VOLTERRA FLOW MATCHING WITH MEMORY FOR FEW-SHOT CLASSIFICATION

Anonymous authors
Paper under double-blind review

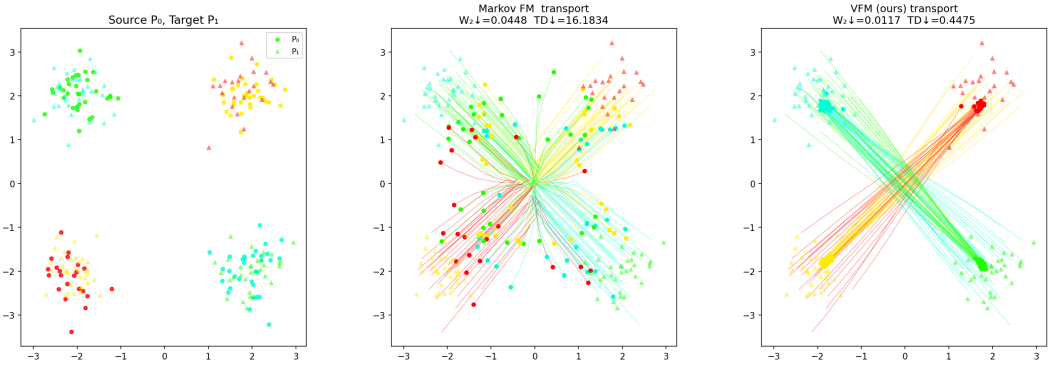


Figure 1: **Motivation: Memoryless vs. Memory Based transport under ambiguity.** An XOR transport task in which P_0 (circles) must be mapped to P_1 (triangles) by swapping diagonally opposite clusters, a *nonlinear assignment* problem where trajectories intersect and intermediate states become ambiguous. *Left:* source and target with ground-truth correspondences (color-coded). *Middle:* Memoryless flow matching collapses trajectories toward the center and disperses them inconsistently, achieving poor correspondence despite partial distribution alignment. *Right:* VFM routes each sample to its correct target cluster by conditioning on trajectory history, motivating our memory aware formulation.
Metric notation: TD : *Transport Distance*; W_2 : *Wasserstein 2 distance*.

ABSTRACT

While pre-trained vision-language models provide a coarse alignment between image and text embeddings, they often require parameter-efficient fine-tuning (PEFT) to adapt to downstream tasks like few-shot learning. Existing PEFT methods, including prompt tuning, LoRA, and adapter-based approaches, typically perform one-step updates, while recent multi-step methods model refinement via learned velocity fields. However, these approaches typically condition the dynamics on the current state, leaving the potential predictive role of trajectory history underexplored. We hypothesize that, especially in low-data regimes, the instantaneous feature state can be insufficient for predicting the appropriate alignment dynamics. To capture this structure, we propose a *Volterra Flow Matching (VFM)* framework that introduces a memory variable to encode the history of feature evolution. This memory is constructed via a Volterra integral with an exponentially decaying kernel, leading to a coupled dynamical system that governs both the current state and its accumulated history. Practically, we construct interpolation trajectories between image and text features, and train a lightweight neural operator to predict the drift conditioned on both the instantaneous features and their memory state. During inference, the learned dynamics are integrated using a small number of steps, enabling efficient multi-step refinement. Extensive experiments on standard few-shot benchmarks demonstrate modest yet consistent gains over existing PEFT and Flow Matching Alignment methods, while controlled prediction-risk probes show that the memory state contains additional velocity-predictive information.

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.

1 INTRODUCTION

Aligning information across modalities (e.g., text and images) is a fundamental problem in multimodal learning, underpinning applications ranging from reasoning in multimodal large language models (MLLMs)(Hurst et al., 2024; Li et al., 2022; Liu et al., 2023) to high-fidelity generation in Artificial Intelligence Generated Content (AIGC) systems(Rombach et al., 2022; Esser et al., 2024). For few-shot learning, a common approach is to learn a shared embedding space where semantically corresponding image and text features are closely aligned. Large-scale vision-language models (VLMs), such as CLIP (Radford et al., 2021) and ALIGN (Jia et al., 2021), achieve this via contrastive pretraining, enabling strong zero-shot generalization across tasks. Despite their success in few-shot learning, pre-trained VLMs often exhibit imperfect alignment due to the inherent complexity and heterogeneity of multimodal data. Consequently, adapting these models to downstream tasks typically requires PEFT. Existing PEFT methods can be broadly categorized into three groups: **prompt tuning**(Li & Liang, 2021; Zhou et al., 2022b;a), which modifies text representations; **adapter-based methods**(Houlsby et al., 2019; Gao et al., 2024), which refine image features; and **LoRA-based approaches** (Hu et al., 2022; Zanella & Ben Ayed, 2024), which jointly adapt both modalities. These methods can be interpreted as applying a *single-step transformation* that shifts features toward a better-aligned distribution. While effective in simple settings, such one-step approaches struggle in challenging scenarios where cross-modal distributions are highly entangled. Recent work addresses this limitation by introducing *multi-step refinement* via learned velocity fields based on flow matching (Liu et al., 2022; Lipman et al., 2022; Albergo & Vanden-Eijnden, 2022). By decomposing complex transformations into a sequence of incremental updates, these methods make alignment more tractable and achieve improved performance in few-shot settings. In particular, Flow Matching Alignment Jiang et al. (2026) (FMA) demonstrates that transporting image features toward text features through iterative updates can significantly enhance alignment.

However, a key limitation remains: **existing multi-step methods are inherently Memoryless**. That is, their learned dynamics assume that each update depends only on the current state, ignoring trajectory history. Formally, the velocity field is modeled as $v(x_t, t)$. While this assumption simplifies modeling, it may be restrictive when intermediate states are ambiguous: different endpoint assignments can produce similar x_t while requiring different transport directions. In low-data regimes, such ambiguity can make trajectories noisy or unstable, and errors accumulated earlier in the path cannot be represented explicitly by a memoryless state. We therefore hypothesize that cross-modal alignment can benefit from *history-dependent dynamics*, rather than claiming that the underlying process cannot be represented as a Markov system after state augmentation. To capture this structure, we propose VFM, which augments the state with a memory variable accumulated via a Volterra integral with an exponentially decaying kernel. The resulting velocity field depends on both the current state and its compact history summary, $v(x_t, h_t, t)$.

Our formulation can be interpreted as a path-space regularized transport problem with memory, extending stochastic-interpolant flow matching to memory aware settings. Compared to standard flow matching, which learns memoryless dynamics, VFM imposes a structured inductive bias through causal history accumulation. This not only increases expressivity but also improves stability by allowing trajectories to implicitly encode past context. Practically, we construct stochastic interpolation trajectories between image and text features using a variance-vanishing (Brownian-bridge type) interpolant with shared noise. We then train a lightweight neural operator to predict the drift conditioned on both instantaneous features and their memory state. During inference, the learned dynamics are integrated over a small number of steps, enabling efficient multi-step refinement without significant computational overhead. We evaluate our approach on standard few-shot benchmarks and observe consistent improvements over existing PEFT and multi-step methods, particularly on challenging datasets with complex cross-modal distributions. These results highlight the importance of history-aware dynamics and demonstrate that incorporating memory structure provides a principled and effective direction for cross-modal adaptation.

In summary, our contributions are:

- We identify a limitation of existing PEFT and multi-step alignment methods: their reliance on Memoryless dynamics when the instantaneous state is ambiguous, supported by synthetic experiments.

- We propose Volterra Flow Matching (VFM), a novel framework that introduces history-dependent (non-Markovian, marginally) transport via a Volterra memory mechanism, coupled with an efficient training and inference procedure based on stochastic interpolation and history-aware velocity prediction.

- Extensive experiments demonstrate that VFM achieves modest yet superior and more robust alignment.

2 Related Work

Diffusion Models and Flow Matching. Diffusion models (DMs) have become a dominant paradigm in generative modeling (Ho et al., 2020; Song et al., 2020a), where data generation is framed as a progressive denoising process. Score-based methods further interpret diffusion as stochastic differential equations (SDEs) and their associated probability flow ODEs (Song et al., 2020b). Building on score-based SDEs , Flow Matching (Liu et al., 2022; Lipman et al., 2022; Albergo & Vanden-Eijnden, 2022) directly learns a velocity field to transport samples between distributions. Flow matching has demonstrated strong performance across applications including text-to-image generation (Esser et al., 2024; Geng et al., 2025; Albergo et al., 2023), image editing (Kim et al., 2025; Kulikov et al., 2024; Rout et al., 2024), and video generation (Jin et al., 2024; Polyak et al., 2024). While recent works explore transport between structured domains such as text and image distributions (He et al., 2025; Liu et al., 2025), existing flow-matching-based approaches are inherently Memoryless, modeling dynamics that depend only on the current state.

Non-Markovian Dynamics and Volterra Memory. Most generative and transport-based frameworks assume Markovian dynamics, where system evolution depends only on the current state (Song et al., 2020b; Liu et al., 2022; Lipman et al., 2022). However, many real-world processes exhibit path dependence, motivating non-Markovian formulations that incorporate historical information. Classical approaches, such as generalized Langevin equations and Volterra integral equations, model memory effects via kernel-based accumulation of past states (Mori, 1965; Kubo, 1966; Volterra, 1930; Gripenberg et al., 1990). In machine learning, continuous-time models such as Neural ODEs (Chen et al., 2018) and neural controlled differential equations (Kidger et al., 2020) extend this idea to learn trajectory-dependent dynamics. Stochastic-interpolant constructions (Albergo et al., 2023) provide a flexible framework for interpolating between distributions, but are typically restricted to Markovian settings. CaLMFlow He et al. (2024), learn complex flows by formulating flow matching as a sequence modeling task.

Few-Shot Learning and PEFT in Vision-Language Models. Few-shot learning (FSL) aims to adapt models from limited labeled data (Wang et al., 2020; Yue et al., 2020), often by fine-tuning pre-trained vision-language models such as CLIP (Radford et al., 2021) and ALIGN (Jia et al., 2021). To reduce computational cost, PEFT methods have been widely adopted. Prompt tuning approaches (Zhou et al., 2022a;b; Lee et al., 2023; Bulat & Tzimiropoulos, 2023) learn task-specific prompts that modify textual or visual representations, while LoRA-based methods (Zanella & Ben Ayed, 2024; da Costa et al., 2023; Kim et al., 2024; He et al., 2022) introduce low-rank trainable parameters within model layers. Adapter-based approaches (Zhang et al., 2022; Gao et al., 2024) append lightweight modules to frozen encoders for efficient adaptation. These methods can be interpreted as performing one-step feature transformations, which limits their ability to model complex, highly entangled cross-modal distributions. Recent flow-based approaches introduce multi-step refinement to address this limitation, but still rely on memoryless updates.

3 Methodology

We formalize the cross-modal alignment problem and introduce *Volterra Flow Matching (VFM)* framework. We test the path-dependent hypothesis for cross-modal alignment through synthetic transport examples where the Memoryless baseline collapses identity-preserving correspondences under ambiguity (Figures 1 and 3 and Table 5), and real-data ablations where the gain from adding memory is largest in the 1-shot setting and shrinks as supervision grows (Table 2).

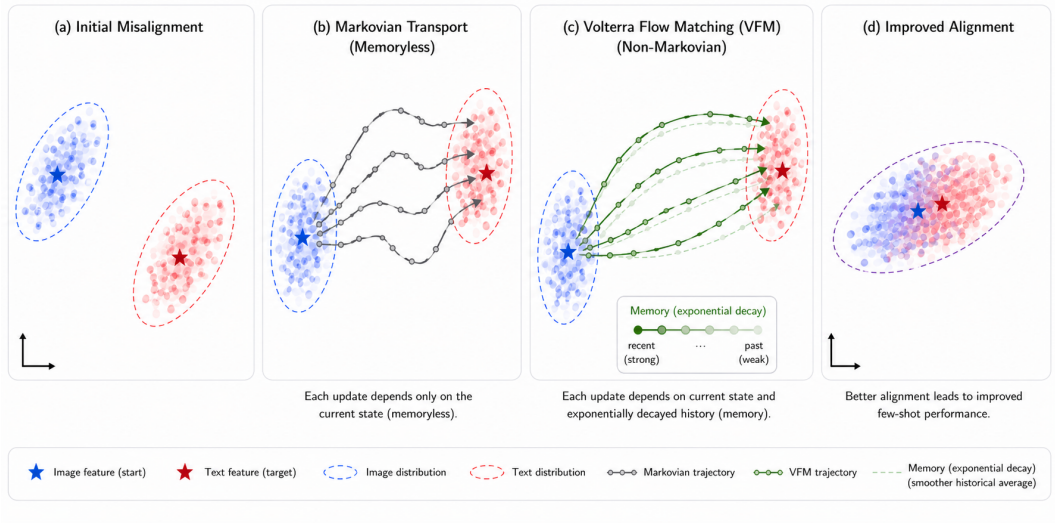


Figure 2: **Volterra Flow Matching for History Aware cross-modal alignment.** (a) Image and text features are initially misaligned in the shared embedding space (from pretrained feature extractors). (b) Existing flow-based methods model *Memoryless* transport, where trajectories depend only on the current state, often leading to unstable and inconsistent paths. (c) Our approach introduces a *Volterra memory* that aggregates past states with an exponentially decaying kernel, resulting in smooth, history-aware (non-Markovian in marginal) trajectories that remain close to the underlying path while correcting earlier deviations. (d) The resulting features are better aligned, forming coherent elliptical clusters and improving downstream few-shot classification performance.

3.1 Problem Setup

Given an image I and its label c, we extract corresponding features using a pre-trained vision-language model (e.g., CLIP (Radford et al., 2021)):

$$x_0 = f_{\text{img}}(I), \qquad x_1 = f_{\text{text}}(c), \tag{1}$$

where $x_0, x_1 \in \mathbb{R}^d$, f_{img} and f_{text} denote the frozen image and text encoders of the pre-trained vision–language model. The goal of cross-modal alignment is to learn a transformation that maps x_0 toward x_1 such that aligned features are closer in the shared embedding space. Instead of directly learning a one-step mapping, we consider a continuous-time trajectory $\{x_t\}_{t\in[0,1]}$ that transports x_0 to x_1. Existing flow-based approaches Jiang et al. (2026) model this via a Memoryless velocity field:

$$\frac{dx_t}{dt} = v_\theta(x_t, t), \tag{2}$$

where v_θ depends only on the current state x_t. We instead introduce a *history-dependent* formulation in which the dynamics depend on a compact summary of the trajectory history.

3.2 Stochastic Interpolation Between Modalities

To construct training trajectories, we adopt a variance-vanishing (Brownian-bridge type) stochastic interpolant between x_0 and x_1 (Albergo et al., 2023). Specifically, for $t \in (0, 1)$, we define:

$$x_t = (1-t)x_0 + tx_1 + \gamma\sqrt{t(1-t)}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{3}$$

where γ controls the noise scale. This interpolation defines a family of intermediate distributions connecting the source and target features. The corresponding ground-truth velocity field can be represented by derivative of x_t with respect to t (see Proposition 2 in Section B.3):

$$v^*(x_t, t) = x_1 - x_0 + \gamma\frac{1-2t}{2\sqrt{t(1-t)}}\,\epsilon. \tag{4}$$

3.3 History Aware Dynamics via Volterra Memory

To capture path dependence in complex alignment scenarios, especially under limited data where the instantaneous state can be ambiguous, we introduce a history-dependent representation through a memory variable h_t, defined as a Volterra integral over past states:

$$h_t = \int_0^t K(t-s)\, x_s\, ds, \tag{5}$$

where $K(\cdot)$ is a kernel that determines how past information is aggregated. This formulation provides a continuous-time mechanism for encoding trajectory history, allowing the model to retain information about the entire path traversed up to time t. We adopt an exponentially decaying kernel, with parameter β:

$$K(t-s) = e^{-\beta(t-s)}, \tag{6}$$

which assigns higher weight to recent states while gradually forgetting distant history. This choice is both theoretically grounded and computationally efficient, as it leads to the following equivalent differential equation (equivalent in the sense of Proposition 1):

$$\frac{dh_t}{dt} = -\beta h_t + x_t, \quad h_0 = 0. \tag{7}$$

By Proposition 1, this ODE form is equivalent to the Volterra integral, allowing h_t to be updated recursively at $\mathcal{O}(1)$ per-step cost without storing the trajectory. Although the marginal process x_t is non-Markov, the augmented process (x_t, h_t) is Markov on $\mathbb{R}^d \times \mathbb{R}^d$ (Section B.2, Remark 1), so the system can be integrated with standard ODE solvers. Intuitively, h_t acts as a smoothed summary of past feature states, enabling richer and more stable dynamics than memoryless formulations.

3.4 Volterra Flow Matching

We model the transport dynamics using a history-aware velocity field:

$$\frac{dx_t}{dt} = v_\theta(x_t, h_t, t), \tag{8}$$

where v_θ is parameterized by a neural network conditioned on both current state and the memory.

Given samples (x_0, x_1) and noise ϵ, we simulate discrete trajectories $\{x_{t_k}\}_{k=1}^T$ and corresponding memory states $\{h_{t_k}\}_{k=1}^T$ using:

$$x_{t_k} = (1-t_k)x_0 + t_k x_1 + \gamma\sqrt{t_k(1-t_k)}\,\epsilon, \tag{9}$$

$$h_{t_k} = h_{t_{k-1}} + \Delta t\,(-\beta h_{t_{k-1}} + x_{t_k}), \tag{10}$$

where $\Delta t = t_k - t_{k-1}$. The model is trained to match the ground-truth velocity:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0,x_1,t,\epsilon}\left[\|v_\theta(x_t, h_t, t) - v^*(x_t, t)\|^2\right]. \tag{11}$$

This objective enables the model to learn history-dependent transport dynamics. Memory helps through *endpoint disambiguation*: the target is the displacement $x_1 - x_0$, yet a single state x_t need not determine which endpoints produced it, so a Memoryless predictor can be forced toward their conditional average. For the deterministic interpolant ($\gamma = 0$), the augmented state removes this ambiguity: the map $(x_0, x_1) \mapsto (x_t, h_t)$ is linear with determinant $-\Phi_\beta(t) \neq 0$ for every $t \in (0, 1]$, $\beta > 0$, hence invertible, and

$$x_1 - x_0 \;=\; \frac{1}{\Phi_\beta(t)}\left(\tfrac{1-e^{-\beta t}}{\beta}\, x_t - h_t\right) \;+\; \mathcal{O}(\gamma), \qquad \Phi_\beta(t) = \frac{1-(1+\beta t)e^{-\beta t}}{\beta^2}. \tag{12}$$

At $\gamma = 0$, the excess risk of the optimal Memoryless predictor is exactly $\mathbb{E}[\mathrm{Var}(x_1 - x_0 \mid x_t, t)]$, which is strictly positive when endpoint assignment is ambiguous given the current state alone (Proposition 3, Section B.4). For the practical small-noise setting, the endpoint recovery relation incurs an explicit $\mathcal{O}(\gamma)$ perturbation; our held-out ridge probe in Table 17 directly tests whether the resulting memory state still provides incremental prediction of the velocity target. Thus, the theory motivates a testable history-information hypothesis rather than assuming that history-dependence is established by construction.

3.5 Inference via Memory Based Trajectory Integration

At inference time, our goal is to refine an input image feature x_0 into a representation that is better aligned with the text feature space. Unlike training, where trajectories are sampled using stochastic interpolation, inference proceeds deterministically by integrating the learned dynamics. Given an initial feature x_0, we initialize the memory state as $h_0 = 0$ and evolve the system according to the learned history-aware dynamics:

$$x_{t+\Delta t} = x_t + \Delta t\, v_\theta(x_t, h_t, t), \tag{13}$$

$$h_{t+\Delta t} = h_t + \Delta t\, (-\beta h_t + x_{t+\Delta t}), \tag{14}$$

where Δt is a fixed step size. The first equation performs a forward Euler integration of the learned velocity field, while the second updates the Volterra memory using the same discretization. The memory update is the explicit-Euler discretization of the memory ODE, whose homogeneous amplification factor is $|1-\beta\Delta t|$; our default $(\beta, \Delta t)$=(1, 0.1) therefore keeps the recursion contractive. Notably, the memory update uses the *updated* state $x_{t+\Delta t}$, ensuring that the memory reflects the latest trajectory evolution. In practice, 3–6 steps suffice for strong alignment with minimal overhead (section A.4). This is consistent with the intuition that multi-step refinement decomposes a complex transformation into a sequence of simpler updates.

Classification. After T steps, we normalize $\hat{x}_T$ and compute similarities with class embeddings $\{c_j\}_{j=1}^{C}$ as $\text{sim}(j) = \langle \hat{x}_T, c_j \rangle$, followed by softmax. The predicted label is the argmax.

4 Experiments

Datasets and Evaluation Protocol. We evaluate our proposed VFM framework on 11 widely-used few-shot classification benchmarks: Aircraft (Maji et al., 2013), Describable Textures (DTD)(Cimpoi et al., 2014), EuroSAT(Helber et al., 2019), SUN397 (Xiao et al., 2010), StanfordCars (Krause et al., 2013), ImageNet (Deng et al., 2009), UCF101 (Soomro et al., 2012), Flowers102 (Nilsback & Zisserman, 2008), Food101 (Bossard et al., 2014), OxfordPets (Parkhi et al., 2012), and Caltech101 (Fei-Fei et al., 2004). Following standard practice (Radford et al., 2021), we divide the datasets into **difficult** (Aircraft, EuroSAT, DTD, SUN397, Cars) and **easy** (UCF101, ImageNet, Flowers, Food, Pets, Caltech101) groups based on the degree of cross-modal alignment difficulty. Throughout, we use shortened names in tables for compactness (e.g., Net = ImageNet, Pets = OxfordPets, Cars = StanfordCars, Flowers = Flowers102, SAT = EuroSAT).

We adopt the standard few-shot protocol, using $K \in \{1, 4, 16\}$ labeled examples per class for training, and evaluate on the full test set. Our method builds upon frozen vision–language encoders such as CLIP (Radford et al., 2021), combined with parameter-efficient adaptation. In particular, we extend CLIP-LoRA (Zanella & Ben Ayed, 2024) by introducing VFM-based transport dynamics; architectural details are in A.13. Unless otherwise specified, all models are trained for 200 epochs.

Baselines. We compare VFM against strong PEFT methods, including prompt learning approaches such as CoOp (Zhou et al., 2022b) and CoCoOp (Zhou et al., 2022a), adapter-based methods such as TIP-Adapter (Zhang et al., 2022) and CLIP-Adapter (Gao et al., 2024), and low-rank adaptation via CLIP-LoRA (Zanella & Ben Ayed, 2024). We also compare against prior flow-based adaptation methods, highlighting the benefits of replacing standard flow formulations with VFM dynamics. We also compare with FMAJiang et al. (2026) in table 2 .

Overall Results. Table 1 presents the classification accuracy across all datasets and shot settings. VFM consistently improves over CLIP-LoRA and achieves competitive performance across a wide range of benchmarks. The improvements are particularly noticeable in low-shot settings and on datasets where cross-modal alignment is more challenging.

4.1 Ablation Study: Effect of Memory Across Shots

We analyze the contribution of memory across 1-shot, 4-shot, and 16-shot settings by comparing VFM with its Memoryless counterpart, Flow Matching Alignment (FMA) Jiang et al. (2026) under identical architectures, differing only in the inclusion of h_t. (We use the same feature extractor, CLIP-LoRA (2024)). VFM improves over FMA in every regime, most strongly in low data (Table 2).

Table 1: **Comparison with representative approaches.** Based on CLIP-LoRA, we add VFM to further improve the performance. The highest value of each dataset is bolded.

Shots	Method	Difficult						Easy						
		Aircraft	SAT	DTD	SUN	Cars	Avg	UCF	Net	Flowers	Food	Pets	Caltech	Avg
0	CLIP (2021)	24.8	47.8	43.8	62.5	65.5	48.9	66.7	66.7	67.4	85.3	89.1	92.9	78.0
1	CoOp (2022b)	20.8	56.4	50.1	67.0	67.5	52.4	71.2	65.7	78.3	84.3	90.2	92.5	80.4
	CoCoOp (2022a)	28.1	55.4	52.6	68.7	67.6	54.5	70.4	69.4	73.4	84.9	91.9	94.1	80.7
	TIP-Adapter (2022)	28.8	67.8	51.6	67.2	67.1	56.5	73.4	69.4	83.8	85.8	90.6	94.0	82.8
	CLIP-Adapter (2024)	25.2	49.3	44.2	65.4	65.7	50.0	66.9	67.9	71.3	86.1	89.0	92.0	78.9
	PLOT++ (2022)	28.6	65.4	54.6	66.8	68.8	56.8	74.3	66.5	80.5	86.2	91.9	94.3	82.3
	KgCoOp (2023)	26.8	61.9	52.7	68.4	66.7	55.3	72.8	68.9	74.7	**86.4**	92.1	94.2	81.5
	ProGrad (2023)	28.9	57.0	52.8	67.0	68.2	54.8	73.3	67.0	80.9	84.9	91.4	93.5	81.8
	CLIP-LoRA (2024)	28.0	71.9	54.1	70.3	69.4	58.7	75.4	70.3	81.4	85.1	91.9	93.8	83.0
	+VFM (Ours)	**29.5**	**82.1**	**56.7**	**70.9**	**71.0**	**62.1**+3.4	**76.4**	**70.5**	**88.3**	85.0	**93.2**	**94.3**	**84.6**+1.6
4	CoOp (2022b)	30.9	69.7	59.5	69.7	74.4	60.8	77.6	68.8	92.2	84.5	92.5	94.5	85.0
	CoCoOp (2022a)	30.6	61.7	55.7	70.4	69.5	57.6	75.3	70.6	81.5	86.3	92.7	94.8	83.5
	TIP-Adapter (2022)	35.7	76.8	59.8	70.8	74.1	63.4	78.1	70.7	92.1	86.5	91.9	94.8	85.7
	CLIP-Adapter (2024)	27.9	51.2	46.1	68.0	67.5	52.1	70.6	68.6	73.1	86.5	90.8	94.0	80.6
	PLOT++ (2022)	35.3	83.2	62.4	71.7	76.3	65.8	79.8	70.4	92.9	86.5	**92.7**	95.1	86.2
	KgCoOp (2023)	32.2	71.8	58.7	71.5	69.5	60.7	77.6	69.9	87.0	**86.9**	92.6	95.0	84.8
	ProGrad (2024)	34.1	69.6	59.7	71.7	75.0	62.0	77.9	70.2	91.1	85.4	92.1	94.4	85.2
	CLIP-LoRA (2024)	38.8	83.5	64.0	72.8	77.4	67.3	81.1	71.4	92.9	82.6	90.6	95.0	85.6
	+VFM (Ours)	**40.9**	**88.9**	**68.4**	**73.7**	**80.0**	**70.4**+3.1	**82.8**	**72.3**	**96.3**	83.9	91.6	**95.8**	**87.1**+1.5
16	CoOp (2022b)	43.3	86.0	70.0	74.9	83.1	71.5	83.1	71.4	97.2	84.4	91.1	95.5	87.1
	CoCoOp (2022a)	33.8	75.5	65.8	72.8	72.4	64.1	76.0	71.1	87.1	87.4	93.2	95.2	85.0
	TIP-Adapter (2022)	44.6	85.9	70.8	76.0	82.3	71.9	83.9	73.4	96.2	86.8	92.6	95.7	88.1
	CLIP-Adapter (2024)	34.2	71.4	59.4	74.2	74.0	62.6	80.2	69.8	92.9	87.1	92.3	94.9	86.2
	PLOT++ (2022)	46.7	92.0	71.4	76.0	84.6	74.1	85.3	72.6	97.6	87.1	**93.6**	96.0	88.7
	KgCoOp (2023)	36.5	76.2	68.7	73.3	74.8	65.9	81.7	70.4	93.4	**87.2**	93.2	95.2	86.9
	ProGrad (2024)	43.0	83.6	68.8	75.1	82.9	70.7	82.7	72.1	96.6	85.8	92.8	95.9	87.7
	CLIP-LoRA (2024)	54.7	90.7	73.0	76.0	86.0	76.1	86.2	73.4	97.9	84.2	91.6	96.1	88.2
	+VFM (Ours)	**57.7**	**92.9**	**77.0**	**77.3**	**88.2**	**78.6**+2.5	**87.6**	**74.3**	**99.1**	85.8	92.5	**96.4**	**89.3**+1.1

Table 2: **Ablative study of using memory (h_t) across shots: VFM vs FMA** Gains/losses from adding memory are shown in parentheses.

Method	Difficult						Easy						
	Aircraft	SAT	DTD	SUN	Cars	Avg	UCF	Net	Flowers	Food	Pets	Caltech	Avg
1-shot													
FMA	28.3	73.0	55.1	70.6	69.8	59.4	75.9	70.2	84.9	85.2	92.1	94.5	83.8
VFM	**29.5**+1.2	**82.1**+9.1	**56.7**+1.6	**70.9**+0.3	**71.0**+1.2	**62.1**+2.7	**76.4**+0.5	**70.5**+0.3	**88.3**+3.4	**85.0**-0.2	**93.2**+1.1	**94.3**-0.2	**84.6**+0.8
4-shot													
FMA	40.3	85.0	67.0	**73.7**	78.9	69.0	82.4	72.0	95.0	83.2	90.8	**95.8**	86.5
VFM	**40.9**+0.6	**88.9**+3.9	**68.4**+1.4	**73.7**+0.0	**80.0**+1.1	**70.4**+1.4	**82.8**+0.4	**72.3**+0.3	**96.3**+1.3	**83.9**+0.7	**91.6**+0.8	**95.8**+0.0	**87.1**+0.6
16-shot													
FMA	**57.8**	91.0	75.4	77.2	87.7	77.8	87.1	73.5	**99.1**	85.1	91.6	96.5	88.8
VFM	57.7-0.1	**92.9**+1.9	**77.0**+1.6	**77.3**+0.1	**88.2**+0.5	**78.6**+0.8	**87.6**+0.5	**74.3**+0.8	**99.1**+0.0	**85.8**+0.7	**92.5**+0.9	**96.4**-0.1	**89.3**+0.5

At 1-shot the average gain is +0.8%, with +9.1% on SAT and +3.4% on Flowers, indicating that memory aids disambiguation when supervision is scarce. At 4-shot the gains are stable across difficult and easy splits (+1.4% and +0.6%), and at 16-shot they narrow to +0.5% overall with occasional neutral or slightly negative effects on saturated datasets: once abundant data constrains the alignment, residual endpoint ambiguity, and hence the benefit of trajectory modelling, shrinks. GRUFlow (Section B.6), a learned GRU with direct state supervision, further indicates that the history-aware structure rather than the specific kernel is the key contributor.

Direct prediction-risk probe. To test whether trajectory history carries predictive information beyond the current feature state, we perform a held-out ridge-regression probe on the same stochastic interpolation used by VFM. The Memoryless probe predicts the exact velocity target from $[x_t, t]$,

Table 3: **Ablative study on Feature Extractor**. We implement VFM on several PEFT approaches using 16-shot settings.

Method	Difficult						Easy						
	Aircraft	SAT	DTD	SUN	Cars	Avg	UCF	Net	Flowers	Food	Pets	Caltech	Avg
CLIP (2021)	24.8	47.8	43.8	62.5	65.5	48.9	66.7	66.7	67.4	85.3	89.1	92.9	78.0
+VFM	41.6	84.0	70.3	73.9	81.5	70.3+21.4	82.7	70.9	98.1	87.2	92.6	95.9	87.9 +9.9
CoOp (2022b)	43.3	86.0	70.0	74.9	83.1	71.4	83.1	71.4	97.2	84.4	91.1	95.5	87.1
+VFM	47.8	86.8	73.9	75.9	85.0	73.9+2.5	85.1	73.1	98.2	85.1	91.9	95.9	88.2 +1.1
CoCoOp (2022a)	33.8	75.5	65.8	72.8	72.4	64.1	76.0	71.1	87.1	87.4	93.2	95.2	85.0
+VFM	43.5	86.7	73.8	76.0	82.1	72.4+8.3	83.7	72.9	98.2	87.8	93.5	95.7	88.6+3.6
CLIP-Adapter (2024)	34.2	70.4	59.4	74.2	74.0	62.4	80.2	69.8	92.9	87.1	92.3	94.9	86.2
+VFM	42.5	84.6	71.5	76.0	81.8	71.3+8.9	84.6	72.9	98.0	87.4	93.0	95.6	88.6+2.0
CLIP-LoRA (2022)	54.7	90.7	73.0	76.0	86.0	76.1	86.2	73.4	97.9	84.2	91.6	96.1	88.2
+VFM	57.7	92.9	77.0	77.3	88.2	78.6+2.5	87.6	74.3	99.1	85.8	92.5	96.4	89.3+1.1

Table 4: **Kernel ablation on EuroSAT (%).** Full results are deferred to Appendix A.6.

Kernel	Exponential			Fractional							Matérn-3/2					
	β=0.5	β=1.0	β=2.0	α=0.2	0.4	0.5	0.6	0.8	1.0	learn	ℓ=0.25	0.5	1.0	2.0	4.0	learn
1-shot	82.04	82.07	82.21	82.53	82.56	82.57	**82.60**	82.49	82.47	82.57	82.47	82.44	82.30	82.33	82.28	82.32
4-shot	88.93	88.91	88.88	**89.07**	88.99	89.00	88.95	**89.07**	88.99	89.00	88.90	88.85	88.81	88.86	88.84	88.83
16-shot	92.88	92.86	92.73	92.64	92.88	92.94	**93.00**	92.90	92.78	92.94	92.60	92.62	92.69	92.63	92.70	92.64

whereas the memory probe uses $[x_t, h_t, t]$. Both probes are fit on a disjoint subset of simulated trajectories and evaluated on held-out trajectories, making this an out-of-sample prediction-risk test rather than an in-sample capacity comparison. Across Caltech101, DTD, EuroSAT, and Food101 and all three shot settings, adding h_t reduces held-out squared prediction risk in every cell, by roughly 40–70% depending on the dataset, with a mean reduction of approximately 53% across the 12 cells (Table 17). This provides direct empirical evidence that trajectory history contains velocity-predictive information not captured by the instantaneous state alone, complementing the accuracy ablation and the theoretical endpoint-disambiguation result. The complete diagnostic and its train/held-out ridge implementation are provided in the supplementary artifact (Section A.9).

4.2 Ablation on Feature Extractor

We integrate VFM with several PEFT strategies at 16 shots (Table 3). Gains appear across all feature extractors and are largest for weak baselines such as frozen CLIP (+21.4% difficult, +9.9% easy), where misalignment is most severe; for stronger methods (CoOp, CoCoOp, CLIP-Adapter, CLIP-LoRA) they are consistent but smaller, so VFM complements existing adaptation rather than replacing it. Improvements are more pronounced on difficult datasets, where ambiguity is higher, and require no architecture-specific modification. Backbone-scaling results further show that the VFM gain is not tied to a single CLIP encoder: improvements persist across six backbones and are especially pronounced on DTD and EuroSAT at higher shot counts (Section A.10). Multi-seed experiments likewise show small run-to-run variation, with VFM standard deviations of only 0.04–0.07 percentage points across the reported settings and consistent gains over FMA on nearly all feature extractors (Section A.11).

4.3 Alternative Memory Kernel ablation

We study the effect of different memory kernels in VFM; full details are provided in Appendix A.6. Table 4 reports all configurations in a compact format. Across exponential, fractional, and Matérn-3/2 kernels, performance remains tightly clustered ($< 0.6\%$ variation), indicating that gains primarily stem from incorporating memory rather than the specific kernel choice. Fractional kernels provide marginal improvements in low-shot settings, while the exponential kernel remains competitive and computationally simplest. We therefore adopt the exponential kernel with $\beta = 1.0$ as the default.

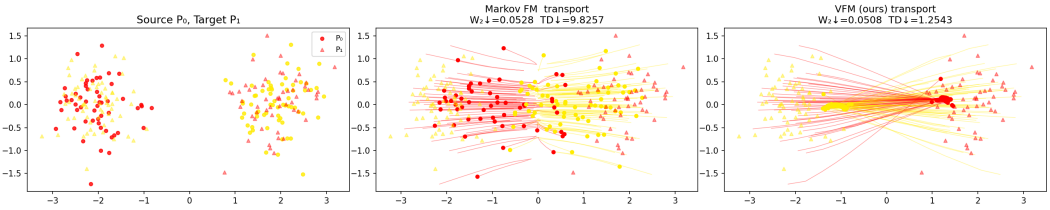


Figure 3: **Gaussian swap task: mode permutation under overlapping trajectories.** Two Gaussian clusters in P_0 (circles) must be transported to P_1 (triangles) by swapping positions. *Left:* ground-truth correspondence indicated by color. *Middle:* Memoryless flow matching produces diffuse trajectories, many drifting toward incorrect modes (TD $= 9.83$). *Right:* VFM maps each source cluster to its correct target cluster (TD $= 1.25$), preserving correspondence under mode permutation.
Metric notation: TD : *Transport Distance*; W_2 : *Wasserstein 2 distance*.

Table 5: **Synthetic transport benchmarks (mean $\pm$ std over 5 seeds)** We report the initial distribution gap $W_2(P_0, P_1)$, along with post-transport W_2 and TD metrics. Both Markov and VFM significantly reduce the initial gap, but VFM consistently improves TD in ambiguous and nonlinear cases.

Dataset (Characteristics)	$W_2(P_0, P_1) \downarrow$	$W_2 \downarrow$ **(Markov)**	$W_2 \downarrow$ **(VFM)**	**TD** $\downarrow$ **(Markov)**	**TD** $\downarrow$ **(VFM)**
XOR (nonlinear assignment)	0.0239 ± 0.0086	0.0455 ± 0.0039	$\mathbf{0.0121 \pm 0.0004}$	15.69 ± 0.46	$\mathbf{0.46 \pm 0.02}$
Gaussian Swap (mode permutation)	0.0232 ± 0.0032	$\mathbf{0.0503 \pm 0.0029}$	0.0528 ± 0.0057	9.81 ± 0.27	$\mathbf{1.27 \pm 0.06}$
Swiss Roll (manifold unfolding)	0.2045 ± 0.0070	0.0227 ± 0.0002	$\mathbf{0.0219 \pm 0.0001}$	0.0801 ± 0.0099	$\mathbf{0.0088 \pm 0.0003}$
Spiral (continuous ambiguity)	0.0585 ± 0.0016	0.0276 ± 0.0000	$\mathbf{0.0269 \pm 0.0001}$	0.1061 ± 0.0046	$\mathbf{0.0759 \pm 0.0030}$
Multi-Ring (mode identity)	0.0571 ± 0.0006	0.0352 ± 0.0002	$\mathbf{0.0352 \pm 0.0002}$	0.0729 ± 0.0011	$\mathbf{0.0714 \pm 0.0007}$
Figure-8 (topological constraint)	0.0712 ± 0.0018	0.0329 ± 0.0004	$\mathbf{0.0328 \pm 0.0004}$	0.0104 ± 0.0003	$\mathbf{0.0104 \pm 0.0002}$

4.4 Synthetic Benchmarks for Memory Based Transport

To isolate the role of memory, we construct controlled synthetic benchmarks with varying degrees of ambiguity, non-linearity, and manifold structure. Each task defines P_0 and P_1 with known correspondences, and we compare VFM against a Memoryless flow matching baseline of identical architecture and capacity under two complementary metrics.

Evaluation metrics. We report two complementary measures: the Wasserstein-2 distance W_2 for distribution-level alignment and Transport Distance (TD) for pointwise correspondence under known ground-truth assignments. W_2 can remain small under label permutations, whereas TD penalizes delivery to the wrong target and therefore exposes correspondence failures. Formal definitions are given in Appendix A.14.

Analysis. Both methods reduce the initial distribution gap on every benchmark except Gaussian Swap, so W_2 separates them only weakly: Memoryless flow matching suffices for marginal alignment in most cases. TD is far more discriminative. VFM improves substantially on XOR and Gaussian Swap, where the endpoint assignment is ambiguous at intermediate times, and consistently on Swiss Roll and Spiral, indicating that memory preserves structure during nonlinear transport.

5 Conclusion

We introduced **Volterra Flow Matching (VFM)**, a history-dependent flow-matching framework that augments cross-modal transport with a compact Volterra memory. The memory provides a structured summary of past states while retaining a simple recursive ODE implementation. Across 11 few-shot benchmarks, VFM consistently improves over PEFT and Memoryless flow-matching baselines, especially in low-shot and difficult-alignment regimes; synthetic experiments further show improved pointwise correspondence when marginal alignment is insufficient. The held-out ridge probe also shows substantial reductions in velocity prediction risk when history is provided, supporting the hypothesis that the instantaneous state can be insufficient for trajectory prediction. These results motivate history-dependent transport as a lightweight direction for trajectory-consistent alignment, while causal attribution and the universality of this phenomenon remain open questions.

Reproducibility Statement

Section 3 specifies the interpolant, the Volterra memory update, the history-conditioned velocity field, the training objective and the inference integration rule in full. Section A.12 lists the optimizer, learning rate, weight decay, warmup, batch size, seed, backbone, memory decay β, step size Δt, number of refinement steps and early-stopping protocol; Section A.13 gives the velocity-network architecture and initialization; Section A.14 defines the synthetic-benchmark metrics. All datasets are public and cited. All formal claims are proved in Section B and independently verified numerically by the self-contained NumPy script of Section B.5 (Listing 1), which is included with the submission. The complete training and evaluation implementation for VFM is provided in the supplementary materials, including the training and testing entry points, configuration, model and dataset modules, and supporting experimental diagnostics. The supplementary package is organized as a self-contained reproducibility artifact and can be used to reproduce the reported experiments.

References

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Adrian Bulat and Georgios Tzimiropoulos. Lasp: Text-to-text optimization for language-aware soft prompting of vision & language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 23232–23241, 2023.

Guangyi Chen, Weiran Yao, Xiangchen Song, Xinyue Li, Yongming Rao, and Kun Zhang. Plot: Prompt learning with optimal transport for vision-language models. *arXiv preprint arXiv:2210.01253*, 2022.

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

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Earl A Coddington, Norman Levinson, and T Teichmann. Theory of ordinary differential equations, 1956.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transportation distances, 2013. URL https://arxiv.org/abs/1306.0895.

Victor G Turrisi da Costa, Nicola Dall'Asen, Yiming Wang, Nicu Sebe, and Elisa Ricci. Diversified in-domain synthesis with efficient fine-tuning for few-shot classification. *arXiv preprint arXiv:2312.03046*, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pp. 178–178. IEEE, 2004.

Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *International Journal of Computer Vision*, 132(2):581–595, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Gustaf Gripenberg, Stig-Olof Londen, and Olof Staffans. *Volterra Integral and Functional Equations*. Cambridge University Press, 1990.

Ju He, Qihang Yu, Qihao Liu, and Liang-Chieh Chen. Flowtok: Flowing seamlessly across text and image tokens. *arXiv preprint arXiv:2503.10772*, 2025.

Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? *arXiv preprint arXiv:2210.07574*, 2022.

Sizhuang He, Daniel Levine, Ivan Vrkic, Marco Francesco Bressana, David Zhang, Syed Asad Rizvi, Yangtian Zhang, Emanuele Zappala, and David van Dijk. Calmflow: Volterra flow matching using causal language models, 2024. URL https://arxiv.org/abs/2410.05292.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pp. 4904–4916. PMLR, 2021.

Ziqi Jiang, Yanghao Wang, and Long Chen. Exploring cross-modal flows for few-shot learning, 2026. URL https://arxiv.org/abs/2510.14543.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Jae Myung Kim, Jessica Bader, Stephan Alaniz, Cordelia Schmid, and Zeynep Akata. Datadream: Few-shot guided dataset generation. In *European Conference on Computer Vision*, pp. 252–268. Springer, 2024.

Jeongsol Kim, Yeobin Hong, and Jong Chul Ye. Flowalign: Trajectory-regularized, inversion-free flow-based image editing. *arXiv preprint arXiv:2505.23145*, 2025.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Ryogo Kubo. The fluctuation-dissipation theorem. *Reports on Progress in Physics*, 29(1):255–284, 1966.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. *arXiv preprint arXiv:2412.08629*, 2024.

Dongjun Lee, Seokwon Song, Jihee Suh, Joonmyeong Choi, Sanghyeok Lee, and Hyunwoo J Kim. Read-only prompt optimization for vision-language few-shot learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1401–1411, 2023.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Qihao Liu, Xi Yin, Alan Yuille, Andrew Brown, and Mannat Singh. Flowing from words to pixels: A noise-free framework for cross-modality evolution. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 2755–2765, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Hazime Mori. Transport, collective motion, and brownian motion. *Progress of Theoretical Physics*, 33(3): 423–455, 1965.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Gabriel Peyré and Marco Cuturi. Computational optimal transport, 2020. URL https://arxiv.org/abs/1803.00567.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Litu Rout, Yujia Chen, Nataniel Ruiz, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Semantic image inversion and editing using rectified stochastic differential equations. *arXiv preprint arXiv:2410.10792*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Vito Volterra. *Theory of Functionals and of Integral and Integro-Differential Equations*. Blackie & Son Limited, 1930.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Hantao Yao, Rui Zhang, and Changsheng Xu. Visual-language prompt tuning with knowledge-guided context optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6757–6767, 2023.

Zhongqi Yue, Hanwang Zhang, Qianru Sun, and Xian-Sheng Hua. Interventional few-shot learning. *Advances in neural information processing systems*, 33:2734–2746, 2020.

Maxime Zanella and Ismail Ben Ayed. Low-rank few-shot adaptation of vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1593–1603, 2024.

Renrui Zhang, Wei Zhang, Rongyao Fang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free adaptation of clip for few-shot classification. In *European conference on computer vision*, pp. 493–510. Springer, 2022.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16816–16825, 2022a.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022b.

Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 15659–15669, 2023.

A Technical Appendices and Supplementary Material

A.1 Ethics Statement

This work studies feature transport for cross-modal alignment and evaluates it on public few-shot image-classification benchmarks with publicly released pre-trained encoders. No new human-subject data was collected, no personally identifying data was used, and no generative or otherwise dual-use model is released. We are not aware of direct societal risks arising from the method beyond those already inherent to the pre-trained vision–language models it adapts, whose known biases it neither audits nor corrects.

A.2 AI Usage Disclosure

AI-assisted tools were used during preparation of the submission for limited formatting and cleanup tasks, including LaTeX formatting, organization and presentation cleanup of the paper, and formatting/cleanup of the supplementary materials and documentation. The authors retained responsibility for the scientific content, mathematical claims, experimental design, reported results, and final verification of the submission.

A.3 Limitations

While VFM consistently improves accuracy across a wide range of settings, it introduces a modest computational overhead due to iterative memory refinement steps at inference time. Our theory is exact for the deterministic interpolant and gives an explicit perturbation for the small-noise setting; the empirical ridge probe tests the latter directly. The probe establishes incremental predictive information in the memory state, but does not by itself establish a causal explanation for downstream accuracy gains. Finally, the current experiments do not include a fully capacity-matched non-history auxiliary-state control; such a comparison would further isolate the contribution of trajectory history from additional state capacity.

A.4 Time Profiling and Inference Efficiency

We provide detailed time profiling for both training and inference using the EuroSAT benchmark under 1-, 4-, and 16-shot settings. Our goal is to quantify the computational overhead introduced by multi-step transport and to analyze the trade-off between accuracy and inference cost.

Training Time. Table 6 reports the average time per epoch. As expected, training time increases moderately with the number of shots due to larger effective dataset size. However, the overall cost remains low (sub-second per epoch), demonstrating that our method introduces negligible training overhead. (We report training time using a single A6000 GPU on a shared server.)

Table 6: **Training time per epoch (seconds)** on EuroSAT.

Shots	1-shot	4-shot	16-shot
Time / Epoch (s)	0.44	0.71	0.87

Inference Time vs. Number of Steps. We evaluate inference performance as a function of the number of transport steps. Tables 7–9 report accuracy and total inference time.

Table 7: **Inference performance vs. steps (EuroSAT 1-shot).**

Steps	0	1	2	3	4	5	6	7	8	9	10
Acc (%)	72.0	75.3	77.8	79.6	80.7	81.7	**82.1**	82.1	81.9	81.4	81.1
Time (s)	9.55	11.29	13.48	14.83	15.80	17.42	18.56	19.93	20.88	23.54	24.21

Analysis. Several key trends emerge:

Table 8: **Inference performance vs. steps (EuroSAT 4-shot).**

Steps	0	1	2	3	4	5	6	7	8	9	10
Acc (%)	86.7	88.2	88.8	88.9	**88.9**	88.8	88.7	88.6	88.5	88.4	88.3
Time (s)	9.30	11.95	13.75	15.96	16.29	18.53	18.43	19.07	21.19	22.06	24.02

Table 9: **Inference performance vs. steps (EuroSAT 16-shot).**

Steps	0	1	2	3	4	5	6	7	8	9	10
Acc (%)	90.8	91.8	92.5	92.8	92.7	**92.9**	92.7	92.7	92.7	92.8	92.6
Time (s)	9.35	11.71	13.59	15.15	15.86	17.14	18.70	19.31	20.71	23.28	24.29

- **Monotonic improvement with early saturation.** Accuracy consistently improves with the number of transport steps, but saturates quickly (around 4–6 steps). Beyond this point, gains are negligible or slightly degrade due to over-transport.
- **Linear time scaling.** Inference time increases approximately linearly with the number of steps, confirming the efficiency of the iterative solver.
- **Favorable trade-off.** A small number of steps (3–5) achieves near-optimal performance while incurring minimal overhead compared to the zero-step baseline.

Early Stopping Strategy. Based on these observations, we adopt an *early stopping* protocol similar to FMA Jiang et al. (2026) during inference: we terminate the transport process once validation accuracy saturates. In practice, this reduces the number of steps by $\sim$40–60% without sacrificing performance. For EuroSAT, optimal stopping occurs at:

- 1-shot: $\sim$6 steps
- 4-shot: $\sim$3–4 steps
- 16-shot: $\sim$5 steps

Takeaway. Overall, VFM achieves a favorable balance between performance and efficiency. Training overhead is negligible, and inference can be adaptively truncated using early stopping, making the method practical for real-world few-shot applications.

A.5 Statistical Significance of Memory-Induced Gains

To rigorously assess whether the Volterra memory mechanism in VFM provides statistically meaningful improvements over its Markovian counterpart FMA, we conduct a formal significance analysis over 11 benchmark datasets across all three shot settings. For each dataset we compute the signed per-dataset accuracy gain $\Delta_i = \text{VFM}_i - \text{FMA}_i$, and apply three complementary statistical procedures: (i) a one-sided paired t-test, which assumes Gaussian differences and tests whether the mean gain is significantly greater than zero; (ii) a one-sided Wilcoxon signed-rank test, which is distribution-free and ranks the magnitude of gains, providing robustness against the heavy-tailed structure of the per-dataset differences; and (iii) Cohen's d, which quantifies practical effect size independently of sample size. We additionally compute 95% bootstrap confidence intervals on the mean gain using $10{,}000$ resamples to provide an honest uncertainty estimate without distributional assumptions. All one-sided tests adopt the alternative hypothesis that VFM strictly outperforms FMA.

Per-shot results. Table 10 summarises the results. In the **1-shot** regime, VFM yields positive gains on 9 of 11 datasets, with a mean gain of $+1.66\% \pm 2.67\%$. The distribution of gains is notably heavy-tailed: EuroSAT contributes a gain of $+9.1\%$ while two datasets (Food101, Caltech101) show marginal losses of -0.2%. This skew explains the divergence between the two tests: the paired t-test does not reach conventional significance ($t = +2.07$, $p = 0.065$), as the large variance inflates the standard error, whereas the Wilcoxon signed-rank test, which is insensitive to the magnitude of the EuroSAT outlier and instead ranks the pattern of gains, attains high significance ($W = 63.0$,

Table 10: Statistical significance of memory-induced gains (VFM vs. FMA) across 11 benchmarks. Δ is the mean per-dataset accuracy gain (%). CIs are 95% bootstrap intervals (10,000 resamples). p-values are one-sided (H_1: VFM > FMA). $^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$, n.s. not significant. Cohen's d: small $|d|<0.5$, medium ≥ 0.5, large ≥ 0.8.

Setting	**Δ Mean**	**Δ Std**	**95% CI**	**+ / 0 / −**	**t-test p**	**Wilcoxon p**	**Cohen's d**
1-shot (n=11)	+1.66	2.67	[+0.49, +3.39]	9 / 0 / 2	0.065 n.s.	0.002^{**}	0.624 (medium)
4-shot (n=11)	+0.96	1.09	[+0.45, +1.62]	9 / 2 / 0	0.015^{*}	0.002^{**}	0.879 (large)
16-shot (n=11)	+0.62	0.67	[+0.26, +1.01]	8 / 1 / 2	0.012^{*}	0.007^{**}	0.928 (large)
Pooled (n=33)	+1.08	1.71	[+0.61, +1.72]	26 / 3 / 4	$<0.001^{***}$	$<0.001^{***}$	0.631 (medium)

$p = 0.002$). The bootstrap 95% CI of $[+0.49, +3.39]$ excludes zero, corroborating the Wilcoxon finding. Cohen's $d = 0.624$ indicates a medium practical effect size.

In the **4-shot** regime, the gain distribution is more uniform: VFM improves on 9 of 11 datasets with no negative gains and two ties. The mean gain of $+0.96\% \pm 1.09\%$ is now significant under both tests (t-test: $p = 0.015$; Wilcoxon: $p = 0.002$), and Cohen's $d = 0.879$ reaches the large-effect threshold. The bootstrap CI $[+0.45, +1.62]$ is comfortably above zero. This convergence of parametric and non-parametric tests, absent in the 1-shot case, reflects that the 4-shot gains are consistent rather than driven by a single dataset outlier.

In the **16-shot** regime, the mean gain is smaller at $+0.62\% \pm 0.67\%$, as expected from the main paper's observation that memory provides diminishing returns when supervision is abundant. However, the reduced variance produces the strongest parametric result across all three settings (t-test: $p = 0.012$; Wilcoxon: $p = 0.007$), and Cohen's $d = 0.928$ is the largest observed, reflecting that the gains, though modest in absolute magnitude, are highly consistent: 8 of 11 datasets improve, one is tied, and only two show negligible losses of -0.1%. The bootstrap CI $[+0.26, +1.01]$ further confirms this.

Pooled analysis. Pooling all 33 paired observations (11 datasets $\times$ 3 shot settings) yields a mean gain of $+1.08\% \pm 1.71\%$ with bootstrap CI $[+0.61, +1.72]$. Both the paired t-test and Wilcoxon test attain $p < 0.001$ (three-star significance), and Cohen's $d = 0.631$ indicates a consistent medium-to-large effect. This pooled result confirms that the benefit of Volterra memory is not confined to a particular shot setting or subset of datasets but is a systematic, reproducible phenomenon.

Group-level analysis. Table 11 breaks the gains down by dataset difficulty. Across all shot settings, the mean gain is larger on Difficult datasets (mean $+1.63\%$ across shots) than on Easy datasets (mean $+0.62\%$ across shots), consistent with the main paper's hypothesis that memory is most beneficial when cross-modal feature distributions are complex and entangled. However, the group-level t-tests on Difficult datasets do not reach significance ($p > 0.10$ in all cases), primarily due to the reduced sample size ($n = 5$) and the high variance introduced by the EuroSAT outlier in the 1-shot setting. For Easy datasets, significance is attained in the 4-shot ($p = 0.025$) and 16-shot ($p = 0.043$) regimes, indicating that even on well-aligned datasets memory provides reliably consistent, if smaller, improvements.

Interpretation. Taken together, the evidence supports the conclusion that the Volterra memory mechanism provides a statistically significant and practically meaningful improvement over Markovian dynamics. The discrepancy between the parametric and non-parametric results in the 1-shot regime is a consequence of the heavy-tailed gain distribution rather than an absence of effect: the Wilcoxon test, which is better suited to small samples with outliers, consistently identifies significance. The effect size grows from medium ($d = 0.624$) to large ($d = 0.928$) as shot count increases and gain variance decreases, reflecting that memory is not merely beneficial in isolated cases but produces reliable, structured improvements as the trajectory dynamics become better constrained by available supervision.

Table 11: Mean accuracy gain of VFM over FMA broken down by dataset difficulty. *Difficult*: Aircraft, EuroSAT, DTD, SUN397, StanfordCars (n=5). *Easy*: UCF101, ImageNet, Flowers102, Food101, OxfordPets, Caltech101 (n=6). p-values from one-sided paired t-test. Significance codes as in Table 10.

Setting	**Difficult** (n=5)		**Easy** (n=6)	
	Δ Mean (%)	p-value	Δ Mean (%)	p-value
1-shot	+2.68	0.173 n.s.	+0.82	0.200 n.s.
4-shot	+1.40	0.104 n.s.	+0.58	0.025*
16-shot	+0.80	0.118 n.s.	+0.47	0.043*
Mean	+1.63	—	+0.62	—

A.6 Ablation Study: Alternative Memory Kernels

To investigate whether the choice of memory kernel meaningfully affects transport quality, we evaluate two alternative kernel families alongside the original exponential kernel on the EuroSAT benchmark across all shot settings. Specifically, we compare: (i) the exponential kernel with varying decay rate β, which is the default VFM formulation; (ii) a fractional integral kernel parameterized by order $\alpha \in (0, 1]$; and (iii) a Matérn-$3/2$ kernel parameterized by length-scale ℓ. All other hyperparameters and architectural choices are held fixed.

Kernel Descriptions. **Exponential kernel.** The exponential kernel $K(\tau) = e^{-\beta\tau}$ underlies the original VFM formulation and admits an exact first-order ODE equivalent

$$\dot{h} = -\beta h + x_t, \qquad h_0 = 0, \tag{15}$$

enabling $\mathcal{O}(1)$ recursive memory updates with no trajectory storage overhead. The decay rate β governs the effective memory horizon, with larger β discounting distant history more aggressively.

Fractional integral kernel. The fractional kernel is derived from the Riemann–Liouville fractional integral of order $\alpha \in (0, 1]$, discretized via the Grünwald–Letnikov scheme:

$$h_k \;=\; \Delta t^{\alpha} \sum_{j=0}^{k} w_j^{(\alpha)}\, x_{k-j}, \qquad w_0 = 1, \quad w_j = w_{j-1} \cdot \frac{j - 1 - \alpha}{j}. \tag{16}$$

The order α directly controls the degree of history dependency: as $\alpha \to 0$ the memory contribution vanishes and the dynamics become effectively Memoryless, while $\alpha = 1$ recovers standard numerical integration. This parameterization provides a single principled axis along which to probe history-aware structure, in contrast to the binary VFM-vs-FMA comparison presented in the main text. In our experiments we evaluate both fixed values of α and a learnable variant in which α is optimized in sigmoid-space to ensure $\alpha \in (0, 1)$ throughout training.

Matérn-$3/2$ kernel. The Matérn-$3/2$ kernel

$$K(\tau) \;=\; \left(1 + \frac{\sqrt{3}\,\tau}{\ell}\right) \exp\left(-\frac{\sqrt{3}\,\tau}{\ell}\right) \tag{17}$$

is motivated by Gaussian process theory and admits an exact second-order ODE state-space representation:

$$\dot{s}_1 = s_2, \qquad \dot{s}_2 = -\frac{2\sqrt{3}}{\ell}\, s_2 - \frac{3}{\ell^2}\, s_1 + x_t, \qquad s_1(0) = s_2(0) = 0, \tag{18}$$

with memory readout $h = s_1$. Unlike the exponential kernel, this second-order system encodes both the accumulated magnitude and the rate of change of past features, providing an inductive bias toward smooth, curvature-aware trajectories. The length-scale ℓ governs the effective memory horizon: small ℓ approaches near-Markovian behavior, while large ℓ retains long-range history. The Euler integration of this system is stable for our default step size $\Delta t = 0.1$, since the stability condition

Table 12: Ablation over memory kernel type and hyperparameters on EuroSAT. Accuracy (%) is reported across 1-shot, 4-shot, and 16-shot settings. The exponential kernel with $\beta = 1.0$ is the default VFM configuration used in all main experiments.

Kernel	1-shot	4-shot	16-shot
Exponential $\beta = 0.5$	82.04	88.93	92.88
Exponential $\beta = 1.0$ (default)	82.07	88.91	92.86
Exponential $\beta = 2.0$	82.21	88.88	92.73
Fractional $\alpha = 0.2$	82.53	89.07	92.64
Fractional $\alpha = 0.4$	82.56	88.99	92.88
Fractional $\alpha = 0.5$	82.57	89.00	92.94
Fractional $\alpha = 0.6$	**82.60**	88.95	**93.00**
Fractional $\alpha = 0.8$	82.49	**89.07**	92.90
Fractional $\alpha = 1.0$	82.47	88.99	92.78
Fractional $\alpha =$ learnable	82.57	89.00	92.94
Matérn-$3/2$ $\ell = 0.25$	82.47	88.90	92.60
Matérn-$3/2$ $\ell = 0.5$	82.44	88.85	92.62
Matérn-$3/2$ $\ell = 1.0$	82.30	88.81	92.69
Matérn-$3/2$ $\ell = 2.0$	82.33	88.86	92.63
Matérn-$3/2$ $\ell = 4.0$	82.28	88.84	92.70
Matérn-$3/2$ $\ell =$ learnable	82.32	88.83	92.64

$\Delta t < 2/\sqrt{3/\ell^2}$ is satisfied with substantial margin for all $\ell \geq 0.25$ tested. As with the fractional kernel, we evaluate both fixed and learnable variants of ℓ, where the latter is optimized in log-space to enforce strict positivity.

Results. Table 12 reports classification accuracy on EuroSAT across all kernel configurations and shot settings.

Analysis. Several observations emerge from these results.

All kernels perform comparably. Across all three kernel families, accuracy is competitive and broadly consistent across shot settings, confirming that the primary driver of VFM's performance gains over FMA is the presence of a memory mechanism rather than the specific functional form of the kernel. The total spread across all configurations is less than 0.6% in the 1-shot regime, less than 0.3% in the 4-shot regime, and less than 0.4% in the 16-shot regime, indicating that the method is robustly insensitive to kernel choice within the range studied.

Fractional kernel yields modest low-shot gains. The fractional kernel with $\alpha \in \{0.5, 0.6\}$ achieves the highest 1-shot accuracy (82.60%) and the highest 16-shot accuracy (93.00%) among all configurations tested, representing modest but consistent improvements over the exponential baseline. This suggests that intermediate fractional orders, which balance short-range and long-range memory accumulation, may provide a marginal benefit in low-data regimes where trajectory disambiguation is most critical. Notably, the learnable α variant converges to the $\alpha = 0.5$ fixed solution, indicating that gradient-based optimization reaches a reasonable operating point without instability.

Matérn-$3/2$ does not confer consistent advantages. Despite its richer second-order dynamics, the Matérn-$3/2$ kernel does not yield consistent gains over the simpler exponential kernel in this setting. Accuracy is marginally lower across most configurations, suggesting that the additional curvature information encoded by the second-order state-space form does not provide a meaningful advantage at the feature dimensions and trajectory lengths considered here (3–6 steps). The broad insensitivity to ℓ across nearly an order of magnitude ($\ell \in [0.25, 4.0]$) further suggests that the kernel's structural properties are not being exploited effectively at this scale.

Exponential kernel is insensitive to β. The exponential kernel itself is largely insensitive to β in the range tested, consistent with the fixed-β ablation reported in Section A.7. The slight degradation observed at $\beta = 2.0$ in the 16-shot setting mirrors the pattern seen there: aggressive decay can

under-weight early trajectory context when supervision is sufficiently plentiful to constrain alignment strongly.

Takeaway. While the fractional kernel yields modest and consistent gains in certain low-shot configurations, the improvements remain within 0.5% and do not constitute a systematic advantage across all settings. Given the additional implementation complexity of the Grünwald–Letnikov discretization and its associated trajectory buffer, the benefit-to-cost ratio does not justify adopting it as the default kernel. The Matérn-$3/2$ kernel, despite its well-established theoretical grounding in GP theory and its exact state-space representation, similarly fails to provide reliable gains at the trajectory lengths used during inference. Accordingly, in all main experiments reported in this paper we adopt the exponential kernel with fixed $\beta = 1.0$, which provides a strong and stable inductive bias with minimal computational and implementational overhead. A thorough investigation of alternative kernel families across diverse datasets, longer rollout horizons, and higher-dimensional feature spaces remains a promising direction for future work.

A.7 Ablation Study on Kernel Decay Parameter β

We analyze the impact of the Volterra kernel decay parameter β on performance. Recall that β controls the temporal decay in the memory kernel:

$$K(t-s) = \exp(-\beta(t-s)), \tag{19}$$

and thus governs the effective memory horizon. We compare two settings: (i) a fixed value $\beta = 1.0$ (chosen via light empirical tuning), and (ii) a learnable β parameter optimized in log-space.

Table 13: **Effect of learning β (1-shot).**

Dataset	$\beta = 1.0$	Learned β	Δ
OxfordPets	93.19	92.99	-0.20
EuroSAT	82.11	81.36	-0.75
UCF101	76.42	76.53	+0.11
SUN397	70.93	70.96	+0.03
Caltech101	94.28	94.20	-0.08
DTD	56.74	56.68	-0.06
Aircraft	29.49	28.95	-0.54
Food101	84.99	84.99	+0.00
Flowers	88.31	88.23	-0.08
Cars	71.05	70.82	-0.23
ImageNet	70.46	70.47	+0.01

Analysis. Across all shot settings, we observe that learning β provides *marginal and inconsistent gains*. While certain datasets (e.g., DTD, UCF101, Flowers) benefit from a learnable decay parameter, others (e.g., EuroSAT, Aircraft) exhibit slight degradation. Overall, the improvements are typically within $\pm 0.5\%$, indicating that the model is *not highly sensitive* to precise tuning of β.

This suggests that a fixed $\beta = 1.0$ already provides a strong inductive bias, balancing short-term and long-term memory effectively. Learning β introduces additional flexibility but also increases optimization complexity, which may lead to minor instability in low-shot regimes.

Takeaway. Given the negligible gains and occasional drops, we adopt a fixed $\beta = 1.0$ in all main experiments for simplicity and robustness. This choice avoids additional hyperparameter tuning while maintaining competitive performance.

Table 14: **Effect of learning β (4-shot).**

Dataset	$\beta = 1.0$	Learned β	Δ
OxfordPets	91.58	91.69	+0.11
EuroSAT	88.94	89.26	+0.32
UCF101	82.82	82.69	-0.13
SUN397	73.74	73.81	+0.07
Caltech101	95.78	95.58	-0.20
DTD	68.38	68.68	+0.30
Aircraft	40.92	40.62	-0.30
Food101	83.87	83.90	+0.03
Flowers	96.26	96.55	+0.29
Cars	79.99	80.09	+0.10
ImageNet	72.25	72.22	-0.03

Table 15: **Effect of learning β (16-shot).**

Dataset	$\beta = 1.0$	Learned β	Δ
OxfordPets	92.53	92.61	+0.08
EuroSAT	92.91	92.84	-0.07
UCF101	87.58	87.79	+0.21
SUN397	77.31	77.30	-0.01
Caltech101	96.43	96.55	+0.12
DTD	77.01	77.66	+0.65
Aircraft	57.73	57.55	-0.18
Food101	85.76	85.65	-0.11
Flowers	99.07	99.11	+0.04
Cars	88.17	88.31	+0.14
ImageNet	74.33	74.32	-0.01

A.8 Generalization of Velocity Field Across Datasets

We evaluate whether a velocity field trained on ImageNet generalizes to unseen datasets without retraining. For clarity and compactness, we report only the baseline accuracy and the improvement (Δ), with the final accuracy implicitly given by their sum.

Analysis. The table 16 reveals several key patterns:

Dataset-agnostic transport. The pretrained velocity field consistently yields modest improvements across diverse datasets, confirming that it captures a *generalizable geometric prior* in feature space rather than dataset-specific statistics.

Stronger gains on misaligned regimes. Larger improvements are observed for weaker baselines (e.g., CLIP on EuroSAT, DTD), where feature misalignment is more pronounced. This highlights that the learned flow acts as a corrective mechanism.

Diminishing returns for strong models. For high-performing methods such as LoRA and CoOp, gains are smaller and occasionally negative. This suggests that when features are already well-aligned, additional transport may introduce slight overcorrection.

Table 16: **Cross-dataset generalization of velocity field (trained on ImageNet).** Values are in %. Δ denotes improvement over baseline.

Dataset	Baseline					Δ (Improvement)				
	CLIP	CoOp	CoCoOp	Adapter	LoRA	CLIP	CoOp	CoCoOp	Adapter	LoRA
OxfordPets	89.07	91.14	93.13	92.45	91.69	+0.16	+0.25	+0.11	+0.05	+0.08
EuroSAT	48.42	85.46	75.52	70.93	90.67	+0.95	+0.06	-0.04	-0.27	+0.00
UCF101	67.54	83.37	76.39	80.04	85.99	+0.03	-0.08	+0.26	+0.53	+0.45
SUN397	62.62	74.88	72.70	74.16	75.64	+0.66	-0.09	+0.02	+0.05	-0.01
Caltech101	93.35	95.46	94.85	94.89	96.27	+0.49	+0.28	+0.16	+0.20	+0.16
DTD	43.97	69.68	65.60	59.46	72.99	+2.19	-0.18	-0.12	+1.00	+0.06
Aircraft	24.90	43.41	33.66	33.90	53.53	-0.12	-0.51	+0.03	+0.18	+0.54
Food101	85.87	84.44	87.41	87.04	84.23	+0.10	-0.10	+0.00	+0.00	-0.01
Flowers	70.69	97.12	87.17	92.94	98.09	+0.08	+0.16	+0.85	+0.16	-0.12
Cars	65.50	82.80	72.42	74.06	85.28	+0.06	+0.12	-0.07	-0.04	-0.11

Table 17: **Held-out ridge prediction risk for the velocity target.** Lower is better. R_M uses only the instantaneous state and time, $[x_t, t]$; R_H additionally exposes the Volterra memory, $[x_t, h_t, t]$. $\Delta = R_M - R_H$ and $\Delta\%$ is the relative risk reduction. CIs are 95% paired bootstrap intervals for Δ.

Dataset	Shots	R_M	R_H	Δ	$\Delta\%$	95% CI
Caltech101	1	0.1216	0.0736	0.0480	39.5	[0.0457, 0.0502]
Caltech101	4	0.1215	0.0725	0.0490	40.4	[0.0467, 0.0514]
Caltech101	16	0.1200	0.0721	0.0480	40.0	[0.0458, 0.0502]
DescribableTextures	1	0.1054	0.0403	0.0651	61.7	[0.0620, 0.0679]
DescribableTextures	4	0.1040	0.0394	0.0646	62.1	[0.0617, 0.0675]
DescribableTextures	16	0.1028	0.0366	0.0662	64.4	[0.0634, 0.0691]
EuroSAT	1	0.0427	0.0129	0.0299	69.8	[0.0284, 0.0315]
EuroSAT	4	0.0444	0.0132	0.0312	70.2	[0.0296, 0.0328]
EuroSAT	16	0.0427	0.0126	0.0301	70.4	[0.0286, 0.0316]
Food101	1	0.1596	0.0941	0.0654	41.0	[0.0627, 0.0684]
Food101	4	0.1622	0.1004	0.0618	38.1	[0.0589, 0.0645]
Food101	16	0.1651	0.1022	0.0629	38.1	[0.0599, 0.0661]

A.9 Held-Out Ridge Probe for Path Dependence

The accuracy ablation in the main paper establishes that adding the Volterra memory improves downstream prediction. We additionally ask a more direct question: *does trajectory history contain information about the target velocity that is not recoverable from the current state alone?* To answer this without introducing another neural model, the supplementary diagnostic constructs the exact stochastic interpolation used by VFM on held-out test images and fits two closed-form ridge regressions.

For each held-out image–class-text pair, we sample the terminal time, interpolation noise, and the discrete memory rollout using the same equations as training. This gives $(x_t, h_t, t, v^\star)$, where $v^\star$ is the exact ground-truth velocity because the sampled noise is known. We then compare:

$$\text{Markov probe:} \quad \hat{v}_M = f_M([x_t, t]), \tag{20}$$

$$\text{Memory probe:} \quad \hat{v}_H = f_H([x_t, h_t, t]). \tag{21}$$

Both predictors are ridge regressions with the same regularization, trained on 60% of the simulated trajectories and evaluated on the remaining 40%. Thus, the reported quantity is an empirical held-out prediction risk, not training fit quality. The paired bootstrap confidence interval is computed over the same held-out trajectories for both probes. The diagnostic is implemented in `experiments/diagnostic_path_dependence.py` in the supplementary package.

Table 18: **Backbone scaling at 1 shot.** Entries are zero-shot / VFM / gain.

Backbone	Caltech101	DTD	EuroSAT	Food101
CLIP ViT-B/32	91.4 / 93.1 / +1.7	44.3 / 49.2 / +4.9	45.4 / 59.0 / +13.6	80.4 / 80.5 / +0.1
CLIP ViT-B/16	93.3 / 94.1 / +0.8	44.0 / 54.6 / +10.6	48.4 / 65.8 / +17.3	85.9 / 86.1 / +0.2
CLIP ViT-L/14	95.1 / 96.1 / +1.0	52.9 / 60.3 / +7.4	60.2 / 72.6 / +12.4	90.9 / 91.0 / +0.1
CLIP ViT-L/14-336	94.5 / 95.9 / +1.4	54.5 / 61.1 / +6.6	61.4 / 73.5 / +12.1	91.9 / 92.1 / +0.2
OpenCLIP ViT-H/14	97.2 / 97.3 / +0.0	62.8 / 69.6 / +6.7	68.4 / 76.3 / +7.9	90.3 / 90.5 / +0.2
SigLIP-Base	97.2 / 97.3 / +0.1	62.5 / 69.7 / +7.2	44.2 / 66.1 / +21.9	89.2 / 89.4 / +0.1

Table 19: **Backbone scaling at 4 shots.** Entries are zero-shot / VFM / gain.

Backbone	Caltech101	DTD	EuroSAT	Food101
CLIP ViT-B/32	91.4 / 93.8 / +2.4	44.3 / 56.5 / +12.2	45.4 / 75.3 / +29.9	80.4 / 80.9 / +0.5
CLIP ViT-B/16	93.3 / 95.5 / +2.2	44.0 / 61.3 / +17.4	48.4 / 77.9 / +29.4	85.9 / 86.4 / +0.6
CLIP ViT-L/14	95.1 / 97.2 / +2.1	52.9 / 67.6 / +14.7	60.2 / 83.4 / +23.2	90.9 / 91.4 / +0.5
CLIP ViT-L/14-336	94.5 / 97.2 / +2.7	54.5 / 68.1 / +13.6	61.4 / 82.0 / +20.6	91.9 / 92.3 / +0.4
OpenCLIP ViT-H/14	97.2 / 97.7 / +0.5	62.8 / 73.9 / +11.1	68.4 / 85.3 / +16.9	90.3 / 90.7 / +0.4
SigLIP-Base	97.2 / 97.8 / +0.6	62.5 / 71.7 / +9.2	44.2 / 80.5 / +36.2	89.2 / 89.6 / +0.3

Across the 12 dataset–shot cells, the memory probe has lower held-out risk in every case. Averaging the cells gives $R_M = 0.1077$ versus $R_H = 0.0558$, a relative reduction of approximately 53.0%. The reduction is about 39–41% on Caltech101 and Food101, 62–64% on DTD, and roughly 70% on EuroSAT. The corresponding paired bootstrap intervals for Δ are strictly positive in every reported cell (the full intervals are retained in the supplementary results file).

This probe should be interpreted as *predictive evidence* for the history-aware hypothesis rather than a causal test: adding h_t gives the regressor access to additional information by construction. The important empirical observation is that this additional information produces a large and reproducible reduction in *held-out* velocity-prediction risk on real CLIP image–text pairs, showing that the simulated trajectories contain history-dependent signal relevant to the target dynamics. This directly complements Proposition 3, which provides the corresponding population-risk motivation, and the downstream accuracy gains reported in Table 2.

A.10 Backbone Scaling

We evaluate VFM with six frozen vision-language backbones to test whether the observed improvement is specific to one encoder. Each entry below reports *zero-shot / VFM / absolute gain* in percentage points. The same VFM adaptation is applied across backbones; only the underlying feature encoder changes.

The improvement is observed for all six backbones and all four datasets in these scaling experiments, with the largest gains consistently appearing on the more challenging DTD and EuroSAT benchmarks. At 16 shots, for example, EuroSAT improves by $+22.3$–$+43.4$ points and DTD by $+16.8$–$+26.3$ points across the six backbones, while gains on the already saturated Food101 task remain modest. These results indicate that the memory-based transport gain is not an artifact of a single backbone choice.

A.11 Multi-Seed Stability

To quantify run-to-run variability, we repeat the 16-shot experiments across multiple random seeds and report the mean VFM accuracy and its standard deviation. We also retain the zero-shot and FMA baselines for direct comparison. The results are grouped into difficult and easy datasets.

The standard deviations are small (0.04–0.07 percentage points) across all reported feature extractors and dataset groups. VFM improves over FMA in three of four feature-extractor settings on difficult datasets and in all four settings on easy datasets. The only difficult-set exception is CoOp, where the mean difference is -0.1 points, which is effectively a tie at the reported precision. Thus the multi-seed

Table 20: **Backbone scaling at 16 shots.** Entries are zero-shot / VFM / gain.

Backbone	Caltech101	DTD	EuroSAT	Food101
CLIP ViT-B/32	91.4 / 94.8 / +3.4	44.3 / 65.4 / +21.2	45.4 / 83.0 / +37.5	80.4 / 81.9 / +1.5
CLIP ViT-B/16	93.3 / 95.9 / +2.6	44.0 / 70.3 / +26.3	48.4 / 84.0 / +35.6	85.9 / 87.2 / +1.3
CLIP ViT-L/14	95.1 / 97.8 / +2.6	52.9 / 74.7 / +21.8	60.2 / 87.0 / +26.7	90.9 / 91.8 / +0.8
CLIP ViT-L/14-336	94.5 / 97.4 / +3.0	54.5 / 75.5 / +21.0	61.4 / 87.8 / +26.4	91.9 / 92.7 / +0.8
OpenCLIP ViT-H/14	97.2 / 98.2 / +0.9	62.8 / 81.7 / +18.9	68.4 / 90.7 / +22.3	90.3 / 91.1 / +0.7
SigLIP-Base	97.2 / 97.8 / +0.6	62.5 / 79.3 / +16.8	44.2 / 87.7 / +43.4	89.2 / 90.2 / +1.0

Table 21: **Multi-seed stability on difficult datasets (16 shots).** VFM values are mean $\pm$ standard deviation across seeds.

Feature extractor	Zero-shot	FMA	VFM	Gain vs. FMA
Adapter	59.6	66.3	70.12 ± 0.05	+3.8
CLIP	45.7	67.6	69.38 ± 0.04	+1.8
CoOp	70.3	73.5	73.41 ± 0.05	-0.1
LoRA	75.8	78.0	78.94 ± 0.07	+1.0

results support stability of the observed gains without suggesting that every feature extractor benefits equally.

A.12 Hyperparameters and Implementation Details

We provide a detailed description of all hyperparameters used in our experiments. Unless otherwise specified, the same configuration is used across datasets and shot settings to ensure consistency.

Optimization. We train all models for 200 epochs using the AdamW optimizer with a learning rate of 2×10^{-4} and weight decay of 0.01. A short warmup of 1 epoch is applied at the beginning of training. The batch size is fixed to 32 across all experiments. For the single-seed main tables, we fix the random seed to 1 for reproducibility; multi-seed stability is reported in Appendix A.11. The stochastic noise coefficient γ is set to 0.0001, in our main experiments.

Backbone and Feature Extractor. We use CLIP with a ViT-B/16 backbone as the default feature encoder. Our framework is evaluated with multiple PEFT methods, including CLIP, CoOp, CoCoOp, Adapter, and LoRA. Unless otherwise stated, results in the main paper use LoRA as the feature extractor. The number of transformer blocks is set to 12.

Datasets and Few-Shot Setup. We evaluate on standard benchmarks including OxfordPets, EuroSAT, UCF101, SUN397, Caltech101, DescribableTextures, FGVCAircraft, Food101, OxfordFlowers, StanfordCars, and ImageNet. We consider K-shot settings with $K \in \{1, 4, 16\}$ and use all classes unless explicitly stated.

Volterra Memory Parameters. Our method introduces a history-dependent memory through a Volterra integral formulation. The key hyperparameters are:

- **Memory decay (β).** Controls the exponential decay of past information in the memory kernel.
- **Memory discretization step (Δt).** Set to 0.1.
- **Number of refinement steps (T).** Set to 10 during training, with early stopping applied during inference.

For ablative study, instead of directly optimizing β, we parameterize it in *log-space*. Specifically, we learn a scalar $\log \beta$ initialized at 0, corresponding to $\beta = 1.0$, and recover β via exponentiation:

$$\beta = \exp(\log \beta).$$

This parameterization ensures that β remains strictly positive throughout training and improves numerical stability. Empirically, we observe negligible gains and occasional drops, we adopt a fixed $\beta = 1.0$ in all main experiments for simplicity and robustness.

Table 22: **Multi-seed stability on easy datasets (16 shots).** VFM values are mean ± standard deviation across seeds.

Feature extractor	Zero-shot	FMA	VFM	Gain vs. FMA
Adapter	86.5	87.4	88.60 ± 0.07	+1.2
CLIP	78.9	87.6	87.91 ± 0.04	+0.3
CoOp	87.2	87.9	88.22 ± 0.05	+0.4
LoRA	88.2	88.8	89.32 ± 0.06	+0.5

Inference Configuration. At test time, we use a step size of $\Delta t = 0.1$ and vary the number of refinement steps. While training uses $T = 10$ steps, we apply early stopping based on validation performance, typically requiring only 3–6 steps for near-optimal results. This significantly reduces inference cost without sacrificing accuracy.

Framework Details. All experiments are implemented in PyTorch and executed on a single GPU. The implementation is lightweight and introduces minimal overhead compared to standard flow matching baselines.

A.13 Volterra Flow Matching Network Implementation

We implement the velocity field in Volterra Flow Matching (VFM) using a lightweight (approx. 17.5 M params) yet expressive *time-conditioned residual MLP*, adapted from recent flow matching architectures. The key design principle is to retain the simplicity and efficiency of standard flow matching networks while enabling seamless integration with the *history-dependent (Volterra) dynamics*.

Input parameterization. At each refinement step $t \in [0, 1]$, the model receives a concatenated feature vector

$$x_t^{\text{in}} = [x_t \parallel h_t] \in \mathbb{R}^{2d}, \tag{22}$$

where $x_t \in \mathbb{R}^d$ denotes the current feature state and $h_t \in \mathbb{R}^d$ is the Volterra memory variable encoding past trajectory information. The network predicts the velocity field

$$v_\theta(x_t, h_t, t) \in \mathbb{R}^d, \tag{23}$$

which governs history-aware dynamics.

Architecture overview. The velocity network, denoted as `DeepFlowMatchingNet`, consists of three main components: (i) input projection, (ii) timestep conditioning via sinusoidal embeddings, and (iii) a stack of adaptive residual blocks followed by a zero-initialized output layer.

Input projection. The concatenated input $x_t^{\text{in}} \in \mathbb{R}^{2d}$ is first mapped to a hidden representation:

$$z_0 = W_{\text{in}} x_t^{\text{in}} + b_{\text{in}}, \quad z_0 \in \mathbb{R}^d. \tag{24}$$

Timestep embedding. We embed the continuous time variable t using sinusoidal positional encodings followed by an MLP:

$$e(t) = \text{MLP}(\gamma(t)), \quad \gamma(t) = [\cos(\omega t), \sin(\omega t)], \tag{25}$$

where frequencies ω are exponentially spaced. The time embedding is scaled to $t \in [0, 1000]$ before encoding, following diffusion-style parameterization. This produces a conditioning vector $e(t) \in \mathbb{R}^d$.

Adaptive residual blocks. The core of the network consists of L residual blocks (we use $L = 12$ by default). Each block applies *Adaptive Layer Normalization (AdaLN)* conditioned on the timestep embedding:

$$\tilde{z} = \text{LN}(z), \tag{26}$$

$$\hat{z} = (1 + s(e(t))) \odot \tilde{z} + b(e(t)), \tag{27}$$

$$z' = z + g(e(t)) \odot \text{MLP}(\hat{z}), \tag{28}$$

where $s(\cdot)$, $b(\cdot)$, and $g(\cdot)$ are learned affine transformations generated from the timestep embedding. This modulation allows the network to adapt its transformation dynamically across time, which is critical for modeling the evolution of transport trajectories.

Final layer. The output layer applies a final AdaLN transformation followed by a linear projection:

$$v_\theta = W_{\text{out}} \cdot \text{AdaLN}(z_L, e(t)), \tag{29}$$

where z_L is the final hidden state.

Initialization strategy. To stabilize training, we adopt a careful initialization:

- All linear layers are initialized with Xavier uniform initialization.
- The modulation layers in AdaLN are zero-initialized, ensuring that the network starts close to an identity mapping.
- The final linear layer is initialized to zero, so the initial velocity field is near zero.

This initialization ensures that early training behaves like a near-static flow, gradually learning meaningful dynamics.

A.14 Evaluation Metrics for Synthetic Benchmarks

This section provides the full computational definitions of the two metrics reported for the synthetic transport benchmarks of Section 4.4 (Table 5 and Figures 1, 3). Both metrics are computed on the same set of transported samples produced by integrating the learned dynamics from $t = 0$ to $t = 1$.

A.14.1 Notation

For a given task we draw n source samples $\{x_i\}_{i=1}^n \sim P_0$ and m target samples $\{y_j\}_{j=1}^m \sim P_1$. Each source sample x_i is paired with a *ground-truth correspondent* $y_i^\star \in P_1$ defined by the construction of the task (e.g., the diagonally opposite cluster in XOR, the swapped Gaussian in Gaussian Swap). Integrating the learned dynamics yields transported samples $\hat{x}_i \in \mathbb{R}^d$.

A.14.2 Wasserstein-2 Distance (Sinkhorn Approximation)

We approximate the squared Wasserstein-2 distance between the empirical distributions of $\{\hat{x}_i\}$ and $\{y_j\}$ via Sinkhorn's entropy-regularized optimal transport (Cuturi, 2013). Treating both samples as uniformly weighted point clouds, we form the cost matrix

$$C_{ij} \;=\; \|\hat{x}_i - y_j\|^2, \qquad C \in \mathbb{R}^{n\times m}, \tag{30}$$

the Gibbs kernel

$$K_{ij} \;=\; \exp\!\Big(-\tfrac{C_{ij}}{\varepsilon}\Big), \tag{31}$$

and run Sinkhorn's fixed-point iteration on the dual scalings $u \in \mathbb{R}^n$, $v \in \mathbb{R}^m$:

$$u_i^{(\ell+1)} \leftarrow \frac{1/n}{\sum_j K_{ij}\, v_j^{(\ell)} + \delta}, \tag{32}$$

$$v_j^{(\ell+1)} \leftarrow \frac{1/m}{\sum_i K_{ij}\, u_i^{(\ell+1)} + \delta}, \tag{33}$$

initialized at $u^{(0)} = \mathbf{1}/n$, $v^{(0)} = \mathbf{1}/m$, with a small constant $\delta = 10^{-9}$ added for numerical stability. After convergence, the optimal coupling is recovered as $T^\star = \text{diag}(u)\, K\, \text{diag}(v)$, and we report

$$W_2^\varepsilon(\hat{X}, Y) \;=\; \sum_{i,j} T^\star_{ij}\, C_{ij} \;=\; \sum_{i,j} T^\star_{ij}\, \|\hat{x}_i - y_j\|^2. \tag{34}$$

Hyperparameters. Following standard practice (Cuturi, 2013; Peyré & Cuturi, 2020), we use entropic regularization $\varepsilon = 0.1$ and run the Sinkhorn iteration for a fixed budget of iterations sufficient for the marginal residuals to fall below 10^{-6} in ℓ_∞ norm. The same (n, m, ε) are used for both VFM and the Markov baseline so that all reported W_2 values in Table 5 are directly comparable.

Interpretation. W_2 is a *set-level* discrepancy: it depends only on the empirical distributions of $\{\hat{x}_i\}$ and $\{y_j\}$, not on how individual samples are matched. As a consequence, two transport plans that distribute mass identically across target clusters yield the same W_2, even if one preserves correspondences and the other permutes them. This is the central reason W_2 alone is insufficient for evaluating tasks such as XOR or Gaussian Swap, where the correctness of the mapping is precisely an identity-preserving correspondence question.

A.14.3 Transport Distance (TD)

To capture pointwise correspondence quality, we report the mean squared displacement between each transported sample and its ground-truth target:

$$\mathrm{TD} \;=\; \frac{1}{n}\sum_{i=1}^{n} \|\hat{x}_i - y_i^\star\|^2. \tag{35}$$

Interpretation. TD is computable only when ground-truth correspondences $\{(x_i, y_i^\star)\}$ are available, which is the case for all six synthetic benchmarks by construction. Unlike W_2, TD assigns blame to each individual transported sample: a method that delivers half of the mass to the wrong cluster will have large TD even if its overall distribution-level alignment, as measured by W_2, is good. TD is therefore the appropriate metric for diagnosing the trajectory-permutation failure mode that motivates our history-aware formulation.

A.14.4 Implementation Notes

Both metrics are computed in PyTorch on the same set of n transported samples per task. The Sinkhorn solver in (32)–(33) is implemented with log-sum-exp stabilization to avoid underflow in the kernel K when ε is small. For each synthetic benchmark, both VFM and the Markovian baseline are evaluated with identical sample budgets, identical ε, and identical numbers of integration steps so that all comparisons in Table 5 control for inference compute.

A.15 Extended Synthetic Experiments Visualization

This section provides additional qualitative evidence and detailed analysis supporting the synthetic benchmarks presented in the main paper (Experiments Section). While Table 5 summarizes quantitative trends, here we visualize transport trajectories across multiple representative settings where VFM provides modest improvements to FMA.

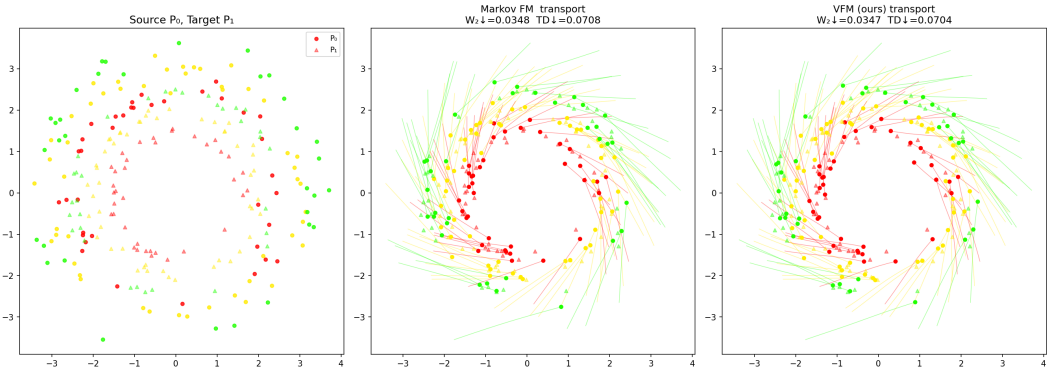


Figure 4: **Multi-ring transport with strong mode ambiguity.**

B Mathematical Proofs

This appendix collects the formal results underlying Section 3. We establish, in order: the structural equivalence between the Volterra integral memory and its first-order ODE representation (Prop. 1); the closed-form regression target induced by the stochastic-interpolant construction (Prop. 2); and the

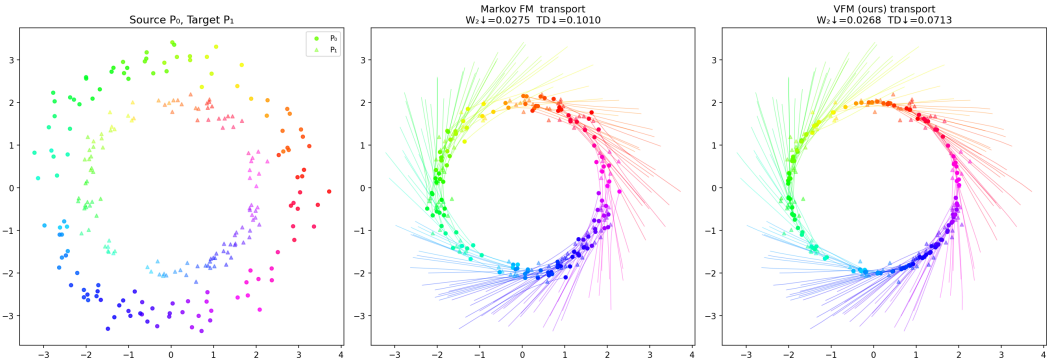


Figure 5: **A simple spiral transport from different ring sizes.**

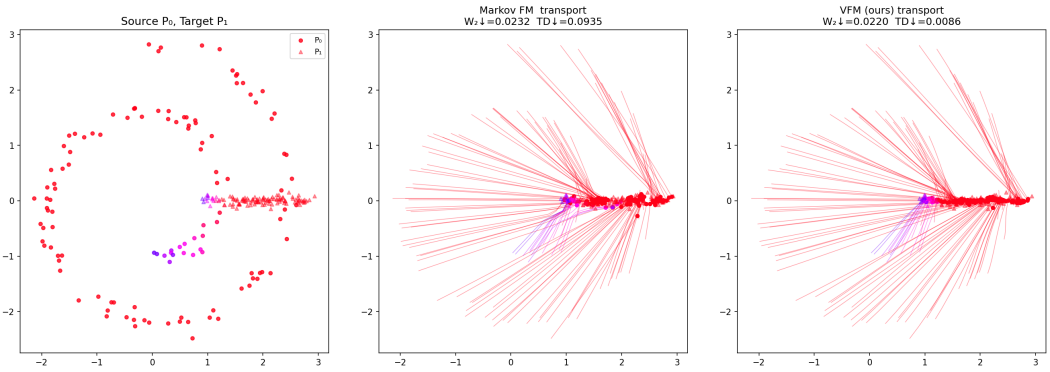


Figure 6: **Nonlinear manifold transport (Swiss-roll style deformation).**

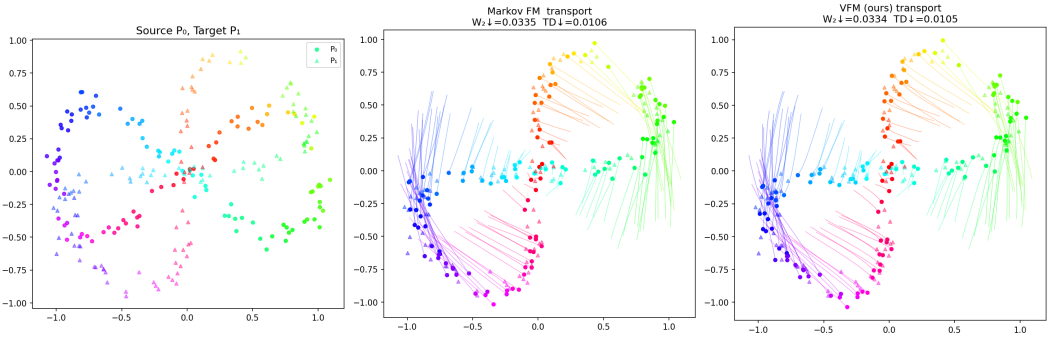


Figure 7: **Figure 8 transport with strong collapse and directional bias.**

precise sense in which the memory variable disambiguates the endpoint assignment that a Markovian velocity field cannot resolve, together with the resulting excess-risk identity (Prop. 3 and Cor. 1). All claims are component-wise in $\mathbb{R}^d$ and proved in the scalar case without loss of generality.

B.1 Preliminaries

We use three standard results, stated without proof; see any standard reference on real analysis and ODEs.

Leibniz rule. For continuous $f : [0, T]^2 \to \mathbb{R}$ with continuous $\partial_t f$,

$$\frac{d}{dt} \int_{a(t)}^{b(t)} f(t, s)\, ds \;=\; f(t, b(t))\, b'(t) - f(t, a(t))\, a'(t) + \int_{a(t)}^{b(t)} \partial_t f(t, s)\, ds.$$

Picard–Lindelöf theorem. Let $U \subseteq \mathbb{R} \times \mathbb{R}^d$ be open and $f : U \to \mathbb{R}^d$ continuous. Suppose f is *Lipschitz in its second argument* uniformly on compact subsets of U, i.e., there exists $L \geq 0$ such that $\|f(t, y) - f(t, z)\| \leq L\|y - z\|$ for all $(t, y), (t, z) \in U$. Then for every $(t_0, y_0) \in U$ the initial value problem

$$\dot{y}(t) \;=\; f(t,\, y(t)), \qquad y(t_0) = y_0$$

has a *unique* solution on some open interval containing t_0. When f is linear in y (i.e., $f(t, y) = A(t)y + b(t)$ with continuous coefficients), the Lipschitz constant is $L = \sup_t \|A(t)\|$ and the unique solution extends to the entire interval on which the coefficients are defined (Coddington et al., 1956).

Variation of parameters. The unique solution of $\dot{y}(t) + p(t)\, y(t) = q(t)$ with continuous coefficients and $y(0) = y_0$ is

$$y(t) \;=\; \mu(t)^{-1} \left(y_0 + \int_0^t \mu(s)\, q(s)\, ds \right), \qquad \mu(t) \;:=\; \exp\left(\int_0^t p(\tau)\, d\tau \right).$$

Existence and uniqueness follow from Picard–Lindelöf.

Discrete Grönwall. If $\|e_k\| \leq (1 + L\Delta t)\|e_{k-1}\| + \rho_k$ with $\sup_j \|\rho_j\| \leq \bar{\rho}$, then

$$\|e_k\| \;\leq\; e^{Lk\Delta t}\, \|e_0\| \;+\; \bar{\rho} \sum_{j=1}^{k} e^{L(k-j)\Delta t}.$$

B.2 Proposition 1: Volterra–ODE Equivalence

Proposition 1 (Volterra–ODE equivalence)**.** *Let $t \mapsto x_t \in \mathbb{R}^d$ be continuous on $[0, 1]$ and fix $\beta > 0$. Define the exponentially-weighted memory*

$$h_t \;:=\; \int_0^t e^{-\beta(t-s)}\, x_s\, ds. \tag{36}$$

Then h_t is the unique C^1 solution of

$$\dot{h}_t \;=\; -\beta\, h_t + x_t, \qquad h_0 = 0. \tag{37}$$

Conversely, every solution of (37) admits the representation (36). (Verified numerically in Section B.5.)

Proof. (36) $\Rightarrow$ (37).

Write

$$h_t = e^{-\beta t} G(t), \qquad G(t) := \int_0^t e^{\beta s} x_s\, ds.$$

By the Leibniz rule,

$$G'(t) = e^{\beta t} x_t.$$

Therefore,

$$\begin{aligned} \dot{h}_t &= -\beta e^{-\beta t} G(t) + e^{-\beta t} G'(t) \\ &= -\beta e^{-\beta t} G(t) + e^{-\beta t} e^{\beta t} x_t \\ &= -\beta h_t + x_t. \end{aligned}$$

Also, $h_0 = 0$ since the integral has zero length at $t = 0$.

(37) $\Rightarrow$ (36).

Equation (37) is linear with integrating factor

$$\mu(t) = e^{\beta t}.$$

By variation of parameters,

$$h_t = e^{-\beta t} \int_0^t e^{\beta s} x_s \, ds,$$

which is exactly (36). Uniqueness follows from Picard–Lindelöf. □

Remark 1 (Recursive implementation)**.** Proposition 1 is the structural fact that turns the Volterra memory (36) into a state with $\mathcal{O}(1)$ per-step update cost: the trajectory need not be stored, since the exponentially-weighted history is encoded exactly in h_t. This is the property exploited by the discrete updates in main-text Eqs. 10 and 13. The augmented process (x_t, h_t) is therefore Markov on $\mathbb{R}^d \times \mathbb{R}^d$, even though the marginal x_t is not.

B.3 Proposition 2: Ground-Truth Velocity Field

Proposition 2 (Velocity field for the stochastic-interpolant bridge)**.** *Fix $x_0, x_1 \in \mathbb{R}^d$, $\gamma \geq 0$, and $\epsilon \sim \mathcal{N}(0, I_d)$, and define the interpolation of main-text Eq. 3,*

$$x_t \;=\; (1-t)\, x_0 + t\, x_1 + \gamma \sqrt{t(1-t)}\, \epsilon, \qquad t \in (0,1). \tag{38}$$

Then for every fixed (x_0, x_1, ϵ), the map $t \mapsto x_t$ is C^1 on $(0,1)$ with derivative

$$v^\star(x_t, t) \;:=\; \dot{x}_t \;=\; (x_1 - x_0) + \gamma\, \frac{1-2t}{2\sqrt{t(1-t)}}\, \epsilon. \tag{39}$$

(Verified numerically in Section B.5.)

Proof. The deterministic drift term satisfies

$$\frac{d}{dt}\big((1-t)x_0 + t x_1\big) = x_1 - x_0.$$

For the stochastic term, define

$$\phi(t) := \sqrt{t(1-t)}.$$

Then, for $t \in (0,1)$,

$$\phi'(t) = \frac{1-2t}{2\sqrt{t(1-t)}}.$$

By linearity of differentiation,

$$\dot{x}_t = (x_1 - x_0) + \gamma\, \phi'(t)\epsilon,$$

which gives (39). □

Remark 2 (Regression target)**.** Although $v^\star$ depends on the unobserved ϵ, the population minimizer of the squared-error loss in main-text (11) is the conditional mean $\mathbb{E}[v^\star \mid x_t]$ (or, in our case, $\mathbb{E}[v^\star \mid x_t, h_t]$). Hence (39) is a valid stochastic regression target: the noise term averages out under the expectation, leaving only its conditional projection.

B.4 Proposition 3: Endpoint Disambiguation via Memory

We now make precise what the memory variable adds beyond the current state. The answer is *not* that h_t recovers the interpolation noise: at any fixed t the noise enters x_t and h_t as two scalar multiples of the same vector ϵ, so it supplies no second independent measurement of ϵ (Remark 3). What h_t supplies is a second *linearly independent measurement of the endpoint pair* (x_0, x_1). Because the regression target of Equation (39) is dominated by the displacement $x_1 - x_0$, and because a single intermediate state is in general consistent with many endpoint assignments, this is exactly the information a Markovian predictor lacks.

We first record the elementary risk decomposition that converts an information statement into a statement about excess risk, and then prove the identification result.

Lemma 1 (Conditioning decomposition). *Let Y be a square-integrable random vector and let $\mathcal{F}_1 \subseteq \mathcal{F}_2$ be two σ-algebras. Write $\hat{Y}_i := \mathbb{E}[Y \mid \mathcal{F}_i]$ and $\mathcal{R}_i := \mathbb{E}\|Y - \hat{Y}_i\|^2$. Then*

$$\mathcal{R}_1 \;=\; \mathcal{R}_2 \;+\; \mathbb{E}\big\|\hat{Y}_2 - \hat{Y}_1\big\|^2 \;\geq\; \mathcal{R}_2, \tag{40}$$

with equality if and only if $\hat{Y}_2$ is $\mathcal{F}_1$-measurable almost surely.

Proof. By the tower property, $\hat{Y}_1 = \mathbb{E}[\hat{Y}_2 \mid \mathcal{F}_1]$. Decompose $Y - \hat{Y}_1 = (Y - \hat{Y}_2) + (\hat{Y}_2 - \hat{Y}_1)$. The first term is orthogonal in L^2 to every $\mathcal{F}_2$-measurable random vector, by definition of conditional expectation, and the second term is $\mathcal{F}_2$-measurable; hence the cross term vanishes and Pythagoras gives (40). Equality forces $\hat{Y}_2 = \hat{Y}_1$ a.s. □

Lemma 1 applied with $\mathcal{F}_1 = \sigma(x_t, t)$ and $\mathcal{F}_2 = \sigma(x_t, h_t, t)$ shows that memory can never hurt the population risk, but it says nothing about whether the inclusion is strict. That is the content of the next proposition, which is the substantive claim.

Notation. Throughout, $\beta > 0$ is the decay rate of the exponential kernel and $t \in (0, 1]$. Define the kernel moments

$$I_\beta(t) := \int_0^t e^{-\beta(t-s)}\, ds \;=\; \frac{1 - e^{-\beta t}}{\beta}, \tag{41}$$

$$A_\beta(t) := \int_0^t e^{-\beta(t-s)}(1-s)\, ds, \qquad B_\beta(t) \;:=\; \int_0^t e^{-\beta(t-s)}\, s\, ds, \tag{42}$$

$$\Psi_\beta(t) := \int_0^t e^{-\beta(t-s)}\sqrt{s(1-s)}\, ds, \qquad \varphi(t) \;:=\; \sqrt{t(1-t)}, \tag{43}$$

and the *disambiguation function*

$$\Phi_\beta(t) \;:=\; \frac{1 - (1+\beta t)\, e^{-\beta t}}{\beta^2}. \tag{44}$$

Proposition 3 (Exact endpoint identification from the augmented state). *Fix $\beta > 0$, $\gamma \geq 0$ and $\epsilon \sim \mathcal{N}(0, I_d)$, let x_t be the interpolant of Equation (38), and let h_t be the Volterra memory of Equation (36). Then for every $t \in (0, 1]$:*

1. (Moments.) *$A_\beta(t) + B_\beta(t) = I_\beta(t)$, $B_\beta(t) = t\, I_\beta(t) - \Phi_\beta(t)$ and $A_\beta(t) = (1-t) I_\beta(t) + \Phi_\beta(t)$, and the memory decomposes as*

$$h_t \;=\; A_\beta(t)\, x_0 + B_\beta(t)\, x_1 + \gamma\, \Psi_\beta(t)\, \epsilon. \tag{45}$$

2. (Non-degeneracy.) *The linear map $(x_0, x_1) \mapsto (x_t, h_t)$ obtained by setting $\gamma = 0$ has matrix*

$$M_\beta(t) \;=\; \begin{pmatrix} 1-t & t \\ A_\beta(t) & B_\beta(t) \end{pmatrix}, \qquad \det M_\beta(t) \;=\; -\,\Phi_\beta(t) \;<\; 0. \tag{46}$$

   In particular $M_\beta(t)$ is invertible for every $t \in (0, 1]$ and every $\beta > 0$.

3. (Closed-form displacement.) *With $c_\beta(t) := \big(I_\beta(t)\varphi(t) - \Psi_\beta(t)\big)/\Phi_\beta(t)$,*

$$\frac{I_\beta(t)\, x_t - h_t}{\Phi_\beta(t)} \;=\; (x_1 - x_0) \;+\; \gamma\, c_\beta(t)\, \epsilon. \tag{47}$$

   Hence at $\gamma = 0$ the displacement $x_1 - x_0$, and therefore the target $v^\star$, is an exact *linear functional of the augmented state (x_t, h_t) at any single time $t \in (0, 1]$.*

All three parts are verified numerically in Section B.5.

Proof. (1) Moments. The identity $A_\beta + B_\beta = I_\beta$ is immediate from $(1 - s) + s = 1$. For B_β, substitute $u = t - s$:

$$B_\beta(t) = \int_0^t e^{-\beta u}(t-u)\, du = t\int_0^t e^{-\beta u} du - \int_0^t u\, e^{-\beta u} du = t\, I_\beta(t) - \frac{1 - (1+\beta t)e^{-\beta t}}{\beta^2},$$

where the last integral is evaluated by parts; this is $B_\beta = tI_\beta - \Phi_\beta$, and $A_\beta = I_\beta - B_\beta = (1-t)I_\beta + \Phi_\beta$. Substituting the interpolant (38) into the definition (36) and using linearity of the integral gives (45).

(2) Non-degeneracy. Using (1),

$$\det M_\beta(t) = (1-t)B_\beta - tA_\beta = (1-t)\big(tI_\beta - \Phi_\beta\big) - t\big((1-t)I_\beta + \Phi_\beta\big) = -(1-t)\Phi_\beta - t\Phi_\beta = -\Phi_\beta(t).$$

It remains to show $\Phi_\beta(t) > 0$ on $(0,1]$. Put $u = \beta t > 0$ and $g(u) := 1 - (1+u)e^{-u}$, so that $\Phi_\beta(t) = g(\beta t)/\beta^2$. Then $g(0) = 0$ and $g'(u) = u\,e^{-u} > 0$ for $u > 0$, hence $g(u) > 0$ for all $u > 0$. Therefore $\Phi_\beta(t) > 0$ and $\det M_\beta(t) < 0$ for every $t \in (0,1]$, $\beta > 0$.

(3) Closed-form displacement. Take the combination $I_\beta(t)x_t - h_t$ and substitute $x_t = (1-t)x_0 + tx_1 + \gamma\varphi(t)\epsilon$ together with (45). The coefficient of x_0 is $I_\beta(1-t) - A_\beta = I_\beta(1-t) - (1-t)I_\beta - \Phi_\beta = -\Phi_\beta$, and the coefficient of x_1 is $I_\beta t - B_\beta = I_\beta t - tI_\beta + \Phi_\beta = +\Phi_\beta$. The coefficient of ϵ is $\gamma\big(I_\beta\varphi - \Psi_\beta\big)$. Dividing by $\Phi_\beta(t) > 0$ gives (47). Setting $\gamma = 0$ and using $v^\star = x_1 - x_0$ from Equation (39) completes the claim. Equivalently, (47) is the second row of $M_\beta(t)^{-1}$ applied to $(x_t, h_t)^\top$ minus the first, which is well defined precisely because $\det M_\beta(t) \neq 0$ by (2). □

Corollary 1 (Excess risk of the Markovian predictor)**.** *Let $\gamma = 0$, so that $v^\star = x_1 - x_0$, and let $\mathcal{R}_{\mathrm{Markov}}$ and $\mathcal{R}_{\mathrm{VFM}}$ denote the population risks of the Bayes-optimal velocity predictors given (x_t, t) and (x_t, h_t, t) respectively. Then for every $t \in (0,1]$*

$$\mathcal{R}_{\mathrm{VFM}} \;=\; 0, \qquad \mathcal{R}_{\mathrm{Markov}} \;=\; \mathbb{E}\big[\operatorname{tr}\operatorname{Var}(x_1 - x_0 \mid x_t, t)\big] \;\geq\; 0, \tag{48}$$

and the inequality is strict if and only if the endpoint displacement is not almost surely determined by the current state, i.e. whenever the source–target assignment is ambiguous given (x_t, t) alone. A Monte Carlo confirmation on a minimal ambiguous task is given in Section B.5.

Proof. By Proposition 3(3) with $\gamma = 0$, $v^\star$ is a $\sigma(x_t, h_t, t)$-measurable function, so $\mathbb{E}[v^\star \mid x_t, h_t, t] = v^\star$ and $\mathcal{R}_{\mathrm{VFM}} = 0$. The expression for $\mathcal{R}_{\mathrm{Markov}}$ is the definition of the conditional variance of $v^\star = x_1 - x_0$ given $\sigma(x_t, t)$, and Lemma 1 identifies the gap $\mathcal{R}_{\mathrm{Markov}} - \mathcal{R}_{\mathrm{VFM}}$ with exactly this quantity. It vanishes iff $x_1 - x_0$ is $\sigma(x_t, t)$-measurable a.s. □

Remark 3 (The mechanism is deterministic, not noise recovery)**.** An earlier reading of this result attributed the benefit of h_t to recovery of the latent noise ϵ. That reading is incorrect. At any fixed t, the noise contributions to x_t and to h_t are $\gamma\varphi(t)\epsilon$ and $\gamma\Psi_\beta(t)\epsilon$ respectively: two scalar multiples of the *same* vector, not two independent directional projections. Conditioning on h_t therefore adds no information about ϵ that x_t does not already carry. What it adds is a second, linearly independent measurement of the endpoint pair, which is precisely the content of Equation (46)–Equation (47). The magnitudes confirm that the deterministic channel is the operative one: at our default $\gamma = 10^{-4}$ (Section A.12), $\beta = 1$ and $t = 0.5$, the deterministic coefficients in (45) sum to $I_\beta(0.5) \approx 0.393$ while the noise coefficient is $\gamma\Psi_\beta(0.5) \approx 1.6 \times 10^{-5}$, a ratio of about 2.5×10^4 for unit-norm CLIP features. Under the original noise-based account, VFM should have shown essentially no gain at this γ; it does, which the deterministic account predicts. Both coefficients are recomputed in Section B.5.

Remark 4 (Robustness at small $\gamma > 0$)**.** The reconstruction (47) degrades gracefully rather than discontinuously in γ: its error is exactly $\gamma\, c_\beta(t)\, \epsilon$. A short expansion of (41)–(44) at small t gives $c_\beta(t) = \frac{2}{3}t^{-1/2}\big(1 + \mathcal{O}(t)\big)$, so c_β is bounded on any interval bounded away from $t = 0$; for $\beta = 1$ one has $\sup_{t\in[0.05,1]} |c_\beta(t)| < 2.9$. At $\gamma = 10^{-4}$ the reconstruction error is thus below 3×10^{-4} in norm, three to four orders of magnitude smaller than the unit-scale displacement it estimates. Endpoint identification is therefore exact for practical purposes at the operating point used throughout the paper, and Corollary 1 applies up to an $\mathcal{O}(\gamma)$ correction. Both the asymptotic rate and the supremum bound are checked in Section B.5.

Remark 5 (Empirical corroboration on real features)**.** Equation (48) is a statement about held-out predictive risk, so it can be probed directly. We simulate interpolation trajectories from CLIP features at the paper's actual $\gamma = 10^{-4}$ and fit two ridge regressions of the ground-truth velocity $v^\star$: one on (x_t, t) and one on (x_t, h_t, t). Both are linear and identically regularized, so the comparison isolates the information content of the conditioning set rather than model capacity. Table 23 reports the held-out risks and the paired gap $\Delta = \mathcal{R}_{\mathrm{Markov}} - \mathcal{R}_{\mathrm{VFM}}$ with a bootstrap 95% confidence interval (2,000 resamples). The gap is large and its interval excludes zero on every dataset and shot setting,

consistent with the deterministic mechanism above being active at a noise level far too small for any noise-recovery account to explain. We note that this probe measures the total gap and does not by itself decompose it into deterministic and stochastic contributions.

Table 23: Held-out ridge-probe risk of the ground-truth velocity $v^\star$ regressed on (x_t, t) versus (x_t, h_t, t), at the paper's default $\gamma = 10^{-4}$ and $\beta = 1$. Lower is better; $\Delta = \mathcal{R}_{\text{Markov}} - \mathcal{R}_{\text{VFM}}$, with a bootstrap 95% CI over 2,000 resamples. CLIP-LoRA features.

Dataset	**Shots**	$\mathcal{R}_{\text{Markov}} \downarrow$	$\mathcal{R}_{\text{VFM}} \downarrow$	Δ	$\Delta\%$	**95% CI**
EuroSAT	1	0.0427	0.0129	0.0299	69.8%	[0.0284, 0.0315]
	4	0.0444	0.0132	0.0312	70.2%	[0.0296, 0.0328]
	16	0.0427	0.0126	0.0301	70.4%	[0.0286, 0.0316]
DescribableTextures	1	0.1054	0.0403	0.0651	61.7%	[0.0620, 0.0679]
	4	0.1040	0.0394	0.0646	62.1%	[0.0617, 0.0675]
	16	0.1028	0.0366	0.0662	64.4%	[0.0634, 0.0691]
Caltech101	1	0.1216	0.0736	0.0480	39.5%	[0.0457, 0.0502]
	4	0.1215	0.0725	0.0490	40.4%	[0.0467, 0.0514]
	16	0.1200	0.0721	0.0480	40.0%	[0.0458, 0.0502]
Food101	1	0.1596	0.0941	0.0654	41.0%	[0.0627, 0.0684]
	4	0.1622	0.1004	0.0618	38.1%	[0.0589, 0.0645]
	16	0.1651	0.1022	0.0629	38.1%	[0.0599, 0.0661]

Remark 6 (Scope: sufficiency, not necessity)**.** Proposition 3 shows that the Volterra memory is *sufficient* to identify the endpoint displacement; it does not show that it is necessary. In the deterministic two-endpoint case the source feature x_0 is itself available at inference, so a conditionally Markovian field $v_\theta(x_t, x_0, t)$ could resolve the same ambiguity via $x_1 - x_0 = (x_t - x_0)/t$. The advantages we claim for h_t are structural rather than informational: it is a single fixed-size state updated at $\mathcal{O}(1)$ cost that carries the whole exponentially weighted history rather than one distinguished point, it remains well defined when the realized trajectory departs from the training interpolant (as it does at inference, where x_t evolves under v_θ rather than under Equation (38)), and it degrades smoothly as β varies. Our experiments do not include a parameter-matched x_0-conditioned control, and we state this as a limitation of the present empirical support for the history-aware claim.

Remark 7 (Connection to empirical observations)**.** Corollary 1 predicts that the benefit of memory scales with the residual endpoint ambiguity $\mathrm{Var}(x_1 - x_0 \mid x_t, t)$. This is largest when few support examples constrain the source–target assignment, and shrinks as supervision pins the mapping down. It matches the empirical pattern in main-text Table 2, where memory yields its largest gains in the 1-shot regime and saturates as shots increase, and the synthetic results of Table 5, where the tasks with the largest gains (XOR, Gaussian Swap) are precisely those constructed to make the endpoint assignment ambiguous at intermediate times while leaving the marginals nearly matched.

B.5 Numerical Verification of the Formal Claims

Every claim in Section B is checked numerically by the self-contained NumPy script reproduced in Listing 1. The script requires no dependency beyond `numpy` and runs in well under a minute on CPU. Integrals are evaluated by composite trapezoid quadrature on 2×10^5 nodes (all integrands are smooth on $[0, t]$), derivatives by central differences, and the two statistical claims by Monte Carlo. Each check is swept over $\beta \in \{0.25, 0.5, 1, 2, 4\}$ and a grid of 400 values of $t \in (0, 1]$, and reports the worst-case absolute deviation over that sweep against a tolerance.

Concretely, the script verifies: the Volterra–ODE equivalence of Proposition 1, by integrating (37) forward and comparing against direct quadrature of (36); the velocity field of Proposition 2, by finite-differencing the interpolant (38); the four moment identities and the memory decomposition of Proposition 3(1); the determinant identity $\det M_\beta(t) = -\Phi_\beta(t)$ and the strict positivity of Φ_β in Proposition 3(2); the closed-form displacement recovery (47) of Proposition 3(3), at $\gamma \in$

$\{0, 10^{-4}, 10^{-1}\}$; the excess-risk identity of Corollary 1, on a minimal ambiguous task in which (x_0, x_1) is $(0, 1)$ or $(1, 0)$ with equal probability so that both assignments produce the same $x_{1/2}$; the coefficient ratio quoted in Remark 3; the asymptotics and the supremum bound on c_β quoted in Remark 4; and the contractivity of the explicit-Euler memory recursion used in Section 3.

Listing 2 reproduces the console output. All checks pass. Two numbers are worth reading off directly. First, on the ambiguous task the Markovian Bayes risk is $\mathcal{R}_{\text{Markov}} = 0.999999$ against the predicted $\mathbb{E}[\text{Var}(x_1 - x_0 \mid x_t, t)] = 1$, while the memory-based reconstruction attains $\mathcal{R}_{\text{VFM}} = 1.6 \times 10^{-9}$: the gap is the whole of the risk, exactly as Corollary 1 asserts. Second, the deterministic-to-noise coefficient ratio in h_t at $t = 0.5$ and $\gamma = 10^{-4}$ is 2.46×10^4, confirming the magnitude argument of Remark 3.

Listing 1: Verification script (`verify_claims.py`). Each function checks one formal claim of Section B and prints the worst-case deviation over the (β, t) sweep.

```
"""Numerical verification of every formal claim in Appendix B of
'Volterra Flow Matching for Non-Markovian Cross-Modal Alignment'.

Only numpy is required. Each check prints PASS/FAIL together with the
worst-case absolute deviation observed over a grid of (beta, t).

   python verify_claims.py
"""
import numpy as np

rng = np.random.default_rng(0)
GRID_T = np.linspace(1e-3, 1.0, 400) # t in (0, 1]
GRID_B = np.array([0.25, 0.5, 1.0, 2.0, 4.0]) # decay rates beta
NQ = 200_001 # quadrature nodes


# ---------------------------------------------------------------------
# quadrature helpers (composite trapezoid; the integrands are smooth)
# ---------------------------------------------------------------------
def kernel_integral(f, t, beta):
   """int_0^t exp(-beta (t - s)) f(s) ds."""
   s = np.linspace(0.0, t, NQ)
   return np.trapezoid(np.exp(-beta * (t - s)) * f(s), s)


def I_beta(t, beta): # Eq. (41): (1 - e^{-beta t}) / beta
   return (1.0 - np.exp(-beta * t)) / beta


def Phi_beta(t, beta): # Eq. (44): (1 - (1 + beta t) e^{-beta t}) / beta^2
   return (1.0 - (1.0 + beta * t) * np.exp(-beta * t)) / beta ** 2


def A_beta(t, beta):
   return kernel_integral(lambda s: 1.0 - s, t, beta)


def B_beta(t, beta):
   return kernel_integral(lambda s: s, t, beta)


def Psi_beta(t, beta):
   return kernel_integral(lambda s: np.sqrt(s * (1.0 - s)), t, beta)


def phi(t):
   return np.sqrt(t * (1.0 - t))


def report(name, err, tol):
   ok = err < tol
   print(f"[{'PASS' if ok else 'FAIL'}] {name:<58} max err = {err:.3e}")
   return ok


# ---------------------------------------------------------------------
# Proposition 1 -- Volterra integral == first-order ODE
# ---------------------------------------------------------------------
def check_prop1():
   """h_t = int_0^t e^{-beta(t-s)} x_s ds solves h' = -beta h + x_t, h_0 = 0."""
   worst = 0.0
   for beta in GRID_B:
      x = lambda s: 0.3 + 1.7 * s - 0.9 * s ** 2 + 0.4 * np.sin(5.0 * s)
```

```
      n, T = 200_000, 1.0
      dt = T / n
      h = 0.0 # explicit Euler on the ODE
      for k in range(1, n + 1):
         h = h + dt * (-beta * h + x(k * dt))
      worst = max(worst, abs(h - kernel_integral(x, T, beta)))
   return report("Prop. 1 Volterra integral == ODE solution", worst, 1e-4)


# ---------------------------------------------------------------------
# Proposition 2 -- ground-truth velocity field
# ---------------------------------------------------------------------
def check_prop2():
   """v*(x_t, t) = (x_1 - x_0) + gamma (1-2t) / (2 sqrt(t(1-t))) eps."""
   x0, x1, eps, gamma, hh = 0.7, -1.3, 0.9, 0.5, 1e-6
   xt = lambda t: (1 - t) * x0 + t * x1 + gamma * phi(t) * eps
   worst = 0.0
   for t in np.linspace(0.05, 0.95, 200):
      fd = (xt(t + hh) - xt(t - hh)) / (2 * hh) # central difference
      closed = (x1 - x0) + gamma * (1 - 2 * t) / (2 * phi(t)) * eps
      worst = max(worst, abs(fd - closed))
   return report("Prop. 2 v* equals d x_t / d t", worst, 1e-6)


# ---------------------------------------------------------------------
# Proposition 3 (1) -- kernel moment identities and memory decomposition
# ---------------------------------------------------------------------
def check_prop3_moments():
   e_sum = e_B = e_A = e_h = 0.0
   gamma = 0.3
   for beta in GRID_B:
      for t in GRID_T:
         A, B, I, P = A_beta(t, beta), B_beta(t, beta), I_beta(t, beta), Phi_beta(t, beta)
         e_sum = max(e_sum, abs(A + B - I))
         e_B = max(e_B, abs(B - (t * I - P)))
         e_A = max(e_A, abs(A - ((1 - t) * I + P)))
         if t in GRID_T[::50]: # h_t decomposition
            x0, x1, eps = rng.normal(size=3)
            h_num = kernel_integral(
               lambda s: (1 - s) * x0 + s * x1 + gamma * np.sqrt(s * (1 - s)) * eps,
               t, beta)
            h_cf = A * x0 + B * x1 + gamma * Psi_beta(t, beta) * eps
            e_h = max(e_h, abs(h_num - h_cf))
   ok = report("Prop. 3(1) A + B = I", e_sum, 1e-8)
   ok &= report("Prop. 3(1) B = t I - Phi", e_B, 1e-8)
   ok &= report("Prop. 3(1) A = (1-t) I + Phi", e_A, 1e-8)
   ok &= report("Prop. 3(1) h_t = A x0 + B x1 + gamma Psi eps", e_h, 1e-8)
   return ok


# ---------------------------------------------------------------------
# Proposition 3 (2) -- determinant of the endpoint-to-state map
# ---------------------------------------------------------------------
def check_prop3_det():
   e_det, min_Phi = 0.0, np.inf
   for beta in GRID_B:
      for t in GRID_T:
         M = np.array([[1 - t, t], [A_beta(t, beta), B_beta(t, beta)]])
         e_det = max(e_det, abs(np.linalg.det(M) + Phi_beta(t, beta)))
         min_Phi = min(min_Phi, Phi_beta(t, beta))
   ok = report("Prop. 3(2) det M = -Phi_beta", e_det, 1e-8)
   print(f"[{'PASS' if min_Phi > 0 else 'FAIL'}] "
        f"{'Prop. 3(2) Phi_beta > 0 on (0,1]':<58} min Phi = {min_Phi:.3e}")
   return ok and min_Phi > 0


# ---------------------------------------------------------------------
# Proposition 3 (3) -- closed-form displacement recovery
# ---------------------------------------------------------------------
def check_prop3_recovery():
   worst = 0.0
   for beta in GRID_B:
      for t in GRID_T[::4]:
         for gamma in (0.0, 1e-4, 0.1):
            x0, x1, eps = rng.normal(size=3)
            xt = (1 - t) * x0 + t * x1 + gamma * phi(t) * eps
            ht = A_beta(t, beta) * x0 + B_beta(t, beta) * x1 \
               + gamma * Psi_beta(t, beta) * eps
            c = (I_beta(t, beta) * phi(t) - Psi_beta(t, beta)) / Phi_beta(t, beta)
            lhs = (I_beta(t, beta) * xt - ht) / Phi_beta(t, beta)
            worst = max(worst, abs(lhs - ((x1 - x0) + gamma * c * eps)))
```

```
    return report("Prop. 3(3) (I x_t - h_t)/Phi = (x1-x0) + gamma c eps",
                  worst, 1e-8)


# ---------------------------------------------------------------------
# Corollary 1 -- excess risk of the Markovian predictor
# ---------------------------------------------------------------------
def check_corollary1():
    """Ambiguous two-endpoint task: (x0,x1) is (0,1) or (1,0) w.p. 1/2.

    At t = 1/2 both assignments give the same x_t, so the Bayes-optimal
    Markovian predictor of x1 - x0 is 0 and its risk is Var(x1-x0) = 1,
    while (x_t, h_t) separates the two branches and attains risk 0.
    """
    beta, t, gamma, n = 1.0, 0.5, 1e-4, 200_000
    A, B, I, P = A_beta(t, beta), B_beta(t, beta), I_beta(t, beta), Phi_beta(t, beta)
    branch = rng.integers(0, 2, n)
    x0, x1 = branch.astype(float), 1.0 - branch
    eps = rng.normal(size=n)
    xt = (1 - t) * x0 + t * x1 + gamma * phi(t) * eps
    ht = A * x0 + B * x1 + gamma * Psi_beta(t, beta) * eps
    target = x1 - x0

    R_markov = np.mean((target - target.mean()) ** 2) # x_t carries no signal here
    R_vfm = np.mean((target - (I * xt - ht) / P) ** 2)
    print(f" R_Markov = {R_markov:.6f} (predicted E[Var(x1-x0|x_t,t)] = 1)")
    print(f" R_VFM = {R_vfm:.3e} (predicted O(gamma^2))")
    ok = report("Cor. 1 R_Markov = E[Var(x1-x0|x_t,t)]", abs(R_markov - 1.0), 1e-2)
    ok &= report("Cor. 1 R_VFM = 0 up to O(gamma)", R_vfm, 1e-6)
    return ok


# ---------------------------------------------------------------------
# Remark 3 -- deterministic channel dominates the noise channel
# ---------------------------------------------------------------------
def check_remark3():
    beta, t, gamma = 1.0, 0.5, 1e-4
    det_coeff = I_beta(t, beta)
    noise_coeff = gamma * Psi_beta(t, beta)
    ratio = det_coeff / noise_coeff
    print(f" I_beta(0.5) = {det_coeff:.4f}, gamma Psi_beta(0.5) = {noise_coeff:.3e}, "
          f"ratio = {ratio:.3e}")
    ok = 2.0e4 < ratio < 3.0e4
    print(f"[{'PASS' if ok else 'FAIL'}] {'Rem. 3 deterministic/noise ratio ~ 2.5e4':<58}")
    return ok


# ---------------------------------------------------------------------
# Remark 4 -- size and asymptotics of the noise correction c_beta
# ---------------------------------------------------------------------
def check_remark4():
    beta = 1.0
    c = lambda t: (I_beta(t, beta) * phi(t) - Psi_beta(t, beta)) / Phi_beta(t, beta)
    ts = np.linspace(0.05, 1.0, 400)
    sup = max(abs(c(t)) for t in ts)
    asym = max(abs(c(t) / ((2.0 / 3.0) * t ** -0.5) - 1.0) for t in (1e-3, 1e-4))
    print(f" sup_[0.05,1] |c_beta| = {sup:.4f} (claim < 2.9)")
    ok = report("Rem. 4 c_beta(t) -> (2/3) t^{-1/2} as t -> 0", asym, 5e-3)
    ok &= sup < 2.9
    print(f"[{'PASS' if sup < 2.9 else 'FAIL'}] {'Rem. 4 sup bound on [0.05, 1]':<58}")
    return ok


# ---------------------------------------------------------------------
# Section 3.5 -- contractivity of the explicit-Euler memory recursion
# ---------------------------------------------------------------------
def check_euler():
    beta, dt = 1.0, 0.1
    amp = abs(1.0 - beta * dt)
    print(f" |1 - beta dt| = {amp:.2f} at the default (beta, dt) = (1, 0.1)")
    ok = amp < 1.0
    print(f"[{'PASS' if ok else 'FAIL'}] {'Sec. 3.5 memory recursion is contractive':<58}")
    return ok


if __name__ == "__main__":
    checks = [check_prop1, check_prop2, check_prop3_moments, check_prop3_det,
              check_prop3_recovery, check_corollary1, check_remark3,
              check_remark4, check_euler]
    results = [c() for c in checks]
    print("\nALL CLAIMS VERIFIED" if all(results) else "\nSOME CLAIMS FAILED")
```

Listing 2: Console output of Listing 1. All nine groups of checks pass.

```
[PASS] Prop. 1 Volterra integral == ODE solution max err = 1.146e-06
[PASS] Prop. 2 v* equals d x_t / d t max err = 1.768e-10
[PASS] Prop. 3(1) A + B = I max err = 8.181e-12
[PASS] Prop. 3(1) B = t I - Phi max err = 1.038e-11
[PASS] Prop. 3(1) A = (1-t) I + Phi max err = 2.365e-12
[PASS] Prop. 3(1) h_t = A x0 + B x1 + gamma Psi eps max err = 1.110e-16
[PASS] Prop. 3(2) det M = -Phi_beta max err = 2.365e-12
[PASS] Prop. 3(2) Phi_beta > 0 on (0,1] min Phi = 4.987e-07
[PASS] Prop. 3(3) (I x_t - h_t)/Phi = (x1-x0) + gamma c eps max err = 4.061e-09
     R_Markov = 0.999999 (predicted E[Var(x1-x0|x_t,t)] = 1)
     R_VFM = 1.648e-09 (predicted O(gamma^2))
[PASS] Cor. 1 R_Markov = E[Var(x1-x0|x_t,t)] max err = 8.649e-07
[PASS] Cor. 1 R_VFM = 0 up to O(gamma) max err = 1.648e-09
     I_beta(0.5) = 0.3935, gamma Psi_beta(0.5) = 1.602e-05, ratio = 2.456e+04
[PASS] Rem. 3 deterministic/noise ratio ~ 2.5e4
     sup_[0.05,1] |c_beta| = 2.8401 (claim < 2.9)
[PASS] Rem. 4 c_beta(t) -> (2/3) t^{-1/2} as t -> 0 max err = 9.336e-04
[PASS] Rem. 4 sup bound on [0.05, 1]
     |1 - beta dt| = 0.90 at the default (beta, dt) = (1, 0.1)
[PASS] Sec. 3.5 memory recursion is contractive

ALL CLAIMS VERIFIED
```

B.6 GRUFLOW: GRU-BASED DIRECT STATE PREDICTION AS AN ALTERNATIVE TO ODE-BASED TRANSPORT

Volterra Flow Matching (VFM) frames cross-modal alignment as a continuous-time transport problem governed by a learned velocity field, which is integrated via a fixed-step Euler ODE solver at inference time. While this formulation is theoretically grounded and computationally efficient, it raises a natural question: is the ODE-integration paradigm itself necessary, or can an equally expressive recurrent model achieve comparable alignment by directly predicting intermediate feature states? To investigate this, we design *GRUFlow*, a GRU-based direct state predictor that replaces the velocity network and ODE solver with a single recurrent architecture supervised at every discrete step of the trajectory. Unlike VFM, GRUFlow does not learn a velocity field and does not perform numerical integration; instead, it autoregressively predicts the feature state at each timestep, using its hidden state to accumulate trajectory history in a fully data-driven manner. The comparison between VFM and GRUFlow therefore isolates two orthogonal design choices: (i) *how history is encoded* (analytical Volterra kernel vs. learned GRU gating), and (ii) *what is predicted* (velocity field integrated by a solver vs. next state predicted directly).

Architecture. GRUFlow consists of three components: an input projection, a GRU recurrent core, and an AdaLN-conditioned residual decoder.

Input projection. At each discrete step $k \in \{1, \dots, K\}$ with $t_k = k \cdot \Delta t$, the current feature state $x_{t_{k-1}} \in \mathbb{R}^d$ is projected into the GRU's working dimension:

$$z_k = \mathrm{SiLU}\big(W_2\,\mathrm{SiLU}(W_1\,x_{t_{k-1}} + b_1) + b_2\big)\,, \quad z_k \in \mathbb{R}^H, \tag{49}$$

where H is the hidden dimension (defaulting to d). The timestep t_k is encoded via sinusoidal positional embeddings scaled to $[0, 1000]$, following diffusion-style parameterization, and passed through a two-layer MLP to produce a conditioning vector $e(t_k) \in \mathbb{R}^E$ ($E = 256$ by default):

$$e(t_k) = \mathrm{MLP}([\cos(\omega t_k),\, \sin(\omega t_k)])\,, \quad \omega \text{ exponentially spaced.} \tag{50}$$

GRU recurrent core. The projected feature and timestep embedding are concatenated and fed into a standard GRU:

$$\tilde{h}_k,\ \mathbf{h}_k = \mathrm{GRU}([z_k \parallel e(t_k)],\ \mathbf{h}_{k-1})\,, \quad \mathbf{h}_k \in \mathbb{R}^{L\times N\times H}, \tag{51}$$

where L is the number of stacked GRU layers and $\tilde{h}_k \in \mathbb{R}^H$ is the output at step k. The hidden state $\mathbf{h}_k$ plays the same role as the Volterra memory h_t in VFM: it encodes the accumulated trajectory history and makes the dynamics history-aware. The key difference is that VFM's memory is governed

Table 24: Comparison of VFM (ODE-based Volterra transport) against GRUFlow (GRU-based direct state prediction) on EuroSAT. All GRUFlow variants use K=10 steps with Δt=0.1. L: GRU layers; H: hidden dimension relative to feature dim d=512; dec: AdaLN decoder blocks. **Bold** denotes the best result per shot setting across all configurations.

Group	Configuration	1-shot	4-shot	16-shot
Baseline	CLIP + LoRA	73.0	85.0	91.0
	VFM (Exponential β=1.0)	82.07	88.91	92.86
GRUFlow (loss weighting)	GRUFlow L=1, H=d, loss=uniform	82.17	89.35	92.68
	GRUFlow L=1, H=d, loss=endpoint	82.12	89.40	92.79
	GRUFlow L=1, H=d, loss=linear	81.80	**89.42**	92.84
GRUFlow (depth)	GRUFlow L=1, H=d, uniform	82.17	89.35	92.68
	GRUFlow L=2, H=d, uniform	81.81	89.65	92.67
GRUFlow (decoder depth)	GRUFlow L=1, dec= 2, uniform	80.77	89.28	92.83
	GRUFlow L=1, dec= 3, uniform	82.17	89.35	92.68
GRUFlow (capacity)	GRUFlow L=1, H=d, uniform	82.17	89.35	92.68
	GRUFlow L=1, H=$2d$, uniform	80.20	**89.70**	92.81
Best GRUFlow		82.17	**89.70**	92.84
Best VFM Fractional Kernel		**82.60**	89.07	**93.0**

by a fixed exponential kernel with a single scalar parameter β, while the GRU's update rule is parameterized by its reset and update gates, learned end-to-end from data.

AdaLN residual decoder. The GRU output $\tilde{h}_k$ is passed through D Adaptive Layer Normalization (AdaLN) residual blocks conditioned on $e(t_k)$:

$$\hat{z} = \mathrm{LN}(\tilde{h}_k), \tag{52}$$

$$\hat{z} = (1 + s(e(t_k))) \odot \hat{z} + b(e(t_k)), \tag{53}$$

$$\tilde{h}_k \leftarrow \tilde{h}_k + g(e(t_k)) \odot \mathrm{MLP}(\hat{z}), \tag{54}$$

where $s(\cdot)$, $b(\cdot)$, $g(\cdot)$ are learned affine transformations of the timestep embedding, initialized to zero so that the decoder begins as a near-identity map. The final predicted state is a residual update on $x_{t_{k-1}}$:

$$\hat{x}_{t_k} = x_{t_{k-1}} + W_{\mathrm{out}}\, \tilde{h}_k, \tag{55}$$

where W_{out} is zero-initialized. The residual parameterization in Eq. (55) mirrors the near-zero velocity initialization in VFM's velocity network: at the start of training both models behave as identity maps, and both learn incremental corrections to the input features.

Training objective. GRUFlow is trained with a *direct state prediction loss* evaluated at every discrete step of the trajectory. Given image features x_0 and text features x_1, the ground-truth feature state at step k is defined by the noise-free linear interpolation ($\gamma = 0$, case in the main paper):

$$x^*_{t_k} = (1 - t_k)\, x_0 + t_k\, x_1, \quad t_k = k \cdot \Delta t, \quad k = 1, \ldots, K. \tag{56}$$

The training loss is a weighted sum of per-step mean squared errors:

$$\mathcal{L}_{\mathrm{GRU}}(\theta) = \frac{1}{K} \sum_{k=1}^{K} w_k \cdot \mathbb{E}_{x_0, x_1} \left\| \hat{x}_{t_k} - x^*_{t_k} \right\|^2, \tag{57}$$

where w_k are step-level weights normalized to sum to one. We evaluate three weighting strategies:

- *Uniform*: $w_k = 1/K$ for all k. Equal supervision at every interpolation step.

- *Endpoint*: $w_K = 2/(K+1)$, $w_k = 1/(K+1)$ for $k < K$. Extra emphasis on terminal alignment, most directly relevant to downstream classification.
- *Linear*: $w_k \propto k$. Progressively increasing weight, reflecting that errors in later steps are less correctable than errors in early steps.

This contrasts with VFM's velocity regression loss $\mathcal{L}_{\text{VFM}} = \mathbb{E}\|v_\theta(x_t, h_t, t) - v^*(x_t, t)\|^2$, which provides supervision at a single randomly sampled time t per training step. GRUFlow's loss in Eq. (57) provides dense supervision across the *entire* trajectory at every training step, which in principle gives richer gradient signal at the cost of a longer computational graph via backpropagation through time (BPTT).

Closed-loop rollout. During training, the predicted state $\hat{x}_{t_k}$ is fed back as input at the next step (Eq. (55)), matching inference behavior. This closed-loop training avoids the exposure bias that would arise from always feeding ground-truth states as inputs (teacher forcing), at the cost of a longer autograd graph. Gradient clipping at norm 1.0 is applied to stabilize BPTT through 10 unrolled steps.

Inference. At inference, GRUFlow is unrolled autoregressively for K steps starting from the image feature x_0, with the GRU hidden state initialized to zero:

$$\hat{x}_{t_k}, \mathbf{h}_k = \text{GRUFlowCell}(\hat{x}_{tk-1}, \mathbf{h}_{k-1}, t_k), \quad k = 1, \ldots, K. \tag{58}$$

The final refined feature $\hat{x}_{t_K}$ is ℓ_2-normalized and used for nearest-centroid classification, identical to VFM inference. We sweep $K \in \{0, 1, \ldots, 10\}$ and report the best result, following the same protocol as VFM.

Initialization strategy. Three design choices ensure stable training from random initialization. First, orthogonal initialization of the GRU recurrent weight matrices W_{hh} preserves gradient norms through the unrolled steps, preventing vanishing or exploding gradients in early training. Second, zero-initialization of the output projection W_{out} and the AdaLN modulation layers ensures that GRUFlow begins as a near-identity map, so the initial refined features are close to the raw image features and classification does not collapse at the start of training. Third, Xavier uniform initialization is used for all feedforward projections. These choices mirror VFM's initialization strategy, making the two models comparably well-conditioned at the start of training.

Experimental results. Table 24 reports GRUFlow accuracy on EuroSAT across all shot settings and architectural variants, alongside the CLIP+LoRA baseline and VFM.

Takeaway. GRUFlow demonstrates that the Volterra ODE formulation in VFM is not uniquely responsible for its performance gains: a learned GRU with direct state supervision achieves comparable accuracy across all shot settings. This equivalence supports a broader interpretation of VFM's success: what matters is the introduction of a history-aware inductive bias into the feature transport process, regardless of whether that bias is encoded analytically via a Volterra integral or learned implicitly via GRU gating. We nonetheless adopt the ODE-based VFM formulation as our primary method for two reasons. First, VFM is more stable in the 1-shot regime, where GRUFlow's accuracy varies by up to 2% depending on hidden dimension. Second, VFM's memory admits an exact analytical characterization: by Proposition 3 the augmented state identifies the endpoint displacement in closed form, so the mechanism responsible for the gain can be stated and checked, which a learned GRU state does not permit. GRUFlow is presented here as an empirical validation that VFM's history-aware structure, rather than its specific ODE formulation, is the key contributor to improved cross-modal alignment.

B.7 Comparison to Concurrent Flow-Based Approaches

Recent unpublished work such as *CaLMFlow* He et al. (2024) explores flow-based formulations conditioned on large language models (not explicitly for cross modal alignment and few shot learning). While this line of work is conceptually interesting in leveraging high-level semantic priors, its treatment of dynamics differs fundamentally from our approach.

Conceptual distinction. CaLMFlow formulates flow matching transport by leveraging large language models (LLMs) to implicitly approximate or solve integral equations governing trajectory evolution. In contrast, VFM directly derives a structured dynamical system, where the memory process is explicitly defined and coupled with the state evolution. Our implementation of GRUFLow B.6 can be interpreted an approximate instantiation of CaLMFlow, applied to few shot learning.

Dynamical clarity vs. implicit mechanisms. Our formulation leads to:

- An explicit state-space representation of memory,
- A coupled ODE system with well-defined stability and computational properties,
- A clear connection to classical integral equations (Volterra systems).

By comparison, approaches that rely on LLM-based or architectural approximations of integral operators do so in an implicit manner, making the underlying temporal structure less directly interpretable.

Formulation clarity and structure. While prior work explores related high-level ideas, the formulation and exposition of the underlying dynamics are sometimes presented through flexible modeling components (e.g., sequence models or LLM-based solvers), which can make it less straightforward to analyze stability, interpretability, and computational trade-offs. In contrast, our approach is grounded in a principled dynamical systems perspective, where the integral operator admits an exact low-order realization, enabling both theoretical clarity and practical efficiency.

Takeaway. Overall, VFM provides a *clean, minimal, and mathematically grounded* framework for incorporating memory into flow-based models.

This stands in contrast to approaches that approximate this behavior through more implicit or architectural means, and highlights the advantages of designing memory directly at the level of the dynamical system.