6e-5	5480.00	1.01	8.9e-5	557.43	1.02	8.0e-5	6280.00	1.02	8.2e-5	426.30	1.01
SAPO	**5.4e-5**	**2766.34**	**2.00**	**5.6e-5**	**193.08**	**2.95**	**5.0e-5**	**1805.50**	**3.55**	**5.2e-5**	**146.59**	**2.94**
SAPO-LODE	5.9e-5	4370.00	1.27	6.1e-5	221.50	2.57	5.5e-5	3560.00	1.80	5.7e-5	174.80	2.47
SAPO-1024	6.5e-5	3415.00	1.62	6.7e-5	207.20	2.75	6.0e-5	2670.00	2.40	6.2e-5	158.90	2.71
SAPO+sparse	7.5e-5	3180.00	1.74	7.7e-5	192.86	2.95	7.0e-5	1940.00	3.30	7.2e-5	145.97	2.95
SAPO-LODE+sparse	7.8e-5	4550.00	1.22	8.0e-5	216.30	2.63	7.2e-5	3720.00	1.72	7.4e-5	171.40	2.52

To isolate the effect of prompt structure, we compare SAPO at the same $P = 16$. Structured prompts reduce training time from $2766.34\,\mathrm{s}$ to $1805.50\,\mathrm{s}$ ($1.53\times$) and test time from $193.08\,\mathrm{s}$ to $146.59\,\mathrm{s}$ ($1.32\times$). In contrast, prompt structuring alone does not reduce LTC training time ($5532.68\,\mathrm{s}$ unstructured versus $6409.54\,\mathrm{s}$ structured). This interaction supports the synergistic design of SAPO: prompt engineering exposes stronger activation locality, while the reduced-order architecture provides the mechanism to exploit it computationally.

Key takeaway. SAPO provides $2.00\times$ and $3.55\times$ training speedups for unstructured and structured prompts, respectively, together with an approximately $2.9\times$ test-time speedup. Within SAPO, prompt structure provides an additional $1.53\times$ training and $1.32\times$ test-time improvement. The ablations further show that removing the linear ODE or eliminating aggressive dimensionality reduction reduces the overall benefit, while additional sparse-NN acceleration provides limited improvement. DejaVu provides only a small test-time gain, supporting the need for reduced-order computation rather than prediction-based sparsity alone.

c) Generality assessment:

Table 4 evaluates SAPO across four RNN-based architectures at sequence lengths 32–128. SAPO improves loss on all four models while delivering 2.2–$2.9\times$ speedup. The improvement is consistent across architectures with fundamentally different recurrence mechanisms: linear attention (RWKV), shared-layer mixture-of-experts (MoEUT), and depth-recurrent latent reasoning (Huginn). Notably, SAPO never degrades accuracy relative to the unoptimized baseline, confirming that the Koopman-projected FF replacement transfers across backbone families without architecture-specific tuning.

Table 4: Generalization to four RNN-based architectures. Speedup ranges across sequence lengths $32 \rightarrow 128$; SAPO preserves or improves loss while delivering consistent speedup.

Architecture	**Speedup** ($32 \rightarrow 128$)	**Base Loss**	**+SAPO Loss**
CoLA (Liu et al., 2025b)	$2.2 \rightarrow 2.7\times$	0.0810	**0.0751**
RWKV (Peng et al., 2023)	$2.3 \rightarrow 2.7\times$	2.9642	**2.8783**
MoEUT (Csordás et al., 2024)	$2.4 \rightarrow 2.7\times$	6.4844	**6.2965**
Huginn (Geiping et al., 2025)	$2.4 \rightarrow 2.9\times$	2.5177	**2.5001**

d) Comparison with SOTA: Table 5 shows that SAPO achieves faster test times than NODE-MOR while maintaining comparable accuracy. Unlike NODE-MOR, SAPO also reduces training time, demonstrating efficiency gains during both training and inference. The MNIST results further show that SAPO extends beyond time-series tasks to image data.

Table 6 compares SAPO with GRUFlow and NODE-VF on the VF-NODE benchmark. Across all tested reduced dimensions, SAPO achieves both lower test error and shorter time per epoch. We sweep the reduced dimension $P \in \{4, 8, 24, 32\}$, yielding $P \times P$ Koopman operators with $P^2 \in \{16, 64, 576, 1024\}$ elements. Increasing P improves approximation quality at the cost of additional computation: test error decreases from 2.72×10^{-3} at $P = 4$ to 1.22×10^{-3} at $P = 32$, while time per epoch increases from $0.0178\,\mathrm{s}$ to $0.1225\,\mathrm{s}$.

In Table 7, all three methods achieve comparable test loss. TEAL provides $2.47\times$ and $2.39\times$ inference speedups on unstructured and structured prompts, respectively, while SAPO achieves approximately $2.95\times$ in both settings. SAPO therefore provides an additional $1.19\times$–$1.23\times$ acceleration

over TEAL while preserving comparable task quality, demonstrating the benefit of reduced-order computation beyond activation sparsification alone.

Table 5: Comparison between NODE-MOR (Lehtimäki et al., 2024) (with varying MOR order) and SAPO on MNIST. TT denotes training time.

Method (MOR order)	Train Acc.	TT / epoch	Test Acc.	Test Time
NODE-512-MOR (100)	97%	$17.4s \pm 1.3$	76%	$0.09s \pm 0.03$
NODE-512-MOR (200)	97%	$17.4s \pm 1.3$	81%	$0.09s \pm 0.03$
NODE-512-MOR (500)	97%	$17.4s \pm 1.3$	93%	$0.09s \pm 0.03$
SAPO H=512,P=16	**94%**	$\mathbf{13.3s \pm 1.5}$	**92%**	$\mathbf{0.03s \pm 0.01}$

Table 6: Comparison of NODE-VF, GRUFlow, and SAPO on the VF-NODE-ICLR2025 dataset (Zhao et al., 2025a). T/Ep denotes time per epoch; lower test error is better.

Model	**T/Ep (s) ↓**	**Test Error ↓**
GRUFlow	0.3691	1.63E+00
NODE-VF	0.1894	3.54E−02
SAPO ($P = 4$)	**0.0178**	2.72E−03
SAPO ($P = 8$)	0.0319	2.44E−03
SAPO ($P = 24$)	0.0972	1.33E−03
SAPO ($P = 32$)	0.1225	**1.22E−03**

Table 7: Comparison with activation-sparsity acceleration under unstructured and structured AID prompts. SP denotes speedup over LTC.

	Unstructured			**Structured**		
Method	**Loss ↓**	**Inference Time (s) ↓**	**SP ↑**	**Loss ↓**	**Inference Time (s) ↓**	**SP ↑**
LTC	8.40e-5	569.25	1.00	7.80e-5	431.25	1.00
TEAL	8.35e-5	230.50	2.47	7.76e-5	180.20	2.39
SAPO	8.31e-5	**193.08**	**2.95**	7.72e-5	**146.59**	**2.94**

e) Sensitivity to reduced dimension P. Table 8 evaluates the effect of the reduced dimension P, where the Koopman operator $A \in \mathbb{R}^{P\times P}$. We evaluate $P \in \{4, 8, 16, 32\}$, corresponding to operator sizes $P^2 \in \{16, 64, 256, 1024\}$. The lowest loss is obtained at $P = 8$, while test time changes only slightly across the tested dimensions. We use $P = 16$ in the main acceleration experiments as the most compact setting, while $P = 8$ provides the lowest loss. Structured prompts remain relatively stable at small P, supporting their suitability for aggressive dimensionality reduction.

Table 8: Sensitivity to reduced dimension P on AID. T denotes test time.

	Structured		**Unstructured**	
$\boldsymbol{P}$	**Loss**	**T (s)**	**Loss**	**T (s)**
4	5.2e-5	146.59	5.6e-5	193.08
8	**5.0e-5**	146.90	**4.8e-5**	193.70
16	5.8e-5	147.60	6.3e-5	194.83
32	6.2e-5	147.74	6.7e-5	195.21

5 Conclusions

SAPO accelerates continuous-depth neural networks by exploiting contextual activation locality induced through prompt engineering. The $H \to P$ transformation replaces H-cell LTC-NN layers with compact $P \ll H$ GRU-based blocks, converting H nonlinear ODEs into a P-dimensional linear ODE. On large-scale foundation models, SAPO achieves $3.55\times$ training speedup for structured prompts and $2.00\times$ for unstructured prompts, together with approximately $2.9\times$ lower aggregate test time while maintaining comparable task quality. These results demonstrate that prompt structure quantifiably determines acceleration magnitude, establishing prompt engineering as a mechanism for architectural efficiency in embodied continuous LLMs.

AI USE STATEMENT

We used generative AI tools to assist with language editing, organization, and presentation of the manuscript. We also used generative AI tools to assist with code development and debugging. All AI-assisted text, code, citations, and technical content were reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve experiments with human subjects or the collection of private user data. We use existing public datasets and simulation-based experiments. Although autonomous insulin delivery is used as an embodied-AI case study, SAPO is an AI efficiency method and is not intended to provide medical advice or replace clinical decision making. The reported experiments should therefore not be interpreted as clinical validation of an autonomous insulin delivery system.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce our experiments throughout the main paper and appendix. Section 4 describes the evaluation settings, datasets, baselines, and hardware. Appendix A.5 describes the datasets, Appendix A.6 provides implementation details and hyperparameters, and Appendix A.8 provides the theoretical derivations and proofs. Appendix A.10 reports repeated-run uncertainty and additional experimental results. We also identify the public datasets and baseline implementations used in our experiments.

REFERENCES

Liquid AI. Liquid foundation models: Our first series of generative ai models, September 30 2024.

Liquid AI. Introducing lfm2: The fastest on-device foundation models on the market, July 10 2025.

Julius Aka, Johannes Brunnemann, Jörg Eiden, Arne Speerforck, and Lars Mikelsons. Balanced neural ODEs: Nonlinear model order reduction and koopman operator approximations. In *The Thirteenth International Conference on Learning Representations*, 2025.

M. Arioli, B. Codenotti, and C. Fassino. The padé method for computing the matrix exponential. *Linear Algebra and its Applications*, 240:111–130, 1996. ISSN 0024-3795. doi: https://doi.org/10.1016/0024-3795(94)00190-1. URL https://www.sciencedirect.com/science/article/pii/0024379594001901.

Ayan Banerjee, Aranyak Maity, Payal Kamboj, and Sandeep KS Gupta. Cps-llm: large language model based safe usage plan generator for human-in-the-loop human-in-the-plant cyber-physical system. *AAAI CAIPI workshop, arXiv preprint arXiv:2405.11458*, 2024.

Richard N. Bergman, Y. Ziya Ider, Charles R. Bowden, and Claudio Cobelli. Quantitative estimation of insulin sensitivity. *American Journal of Physiology-Endocrinology and Metabolism*, 236(6): E667–E677, 1979. doi: 10.1152/ajpendo.1979.236.6.E667.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Luis M Candanedo and Véronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and buildings*, 112:28–39, 2016.

Julyan HE Cartwright and Oreste Piro. The dynamics of runge–kutta methods. *International Journal of Bifurcation and Chaos*, 2(03):427–449, 1992.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Yuli Chen, Bo Cheng, Jiale Han, Yingying Zhang, Yingting Li, and Shuhao Zhang. DLP: Dynamic layerwise pruning in large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7934–7956. PMLR, 2025.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of EMNLP*, 2021.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. Association for Computational Linguistics, October 2014. doi: 10.3115/v1/D14-1179.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Rei Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D. Manning. MoEUT: Mixture-of-experts universal transformers. In *NeurIPS*, 2024. URL https://arxiv.org/abs/2405.16039.

Chiara Dalla Man, Francesco Micheletto, Dayu Lv, Marc Breton, Boris Kovatchev, and Claudio Cobelli. The UVA/PADOVA type 1 diabetes simulator: New features. *Journal of Diabetes Science and Technology*, 8(1):26–34, 2014. doi: 10.1177/1932296813514502.

Dheeru Dua, Casey Graff, et al. Uci machine learning repository, 2017. *URL http://archive. ics. uci. edu/ml*, 7(1):62, 2017.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Giorgio Gnecco, Marcello Sanguineti, et al. Approximation error bounds via rademacher complexity. *Applied Mathematical Sciences*, 2:153–176, 2008.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks, 2021.

David Haussler and Manfred Warmuth. The probably approximately correct (pac) and other learning models. *The Mathematics of Generalization*, pp. 17–36, 2018.

Weizhong Huang, Yuxin Zhang, Xiawu Zheng, Fei Chao, and Rongrong Ji. Determining layer-wise sparsity for large language models through a theoretical perspective. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 26175–26201. PMLR, 2025.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677. Association for Computational Linguistics, 2024.

Bernard O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931. doi: 10.1073/pnas.17.5.315.

Ruihang Lai, Junru Shao, Siyuan Feng, Steven Lyubomirsky, Bohan Hou, Wuwei Lin, Zihao Ye, Hongyi Jin, Yuchen Jin, Jiawei Liu, Lesheng Jin, Yaxing Cai, Ziheng Jiang, Yong Wu, Sunghyun Park, Prakalp Srivastava, Jared Roesch, Todd C. Mowry, and Tianqi Chen. Relax: Composable abstractions for end-to-end dynamic machine learning. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '25, pp. 998–1013. ACM, March 2025. doi: 10.1145/3676641.3716249. URL http://dx.doi.org/10.1145/3676641.3716249.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pp. 598–605. Morgan Kaufmann, 1990.

Mikko Lehtimäki, Lassi Paunonen, and Marja-Leena Linne. Accelerating neural odes using model order reduction. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):519–531, January 2024. doi: 10.1109/TNNLS.2022.3175757.

Jörg Liesen and Zdeněk Strakoš. *Krylov Subspace Methods: Principles and Analysis*. Oxford University Press, 2013. ISBN 9780199655410.

Yen Ting Lin, Yifeng Tian, Daniel Livescu, and Marian Anghel. Data-driven learning for the mori–zwanzig formalism: A generalization of the koopman learning framework. *SIAM Journal on Applied Dynamical Systems*, 20(4):2558–2601, 2021. doi: 10.1137/21M1401759.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. Deja vu: Contextual sparsity for efficient LLMs at inference time. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22137–22176. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/liu23am.html.

Ziyue Liu, Ruijie Zhang, Zhengyang Wang, Mingsong Yan, Zi Yang, Paul D. Hovland, Bogdan Nicolae, Franck Cappello, Sui Tang, and Zheng Zhang. CoLA: Compute-efficient pre-training of LLMs via low-rank activation. Technical report, Suzhou, China, November 2025b.

Yuqi Luo, Chenyang Song, Xu Han, Yingfa Chen, Chaojun Xiao, Xiaojun Meng, Liqun Deng, Jiansheng Wei, Zhiyuan Liu, and Maosong Sun. Sparsing law: Towards large language models with greater activation sparsity. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41311–41330. PMLR, 2025.

Seyed-Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C. del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. ReLU strikes back: Exploiting activation sparsity in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Yurii E. Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981. Association for Computational Linguistics, 2024.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of NAACL-HLT*, 2021.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, et al. RWKV: Reinventing RNNs for the transformer era. In *Findings of EMNLP*, 2023. URL https://aclanthology.org/2023.findings-emnlp.936/.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Nur Adila Faruk Senan. A brief introduction to using ode45 in matlab. *University of California at Berkeley, USA*, 2007.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR, 2024.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Petra Wagner, Zofia Malisz, and Stefan Kopp. Gesture and speech in interaction: An overview, 2014.

Hongjue Zhao, Yuchen Wang, Hairong Qi, Zijie Huang, Han Zhao, Lui Sha, and Huajie Shao. Accelerating neural odes: A variational formulation-based approach. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Jialin Zhao, Yingtao Zhang, and Carlo Vittorio Cannistraci. Pivoting factorization: A compact meta low-rank representation of sparsity for efficient inference in large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 77926–77947. PMLR, 2025b.

Wenting Zhao et al. Wildchat: 1m chatgpt interaction logs in the wild. In *International Conference on Learning Representations*, 2024.

Haizhong Zheng, Xiaoyan Bai, Xueshen Liu, Z. Morley Mao, Beidi Chen, Fan Lai, and Atul Prakash. Learn to be efficient: Build structured sparsity in large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

A Appendix

A.1 Limitations

This work focuses on computational efficiency and does not provide a comprehensive assessment of the broader societal impacts. Improved efficiency may reduce computational and energy costs, but its overall impact depends on how the resulting systems are deployed and used. Issues such as environmental impact, accessibility, fairness, privacy, and responsible deployment are beyond the scope of this work and warrant further study.

A.2 Related Work

Activation and contextual sparsity in LLMs. Activation sparsity has become an important direction for reducing LLM inference cost. DejaVu (Liu et al., 2023) introduced contextual sparsity, showing that small input-dependent subsets of attention heads and MLP parameters can approximate dense computation and can be predicted online. ReLU Strikes Back (Mirzadeh et al., 2024) showed that ReLU-based LLMs exhibit exploitable activation sparsity and neuron reuse across generated tokens, enabling substantial reductions in inference computation. More recently, Learn-To-be-Efficient (LTE) (Zheng et al., 2024) explicitly trains LLMs to develop more structured activation sparsity, demonstrating that sparsity can be amplified rather than only exploited after training. TEAL (Liu et al., 2025a) instead introduces training-free magnitude-based activation sparsification across the model and demonstrates wall-clock decoding acceleration. Sparsing Law (Luo et al., 2025) further studies how activation sparsity varies with model and training factors and provides a systematic characterization of this property.

SAPO differs from these approaches in the mechanism used to expose sparsity. Prior work primarily predicts existing sparsity, modifies activation functions, changes model training, or removes low-magnitude activations. SAPO instead studies prompt structure as a computational variable. In particular, heterogeneous prompts containing text, numerical values, sensor streams, and equations can be explicitly organized to expose stronger contextual activation locality, which is subsequently exploited through reduced-order computation.

Prompt compression and input-aware efficiency. A complementary line of work reduces LLM computation by optimizing the input itself. LongLLMLingua (Jiang et al., 2024) compresses and reorganizes long prompts to retain important information while reducing inference cost and latency. LLMLingua-2 (Pan et al., 2024) formulates task-agnostic prompt compression as token classification and uses a lightweight encoder to remove redundant information while preserving prompt fidelity. Natural-language prompt compression has also been studied as a learned compression problem that preserves transferability across different LLMs.

These methods primarily improve efficiency by reducing the number of tokens processed by the LLM. SAPO addresses a different dimension of input-aware efficiency. Rather than removing prompt information, SAPO preserves the semantic content and organizes heterogeneous components to alter the resulting activation structure. Prompt engineering therefore serves as a mechanism for exposing activation locality and improving the effectiveness of the subsequent reduced-order transformation.

Input-dependent sparse computation. Recent work has also shown that the appropriate computational pattern can depend strongly on the current input. QUEST (Tang et al., 2024) exploits query-dependent sparsity in long-context attention by selecting important KV-cache pages conditioned on the current query. More generally, these results reinforce the observation that a fixed computational pattern can be suboptimal when different inputs induce different internal behavior. SAPO shares this input-dependent perspective but focuses on continuous-depth activation dynamics rather than attention-token selection. Moreover, SAPO actively structures the prompt to expose a representation that is more amenable to reduced-order computation.

Continuous-depth networks and reduced-order computation. Neural ODEs and liquid neural networks model hidden-state evolution through continuous dynamics (Chen et al., 2018; Hasani et al., 2021), but numerical integration introduces substantial computational overhead. Model-order

reduction has therefore been explored as a mechanism for accelerating Neural ODEs (Lehtimäki et al., 2024). Recent work has continued this direction. Accelerating Neural ODEs via a variational formulation (Zhao et al., 2025a) replaces repeated numerical ODE solves with a variational approximation and demonstrates substantial training and inference acceleration. Balanced Neural ODEs (Aka et al., 2025) studies nonlinear model-order reduction together with Koopman-operator approximations, further connecting continuous-depth models with low-dimensional dynamical representations.

Low-rank and sparsity-aware model compression. Low-rank representations provide another route to efficient LLM inference. Recent methods study compact low-rank representations of sparse models and the interaction between sparsity and low-rank approximation (Zhao et al., 2025b). Other work adapts sparsity across layers using activation information rather than applying a uniform pruning ratio (Chen et al., 2025; Huang et al., 2025). These approaches modify or factorize model parameters. In contrast, SAPO reduces the dimensionality of continuous activation dynamics and studies how prompt-induced activation structure affects the effectiveness of that reduction.

SAPO differs from conventional reduced-order approaches by coupling dimensionality reduction with the structure of the input. Rather than assuming that a fixed reduced representation is equally suitable for all inputs, SAPO shows that prompt engineering can expose stronger activation locality for heterogeneous prompts, making the nonlinear dynamics more amenable to the $H \to P$ transformation.

Position of SAPO. Existing work has independently explored prompt compression, activation sparsity, input-dependent sparse computation, and reduced-order modeling. SAPO connects these directions by treating prompt structure and model reduction as a joint optimization problem. Reduced-order computation provides a common acceleration mechanism for homogeneous and heterogeneous inputs, while prompt engineering amplifies its benefit for heterogeneous prompts by exposing stronger activation locality and lower-dimensional nonlinear structure. In the Table 9, we show the recent related works.

Table 9: Recent related work on efficient LLM computation and continuous-depth model acceleration.

Area	Work	Venue
Contextual sparsity	DejaVu (Liu et al., 2023)	ICML 2023 Oral
Activation sparsity	ReLU Strikes Back (Mirzadeh et al., 2024)	ICLR 2024 Oral
Structured activation sparsity	Learn-To-be-Efficient (LTE) (Zheng et al., 2024)	NeurIPS 2024
Query-dependent sparsity	QUEST (Tang et al., 2024)	ICML 2024
Prompt efficiency	LongLLMLingua (Jiang et al., 2024)	ACL 2024
Prompt efficiency	LLMLingua-2 (Pan et al., 2024)	ACL Findings 2024
Activation sparsity	TEAL (Liu et al., 2025a)	ICLR 2025
Activation sparsity	Sparsing Law (Luo et al., 2025)	ICML 2025
Neural ODE acceleration	Variational Neural ODE (Zhao et al., 2025a)	ICLR 2025
Neural ODE / MOR	Balanced Neural ODEs (Aka et al., 2025)	ICLR 2025
Low-rank + sparsity	PIFA (Zhao et al., 2025b)	ICML 2025
Adaptive sparsity	DLP (Chen et al., 2025)	ICML 2025
Layer-wise sparsity	Layer-wise Sparsity (ATP) (Huang et al., 2025)	ICML 2025

A.3 PRELIMINARIES

Implicit and Measured Dynamics. We partition $h(t)$ into *measured dynamics*, whose components are directly observable, and *implicit dynamics*, whose components are unobserved but influence the measured variables. We assume an observable system in which all state variables affect the measurements, although not all are measured.

CDNN with Continuous Depth Latent Variables. The general CDNN architecture employs continuous depth latent variable (CDLV) nodes, such as NODE or LTC-NN, to model dynamical systems. Training minimizes a task-specific loss (e.g., cross-entropy) often combined with a physics-based loss that enforces known physical laws.

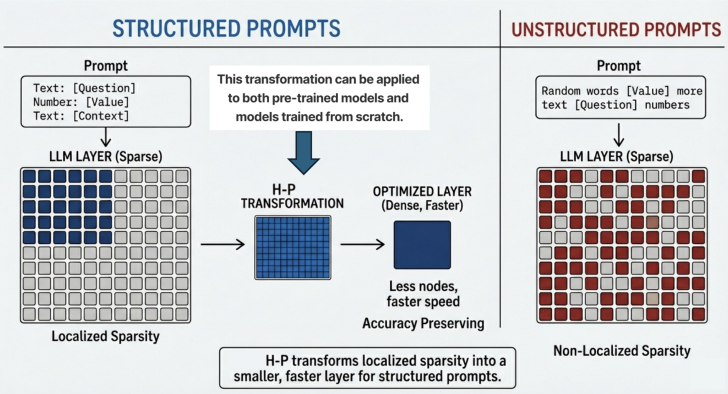


Figure 3: Prompt structure affects contextual activation locality and the effectiveness of the $H \rightarrow P$ reduced-order transformation.

In this work, each CDLV node (Chen et al., 2018) has a forward pass given by Equation 4 over a time horizon τ:

$$h_i^C(t) = h_i^C(t-\tau) + \int_{t-\tau}^{t} f\big(h_i^C(s), i(s), \theta_i, s\big), ds, \tag{4}$$

where $h_i^C(t)$ denotes the ith hidden state. Specific examples of f include $f(h_i^C, i, \theta_i, t) = \tanh\big(W_i[i(t), h_i^C(t)]^\top + b_i\big)$ for NODE architecture (Chen et al., 2018), and $\theta_i = \{W_i, b_i\}$.

General Continuous-Depth Cells. Our theoretical framework applies to any continuous-depth network satisfying Lipschitz continuity and integrability, including Liquid Time Constant networks and continuous-time recurrent neural networks (Hasani et al., 2021).

Probably Approximately Correct (PAC) PAC theory provides statistical equivalence in accuracy for machines of equal expressivity (Haussler & Warmuth, 2018) measured by Rademacher number.

A.4 Guardian Angel Planning problem

Let the state of the HIL control autonomous system (HCAS) be expressed by the state variable X which follows the dynamics in Eqn 5.

$$\dot{X} = f_\omega(X, \pi(X, S_p) + U_{ex}), \tag{5}$$

where $f_\omega(.)$ is any Lipschitz continuous (Nesterov, 1983) function parameterized by coefficient set ω, and $\pi(.,.)$ describes the action of a RM that computes an input to the environment based on the environment state expressed by $X_p \subset X$ and RM configuration set (such as set point) S_p. For convenience in expressing the planning problem and with restricted focus on the case study of AP, we will assume that the environment state is accurately expressed by the HCAS state and hence $X_p = X$. In a HIL-HIP architecture, the input to the environment is given by: $u = \pi(X, S_p) + u_{ex}$, where $u_{ex} \in U_{ex}$ is an external input from the HIL, and S_p can be manually changed by the HIL. A *usage plan*, pl, consists of a temporal sequence of b external inputs ($u_{ex}(q_i)$) at times $0 \leq q_i \leq T_H$ and/or a system configuration changes ($s_p(p_i) \in S_p$) at times $0 \leq p_i \leq T_H$, T_H is the planning horizon. We denote the set of all possible plans as $\mathcal{P}\infty$.

Safety is defined as a logical predicate $Safe(X, f_\omega, pl)$ on the state of the HIP X, for a given dynamics f_ω, and a given usage plan pl. A plan pl is a *safety plan* if the predicate $Safe(X, f_\omega, pl)$ is satisfied at all times $t \in [0, T_H]$. The user burden for executing a plan $pl \in \mathcal{P}\infty$ is simply the length $|pl|$.

Safety tested HCAS: We assume that the HCAS is safety tested, which implies the existence of an "average user" denoted by coefficient distribution $\mathcal{D}(\omega)$, external input distribution $\mathcal{D}(U_{ex})$, and configuration distribution $\mathcal{D}(S_p)$, and a set $\mathcal{P} \subset \mathcal{P}\infty$ of safety plans with respect to the predicate $Safe(X, f_\omega, pl) : \omega \in \mathcal{D}(\omega)$ and $pl \in \mathcal{P}$.

Open World: It is characterized by open world events $(\omega', u'_{ex}, s'_p, X)$, such that either, a) $\omega' \notin \mathcal{D}(\omega)$, or b) $u'_{ex} \notin \mathcal{D}(U_{ex})$, or c) $s'_p \notin \mathcal{D}(S_p)$.

Personalized plan generation in open world: Given an open world event $(\omega', u'_{ex}, s'_p, X)$, find a safety plan $pl \in \mathcal{P}\infty$ with minimum length $|pl|$, such that $Safe(X, f_{\omega'}, pl)$ is satisfied in $[0, T_H]$.

A.5 AID SYSTEM

AID is an exemplary safety critical HIL-HIP HCAS with the open world problem. APs automatically infuse insulin, known as *micro bolus*, to control blood glucose levels around a *set point* S_p, while preventing hypoglycemia when blood glucose level falls below 70 mg/dl. All AID systems that are approved for human use by Food and Drug Administration (FDA) require the human user to provide external insulin in addition to the AID controller input to manage glucose variability due to meal intake, also called *meal bolus* (u_{meal}). This meal bolus is proportional to the carbohydrate content of the meal C, with *carb insulin ratio (CIR)* as the proportionality constant. While administering meal bolus any residual insulin in the body due to past insulin infusion, characterized by insulin on board (IoB) is subtracted (Eqn. 6).

$$u_{meal} = C/CIR - iob, \tag{6}$$

In addition, external insulin, *correction bolus* (u_{corr}), unrelated to meal can also be administered if the CGM reading is greater than the set point. u_{corr} is proportional to the difference between the current glucose value $G(t)$ and the set point, with *insulin sensitivity factor (ISF)* as the proportionality constant shown in Eqn. 7.

$$u_{corr} = \frac{G(t) - S_p}{ISF} - iob. \tag{7}$$

The residual insulin or IOB depends on the insulin pharmacokinetics, (Eqn. 8), obtained from Bergman Minimal Model (BMM) (Bergman et al., 1979), and is difficult for a human to guess.

$$\frac{dy}{dt} = z, \frac{dz}{dt} = -2k_1 z - k_1^2 y + k_1^2 u_{ex}, \frac{diob}{dt} = -niob + p_1(y + I_b), \tag{8}$$

where $X = y, z, iob$, k_1 is the diffusion coefficient for insulin, and n and p_1 are patient specific metrics. Here, we assume that y and z are internal state variables of the BMM and are not measurable.

A safe usage plan for the AID HCAS is a sequence of set point changes, carbohydrate, meal bolus, and correction bolus intake actions such that the safety criteria of percentage time below 70 mg/dl in 24 hrs is less than 4% is satisfied. An example safe usage plan is as follows: "*set point is 110 mg/dL at 6 am; set ISF to 50 at 6 am; set CIR to 15 at 6 am; breakfast with 20 g of carbohydrate at 8:30 am with 1 U of meal bolus; if CGM $>$ 180 mg/dL, take correction bolus from Eqn. 7; lunch with 40 g of carbohydrate at 1 pm with meal bolus in Eqn. 6; if CGM $>$ 210 mg/dL, take correction bolus from Eqn. 7; dinner with 30 g of carbohydrate at 6 pm with meal bolus in Eqn. 6, set set point at 90 mg/dL at 10 pm*" This is deemed safe by a sample patient from the virtual patient registry available in the FDA approved Type 1 Diabetes simulator developed at UVA PADOVA (Dalla Man et al., 2014). This is an example of the average user that follows the constrained distribution manifested by the virtual patient registry. The outcome is measured using two metrics: a) percentage time in range (TIR), $70mg/dl \leq CGM \leq 180mg/dl$, and b) time below range (TBR), when $CGM < 70mg/dl$.

A.6 DATA DESCRIPTION

WildChat: WildChat (Zhao et al., 2024) contains one million real-world user–ChatGPT conversations collected through free access to GPT-3.5 and GPT-4. We use a subset of 650K conversations in our experiments. *Source:* `https://huggingface.co/datasets/allenai/WildChat`

Room Occupancy: This dataset predicts binary room occupancy (occupied vs. unoccupied) using five environmental sensor readings (e.g., temperature, humidity, CO_2) sampled at one-minute intervals (Candanedo & Feldheim, 2016).

Ozone Day Prediction: Consisting of daily environmental features (e.g., wind, solar radiation, temperature), this dataset predicts whether ozone levels exceed a regulatory threshold (Dua et al., 2017). Six years of data are segmented into overlapping 32-day sequences. A significant class imbalance (positive class: 6.31%) is handled via weighted cross-entropy. F1-score is used for evaluation.

Person Activity: This dataset captures time-series recordings of physical activities (e.g., walking, lying, standing) across 25 sessions (Rubanova et al., 2019). We use it for temporal classification, with randomized train-test splits and preprocessed sensor inputs.

Power Consumption: A household energy dataset for hourly power consumption forecasting (Dua et al., 2017). Features include reactive power and sub-metering values. Approximately 1.25% of missing entries are forward-filled. Sequences of length 32 are used, with inputs normalized. Mean squared error is used as the loss and evaluation metric.

Half-Cheetah Kinematics: A synthetic dataset modeling the dynamics of the HalfCheetah-v2 robot in MuJoCo (Brockman et al., 2016; Todorov et al., 2012). It includes 25 trajectories of 1,000 steps each, with 17-dimensional observations. To simulate real-world noise, 5% of actions are randomly corrupted. The model predicts future states using autoregressive MSE loss.

Hand Gesture Segmentation: This dataset comprises seven recordings of human hand gestures, annotated into five temporal segments (rest, preparation, stroke, hold, retraction) (Wagner et al., 2014). Each frame includes 32 motion-sensor features. The goal is sequence-level classification of gesture phases.

All application datasets (except WildChat) are obtained from:`https://github.com/raminmh/liquid_time_constant_networks`

MNIST dataset: Comparison of SAPO with NODE+MOR technique required us to use the MNIST dataset. It is a popular dataset of images of digits and is accessed from `https://git-disl.github.io/GTDLBench/datasets/mnist_datasets/`

NODE+VF dataset: For comparison with NODE+VF we utilized the dataset shared by the authors in (Zhao et al., 2025a). The dataset consists of simulation of nonlinear ordinary differential equations such as Lorenz or Lotka Volterra predator prey systems.

A.7 Implementation Details

Table 10: Hyperparameter configurations across experiments. **Hidden**: hidden layer size; **Ep**: number of epochs or training steps; **LR**: learning rate; **Activation F**: model activation functions.

Table / Figure	Hidden	Ep	LR	Activation
Table 2	1024	100	0.00005	sigmoid + tanh
Table 3	1024	100	0.00005	sigmoid + tanh
Table 7	1024	100	0.00005	sigmoid + tanh
Table 8	1024	100	0.00005	sigmoid + tanh
Table 13	1024	200	0.005	sigmoid + tanh
Table 5	512	150	0.01	tanh
Table 6	1024	5000 steps	0.000001	sigmoid + tanh

In Table 10, we summarize the hyperparameter configurations used across experiments. Tables 2, 3, 7, and 8 use the LFM2-1.2B configuration for the heterogeneous-prompt and AID experiments. Table 13 evaluates six application-specific datasets, while Tables 5 and 6 use the original datasets and experimental settings associated with NODE-MOR and VF-NODE, respectively. TEAL is training-free; the hyperparameters reported for Table 7 correspond to the underlying model configuration used for the controlled comparison.

For the AID ablation study in Table 3, we compare SAPO against the LTC baseline and several architectural and sparsity variants, including LTC+DejaVu, SAPO-LODE, SAPO-1024, SAPO+sparse, and SAPO-LODE+sparse. Structured and unstructured prompts use the same experimental configuration so that differences can be attributed to prompt organization and the corresponding SAPO variants. All experiments use a hidden layer size of $H = 1024$.

The matrix exponential in SAPO operates on the learned Koopman operator $A \in \mathbb{R}^{P\times P}$. We evaluate reduced dimensions $P \in \{4, 8, 24, 32\}$, corresponding to Koopman operator containing $P^2 \in \{16, 64, 576, 1024\}$ elements. The $P = 4$ configuration therefore uses a 4×4 matrix exponential. CUDA kernels are specialized to the corresponding matrix dimensions. The student model used is **LFM2** (from `https://huggingface.co/unsloth/LFM2-1.2B`), while the teacher model is **LLaMA3-8B** (from `https://huggingface.co/meta-llama/Meta-Llama-3-8B`). The primary training logic is implemented in the ablation study script.

Table 13 is generated using the **application comparison** code. All six datasets are trained using PyTorch with custom CUDA kernels to accelerate matrix exponential computation. The hidden

layer size is fixed at 1024, and the number of training epochs is set to 65. The code is based on the repository at `https://github.com/raminmh/liquid_time_constant_networks`.

In Table 5, we compare **SAPO** with **NODE-MOR**, following the setup described in (Lehtimäki et al., 2024). The source code for these experiments is available at `https://codeocean.com/capsule/9516453/tree/v1`.

Table 6 presents a comparison between **SAPO** and **VF-NODE**, which utilizes a spline-based variational loss formulation. The training is conducted for 5000 steps using 100 data points, with a noise level of 0.01 and an 80% training split. Different P values (matrix sizes) are evaluated to study their impact on the matrix exponential function. The code is adapted from `https://github.com/ZhaoHongjue/VF-NODE-ICLR2025`, with the main change being the replacement of GRUFlow logic with SAPO while keeping the rest of the configuration unchanged.

Table 7 compares **SAPO** with **TEAL** (Liu et al., 2025a), a training-free magnitude-based activation sparsification method. We adapt the authors' public implementation to the LFM2-1.2B AID setting. Following TEAL, activation histograms are first collected to calibrate magnitude thresholds, after which activations below the calibrated threshold are removed during inference. We use uniform activation sparsity and FP16 inference, following the released TEAL inference implementation. Structured and unstructured AID prompts are evaluated separately using the same model, evaluation samples, batch size, generation settings, and NVIDIA A100 GPU used for SAPO. We report test loss and end-to-end inference time, with speedup computed relative to the corresponding dense LTC baseline. The TEAL implementation is adapted from the authors' public repository: `https://github.com/FasterDecoding/TEAL`.

A.8 Time and Energy Efficiency of SAPO

We compare the scalar multiplications required for an H-cell CDLV layer versus the $H \to P$ transformation in SAPO.

CDLV Layer (LTC-NN): For an H-cell LTC-NN with input size N, multiplying the weight matrix with the input requires NH multiplications. Each $tanh$ activation involves two matrix exponentiations, and using a Padé approximation of degree d, this requires $2Hd$ multiplications. Thus, one evaluation of the function f costs $NH+2Hd$. For N_s ODE solver steps, f is computed N_s times. Additionally, at step i, a weighted sum of the previous $i-1$ steps must be taken, incurring $HN_s(N_s-1)/2$ multiplications. In total, an H-cell LTC-NN layer requires: $N_sNH + 2N_sHd + HN_s(N_s-1)/2$ multiplications in the forward pass.

$H \to P$ transformation (SAPO): For SAPO, the GRU requires $3NP + 3P^2 + 3P$ multiplications. Matrix exponentiation with a Padé approximation of degree d adds dP^3 multiplications, and the dense layer contributes PH. The total is $3NP+3P^2+3P+dP^3+PH$ multiplications. The naive speedup estimate of SAPO over an LTC-NN based CDLV layer is:

$$\sigma_{SAPO}(LTC\text{-}NN) = \frac{N_sNH + 2N_sHd + HN_s(N_s-1)/2}{3NP + 3P^2 + 3P + dP^3 + PH} \tag{9}$$

This estimate is conservative, since GPU kernel optimizations exist for Padé-based matrix exponentiation but not for ODE solvers such as ODE45. As P increases, the speedup decreases, converging to the cost of a full H-cell CDLV layer. In practice, small $P < \sqrt{H}$ achieves comparable or superior accuracy while delivering speedups. Moreover, as the number of solver steps N_s increases, the speedup grows as $O(N_s^2)$. SAPO is highly effective for high temporal resolution (large N_s), providing acceleration while maintaining accuracy.

A.9 Proofs of Lemma 1 and Theorem 1

Nonlinear system dynamics. An LTC-NN layer forward pass is equivalent to an H-dimensional nonlinear system defined by

$$\frac{dh_i}{dt} = -h_i(t)/\left(\frac{\rho}{1+\rho f_{NN}(h_i(t), i(t), t, \theta)}\right) + f_{NN}(h_i(t), i(t), t, \theta)(A). \tag{10}$$

where $h_i(t) \in [h_1(t), \ldots, h_H(t)]^\top \in \mathbb{R}^H$ is the state vector, $i(t)$ denotes the N dimensional input, θ and A are parameter matrices, and f_{NN} is a Lipschitz continuous, integrable function such as

perceptron with tanh activation. An LTC-NN is a type of continuous depth latent variable (CDLV) neuron that also includes neural ODE, continuous time recurrent neural networks (CT-RNN) or long short term memory (LSTM). Our acceleration strategy applies to any network with a layer of CDLV nodes.

ODE Solvers. The forward pass of any CDNN layer require integration of the hidden output over a time horizon $N_s\tau$ using ODE solvers, where N_s is the number of steps of integration and τ is the time in each step. ODE solvers fall into two categories: nonlinear and linear. Nonlinear solvers, such as ODE45 (Senan, 2007) or Runge–Kutta methods (Cartwright & Piro, 1992), require iterative integration and are computationally intensive. Linear solvers exploit matrix exponentiation and Krylov-subspace methods for large systems, offering significantly lower computational cost (Liesen & Strakoš, 2013).

Koopman Theory. Koopman theory (Koopman, 1931) represents the evolution of a nonlinear system via an (infinite-dimensional) linear operator. Given a measurement function $\mathcal{G} : \mathbb{R}^H \to \mathbb{R}^P$, the Koopman operator $\mathcal{K}$ satisfies

$$\mathcal{K}\,\mathcal{G}\big(h(t), i(t)\big) = \mathcal{G}\big(f\big(h(t), i(t), \theta, t\big), i(t)\big), \tag{11}$$

with $P \to \infty$ in the general case.

Mori–Zwanzig Formalism. In practice, the Koopman representation is sparse in the observable manifold. Absent explicit sparsity, one can combine Koopman theory with the Mori–Zwanzig formalism (Lin et al., 2021), yielding a decomposition of the observables into measured components $\mathcal{G}_M$ and implicit (unmeasured) components $\mathcal{G}_I$ (see Appendix Section A.3):

$$\begin{bmatrix} \dot{\mathcal{G}}_M \\ \dot{\mathcal{G}}_I \end{bmatrix} = \begin{bmatrix} \mathcal{K}_M & \mathcal{K}_{MI} \\ \mathcal{K}_{IM} & \mathcal{K}_I \end{bmatrix} \begin{bmatrix} \mathcal{G}_M \\ \mathcal{G}_I \end{bmatrix}. \tag{12}$$

Here, $\mathcal{K}_M$ captures the observable dynamics, $\mathcal{K}_I$ the implicit dynamics, and $\mathcal{K}_{IM} = \mathcal{K}_{MI}$ the interactions between the two.

Lemma 2 (CDLV Decomposition)**.** *The representation output of a CDLV layer with H cells can be decomposed into a linear combination of a linear dynamical system and a non-linear convolution function.*

Proof: *The forward pass of CDLV is given by the dynamics $dh/dt = f(h(t), i(t)\,, \theta_i\,, t)$, a nonlinear ODE based system of equations, where $h(t) = \{h_i(t)\}$ is the vector of hidden layer outputs. As such the forward pass is akin to any nonlinear ODE system and the Koopman theory represents the nonlinear dynamics through linear evolution of observables (Koopman, 1931). The MZ formalism can be combined with KO theory to represent it in terms of observable and implicit dynamics.*

Utilizing Laplace transforms a solution for $\mathcal{G}_M$ can be obtained as follows

$$\dot{\mathcal{G}}_M = \underbrace{\mathcal{K}_M\mathcal{G}_M}_{\text{observable dynamics}} + \underbrace{\mathcal{K}_{MI}\int_0^t e^{t-s\mathcal{K}_I}\mathcal{K}_{MI}\mathcal{G}_M(s)ds}_{\text{interaction of implicit with observable dynamics}} \tag{13}$$

$$+ \underbrace{\mathcal{K}_{MI}e^{-t\mathcal{K}_I}\mathcal{G}_I(0)}_{\text{Residual of implicit factors}} \tag{14}$$

Here $h = \mathcal{K}_M\mathcal{G}_M$.

Steady state assumption: The process f that generates the input data can be assumed to be in steady state for time window of $[0, \tau]$. Under this assumption, Equation 13 reduces to the following:

$$\dot{\mathcal{G}}_M = \mathcal{K}_M\mathcal{G}_M + \mathcal{K}_{MI}\int_{-\tau}^t e^{t-s\mathcal{K}_I}\mathcal{K}_{MI}\mathcal{G}_M(s)ds + \mathcal{K}_{MI}e^{-(t+\tau)\mathcal{K}_I}\mathcal{G}_I(-\tau) \tag{15}$$

If $\tau \to \infty$ then the last term of Equation 15 vanishes resulting in the following form:

$$\dot{\mathcal{G}}_M = \underbrace{\mathcal{K}_M\mathcal{G}_M}_{\text{observable dynamics}} + \underbrace{\mathcal{K}_{MI}\int_{-\infty}^t e^{t-s\mathcal{K}_I}\mathcal{K}_{MI}\mathcal{G}_M(s)ds}_{\text{Implicit dynamics interaction}} \tag{16}$$

This steady state assumption thus enables us to learn the steady state effects of unmeasured causal factors through a high dimensional neural network such as gated recurrent unit (GRU) (Cho et al., 2014). The hidden layer output h of the CDLV layer can be obtained by solving Equation 3 to obtain $\mathcal{G}_M$ and then get $h = \mathcal{K}_M \mathcal{G}_{\mathcal{M}}$.

Theorem 1. *$\exists$ W_r, U_r, b_r, W_h, U_h, and b_h such that for any $\epsilon > 0$ and $0 < \alpha < 1$, GRU layer forward pass $h^G(t)$ can approximate implicit dynamics in the MZ approximation of CDLV forward pass $h^N(t)$ with accuracy ϵ, i.e. $Pr(||\dot{h^G}(t) - \dot{\mathcal{G}} - \mathcal{K}_M \mathcal{G}_M|| < \epsilon) > (1 - \alpha)$.*

Proof: *The forward pass of a GRU cell can be rearranged as -*

$$h_i^G(t) = h_i^G(t-1) + (1 - r_i(t-1)) \circ (h_i^G(t-1) - c_i(t-1)) \tag{17}$$

We add a cell k in the GRU architecture that only tracks time t. This means that $W_r(i,k) = 0, U_r(i,k) = 0, W_h(i,k) = 0, U_h(i,k) = 0 \forall i$. Then we have $b_r(i,k) = b_h(i,k) = B_{ik}, \forall i$, where B_{ik} is a large positive value. This forces the output $h_k^G(t) = h_k^G(t-1) + c$, where $c = tanh(B_{ik}) \approx 1$, making it keep time.

Let us consider the term $(1 - r_i(t-1))$. Let for each cell i except the time cell k, $\forall i \neq k$, we have a reset weight $W_r(j,i) = 0, \forall j$, $U_r(j,i) = 0, \forall j \neq k$, $U_r(k,i) = u_{ki}$, $b_r(j,i) = 0, \forall j \neq k$ and $b_r(k,i) = B_{ki}$, a large value, and a sigmoid activation function. Then we have -

$$(1 - r_i(t)) = 1 - \frac{1}{1 + e^{-h_k^G(t)(u_{ki} + B_{ki})}} = 1 - \frac{1}{1 + e^{-t(u_{ki} + B_{ki})}}, \quad \text{since } h_k^G(t) = t, \text{ by design .} \tag{18}$$

For large B, we can approximate $\frac{1}{1+e^{-t(u_{ki}+B_{ki})}} \approx 1 - e^{-t(u_{ki}+B_{ki})}$. Hence, we get,

$$(1 - r_i(t)) = e^{-(u_{ki}+B_{ki})t}, \text{ by design for a large bias } b_r(k,i) = B_{ki}, \text{ and } U_r(k,i) = u_{ki}. \tag{19}$$

We assume rectified linear unit (RELU) activation for the update function. The weight configuration is detailed in Table 11, and the forward pass of each cell in the GRU yields -

Weight	Block 1 Constraint	Block 1 weights	Block 2 Constraint	Block 2 weights
W_r	all entries	0	all entries	0
U_r	all entries	0	$U_r(j,i), j \neq k$	0
			$j = k$	$u_{k,i}^{(2)}$
b_r	all entries	*learned* $B_i^{(1)}$	$i \neq k$	0
			$i = k$	$B_k^{(2)}$
W_h	all entries	learned	all entries	learned
U_h	all entries	0	all entries	learned
b_h	all entries	*learned* $b_i^{(1)}$	all entries	0

Table 11: GRU weight configurations based on Theorem 1

$$\begin{aligned} h_i^G(t) \quad &= h_i^G(t-1) + e^{-(u_{ki}+B_{ki})t} \circ \big[h_i^G(t) - W_h i(t-1) \\ &- \sum_{j=1}^{P} U_h(j,i)\big((1 - e^{-(u_{ki}+B_{ki})t})\big) \circ h_i^G(t-1)\big) - b_h(i)\big] \end{aligned} \tag{20}$$

The GRU cell is a discretization of the differential $\frac{h_i^G(t)}{dt}$. The continuous counterpart of GRU cell forward pass can be expressed as follows:

$$\begin{aligned} \frac{dh_i^G(t)}{dt} &= e^{-(u_{ki}+B_{ki})t} \circ \big[h_i^G(t) - W_h i(t) \\ &- \sum_{j=1}^{P} U_h(j,i)\big((1 - e^{-(u_{ki}+B_{ki})t})\big) \circ h_i^G(t)\big) - b_h(i)\big] \\ &= e^{-(u_{ki}+B_{ki})t} \circ h_i^G(t) - W_h e^{-(u_{ki}+B_{ki})t} \circ i(t) - \\ &\quad \sum_{j=1}^{P} U_h(j,i) e^{-(u_{ki}+B_{ki})t} \circ (1 - e^{-(u_{ki}+B_{ki})t})) \circ h_i^G(t) \\ &- (1 - e^{-(u_{ki}+B_{ki})t})) \circ b_h(i). \end{aligned} \tag{21}$$

The convolution operation $e^{-(u_{ki}+B_{ki})t} \circ (1 - e^{-(u_{ki}+B_{ki})t})) = \frac{1-cosh((u_{ki}+B_{ki})t)}{u_{ki}+B_{ki}}$. *If we assume* $b_h(i) = 0 \forall i$, *then we get -*

$$\begin{aligned} \frac{dh_i^G(t)}{dt} &= e^{-(u_{ki}+B_{ki})t} \circ h_i^G(t) - W_h e^{-(u_{ki}+B_{ki})t} \circ i(t) \\ &- \sum_{j=1}^{P} U_h(j,i) \frac{1 - cosh((u_{ki}+B_{ki})t)}{u_{ki}+B_{ki}} \circ h_i^G(t) \\ &= e^{-(u_{ki}+B_{ki})t} \circ A[h_i^G(t), i(t)]^T \\ &- \sum_{j=1}^{P} U_h(j,i) \frac{1 - cosh((u_{ki}+B_{ki})t)}{u_{ki}+B_{ki}} \circ h_i^G(t), \end{aligned} \tag{22}$$

where A *is a matrix of size* $(P+N) \times (P+N)$ *by concatenating an identity matrix of size* $P \times P$ *in the upper diagonal and the matrix* W_h *of size* $N \times N$ *in the lower diagonal. Expanding the first convolution operation we obtain the following form of the GRU hidden layer -*

$$\begin{aligned} \frac{dh_i^G(t)}{dt} &= \int_0^t e^{-(u_{ki}+B_{ki})(t-z)} A[h_i^G(z), i(z)]^T dz \\ &- \sum_{j=1}^{P} U_h(j,i) \frac{1 - cosh((u_{ki}+B_{ki})t)}{u_{ki}+B_{ki}} \circ h_i^G(t). \end{aligned} \tag{23}$$

For the reset gate, we have assumed that B_{ki} *is a large bias. Through that assumption we can approximate* $\frac{1-cosh((u_{ki}+B_{ki})t)}{(u_{ki}+B_{ki})} \approx -\frac{e^{(u_{ki}+B_{ki})t}}{2(u_{ki}+B_{ki})}$

$$\begin{aligned} \frac{dh_i^G(t)}{dt} &= \underbrace{\int_0^t e^{-(u_{ki}+B_{ki})(t-z)} A[h_i^G(z), i(z)]^T dz}_{\text{Part 1: Input Dependent Implicit Interaction}} \\ &+ \underbrace{\sum_{j=1}^{P} U_h(j,i) \int_0^t \frac{e^{(u_{ki}+B_{ki})(t-z)}}{2(u_{ki}+B_{ki})} h_i^G(z) dz}_{\text{Part II: Input Independent Implicit Interaction}}. \end{aligned} \tag{24}$$

We observe that Equation 3 and 24 are of the same class of functions i.e. a nonlinear polynomial function convolved with an exponential function. Hence both the implicit dynamics interaction part of forward pass of CDLV in Equation 3 and forward pass of GRU 24 have the same Rademacher number(Gnecco et al., 2008). Hence Equation 24 has PAC learning guarantee for learnability of the implicit dynamics interaction part. This implies that $\exists \epsilon > 0, 0 < \alpha < 1$ *such that* $Pr(||\dot{h^G}(t) - \dot{\mathcal{G}} - \mathcal{K}_M \mathcal{G}_M|| < \epsilon) > (1 - \alpha)$.

A.10 LTC-NN Speed up

LTC-NN involves one more multiplication of input with hidden layer to compute the input dependent time constant. Hence the total number of multiplications for the forward pass of a the H cell LTC-NN layer is $N_s(N(H+1) + 2Hd)$. The overall speed up results in the following equation.

$$\sigma_{SAPO}(LTC) = \frac{N_s(NH+N) + 2N_s Hd + HN_s(N_s-1)/2}{3NP + 3P^2 + 3P + dP^3 + PH} \tag{25}$$

A.11 Results tables with uncertainty

In this section, we report statistical uncertainty for the numerical results in the main paper using standard deviation. Tables 17 and 18 report repeated-run uncertainty for the unstructured and structured AID ablations, respectively. The results reproduce the trends in Table 3, with relatively small variation across runs. Tables 19 and 20 provide detailed training-time, energy, and loss results across applications. Finally, Tables 5 and 6 report comparisons with NODE-MOR and VF-NODE, respectively, including uncertainty.

Table 12: Cache hit rates for SAPO and LFM2-1.2B

Cache Level	SAPO (%)	LFM2-1.2B (%)
L1	77.8 ± 22.1	71.3 ± 20.8
L2	97.8 ± 20.04	98.1 ± 21.2

Complementary cache results in Table 12 indicate that both models use the cache hierarchy effectively, with similarly high **L2 hit rates** (SAPO: $\mathbf{97.8 \pm 20.0\%}$, LFM2-1.2B: $\mathbf{98.1 \pm 21.2\%}$) and moderate **L1 hit rates** (SAPO: $\mathbf{77.8 \pm 22.1\%}$, LFM2-1.2B: $\mathbf{71.3 \pm 20.8\%}$). Overall, SAPO provides both lower and more predictable memory transfer times, improving efficiency for large-scale token generation.

Table 13: Comparison of model performance across six applications during inference, including inference time per epoch, energy consumption, and inference loss. The best result per row is shown in **bold**, and the second-best is underlined.

Application	Inference Time / Epoch (s)					Energy / Epoch (J)					Testing Loss				
	SAPO	GRU	NODE	LTC	LSTM	**SAPO**	GRU	NODE	LTC	LSTM	**SAPO**	GRU	NODE	LTC	LSTM
occupancy	**0.31**	0.76	27.15	0.92	0.51	**24.18**	57.76	2172.00	77.28	39.78	**0.13**	0.25	0.25	0.31	0.25
ozone	**0.03**	0.04	2.57	1.21	0.04	**2.70**	3.72	277.56	134.31	4.08	0.13	0.13	0.24	0.13	**0.12**
person	**0.03**	0.58	1.01	2.50	0.04	**2.43**	53.36	112.11	217.5	4.04	**0.45**	**0.45**	0.80	0.46	0.55
power	0.16	0.07	0.36	**0.001**	0.09	13.6	5.88	41.04	**0.078**	7.02	0.03	**0.0022**	0.01	0.009	0.01
har	**0.02**	0.03	0.66	0.09	0.13	**1.60**	2.34	56.10	7.38	10.92	**0.11**	0.21	0.18	0.15	0.18
gesture	**0.0085**	0.01	0.19	0.21	0.01	**0.6545**	0.79	16.72	16.59	0.85	**0.95**	**0.95**	0.98	1.15	1.39

Table 14: Inference time with uncertainty across six applications. Results are reported as mean $\pm$ standard deviation across repeated runs.

Application	SAPO	GRU	NODE	LTC	LSTM
Occupancy	0.31 ± 0.02	0.76 ± 0.04	27.15 ± 1.31	0.92 ± 0.05	0.51 ± 0.03
Ozone	0.03 ± 0.003	0.04 ± 0.004	2.57 ± 0.15	1.21 ± 0.07	0.04 ± 0.004
Person	0.03 ± 0.003	0.58 ± 0.03	1.01 ± 0.06	2.50 ± 0.13	0.04 ± 0.004
Power	0.16 ± 0.01	0.07 ± 0.006	0.36 ± 0.02	0.001 ± 0.0001	0.09 ± 0.008
HAR	0.02 ± 0.002	0.03 ± 0.003	0.66 ± 0.04	0.09 ± 0.008	0.13 ± 0.01
Gesture	0.0085 ± 0.0008	0.01 ± 0.001	0.19 ± 0.01	0.21 ± 0.01	0.01 ± 0.001

Table 15: Inference energy with uncertainty across six applications. Results are reported as mean $\pm$ standard deviation across repeated runs.

Application	SAPO	GRU	NODE	LTC	LSTM
Occupancy	24.18 ± 1.42	57.76 ± 3.21	2172.00 ± 105.4	77.28 ± 4.16	39.78 ± 2.31
Ozone	2.70 ± 0.18	3.72 ± 0.25	277.56 ± 15.2	134.31 ± 7.6	4.08 ± 0.27
Person	2.43 ± 0.16	53.36 ± 2.91	112.11 ± 6.2	217.50 ± 11.8	4.04 ± 0.26
Power	13.60 ± 0.81	5.88 ± 0.37	41.04 ± 2.4	0.078 ± 0.006	7.02 ± 0.43
HAR	1.60 ± 0.11	2.34 ± 0.16	56.10 ± 3.3	7.38 ± 0.49	10.92 ± 0.67
Gesture	0.6545 ± 0.05	0.79 ± 0.06	16.72 ± 1.0	16.59 ± 0.9	0.85 ± 0.06

Table 16: Testing loss with uncertainty across six applications. Results are reported as mean $\pm$ standard deviation across repeated runs.

Application	SAPO	GRU	NODE	LTC	LSTM
Occupancy	0.13 ± 0.01	0.25 ± 0.02	0.25 ± 0.02	0.31 ± 0.03	0.25 ± 0.02
Ozone	0.13 ± 0.01	0.13 ± 0.01	0.24 ± 0.02	0.13 ± 0.01	0.12 ± 0.01
Person	0.45 ± 0.04	0.45 ± 0.04	0.80 ± 0.07	0.46 ± 0.04	0.55 ± 0.05
Power	0.030 ± 0.003	0.0022 ± 0.0002	0.010 ± 0.001	0.009 ± 0.001	0.010 ± 0.001
HAR	0.11 ± 0.01	0.21 ± 0.02	0.18 ± 0.02	0.15 ± 0.01	0.18 ± 0.02
Gesture	0.95 ± 0.08	0.95 ± 0.08	0.98 ± 0.09	1.15 ± 0.10	1.39 ± 0.12

Table 17: Ablation results with uncertainty on unstructured AID. Results are mean ± standard deviation across repeated runs. T denotes total time.

Model	Train Loss	Train T (s)	Test Loss	Test T (s)
LTC	$8.1e{-5} \pm 0.3e{-5}$	5532.68 ± 165.4	$8.4e{-5} \pm 0.3e{-5}$	569.25 ± 14.8
LTC+DejaVu	$8.6e{-5} \pm 0.4e{-5}$	5480.00 ± 160.2	$8.9e{-5} \pm 0.4e{-5}$	557.43 ± 14.1
SAPO	$\mathbf{5.4e{-5} \pm 0.2e{-5}}$	$\mathbf{2766.34 \pm 91.7}$	$\mathbf{5.6e{-5} \pm 0.2e{-5}}$	193.08 ± 5.6
SAPO-LODE	$5.9e{-5} \pm 0.2e{-5}$	4370.00 ± 132.6	$6.1e{-5} \pm 0.2e{-5}$	221.50 ± 6.4
SAPO-1024	$6.5e{-5} \pm 0.3e{-5}$	3415.00 ± 105.3	$6.7e{-5} \pm 0.3e{-5}$	207.20 ± 6.0
SAPO+sparse	$7.5e{-5} \pm 0.3e{-5}$	3180.00 ± 101.5	$7.7e{-5} \pm 0.3e{-5}$	192.86 ± 5.5
SAPO-LODE+sparse	$7.8e{-5} \pm 0.3e{-5}$	4550.00 ± 139.8	$8.0e{-5} \pm 0.3e{-5}$	$\mathbf{191.09 \pm 5.4}$

Table 18: Ablation results with uncertainty on structured AID. Results are mean ± standard deviation across repeated runs. T denotes total time.

Model	Train Loss	Train T (s)	Test Loss	Test T (s)
LTC	$7.5e{-5} \pm 0.3e{-5}$	6409.54 ± 181.6	$7.8e{-5} \pm 0.3e{-5}$	431.25 ± 12.1
LTC+DejaVu	$8.0e{-5} \pm 0.3e{-5}$	6280.00 ± 176.3	$8.2e{-5} \pm 0.3e{-5}$	426.30 ± 11.8
SAPO	$\mathbf{5.0e{-5} \pm 0.2e{-5}}$	$\mathbf{1805.50 \pm 62.4}$	$\mathbf{5.2e{-5} \pm 0.2e{-5}}$	146.59 ± 4.3
SAPO-LODE	$5.5e{-5} \pm 0.2e{-5}$	3560.00 ± 108.5	$5.7e{-5} \pm 0.2e{-5}$	174.80 ± 5.1
SAPO-1024	$6.0e{-5} \pm 0.2e{-5}$	2670.00 ± 83.9	$6.2e{-5} \pm 0.2e{-5}$	158.90 ± 4.7
SAPO+sparse	$7.0e{-5} \pm 0.3e{-5}$	1940.00 ± 65.7	$7.2e{-5} \pm 0.3e{-5}$	145.97 ± 4.2
SAPO-LODE+sparse	$7.2e{-5} \pm 0.3e{-5}$	3720.00 ± 113.6	$7.4e{-5} \pm 0.3e{-5}$	$\mathbf{145.38 \pm 4.1}$

Table 19: Training results with uncertainty (supplementary to Table 13): Comparison of model performance across six applications during training, including training time per epoch and energy consumption. The best result per row is shown in **bold**, and the second-best is underlined.

Application	Training Time / Epoch (s)					Energy / Epoch (J)				
	SAPO	GRUF	NODE	LTC	LSTM	SAPO	GRUF	NODE	LTC	LSTM
occupancy	**0.5 ± 0.05**	0.62 ± 0.06	20.9 ± 2.6	18.42 ± 2.3	0.65 ± 0.06	**39 ± 5**	47.12 ± 4.9	1672 ± 170	1547.28 ± 160.7	50.7 ± 6.3
ozone	**0.1 ± 0.01**	0.2 ± 0.02	2.62 ± 0.25	2.57 ± 0.25	0.2 ± 0.02	**9 ± 1**	18.6 ± 2.0	282.96 ± 29.0	285.27 ± 30.0	20.4 ± 2.1
person	**0.2 ± 0.02**	0.3 ± 0.03	10.3 ± 1.0	10.99 ± 1.1	0.38 ± 0.04	**16.2 ± 2.0**	27.6 ± 3.0	1143.3 ± 120	956.13 ± 100	38.38 ± 4.0
power	**4 ± 0.4**	5.5 ± 0.6	272.6 ± 30.0	210.09 ± 22.0	5.7 ± 0.6	**340 ± 35**	462 ± 48	31076.4 ± 3400	16387.02 ± 1800	444.6 ± 50.0
har	**0.7 ± 0.08**	0.8 ± 0.09	20.8 ± 2.3	24.89 ± 2.7	0.865 ± 0.09	**56 ± 6**	62.4 ± 7.0	1768 ± 190	2040.98 ± 220	72.66 ± 8.0
gesture	**0.1 ± 0.01**	0.143 ± 0.015	3.25 ± 0.35	2.51 ± 0.28	0.152 ± 0.017	**7.7 ± 0.8**	11.297 ± 1.2	286 ± 30	198.29 ± 21	12.92 ± 1.4

Table 20: Training loss with uncertainty (supplementary to Table 13): Comparison of training loss across six applications. The best result per row is shown in **bold**, and the second-best is underlined.

Application	SAPO	GRUF	NODE	LTC	LSTM
occupancy	**0.0096 ± 0.001**	**0.0096 ± 0.001**	0.03 ± 0.004	**0.0096 ± 0.001**	0.01 ± 0.001
ozone	**0.13 ± 0.01**	**0.13 ± 0.01**	0.24 ± 0.03	**0.13 ± 0.01**	**0.13 ± 0.01**
person	**0.37 ± 0.04**	0.39 ± 0.04	0.79 ± 0.09	0.45 ± 0.05	0.5 ± 0.06
power	**0.0013 ± 0.0001**	**0.0013 ± 0.0001**	0.007 ± 0.001	0.009 ± 0.001	**0.0013 ± 0.0001**
har	0.0756 ± 0.008	0.082 ± 0.009	0.0883 ± 0.010	**0.0334 ± 0.004**	0.097 ± 0.011
gesture	0.46 ± 0.05	0.5 ± 0.06	0.92 ± 0.10	0.96 ± 0.11	**0.44 ± 0.05**

Combination of SAPO with quantization. Quantization and Model-Order Reduction (MOR) can be combined with SAPO to obtain inference speedups without degrading accuracy. As shown in Table 21, converting the model from FP32 to INT8 reduces inference time by approximately 11%. MOR is most beneficial when the internal representation is high-dimensional; because SAPO already operates in a compact low-dimensional latent space, additional MOR provides only marginal gains.

Table 21: Inference time comparison for LTC and SAPO models in FP32 and INT8 precision.

Model	Inference time (ms) FP32	Inference time (ms) INT8
LTC	305.58	269.76
SAPO	**3.31**	**2.96**

Prompt Length and Timing Statistics. Table 22 reports input token counts and inference-time statistics for the heterogeneous-prompt experiments. Structured prompts have comparable input

lengths while consistently reducing per-token and E2E latency, showing that the observed speedup is not simply due to shorter prompts.

Table 22: Prompt-length and inference measurement statistics for Table 2. Values report mean ± standard deviation across evaluation samples.

	Input Tokens		Latency (ms/token)		E2E (ms)	
Dataset	**Unstruct.**	**Struct.**	**Unstruct.**	**Struct.**	**Unstruct.**	**Struct.**
AID	248 ± 18	255 ± 19	49.6 ± 1.8	$\mathbf{44.7 \pm 1.6}$	15192 ± 410	$\mathbf{13991 \pm 386}$
FinQA	286 ± 24	294 ± 25	51.8 ± 2.0	$\mathbf{46.7 \pm 1.8}$	14836 ± 438	$\mathbf{13514 \pm 401}$
GSM8K	132 ± 15	138 ± 16	43.9 ± 1.5	$\mathbf{41.2 \pm 1.4}$	10972 ± 315	$\mathbf{10318 \pm 298}$
SVAMP	118 ± 13	124 ± 14	46.8 ± 1.7	$\mathbf{43.1 \pm 1.5}$	11864 ± 342	$\mathbf{10982 \pm 321}$

Table 23: Generalization results with uncertainty across RNN-based architectures. Results are reported as mean ± standard deviation across repeated runs. Speedup ranges from sequence length 32 to 128.

Architecture	**Speedup (32 → 128)**	**Base Loss**	**+SAPO Loss**
CoLA	$2.2\pm0.08 \rightarrow 2.7\pm0.10\times$	0.0810 ± 0.0032	$\mathbf{0.0751\pm0.0028}$
RWKV	$2.3\pm0.09 \rightarrow 2.7\pm0.11\times$	2.9642 ± 0.071	$\mathbf{2.8783\pm0.065}$
MoEUT	$2.4\pm0.10 \rightarrow 2.7\pm0.10\times$	6.4844 ± 0.142	$\mathbf{6.2965\pm0.131}$
Huginn	$2.4\pm0.09 \rightarrow 2.9\pm0.12\times$	2.5177 ± 0.061	$\mathbf{2.5001\pm0.058}$

Table 24: Comparison with uncertainty on the VF-NODE benchmark. Results are reported as mean ± standard deviation across repeated runs. T/Ep denotes time per epoch; lower test error is better.

Model	**T/Ep (s) ↓**	**Test Error ↓**
GRUFlow	0.3691 ± 0.0185	$1.63\text{E}{+00} \pm 0.09\text{E}{+00}$
NODE-VF	0.1894 ± 0.0092	$3.54\text{E}{-02} \pm 0.21\text{E}{-02}$
SAPO ($P = 4$)	$\mathbf{0.0178 \pm 0.0011}$	$2.72\text{E}{-03} \pm 0.18\text{E}{-03}$
SAPO ($P = 8$)	$\underline{0.0319 \pm 0.0017}$	$2.44\text{E}{-03} \pm 0.16\text{E}{-03}$
SAPO ($P = 24$)	0.0972 ± 0.0048	$\underline{1.33\text{E}{-03} \pm 0.09\text{E}{-03}}$
SAPO ($P = 32$)	0.1225 ± 0.0061	$\mathbf{1.22\text{E}{-03} \pm 0.08\text{E}{-03}}$

Table 25: Comparison with activation-sparsity acceleration with uncertainty under unstructured and structured AID prompts. Results are reported as mean ± standard deviation across repeated runs. SP denotes speedup over LTC.

	Unstructured			Structured		
Method	**Loss ↓**	**Time (s) ↓**	**SP ↑**	**Loss ↓**	**Time (s) ↓**	**SP ↑**
LTC	$8.40\text{e}{-5} \pm 0.30\text{e}{-5}$	569.25 ± 14.8	1.00	$7.80\text{e}{-5} \pm 0.30\text{e}{-5}$	431.25 ± 12.1	1.00
TEAL	$8.35\text{e}{-5} \pm 0.28\text{e}{-5}$	230.50 ± 6.7	2.47	$7.76\text{e}{-5} \pm 0.27\text{e}{-5}$	180.20 ± 5.3	2.39
SAPO	$8.31\text{e}{-5} \pm 0.27\text{e}{-5}$	$\mathbf{193.08 \pm 5.6}$	**2.95**	$7.72\text{e}{-5} \pm 0.26\text{e}{-5}$	$\mathbf{146.59 \pm 4.3}$	**2.94**

Table 26: Sensitivity to reduced dimension P with uncertainty on AID. Results are reported as mean ± standard deviation across repeated runs. T denotes test time.

	Structured		Unstructured	
P	**Loss ↓**	**T (s) ↓**	**Loss ↓**	**T (s) ↓**
4	$5.2\text{e}{-5} \pm 0.20\text{e}{-5}$	146.59 ± 4.3	$5.6\text{e}{-5} \pm 0.22\text{e}{-5}$	193.08 ± 5.6
8	$\mathbf{5.0\text{e}{-5} \pm 0.19\text{e}{-5}}$	146.90 ± 4.4	$\mathbf{4.8\text{e}{-5} \pm 0.20\text{e}{-5}}$	193.70 ± 5.7
16	$5.8\text{e}{-5} \pm 0.23\text{e}{-5}$	147.60 ± 4.5	$6.3\text{e}{-5} \pm 0.25\text{e}{-5}$	194.83 ± 5.8
32	$6.2\text{e}{-5} \pm 0.25\text{e}{-5}$	147.74 ± 4.5	$6.7\text{e}{-5} \pm 0.27\text{e}{-5}$	195.21 ± 5.9

To directly measure activation compressibility, we compute the effective rank of the activation matrix. We define r_q as the minimum number of singular components required to capture $q\%$ of the activation energy. Lower r_q indicates that the activation representation is concentrated in a lower-dimensional subspace.

Table 27: Quantization results with uncertainty. Inference time is reported as mean ± standard deviation across repeated runs. Speedup denotes FP32-to-INT8 acceleration.

Model	**FP32 (ms)** ↓	**INT8 (ms)** ↓	**Speedup** ↑
LTC	305.58 ± 8.42	269.76 ± 7.31	$1.13\times$
SAPO	3.31 ± 0.12	2.96 ± 0.10	$1.12\times$

Table 28: Effective rank of activation representations with uncertainty on AID. Results are reported as mean ± standard deviation across prompts. Lower rank indicates greater low-dimensional compressibility.

Prompt	**Locality** ↑	r_{90} ↓	r_{95} ↓	r_{99} ↓
Unstructured	0.354 ± 0.026	78 ± 7	126 ± 10	247 ± 16
Structured	$\mathbf{0.734 \pm 0.025}$	$\mathbf{34 \pm 4}$	$\mathbf{61 \pm 6}$	$\mathbf{137 \pm 11}$

Structured prompts exhibit higher activation locality and lower effective rank across all three energy thresholds. In particular, the number of components required to capture 90% of the activation energy decreases from 78 to 34, while r_{95} decreases from 126 to 61 and r_{99} from 247 to 137. This indicates that structured prompts concentrate activation representations into a smaller subspace, providing additional empirical motivation for the $H \to P$ reduced-order transformation.