PARAMETER-EFFICIENT TEMPORAL DOMAIN GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Temporal domain generalization (TDG) learns from a sequence of observed domains to predict models for unseen future domains. Existing parameter-dynamics methods become prohibitively expensive at modern backbone scales because their temporal encoders and generators operate on the entire adapted parameter state. Replacing full-model updates with LoRA does not remove this failure mode: the dynamics model must still encode and reconstruct every adapter parameter, making the auxiliary model itself prohibitively large for modern backbones. We propose Low-rank Time-varying Core (LoTiC), a parameter-efficient TDG method that decomposes temporal updates into shared low-rank bases and a compact core. The bases define a stable update subspace, while only the core evolves. LoTiC therefore learns temporal dynamics in an $r \times r$ space instead of tracking the full adapter. We justify this factorization through the geometry of parameter increments and support continuous, recurrent, and time-conditioned core models. Experiments span six controlled and real-world benchmarks, three data modalities, classification and generation, and backbones from 66M to 8B parameters. LoTiC ranks first in 14 of 15 settings, including near-perfect future-domain accuracy with an 8B backbone on Rotating 2-Moons. It adds only 0.13–0.51% trainable parameters beyond LoRA and trains 1.8–7.1$\times$ faster than parameter-dynamics baselines. These results establish that temporal parameter dynamics can remain both learnable and efficient as pretrained backbones scale. Code is available at `https://github.com/dfwsds/LoTiC`.

1 INTRODUCTION

Many deployment environments change in an ordered rather than arbitrary manner. News topics follow current events, scientific terminology develops over years, and visual appearance shifts across decades. Domain generalization (DG) (Tremblay et al., 2018; Huang et al., 2020; Du et al., 2020; Qiao et al., 2020; Wang et al., 2022) prepares a model for unseen domains, but usually treats the observed domains as an unordered collection. Temporal domain generalization (TDG) (Li et al., 2020; Nasery et al., 2021; Qin et al., 2022; Bai et al., 2023; Yong et al., 2023; Zeng et al., 2024; Cai et al., 2024; Yu et al., 2025) instead uses their temporal order to learn how the predictive function evolves and to continue that evolution into future domains.

Parameter-dynamics methods provide a direct way to achieve this goal. DRAIN (Bai et al., 2023), Koodos (Cai et al., 2024), and FreKoo (Yu et al., 2025) learn recurrent, continuous-time, or spectral dynamics over a sequence of model parameters. Their temporal state, however, remains tied to the parameters being evolved. This becomes a decisive limitation for pretrained backbones. Replacing full-model updates with LoRA adapters reduces the state substantially, but does not remove the dependence. The auxiliary model must still encode, evolve, and decode the complete adapter. As the backbone grows, the dynamic state grows with it, and a simple temporal law must be recovered from an increasingly large parameter space.

This scaling problem should not imply that temporal structure disappears in large models. Model performance still deteriorates as deployment moves away from the training period (Luu et al., 2022; Lazaridou et al., 2021), while models adapted at nearby times exhibit aligned update directions and coherent weight-space trajectories (Nylund et al., 2024; Dziadzio et al., 2025). Parameter-efficient fine-tuning offers a useful starting point because a pretrained model at time t can be written as $W_{\mathrm{pre}} + \Delta W_t$. Yet independently learned LoRA factors do not provide a stable coordinate system for the update trajectory. Their non-unique factorizations can describe similar updates with different bases, leaving the temporal model to separate genuine evolution from changes of parameterization.

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.

These observations expose three challenges for parameter-efficient TDG. **1) A geometric foundation.** It is unclear whether the temporal geometry of full-model optima is preserved when the model is represented through parameter-efficient updates. **2) A scalable temporal state.** LoRA reduces each update, but the complete adapter can still contain millions of parameters, and its non-unique factors obscure the evolution that should be modeled. **3) A general dynamics interface.** Temporal shifts may be smooth, path-dependent, or directly indexed by time. Supporting these regimes should not require rebuilding the adapter or enlarging the evolved state.

We propose Low-rank Time-varying Core (LoTiC) to make the evolving object explicit and compact. LoTiC learns shared low-rank bases that define a common update subspace across domains and concentrates temporal variation in a small core. Future models are obtained by continuing only the core sequence. This construction reduces the modeled state from the complete LoRA parameter vector to a compact representation and admits different dynamics without changing the adapter. We study continuous flow, recurrent, and direct time-to-core parameterizations. Our analysis further shows why temporal geometry is preserved in the parameter-increment space and when shared bases provide a stable approximation to domain-wise updates.

We evaluate LoTiC against eight baselines on six benchmarks covering controlled and natural shifts, tabular, visual, and textual inputs, and classification and generation. The backbones range from 66M to 8B parameters. LoTiC ranks first in 14 of 15 settings. On the controlled benchmarks, it recovers the prescribed evolution of the decision function and maintains over 97% future-domain accuracy on Rotating 2-Moons even with an 8B backbone after observing only 15 domains. On the real-world benchmarks, it leads in nine of ten settings and improves ROUGE-L F1 by up to 16.3% on temporal news generation. These gains require only 0.13–0.51% more trainable parameters than ordinary LoRA, and LoTiC trains 1.8–7.1$\times$ faster than parameter-dynamics baselines.

Our contributions are threefold.

- We formulate TDG in the parameter-increment space and show that translating a low-dimensional manifold of model optima preserves its dimension and embedded structure.
- We introduce a shared-basis factorization and derive sufficient stability conditions that confine temporal evolution to an $r \times r$ core with LoRA-like overhead.
- We instantiate the core with continuous, recurrent, and time-conditioned dynamics and establish its effectiveness across six benchmarks, eight baselines, and backbones up to 8B parameters.

2 Related Work

Temporal domain generalization. Domain generalization (DG) learns from multiple source domains without access to target-domain data (Tremblay et al., 2018; Huang et al., 2020; Du et al., 2020; Qiao et al., 2020; Wang et al., 2022). TDG additionally uses the chronological order of source domains to generalize to future ones. GI (Nasery et al., 2021) regularizes temporal changes through a Taylor expansion. DRAIN (Bai et al., 2023) recurrently generates future parameters, TKNets (Zeng et al., 2024) linearizes feature dynamics with Koopman theory, Koodos (Cai et al., 2024) models continuous and irregularly sampled parameter trajectories, and FreKoo (Yu et al., 2025) separates their trend and periodic components. These methods differ in their temporal mechanisms, but their encoders or generators remain coupled to the complete parameter state being evolved. This coupling makes parameter dynamics prohibitive for modern pretrained backbones, even when full-model updates are replaced by adapters.

Temporal adaptation of pretrained models. Fine-tuned language models encode temporal information in weight space (Nylund et al., 2024). TEA (Liu et al., 2025) combines temporal experts in a PCA-induced subspace, while TIME (Dziadzio et al., 2025) merges models along initialization and deployment axes. PromptTDG (Hosseini et al., 2025) instead predicts future prompts from their temporal sequence. These approaches show that temporal adaptation can exploit pretrained models, but they do not factor the evolving update into a shared spatial subspace and a compact temporal state. LoTiC addresses this distinction directly.

Parameter-efficient fine-tuning. PEFT adapts pretrained models with a small trainable state. Its main forms are additive modules (Houlsby et al., 2019; Mahabadi et al., 2021), selective updates (Liu et al., 2022; Sung et al., 2021), low-dimensional reparameterizations (Hu et al., 2022; Wu et al.,

2024), and hybrid designs (Mao et al., 2022; Zhou et al., 2024). We build on LoRA (Hu et al., 2022), which represents each weight update with two low-rank factors. LoTiC is not a new static adapter. It introduces a shared coordinate system in which only a compact core evolves over time.

3 Problem Definition

We consider T observed domains indexed by ordered timestamps $t_1 < \cdots < t_T$. Domain $\mathcal{D}_t = \{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^{N_t}$ contains samples from a time-dependent distribution $P_t(X, Y)$. Changes in P_t may affect the input distribution, label prevalence, or input–output relationship. Unlike standard DG, the ordering is informative: training may use only $\{\mathcal{D}_{t_1}, \ldots, \mathcal{D}_{t_T}\}$, whereas data from any future domain $\mathcal{D}_\tau$, $\tau > t_T$, are unavailable.

Let W_{pre} be a pretrained backbone. At time t, an update $\Delta W(t)$ defines the predictor $g_{W_{\text{pre}}+\Delta W(t)}$. The objective of TDG is to learn the evolution of these updates from the observed sequence and produce $\Delta W(\tau)$ for unseen future timestamps $\tau > t_T$. The same formulation covers classification and generation, discrete and continuous timestamps, and different pretrained architectures.

4 Methodology

In this section, we develop LoTiC for parameter-efficient temporal domain generalization. We first motivate a low-dimensional manifold structure for the trajectory of optimal updates under distribution shift. We then introduce a manifold-constrained factorization that separates time-invariant spatial bases from a time-varying temporal core. This makes the temporal state independent of the full adapter dimension. Finally, we describe continuous, recurrent, and time-conditioned parameterizations of the core.

4.1 The Geometry of Parameter Increments for Temporal Domains

Following prior parameter-dynamics TDG (Cai et al., 2024; 2025; Yu et al., 2025), we model temporal shifts as a structured latent process and assume that the corresponding full-parameter optima $\{W_t^*\}_{t=1}^T$ are contained in a low-dimensional embedded manifold $\mathcal{M}$. This assumption captures coherent physical, social, or semantic evolution rather than arbitrary changes between domains. We now state its consequence in the parameter-increment space.

Definition 1 (Parameter increments)**.** *Fix a pretrained initialization $W_{\text{pre}} \in \mathbb{R}^p$. For each time step $t \in [T]$, define the optimal increment*

$$\Delta W_t^* := W_t^* - W_{\text{pre}}. \tag{1}$$

Define the map $\phi : \mathbb{R}^p \to \mathbb{R}^p$ by

$$\phi(W) := W - W_{\text{pre}}. \tag{2}$$

Lemma 1 (Embedded manifolds)**.** *Let $\mathcal{M} \subset \mathbb{R}^p$ be an embedded m-dimensional submanifold. Then $\mathcal{M}' := \phi(\mathcal{M}) = \mathcal{M} - W_{\text{pre}}$ is also an embedded m-dimensional submanifold of $\mathbb{R}^p$.*

Proof. The map $\phi(W) = W - W_{\text{pre}}$ is smooth and bijective with smooth inverse $\phi^{-1}(U) = U + W_{\text{pre}}$; hence ϕ is a diffeomorphism on $\mathbb{R}^p$. The image of an embedded submanifold under a diffeomorphism is an embedded submanifold of the same dimension. Therefore $\mathcal{M}' = \phi(\mathcal{M})$ is an embedded m-dimensional submanifold. □

Lemma 2 (Parameter-increment manifold)**.** *Given $\{W_t^*\}_{t=1}^T \subset \mathcal{M}$ for some embedded m-dimensional submanifold $\mathcal{M} \subset \mathbb{R}^p$. Then $\{\Delta W_t^*\}_{t=1}^T \subset \mathcal{M}'$, where $\mathcal{M}' := \mathcal{M} - W_{\text{pre}}$ is also an embedded m-dimensional submanifold.*

Proof. By Definition 1, $\Delta W_t^* = \phi(W_t^*)$. Since $W_t^* \in \mathcal{M}$ for each t, it follows that $\Delta W_t^* \in \phi(\mathcal{M}) = \mathcal{M}'$. The manifold claim follows from Lemma 1. □

This result motivates working in the parameter-increment space: modeling temporal generalization via ΔW_t provides a principled reparameterization for large pretrained models under temporal domain shifts, and it supports extrapolation by learning trajectories on a low-dimensional manifold.

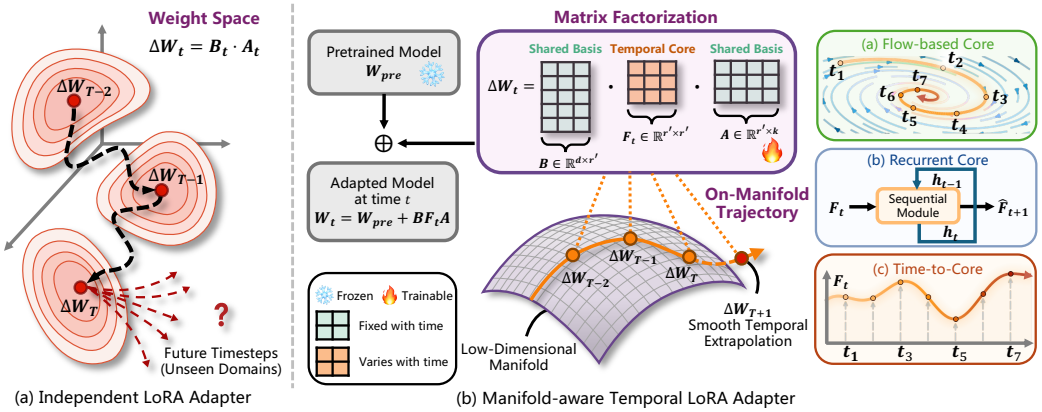


Figure 1: (a) Independent domain-wise LoRA optimization can yield scattered updates without temporal consistency. (b) With shared bases B and A, temporal adaptation is captured by the compact core F_t. The temporal core can be instantiated by different sequence modeling modules.

4.2 Manifold-Constrained Low-Rank Factorization

Prior methods project complete parameter states into compressed representations and learn dynamics between them (Bai et al., 2023; Cai et al., 2024). On modern pretrained backbones, however, the required encoder and decoder remain coupled to the full adapted state and become prohibitively large. Directly generating full weights may also compromise pretrained capabilities (Luo et al., 2025; Li et al., 2024). We instead model temporal dynamics inside the low-rank update space induced by PEFT.

4.2.1 LoRA as a Basis for Temporal Generalization

In a standard PEFT pipeline, each temporal domain is adapted independently with its own LoRA factors (B_t, A_t). While this provides domain-specific flexibility, it imposes no mechanism to coordinate the learned updates across time. For over-parameterized models, the loss landscape typically contains broad sets of near-optimal solutions, where distinct parameter configurations can achieve comparable losses and are connected by low-loss paths (Garipov et al., 2018; Draxler et al., 2018). Consequently, independently optimized LoRA updates $\{(B_t, A_t)\}_{t=1}^{T}$ can be arbitrarily scattered, as illustrated in Fig. 1(a), making future-domain extrapolation unstable and underdetermined.

The issue does not undermine LoRA, but shows that low-rank adaptation alone is insufficient for temporal generalization. Given a pretrained parameter W_{pre}, LoRA represents adaptation by a low-rank update $\Delta W = BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and $\text{rank}(\Delta W) \le r$. Thus, LoRA restricts adaptation to an at-most-rank-r update subspace, providing a natural low-dimensional representation required by temporal relation modeling among domain-specific updates. Together with Lemmas 1–2, this motivates a temporal manifold constrained extension of LoRA for future-domain generalization.

4.2.2 Subspace-Shared Time-Varying Parameterization

Let $\{\Delta W_t = B_t A_t\}_{t=1}^{T}$ be a sequence of LoRA updates where $B_t \in \mathbb{R}^{d\times r}$ and $A_t \in \mathbb{R}^{r\times k}$. Although each update has rank at most r, the union of its factor subspaces can have dimension as large as Tr. Define the global column and row subspaces as $\mathcal{C} = \text{span}(\bigcup_{t=1}^{T} \text{col}(B_t))$ and $\mathcal{R} = \text{span}(\bigcup_{t=1}^{T} \text{row}(A_t))$. Choose fixed matrices $B \in \mathbb{R}^{d\times r'}$ and $A \in \mathbb{R}^{r'\times k}$ whose column and row spaces contain $\mathcal{C}$ and $\mathcal{R}$, respectively, with $r' \ge \max\{\dim(\mathcal{C}), \dim(\mathcal{R})\}$. Since $\text{col}(B_t) \subseteq \text{col}(B)$, there exists a coefficient matrix $C_t \in \mathbb{R}^{r'\times r}$ such that

$$B_t = BC_t \tag{3}$$

Similarly, because $\text{row}(A_t) \subseteq \text{row}(A)$, there exists $D_t \in \mathbb{R}^{r\times r'}$ such that

$$A_t = D_t A. \tag{4}$$

Substituting both factorizations gives

$$\Delta W_t = B_t A_t = (BC_t)(D_t A) = B(C_t D_t)A \tag{5}$$

Defining $F_t := C_t D_t \in \mathbb{R}^{r' \times r'}$ yields the manifold-constrained factorization, which separates the time-varying update into fixed ambient bases and a dynamic temporal core:

$$\Delta W(t) = B \cdot F_t \cdot A \tag{6}$$

As illustrated in Fig. 1(b), the shared bases B and A define a time-invariant update coordinate system, while F_t captures domain-specific adaptation. Future-domain generalization then requires extrapolating only this compact core sequence, as detailed in Sec. 4.4. The induced updates $\Delta W_t = BF_tA$ vary coherently within a shared subspace, avoiding the linear parameter growth of independent domain-wise adapters analyzed in Appendix A.6. The construction is exact when r' spans the union subspaces. In practice, LoTiC sets $r' = r$ and learns a compact shared support; Sec. 4.3 gives sufficient conditions under which the domain-wise factor subspaces remain close to such a support.

4.3 Stability and Justification of the Shared-Basis Form

To justify the shared-basis parameterization $\Delta W_t = BF_tA$, we analyze a sequence of domain-wise LoRA factors connected by gradient updates. Under explicit stability conditions, their factor subspaces do not drift far over time. Concretely, starting from (B_1, A_1), we prove that the components of B_t orthogonal to $\mathrm{col}(B_1)$ and of A_t orthogonal to $\mathrm{row}(A_1)$ remain small throughout the updates, up to a controlled leakage term. This establishes that using fixed shared subspaces for B and A is meaningful, and that temporal variation can be concentrated into a small core F_t.

Definition 2 (Reference subspaces and leakage)**.** *Let $B := B_1$ and $A := A_1$. Define the reference subspaces*

$$\mathcal{S}_B := \mathrm{col}(B_1) \subseteq \mathbb{R}^d, \qquad \mathcal{S}_A := \mathrm{row}(A_1) \subseteq \mathbb{R}^k. \tag{7}$$

Let P_B and P_A be the orthogonal projectors onto $\mathcal{S}_B$ and $\mathcal{S}_A$, respectively. For LoRA factors (B_t, A_t), define

$$e_t^B := \|(I - P_B)B_t\|_F, \qquad e_t^A := \|A_t(I - P_A)\|_F. \tag{8}$$

Lemma 3 (Subspace stability under gradient descent (Preliminary Lemma of Theorem 4))**.** *Suppose the LoRA factors (B_t, A_t) are updated by gradient descent:*

$$B_{t+1} = B_t - \eta \nabla_B \mathcal{J}_t(B_t, A_t), \qquad A_{t+1} = A_t - \eta \nabla_A \mathcal{J}_t(B_t, A_t). \tag{9}$$

Under the stability assumptions 1, 2 and 3 stated in Appendix B.2, if the initial leakage e_1^B, e_1^A and the gradient leakage terms are small, then e_t^B and e_t^A remain uniformly small throughout training. Consequently, for $\Delta W_t := B_t A_t$,

$$\|(I - P_B)\Delta W_t\|_F \le e_t^B \|A_t\|_2, \qquad \|\Delta W_t(I - P_A)\|_F \le \|B_t\|_2 e_t^A. \tag{10}$$

Thus, ΔW_t has small leakage outside the shared subspace pair $(\mathcal{S}_B, \mathcal{S}_A)$, which justifies using fixed shared subspaces while allowing the coordinates inside them to vary over time.

The formal theorem and proof are provided in Appendix B.1. Lemma 3 shows that, under the dissipativity conditions in Assumption 2, the out-of-subspace energies $(e_t^B)^2$ and $(e_t^A)^2$ follow a contractive recursion with bounded additive terms. Hence, if the initialization is aligned ($e_1^B = e_1^A = 0$) and leakage is small, then $\mathrm{col}(B_t)$ stays close to $\mathrm{col}(B_1)$ and $\mathrm{row}(A_t)$ stays close to $\mathrm{row}(A_1)$ for all t, which in turn implies that the induced updates $\Delta W_t = B_t A_t$ largely remain supported on the shared subspaces, as quantified by (10). Therefore, a shared (B, A) with a time-varying core is a consistent structural restriction rather than an ad hoc constraint.

4.4 Instantiations of the Temporal Core

Under the shared-basis factorization $\Delta W_t = BF_tA$, temporal variation is concentrated in the low-dimensional core F_t. When B has full column rank and A has full row rank, the map $F \mapsto BFA$ is injective, so the core provides valid coordinates for the induced update subspace. As illustrated in Fig. 1(b), we parameterize F_t according to the inductive bias of the underlying dynamics and consider three regimes:

- **Flow-based Core:** If the data evolution follows a smooth continuous flow governed by an autonomous linear dynamical system, we use a flow-based core. Specifically, the core evolves as

$$F_t = \exp(tW), \qquad W \in \mathbb{R}^{r' \times r'}. \tag{11}$$

  Here, W is learnable, and $\exp(\cdot)$ denotes the matrix exponential. This form imposes a continuous-time structure on the core trajectory.

- **Recurrent Core:** When temporal evolution is sequential or path-dependent, we parameterize the core with a recurrent model:

$$h_t = \mathrm{RNN}\big(h_{t-1}, \mathrm{vec}(F_{t-1})\big), \qquad \mathrm{vec}(F_t) = W_o h_t. \tag{12}$$

  This formulation captures autoregressive dependencies from previous time steps and is suited to sequential temporal shifts.

- **Time-to-Core:** When temporal domains are indexed by explicit time or other descriptors, we directly parameterize the core as a function of time:

$$\mathrm{vec}(F_t) = f_\psi(t). \tag{13}$$

  Here, f_ψ is a lightweight time-conditioned network that maps the temporal index to F_t.

5 Experiments

We organize our experiments around four questions. **RQ1.** Can LoTiC recover the correct evolution of the decision function? **RQ2.** Can LoTiC generalize across benchmarks, tasks, modalities, and backbones? **RQ3.** Can LoTiC remain parameter efficient as the backbone scales? **RQ4.** How do different temporal parameterizations affect performance? Sensitivity and convergence are summarized in Section 5.5 and detailed in Appendices A.4–A.5, with complexity analysis in Appendix A.6.

Datasets. We use six established temporal benchmarks. Rotating 2-Moons and Rotated MNIST are standard TDG controlled testbeds (Bai et al., 2023; Cai et al., 2024). Yearbook, HuffPost, and arXiv are from Wild-Time (Yao et al., 2022). We further construct a news-continuation task from the temporally indexed WMT News Crawl corpus (Luu et al., 2022). They span tabular, visual, and textual modalities, classification and generation, and controlled and natural shifts over decades. Dataset details and temporal splits are provided in Appendix A.1.

Baselines. We compare LoTiC with eight baselines from three groups. **Practical baselines.** ERM, LastDomain, and IncFT. **Parameter-dynamics TDG methods.** DRAIN (Bai et al., 2023), Koodos (Cai et al., 2024), and FreKoo (Yu et al., 2025) represent state-of-the-art approaches. **Temporal methods for pretrained models.** PromptTDG (Hosseini et al., 2025) and TEA (Liu et al., 2025) are the latest temporal methods designed for pretrained backbones. For scalability, we apply TDG methods to LoRA parameters while retaining their original temporal mechanisms. Baseline definitions and adaptation details are provided in Appendix A.2.

Protocol and metrics. For each dataset, we evaluate backbones at different scales, ranging from 66M to 8B parameters overall. We report error rate (%, lower is better) for classification and ROUGE-L F1 (higher is better) for text generation. Unless stated otherwise, LoTiC uses the flow-based core. All methods are trained on the observed domains and then deployed to all unseen future domains, with the same data splits and training budgets. We repeat each experiment with three seeds and report the mean and standard deviation. Dataset-specific configurations are provided in Appendix A.3.

5.1 Controlled Temporal Extrapolation

Controlled shifts provide a known temporal trajectory that real-world benchmarks typically do not. We consider Rotating 2-Moons, a binary classification problem with two interleaving point clouds, and Rotated MNIST, a visual classification problem with digit images. In both testbeds, consecutive domains differ by an 18° rotation, producing 20 domains over a complete 360° cycle. The first 15 domains are used for training, and the remaining five are held out for future-domain evaluation. Figure 2(a) illustrates the Rotating 2-Moons domain sequence, along which a backbone model must evolve its decision function beyond the observed domains.

Table 1(a) summarizes the performance, while Figure 2(b,c) shows how the learned decision functions evolve. **1) Quantitatively, LoTiC maintains strong performance as the backbone scales to 8B parameters.** LoTiC achieves over 97% future-domain accuracy on Rotating 2-Moons and reduces the strongest baseline's error by 52.9–54.0% on Rotated MNIST. Koodos-LoRA and FreKoo-LoRA already incur double-digit error at 66M and become infeasible at larger scales, whereas LoTiC remains near-perfect even on an 8B backbone after observing only 15 domains. **2) Qualitatively, LoTiC's decision boundary follows the prescribed evolution.** Figure 2(b,c) shows that Koodos-LoRA and

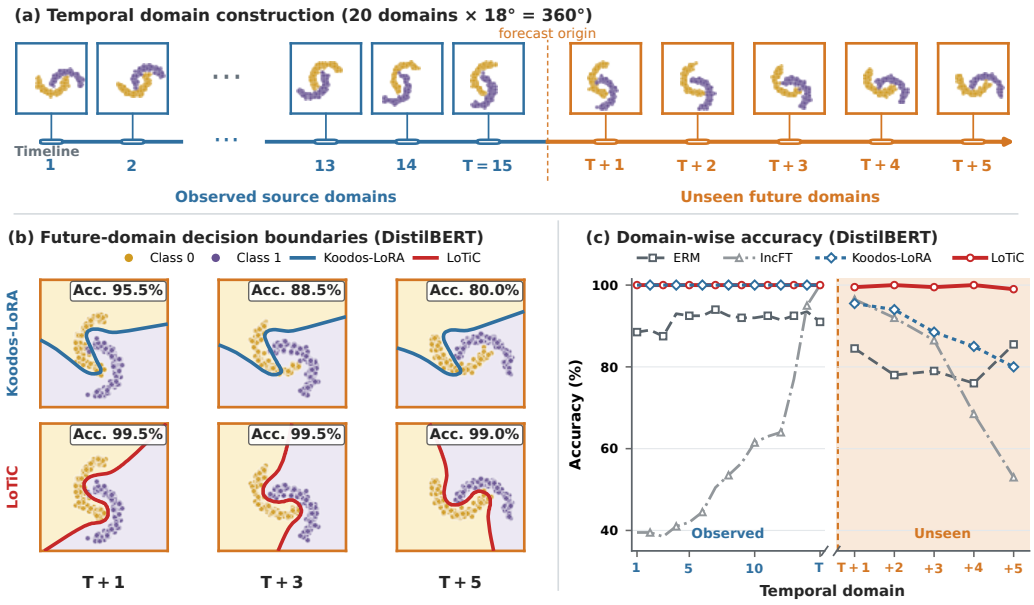


Figure 2: Controlled temporal extrapolation on Rotating 2-Moons. (a) Twenty domains with 15 observed and 5 unseen. (b) Future-domain decision boundaries of Koodos-LoRA and LoTiC. (c) Domain-wise accuracy; the observed and unseen intervals use different x-axis scales.

LoTiC fit the observed domains equally well, and Koodos-LoRA remains accurate at $T + 1$. Yet its boundary then stalls, while LoTiC continues the correct rotation through $T + 5$. The contrast lies in where temporal dynamics are modeled. Koodos-LoRA compresses and evolves the complete adapter, whereas LoTiC isolates the evolution within a compact core shared across domains.

5.2 Generalization Across Benchmarks and Backbones

Table 1(b) evaluates LoTiC under natural temporal shifts across four real-world benchmarks. **1) LoTiC achieves consistent gains across real-world shifts.** It ranks first in nine of ten settings and reduces error by 20.5–37.8% over the strongest baseline on Yearbook. **2) The advantage persists as**

Table 1: Future-domain performance, reported as mean (standard deviation). Classification benchmarks use error rate (%, ↓); WMT uses ROUGE-L F1 (↑). Bold and underlined values are best and second-best. -L denotes LoRA; OOM denotes out of memory.

(a) Controlled benchmarks

Dataset	Backbone	Scale	ERM	Last Domain	IncFT	DRAIN-L	Koodos-L	FreKoo-L	Prompt TDG	TEA	**LoTiC**
Rot-2-Moons	DistilBERT	**66M**	18.5 (1.0)	12.7 (0.9)	28.2 (6.5)	21.4 (5.9)	11.9 (0.5)	11.8 (1.1)	25.9 (4.2)	21.8 (3.5)	**0.6 (0.3)**
	Qwen3	**0.6B**	23.5 (4.4)	13.7 (1.3)	24.7 (1.0)	OOM	OOM	OOM	49.8 (15.9)	25.6 (1.3)	**2.4 (2.5)**
	LLaMA-3	**8B**	21.8 (6.1)	12.7 (0.6)	22.1 (7.2)	OOM	OOM	OOM	23.4 (5.4)	17.3 (1.2)	**0.9 (0.7)**
Rot-MNIST	DINOv2-B	**86M**	20.0 (1.2)	45.9 (2.0)	44.7 (2.0)	61.8 (2.5)	61.3 (5.2)	28.7 (1.0)	19.1 (1.4)	18.7 (0.6)	**8.6 (2.4)**
	DINOv2-L	**0.3B**	16.6 (1.9)	37.5 (3.3)	37.8 (2.7)	OOM	OOM	OOM	15.5 (1.1)	15.6 (0.4)	**7.3 (1.0)**

(b) Real-world benchmarks

Dataset	Backbone	ERM	LastDomain	IncFT	DRAIN-L	Koodos-L	FreKoo-L	PromptTDG	TEA	**LoTiC**
Yearbook	DINOv2-B	13.8 (0.9)	11.8 (1.3)	11.7 (0.3)	17.8 (0.5)	13.5 (0.4)	16.9 (3.4)	13.9 (0.7)	14.4 (0.3)	**9.3 (0.4)**
	DINOv2-L	12.5 (0.9)	9.8 (0.5)	10.9 (1.0)	OOM	OOM	OOM	13.4 (0.3)	13.8 (0.6)	**6.1 (0.6)**
HuffPost	DistilBERT	29.2 (0.2)	29.6 (0.3)	29.1 (0.4)	29.3 (0.6)	30.7 (0.4)	30.7 (0.1)	29.3 (0.2)	30.1 (0.3)	**28.4 (0.2)**
	Qwen3	30.7 (0.5)	30.6 (0.1)	31.8 (0.2)	OOM	OOM	OOM	30.0 (0.0)	29.7 (0.2)	**27.9 (0.5)**
	LLaMA-3	19.3 (0.4)	19.1 (0.0)	19.5 (0.4)	OOM	OOM	OOM	33.7 (21.3)	19.3 (0.1)	**19.0 (0.2)**
arXiv	DistilBERT	54.1 (0.1)	59.9 (0.1)	56.9 (0.1)	57.6 (0.3)	68.8 (8.9)	63.1 (0.5)	52.6 (0.3)	53.4 (0.1)	**51.8 (0.0)**
	Qwen3	53.2 (0.4)	53.5 (0.1)	54.5 (0.1)	OOM	OOM	OOM	52.0 (0.1)	51.1 (0.3)	**48.4 (0.3)**
	LLaMA-3	43.5 (0.3)	42.1 (0.2)	**41.8 (0.2)**	OOM	OOM	OOM	43.1 (0.2)	42.8 (0.2)	42.6 (0.2)
WMT	Qwen3	11.4 (0.2)	9.5 (0.3)	10.7 (0.1)	OOM	OOM	OOM	11.2 (0.2)	11.4 (0.1)	**13.0 (0.0)**
	LLaMA-3	12.2 (0.1)	11.8 (0.2)	11.5 (0.0)	OOM	OOM	OOM	12.3 (0.1)	12.2 (0.2)	**14.3 (0.2)**

the backbone scales. LoTiC remains best on two of the three LLaMA-3 settings and is only 0.8 error points behind the best result on arXiv. Larger pretrained models therefore do not remove the value of modeling temporal evolution. **3) LoTiC transfers across modalities, tasks, and architectures.** The same temporal formulation leads on image classification with DINOv2 and text classification with DistilBERT and Qwen3. On WMT generation, its Qwen3 and LLaMA-3 variants improve ROUGE-L F1 over the strongest baseline by 14.0% and 16.3%, respectively.

5.3 Parameter Efficiency and Scalability

We examine two questions about parameter efficiency. **1)** Can LoTiC improve future-domain performance without the parameter, storage, and runtime costs of existing TDG methods? **2)** Does its resource use continue to scale like ordinary LoRA as the backbone grows?

Table 2: Efficiency on Rot-2-Moons (DistilBERT, 66M).

Method	Total params	Trainable params	Trainable vs. LoRA	Ckpt. (MB)	Train (s)	Test (s)	Test err. (%)
ERM	66.59M	225,026	1.00×	0.900	10	< 0.1	18.5
DRAIN–LoRA	182.01M	115.64M	513.91×	462.6	165.3	< 0.1	21.4
Koodos–LoRA	532.0M	465.65M	2069.30×	1862.6	580	< 0.1	11.9
FreKoo–LoRA	763.0M	696.67M	3095.95×	2786.7	150	< 0.1	11.8
LoTiC	**66.59M**	**226,178**	**1.01×**	**0.905**	**82**	< 0.1	**0.6**

LoTiC substantially reduces the cost of parameter dynamics. Table 2 compares LoTiC with ERM and three parameter-dynamics TDG methods on Rot-2-Moons with DistilBERT. The dominant cost of existing parameter-dynamics methods comes from the machinery used to evolve the adapter. Starting from the same 225K-parameter LoRA adapter, DRAIN–LoRA, Koodos–LoRA, and FreKoo–LoRA expand the trainable state to 514–3,096× that of LoRA. By confining the evolution to a compact core, LoTiC limits the trainable-parameter overhead to only 0.51% and trains 1.8–7.1× faster. Parameter dynamics therefore no longer dominate the resource budget.

LoTiC preserves LoRA-like parameter efficiency across architectures, modalities, and scales. Table 3 shows negligible LoTiC overhead across text encoders, vision transformers, and decoder-only LMs from 66M to 8B. Across this 120× range, the temporal component grows only 5.3×, and its relative overhead falls from 0.51% to 0.13%. We next scale a controlled LLaMA-3 family by depth. Figure 3(a) measures peak GPU memory, while Figure 3(b) measures training time relative to the smallest backbone. For Koodos–LoRA, operating on LoRA rather than full-model parameters already reduces the modeled parameter space substantially. Nevertheless, it still learns dynamics over the entire adapter. Its auxiliary dynamics model therefore grows with the adapter dimension and quickly reaches the VRAM limit. LoTiC, however, closely tracks ERM in both memory and training time. Its scaling limit is therefore set by the underlying LoRA training rather than temporal modeling. Appendix A.6 provides the corresponding complexity analysis.

Table 3: Parameter overhead by backbone.

Backbone	LoRA	**LoTiC**	Additional
DistilBERT (66M)	225,026	**226,178**	+1,152 (0.51%)
DINOv2-B (86M)	450,058	**452,362**	+2,304 (0.51%)
DINOv2-L (0.3B)	1,189,898	**1,194,506**	+4,608 (0.39%)
Qwen3 (0.6B)	1,610,754	**1,616,130**	+5,376 (0.33%)
LLaMA-3 (8B)	4,739,074	**4,745,218**	+6,144 (0.13%)

5.4 Temporal Structure and Parameterization

Table 4 ablates the temporal parameterization on Yearbook with DINOv2-B. **Isolating temporal variation is critical.** Evolving the high-dimensional factors yields 33.3–51.5% error, whereas BF_tA improves static LoRA from 13.8% to 9.3%. Shared B and A retain the update subspace while only F_t evolves. **The factorization supports different temporal models.** Recurrent and Time-to-Core achieve 12.9% and 11.5%, both outperforming static LoRA. Flow performs best at 9.3%, indicating that continuous dynamics match the domain evolution.

Table 4: Yearbook ablation.

Form	Error (%, ↓)
(a) Factorization	
Static BA (LoRA)	13.8 (0.9)
B_tA	33.3
BA_t	51.5
B_tA_t	51.5
$\mathbf{BF_tA}$	**9.3 (0.4)**
(b) Core model	
Flow	**9.3 (0.4)**
Recurrent	12.9
Time-to-Core	11.5 (0.3)

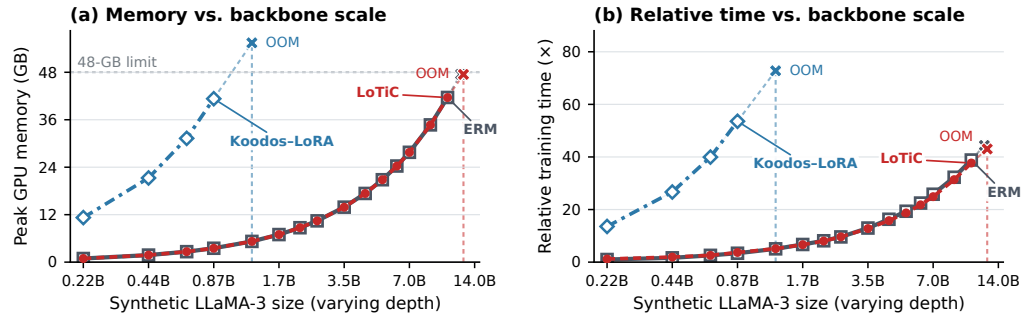


Figure 3: Efficiency at scale on a controlled LLaMA-3 family. Model scale is varied only by increasing the number of identical Transformer blocks. (a) Peak GPU memory. (b) Training time relative to the smallest model. Crosses indicate OOM on a 48-GB GPU.

5.5 Sensitivity and Convergence

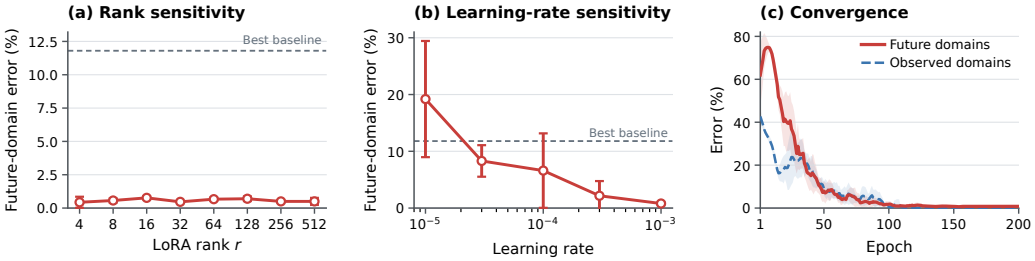


Figure 4: Sensitivity and convergence on Rotating 2-Moons with DistilBERT. Error bars and shading show the standard deviation over three seeds; the dashed line is the strongest baseline.

Across ranks from 4 to 512, LoTiC remains near 1% error, far below the strongest baseline. It is also stable for learning rates from 3×10^{-5} to 10^{-3}. Smaller rates underfit. Errors on observed and future domains decrease together, stabilize near 1%, and show diminishing variation across seeds. The final result therefore depends on neither narrow tuning nor an isolated checkpoint. Details are provided in Appendices A.4–A.5.

6 Conclusion

We introduced LoTiC, a parameter-efficient formulation of temporal domain generalization that confines parameter dynamics to a compact core within shared low-rank bases. This changes the object being evolved from the complete adapter to an $r \times r$ temporal state. Across controlled and real-world shifts, LoTiC generalizes across tasks, modalities, and backbones up to 8B parameters while retaining LoRA-like overhead. Its controlled trajectories and factorization ablations further show that both the shared update subspace and the temporal core are important for learning how a predictor should evolve beyond observed domains. These results show that parameter dynamics need not scale with the backbone to remain effective on future domains.

In accordance with the ICLR 2027 Author Guidelines, the AI use, reproducibility, and ethics statements on this page are excluded from the main-text page limit.

AI USE STATEMENT

Generative AI tools were used for language editing and manuscript preparation, including improving grammar, clarity, readability, and LaTeX formatting. They were not used to generate the research ideas, methodology, theoretical results, experimental design, or scientific conclusions reported in this paper. All AI-assisted language revisions were reviewed and verified by the authors.

REPRODUCIBILITY STATEMENT

We provide the following materials and controls to support independent reproduction of our empirical and theoretical results.

- **Code and configurations.** Training and evaluation code with experiment configurations is available at `https://github.com/dfwsds/LoTiC`.
- **Dataset construction.** Appendix A.1 reports data sources, task definitions, temporal-domain construction, and observed and future splits.
- **Comparison protocol.** All methods use the same temporal splits, preprocessing, backbone, task head, and evaluation protocol. Whenever computationally feasible, LoRA configurations are matched; method-specific settings follow the original implementations. Future-domain data are excluded from training and model selection.
- **Baseline implementations.** Appendix A.2 defines each baseline, its adaptation to pretrained backbones, and its future-domain rollout procedure.
- **Model specification.** Backbone families and scales, prediction heads, LoRA target modules, ranks, scaling factors, precision, and initialization are given in Appendix A.3.
- **Optimization and selection.** We specify the optimizer, schedule, learning rate, batch size, training length, regularization, gradient clipping, and checkpoint rule in Appendix A.3. Hyperparameters are selected using only observed-domain data.
- **Randomness and reporting.** Main benchmark results and sensitivity and convergence analyses use three seeds. We define the metrics and aggregation rule, weight future domains equally, and report means and standard deviations where available.
- **Environment and compute.** Appendix A.3 reports software versions, precision, accelerator type, and memory budget. We additionally report parameter counts, runtime, and peak memory; all OOM results use the same 48 GB device-memory limit.
- **Analyses and proofs.** Appendices A.4–A.6 provide sensitivity, convergence, and complexity analyses. Appendix B states the assumptions and provides the corresponding proofs.

ETHICS STATEMENT

This work develops a methodological and theoretical framework and evaluates it on established public benchmark datasets; it does not involve human-subject research or the collection of sensitive personal data, and we identify no specific ethical concerns.

REFERENCES

Guangji Bai, Chen Ling, and Liang Zhao. Temporal domain generalization with drift-aware dynamic neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

Zekun Cai, Guangji Bai, Renhe Jiang, Xuan Song, and Liang Zhao. Continuous temporal domain generalization. *Advances in Neural Information Processing Systems*, 37:127987–128014, 2024.

Zekun Cai, Yiheng Yao, Guangji Bai, Renhe Jiang, Xuan Song, Ryosuke Shibasaki, and Liang Zhao. Continuous domain generalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning (ICML)*, 2018.

Yingjun Du, Jun Xu, Huan Xiong, Qiang Qiu, Xiantong Zhen, Cees GM Snoek, and Ling Shao. Learning to learn with variational information bottleneck for domain generalization. In *European Conference on Computer Vision*, pp. 200–216, 2020.

Sebastian Dziadzio, Vishaal Udandarao, Karsten Roth, Ameya Prabhu, Zeynep Akata, Samuel Albanie, and Matthias Bethge. How to merge your multimodal models over time? In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 20479–20491, 2025.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Sepidehsadat Hosseini, Mengyao Zhai, Hossein Hajimirsadeghi, and Frederick Tung. Scalable temporal domain generalization via prompting. In *Second Workshop on Test-Time Adaptation: Putting Updates to the Test! at ICML 2025*, 2025.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *16th European Conference on Computer Vision, ECCV 2020*, pp. 124–140. Springer, 2020.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363, 2021.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Sequential learning for domain generalization. In *European conference on computer vision*, pp. 603–619. Springer, 2020.

Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. Revisiting catastrophic forgetting in large language model tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4297–4308, 2024.

Aoming Liu, Kevin Miller, Venkatesh Saligrama, Kate Saenko, Boqing Gong, Ser-Nam Lim, and Bryan A Plummer. Scaling up temporal domain generalization via temporal experts averaging. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 28196–28219, 2025.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A Smith. Time waits for no one! analysis and challenges of temporal misalignment. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5944–5958, 2022.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 565–576, 2021.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. Unipelt: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6253–6264, 2022.

Anshul Nasery, Soumyadeep Thakur, Vihari Piratla, Abir De, and Sunita Sarawagi. Training for the future: A simple gradient interpolation loss to generalize along time. *Advances in Neural Information Processing Systems*, 34:19198–19209, 2021.

Kai Nylund, Suchin Gururangan, and Noah A Smith. Time is encoded in the weights of finetuned language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2571–2587, 2024.

Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12556–12565, 2020.

Tiexin Qin, Shiqi Wang, and Haoliang Li. Generalizing to evolving domains with latent structure-aware sequential autoencoder. In *International Conference on Machine Learning*, pp. 18062–18082. PMLR, 2022.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.

Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 969–977, 2018.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and S Yu Philip. Generalizing to unseen domains: A survey on domain generalization. *IEEE transactions on knowledge and data engineering*, 35(8):8052–8072, 2022.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. Mixture-of-subspaces in low-rank adaptation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7880–7899, 2024.

Huaxiu Yao, Caroline Choi, Bochuan Cao, Yoonho Lee, Pang Wei Koh, and Chelsea Finn. Wild-time: A benchmark of in-the-wild distribution shift over time. In *Advances in Neural Information Processing Systems*, volume 35, pp. 10309–10324, 2022.

LIN Yong, Fan Zhou, Lu Tan, Lintao Ma, Jianmeng Liu, HE Yansu, Yuan Yuan, Yu Liu, James Y Zhang, Yujiu Yang, et al. Continuous invariance learning. In *The Twelfth International Conference on Learning Representations*, 2023.

En Yu, Jie Lu, Xiaoyu Yang, Guangquan Zhang, and Zhen Fang. Learning robust spectral dynamics for temporal domain generalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Qiuhao Zeng, Wei Wang, Fan Zhou, Gezheng Xu, Ruizhi Pu, Changjian Shui, Christian Gagné, Shichun Yang, Charles X Ling, and Boyu Wang. Generalizing across temporal domains with koopman operators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16651–16659, 2024.

Han Zhou, Xingchen Wan, Ivan Vulić, and Anna Korhonen. Autopeft: Automatic configuration search for parameter-efficient fine-tuning. *Transactions of the Association for Computational Linguistics*, 12:525–542, 2024.

A Experimental Details and Analyses

A.1 Datasets

Following established TDG evaluation protocols, we combine the widely used Rotating 2-Moons and Rotated MNIST testbeds (Bai et al., 2023; Cai et al., 2024; 2025; Yu et al., 2025) with three real-world benchmarks from Wild-Time (Yao et al., 2022): Yearbook, HuffPost, and arXiv. We further construct a temporal news-continuation benchmark from the English WMT News Crawl corpus, which has been used to study temporal adaptation in language models (Luu et al., 2022). Together, they cover both controlled and real-world temporal shifts, with inputs ranging from tabular data and images to text, and tasks ranging from binary and large-scale multi-class classification to text generation.

- **Rotating 2-Moons.** Following the controlled setting used in temporal domain generalization (Bai et al., 2023; Cai et al., 2024), we construct 20 temporally ordered domains from the two-moons distribution. Each domain contains 200 class-balanced points independently sampled with Gaussian noise of 0.10. Domain $t \in \{0, \ldots, 19\}$ is obtained by rotating the distribution counter-clockwise by $18^\circ t$ about the origin. We use the first 15 domains for training and the remaining five as unseen future domains, yielding a complete 360° evolution at an 18° temporal resolution.
- **Rotated MNIST.** We construct a visual counterpart of the controlled benchmark from MNIST (LeCun et al., 1998). A fixed random permutation of the original training split is used to select 4,000 non-overlapping images, with 200 images assigned to each of 20 domains. Images in domain t are rotated clockwise by $18^\circ t$ using bilinear interpolation and zero padding. All ten digit classes are retained. As in Rotating 2-Moons, domains 0–14 are observed during training and domains 15–19 are reserved for future-domain evaluation.
- **Yearbook.** Yearbook is a real-world visual benchmark from Wild-Time (Yao et al., 2022) for binary gender classification from 32×32 grayscale yearbook portraits. Its temporal shift reflects changes in appearance and fashion across decades. Our processed set contains 37,189 images collected from 1930 to 2013. We merge every two consecutive years into one temporal domain, producing 42 domains in total: 20 observed domains covering 1930–1969 and 22 unseen domains covering 1970–2013. The observed portion contains 14,156 training and 1,594 validation images, while all 21,439 images from the future years are used for evaluation.
- **HuffPost.** HuffPost (Yao et al., 2022) studies how news topics and language evolve with current events. The input is a news headline and the task is to predict one of 11 categories that occur throughout the collection. The benchmark contains 63,903 examples grouped into seven annual domains from 2012 to 2018. We use 2012–2015 as the observed period, comprising 35,558 training and 3,948 validation examples, and evaluate on all 24,397 examples from the future domains in 2016–2018.
- **arXiv.** The arXiv benchmark (Yao et al., 2022) captures temporal shifts in scientific terminology and topic prevalence. Given a paper title, the task is to predict its primary subject category among 172 classes. We form 16 annual domains spanning 2007–2022 and use a fixed, year-balanced subset of 170,000 examples. The ten domains from 2007–2016 are used for training and validation, and the six domains from 2017–2022 are held out for future-domain evaluation.
- **WMT News Crawl.** We construct a temporal news-continuation task from the English WMT News Crawl corpus, following its use in prior temporal adaptation studies (Luu et al., 2022). Given a 100–200-word news lead, the model predicts the next sentence of 8–26 words. Each year from 2012 to 2021 forms one domain. We use 2012–2017 as observed domains and 2018–2021 as unseen domains.

A.2 Baselines and Comparison Protocol

We compare LoTiC with eight baselines organized into the same three groups used in Section 5. Together, they disentangle the effects of time-agnostic adaptation, temporal ordering, explicit parameter dynamics, and mechanisms designed for pretrained backbones.

Practical baselines. **ERM** trains one LoRA adapter and task head on the union of all observed domains, providing a time-agnostic reference with the same backbone adaptation as LoTiC. **LastDomain** uses the same architecture but trains only on the most recent observed domain, testing whether recency alone explains future performance. **IncFT** updates one adapter sequentially from the earliest

to the latest observed domain without replay. It uses temporal order but does not learn an explicit rule for future model evolution.

Parameter-dynamics TDG methods. **DRAIN (Bai et al., 2023)** uses a recurrent generator to evolve model parameters and rolls the generator forward autoregressively for multiple future domains. **Koodos (Cai et al., 2024)** encodes domain-specific parameters into a latent space, learns their continuous evolution through Koopman dynamics, and decodes future parameters. **FreKoo (Yu et al., 2025)** separates a parameter trajectory into low- and high-frequency components and extrapolates their dynamics independently.

Scaling parameter-dynamics baselines. DRAIN, Koodos, and FreKoo were originally formulated over full-model parameter trajectories, which makes their temporal states and auxiliary dynamics models prohibitively large for pretrained backbones. We therefore freeze the backbone and use the concatenated LoRA parameters of each observed domain as the dynamic state. This sequence is processed by each method's original temporal mechanism, and the predicted state is reshaped into a future LoRA adapter and attached to the same frozen backbone for evaluation. We denote the resulting implementations as DRAIN-LoRA, Koodos-LoRA, and FreKoo-LoRA. Apart from the necessary input and output dimensions, their temporal mechanisms and rollout procedures are retained.

Temporal methods for pretrained models. **PromptTDG (Hosseini et al., 2025)** keeps the backbone frozen, learns domain-specific prompts and a temporal prompt generator, and extrapolates a prompt for each future domain. **TEA (Liu et al., 2025)** constructs temporally ordered domain experts, projects their updates into a low-dimensional coordinate space, and obtains the future model by extrapolating and weighting these experts. These baselines represent temporal adaptation mechanisms developed directly for pretrained models rather than full parameter trajectories.

Comparison protocol. All methods use identical observed/future domain splits, input preprocessing, pretrained backbones, and task heads. Whenever computationally feasible, LoRA-based methods use the same target modules, rank, and scaling factor as LoTiC; only the method-specific temporal component and objective differ. Future-domain samples and labels are never used for training.

A.3 Implementation and Reproducibility

Backbones and prediction heads. We use the public DistilBERT, Qwen3–0.6B, LLaMA-3–8B, and DINOv2-B/L checkpoints. Their backbone parameters remain frozen. For Rotating 2-Moons, a trainable linear layer maps each coordinate to an input embedding. DistilBERT uses its first hidden state and decoder-only backbones use the final hidden state. DINOv2 uses the class token for Rotated MNIST and Yearbook. The text classifiers use the corresponding backbone readout, while generation uses the native causal language-modeling head. Prediction heads are identical across methods using the same backbone.

Dataset-specific configurations. Hyperparameters are selected using validation data from the observed domains. Dataset- and backbone-specific settings are shared across methods, while method-specific configurations follow the original implementations. Across backbone scales, we retain the same adapter and temporal parameterization, adjusting only numerical settings such as the learning rate, precision, batch size, and LoRA scaling factor when needed for stable training or memory. Table 5 gives the shared adapter and base training configurations.

Method-specific configurations. Additional components are configured as follows.

- **LoTiC.** Controlled benchmarks use $F_t = \exp(tW)$ with an independent generator for each adapted projection. We initialize $A \sim \mathcal{N}(0, 0.01^2)$ and set B and W to zero.
- **DRAIN-LoRA.** We use the original LSTM generator and its official initialization on Rotating 2-Moons. The decoder output initialization is scaled by 0.1 elsewhere.
- **Koodos-LoRA and FreKoo-LoRA.** Both use encoder widths $1024/512/128/32$ followed by linear latent dynamics. Koodos uses loss weights $(1, 100, 100)$ and segments of length 10. FreKoo uses threshold 0.9 and weights $(10, 1, 1)$. Adapter and encoder parameters use learning rate 10^{-3}.

Table 5: Dataset-specific configurations. BF16/FP32 denotes BF16 backbone computation with FP32 trainable modules.

Benchmark	Backbone	Targets	r	α	Precision	LR	Epochs	Batch
Rotating 2-Moons	DistilBERT	$q/k/v$	8	64	FP32	10^{-3}	200	200
	Qwen3–0.6B	$q/k/v$	8	64	FP32	10^{-3}	500	200
	LLaMA-3–8B	$q/k/v$	8	128	BF16/FP32	10^{-3}	500	200
Rotated MNIST	DINOv2-B/L	$q/k/v$	8	64	FP32	10^{-3}	300	200
Yearbook	DINOv2-B/L	$q/k/v/o + \mathrm{FFN}$	8	32	FP32	3×10^{-4}	100	128
HuffPost	DistilBERT	$q/k/v/o + \mathrm{FFN}$	8	64	FP32	5×10^{-4}	10	128
	Qwen3–0.6B	$q/k/v/o + \mathrm{FFN}$	8	64	FP32	5×10^{-4}	10	128
	LLaMA-3–8B	$q/k/v/o + \mathrm{FFN}$	8	64	BF16/FP32	5×10^{-5}	10	128
arXiv	DistilBERT	$q/k/v/o + \mathrm{FFN}$	8	64	FP32	5×10^{-4}	5	128
	Qwen3–0.6B	$q/k/v/o + \mathrm{FFN}$	8	64	FP32	5×10^{-4}	5	128
	LLaMA-3–8B	$q/k/v/o + \mathrm{FFN}$	8	64	BF16/FP32	5×10^{-5}	5	128
WMT News Crawl	Qwen3–0.6B	$q/k/v/o + \mathrm{FFN}$	8	64	FP32	5×10^{-4}	2	16
	LLaMA-3–8B	$q/k/v/o + \mathrm{FFN}$	8	64	BF16/FP32	5×10^{-5}	2	16

- **PromptTDG.** Stage 0 trains ERM-LoRA using the corresponding table entry. We then freeze it and learn one prompt token followed by a single-layer, single-head temporal generator.
- **TEA.** We first train ERM-LoRA, and then fine-tune domains in reverse order at one tenth of the base learning rate. We use $c = 0.1$, $\epsilon = 10^{-3}$, five snapshots, ten Gram-PCA components, ARIMA$(1, 1, 1)$ extrapolation, and sharpening factor 5.

Optimization. LoTiC and the LoRA controls use AdamW and a OneCycle schedule. Controlled experiments use weight decay 10^{-4}, gradient clipping at 1.0, and a warm-up fraction of 0.3. Real-world experiments use zero weight decay, clipping at 0.5, and a warm-up fraction of 0.1. We report the final checkpoint and never select an epoch using future-domain performance.

Evaluation and metrics. We report error rate (%, lower is better) on Rotating 2-Moons, Rotated MNIST, Yearbook, HuffPost, and arXiv, and ROUGE-L F1 (higher is better) on WMT News Crawl. Each experiment uses three random seeds. We first compute the metric for every unseen domain and then average domains with equal weight. The tables report the mean and standard deviation.

Software and hardware. We use Python 3.11.16, PyTorch 2.5.1 with CUDA 12.4, Transformers 5.2.0, and NumPy 2.0.1. Experiments are distributed across two servers with seven NVIDIA RTX 6000 Ada GPUs (48 GB each) in total. The primary server has four such GPUs, two Xeon Gold 6348 processors with 56 physical cores in total, and 1024 GB of memory.

A.4 Sensitivity Analysis

We test whether LoTiC depends on narrow hyperparameter tuning by varying the LoRA rank and learning rate on Rotating 2-Moons with DistilBERT. The sweeps cover a 128-fold range of ranks and a 100-fold range of learning rates. Figure 4(a,b) reports future-domain error, with the strongest baseline shown as a common reference.

LoRA rank. LoTiC remains near 1% error as the rank varies from 4 to 512. This 128-fold range covers both highly constrained and overparameterized adapters, yet produces little change relative to the gap from the strongest baseline. The gain is therefore not tied to a narrow choice of rank.

Learning rate. LoTiC remains below the strongest baseline from 3×10^{-5} to 10^{-3}. Smaller learning rates may require longer training horizons, and the degradation at 10^{-5} suggests a risk of underfitting when the update is too small.

A.5 Convergence Analysis

We record observed- and future-domain errors throughout training under the default configuration. Figure 4(c) shows consistent convergence across three seeds. Both errors decrease and stabilize near 1% in later training, while the variation across seeds also narrows. The final performance is therefore sustained over the converged portion of training rather than produced by an isolated checkpoint.

Future-domain curves are used only for retrospective analysis, and all reported results use the final checkpoint.

A.6 Complexity Analysis

We analyze the cost introduced by temporal modeling and omit the frozen backbone and task head, which are shared by all methods. Let M be the number of adapted weight matrices, let the m-th matrix have shape $d_m \times k_m$, and let r denote the LoRA rank. We use T for the number of observed domains.

Adapter and core parameters. The number of parameters in one shared LoRA adapter is

$$P_{\mathrm{LoRA}} = r \sum_{m=1}^{M} (d_m + k_m). \tag{14}$$

For the default flow-based core, LoTiC learns one generator $W_m \in \mathbb{R}^{r\times r}$ for each adapted matrix. Its total parameter count is therefore

$$P_{\mathrm{LoTiC}} = P_{\mathrm{LoRA}} + Mr^2, \qquad \frac{P_{\mathrm{LoTiC}} - P_{\mathrm{LoRA}}}{P_{\mathrm{LoRA}}} = \frac{Mr}{\sum_m (d_m + k_m)}. \tag{15}$$

For square projections of width d, the relative overhead is $r/(2d)$. Increasing depth makes both terms grow linearly with M, while increasing width reduces the relative core overhead. This explains why the measured overhead remains close to LoRA and decreases on wider backbones in Table 3.

Cost of parameter dynamics. Let $p = P_{\mathrm{LoRA}}$, h denote the hidden width of an auxiliary dynamics model, and z its latent dimension. DRAIN-LoRA, Koodos-LoRA, and FreKoo-LoRA all evolve a state in $\mathbb{R}^p$. Their input or output mappings must therefore touch the complete adapter state. LoTiC instead evolves M independent $r \times r$ cores. Table 6 summarizes the resulting leading-order costs.

Table 6: Asymptotic cost of temporal modeling.

Method	Evolved state	Auxiliary parameters	Training trajectory
DRAIN–LoRA	p	$\mathcal{O}(ph + h^2)$	$\mathcal{O}(Tp)$
Koodos–LoRA	p	$\mathcal{O}(ph + z^2)$	$\mathcal{O}(Tp)$
FreKoo–LoRA	p	$\mathcal{O}(ph + z^2)$	$\mathcal{O}(Tp)$
LoTiC (Flow)	r^2 per matrix	Mr^2	$\mathcal{O}(p + Mr^2)$

Thus, applying prior TDG methods to LoRA greatly reduces their state dimension relative to full-model dynamics, but their auxiliary models still scale with the complete adapter. LoTiC changes the modeled object itself. Its temporal component depends only on the low-rank core and is independent of the ambient dimensions d_m and k_m. Recurrent and Time-to-Core variants replace Mr^2 by the parameter count of their lightweight core generator, without returning to the p-dimensional adapter trajectory.

B Theoretical Analysis and Proofs

B.1 Stability and Justification

Assumption 1 (Differentiability and GD dynamics). *For each $t \in [T]$, the objective $\mathcal{J}_t : \mathbb{R}^{d\times r} \times \mathbb{R}^{r\times k} \to \mathbb{R}$ is differentiable. The LoRA factors (B_t, A_t) are updated by gradient descent*

$$B_{t+1} = B_t - \eta \nabla_B \mathcal{J}_t(B_t, A_t), \qquad A_{t+1} = A_t - \eta \nabla_A \mathcal{J}_t(B_t, A_t), \tag{16}$$

with step size $\eta > 0$.

Assumption 1 specifies the optimization dynamics used to analyze the LoRA factors. At each training step t, the objective $\mathcal{J}_t$ may change with the data batch, task, or time index, but it is assumed to be differentiable with respect to both factors B_t and A_t. This allows the updates of B_t and A_t to be written as standard gradient descent steps with a shared step size η. The assumption does not impose convexity or a specific loss form; it only fixes the first-order training rule needed for the later subspace stability analysis.

Definition 3 (Reference shared subspaces). *Let $B := B_1$ and $A := A_1$. Define the orthogonal projectors*

$$P_B := \mathrm{Proj}_{\mathrm{col}(B)} \in \mathbb{R}^{d\times d}, \qquad P_A := \mathrm{Proj}_{\mathrm{col}(A^\top)} \in \mathbb{R}^{k\times k}. \tag{17}$$

Equivalently, $\mathrm{Im}(P_A) = \mathrm{row}(A) \subset \mathbb{R}^k$.

Assumption 2 (Dissipativity of out-of-subspace components). *There exist constants $\alpha_B, \alpha_A \geq 0$ and $\varepsilon_B, \varepsilon_A \geq 0$ such that for all $t \in [T]$,*

$$\big\langle (I - P_B)B_t, (I - P_B)\nabla_B \mathcal{J}_t(B_t, A_t) \big\rangle \geq \alpha_B \|(I - P_B)B_t\|_F^2 - \varepsilon_B, \tag{18}$$

$$\big\langle A_t(I - P_A), \nabla_A \mathcal{J}_t(B_t, A_t)(I - P_A) \big\rangle \geq \alpha_A \|A_t(I - P_A)\|_F^2 - \varepsilon_A, \tag{19}$$

where $\langle X, Y\rangle := \mathrm{tr}(X^\top Y)$.

Assumption 2 requires that the optimization dynamics do not amplify the components of B_t and A_t outside the selected subspaces. Instead, the gradient acts as a restoring force on $(I - P_B)B_t$ and $A_t(I - P_A)$, shrinking them whenever they become large, up to bounded slack terms $\varepsilon_B, \varepsilon_A$. Therefore, the evolution of the in-subspace components and the out-of-subspace components remains approximately decoupled, preserving the intended independence of the structured part from its complement throughout training.

Assumption 3 (Projected gradient growth outside the reference subspaces). *There exist constants $\beta_B, \beta_A \geq 0$ and $\delta_B, \delta_A \geq 0$ such that for all $t \in [T]$,*

$$\|(I - P_B)\nabla_B \mathcal{J}_t(B_t, A_t)\|_F^2 \leq \beta_B \|(I - P_B)B_t\|_F^2 + \delta_B, \tag{20}$$

$$\|\nabla_A \mathcal{J}_t(B_t, A_t)(I - P_A)\|_F^2 \leq \beta_A \|A_t(I - P_A)\|_F^2 + \delta_A. \tag{21}$$

Assumption 3 states that the gradient energy outside the reference subspaces is controlled by the current out-of-subspace energy, up to a bounded residual term. Together with Assumption 2, this yields a stable discrete-time recursion under gradient descent: the out-of-subspace components are not only discouraged by the gradient direction, but their one-step fluctuation is also quantitatively bounded.

Theorem 4 (Stability of (B_t, A_t) subspaces under GD (formal)). *Under Assumptions 1–2 and Assumption 3, define*

$$e_t^B := \|(I - P_B)B_t\|_F, \qquad e_t^A := \|A_t(I - P_A)\|_F, \qquad \Delta W_t := B_t A_t. \tag{22}$$

Suppose the step size $\eta > 0$ satisfies

$$0 \leq \kappa_B := 1 - 2\eta\alpha_B + \eta^2\beta_B < 1, \qquad 0 \leq \kappa_A := 1 - 2\eta\alpha_A + \eta^2\beta_A < 1. \tag{23}$$

Then for all $t \in [T]$,

$$(e_{t+1}^B)^2 \leq \kappa_B (e_t^B)^2 + 2\eta\varepsilon_B + \eta^2\delta_B, \tag{24}$$

$$(e_{t+1}^A)^2 \leq \kappa_A (e_t^A)^2 + 2\eta\varepsilon_A + \eta^2\delta_A. \tag{25}$$

Consequently, for all $t \in [T]$,

$$(e_t^B)^2 \le \kappa_B^{t-1}(e_1^B)^2 + \frac{2\eta\varepsilon_B + \eta^2\delta_B}{1-\kappa_B}\big(1-\kappa_B^{t-1}\big), \tag{26}$$

$$(e_t^A)^2 \le \kappa_A^{t-1}(e_1^A)^2 + \frac{2\eta\varepsilon_A + \eta^2\delta_A}{1-\kappa_A}\big(1-\kappa_A^{t-1}\big). \tag{27}$$

Moreover, the induced update leakage satisfies

$$\|(I-P_B)\Delta W_t\|_F \le e_t^B\|A_t\|_2, \qquad \|\Delta W_t(I-P_A)\|_F \le \|B_t\|_2 e_t^A. \tag{28}$$

The detailed proof is provided in Section B.2.

B.2 Proof of Theorem 4

Proof. We prove the B-part; the A-part follows analogously.

Let $R_B := I - P_B$. By Assumption 3, P_B is an orthogonal projector, hence R_B is also an orthogonal projector:

$$R_B^\top = R_B, \qquad R_B^2 = R_B. \tag{29}$$

By Assumption 1,

$$B_{t+1} = B_t - \eta\nabla_B\mathcal{J}_t(B_t, A_t). \tag{30}$$

Left-multiplying by R_B, we obtain

$$R_B B_{t+1} = R_B B_t - \eta R_B\nabla_B\mathcal{J}_t(B_t, A_t). \tag{31}$$

Taking squared Frobenius norms and expanding,

$$\|R_B B_{t+1}\|_F^2 = \|R_B B_t\|_F^2 - 2\eta\big\langle R_B B_t, R_B\nabla_B\mathcal{J}_t(B_t, A_t)\big\rangle + \eta^2\|R_B\nabla_B\mathcal{J}_t(B_t, A_t)\|_F^2. \tag{32}$$

By Assumption 2, specifically (18),

$$\big\langle R_B B_t, R_B\nabla_B\mathcal{J}_t(B_t, A_t)\big\rangle \ge \alpha_B\|R_B B_t\|_F^2 - \varepsilon_B. \tag{33}$$

Substituting this into (32) yields

$$\begin{aligned}\|R_B B_{t+1}\|_F^2 &\le \|R_B B_t\|_F^2 - 2\eta\big(\alpha_B\|R_B B_t\|_F^2 - \varepsilon_B\big) + \eta^2\|R_B\nabla_B\mathcal{J}_t(B_t, A_t)\|_F^2 \\ &= (1-2\eta\alpha_B)\|R_B B_t\|_F^2 + 2\eta\varepsilon_B + \eta^2\|R_B\nabla_B\mathcal{J}_t(B_t, A_t)\|_F^2.\end{aligned} \tag{34}$$

Next, by Assumption 3, specifically (20),

$$\|R_B\nabla_B\mathcal{J}_t(B_t, A_t)\|_F^2 \le \beta_B\|R_B B_t\|_F^2 + \delta_B. \tag{35}$$

Substituting this bound into (34), we obtain

$$\begin{aligned}\|R_B B_{t+1}\|_F^2 &\le (1-2\eta\alpha_B)\|R_B B_t\|_F^2 + 2\eta\varepsilon_B + \eta^2\big(\beta_B\|R_B B_t\|_F^2 + \delta_B\big) \\ &= \big(1-2\eta\alpha_B + \eta^2\beta_B\big)\|R_B B_t\|_F^2 + 2\eta\varepsilon_B + \eta^2\delta_B.\end{aligned} \tag{36}$$

By the definition of κ_B in (23),

$$\|R_B B_{t+1}\|_F^2 \le \kappa_B\|R_B B_t\|_F^2 + 2\eta\varepsilon_B + \eta^2\delta_B. \tag{37}$$

Since $e_t^B = \|R_B B_t\|_F$, this proves

$$(e_{t+1}^B)^2 \le \kappa_B(e_t^B)^2 + 2\eta\varepsilon_B + \eta^2\delta_B, \tag{38}$$

which is (24).

Now define $x_t := (e_t^B)^2$ and $c_B := 2\eta\varepsilon_B + \eta^2\delta_B$. Then

$$x_{t+1} \le \kappa_B x_t + c_B. \tag{39}$$

Iterating the recursion gives

$$\begin{aligned}x_t &\le \kappa_B^{t-1}x_1 + c_B\sum_{j=0}^{t-2}\kappa_B^j \\ &= \kappa_B^{t-1}x_1 + \frac{c_B}{1-\kappa_B}\big(1-\kappa_B^{t-1}\big),\end{aligned} \tag{40}$$

because $\kappa_B < 1$. Therefore,

$$(e_t^B)^2 \leq \kappa_B^{t-1}(e_1^B)^2 + \frac{2\eta\varepsilon_B + \eta^2\delta_B}{1-\kappa_B}\big(1-\kappa_B^{t-1}\big), \tag{41}$$

which proves (26).

The proof of (25) and (27) is identical with

$$R_A := I - P_A \tag{42}$$

acting on the right, using (19) and (21).

Finally, since

$$(I-P_B)\Delta W_t = (I-P_B)B_tA_t, \tag{43}$$

the submultiplicativity of the Frobenius and spectral norms gives

$$\|(I-P_B)\Delta W_t\|_F \leq \|(I-P_B)B_t\|_F\|A_t\|_2 = e_t^B\|A_t\|_2. \tag{44}$$

Similarly,

$$\Delta W_t(I-P_A) = B_tA_t(I-P_A), \tag{45}$$

hence

$$\|\Delta W_t(I-P_A)\|_F \leq \|B_t\|_2\|A_t(I-P_A)\|_F = \|B_t\|_2 e_t^A. \tag{46}$$

This proves (28). □