VOLT: Token-Level Vector Critics with KAN-LoRA and Liquid-Time LoRA for Test-Time Learning

Anonymous ACL submission

Abstract

Test-time learning that leverages feedback during inference promises to close the gap between pretraining and a model's immediate deployment needs. We present VOLT, a test-time learning framework that issues dense, token-level vector evaluations and turns them into stable, parameter-efficient updates via a Kolmogorov–Arnold Network parameterization of LoRA (KAN-LoRA) together with a Liquid-Time Neural-ODE flavored LoRA (LTM-LoRA). A frozen token-level vector critic issues high-dimensional per-token signals which are projected to scalar advantages and then used to compute compact low-rank updates that evolve either as spline-based factor maps or as continuous trajectories. VOLT integrates critic-guided, token-level supervision with parameter-efficient update families to improve reasoning and code generation under constrained test-time budgets. Our code is available at: `https://anonymous.4open.science/r/tempvo-B4AD/README.md`

Keywords: Test-time learning, token-level critics, LoRA, Kolmogorov–Arnold networks, Neural ODEs, parameter-efficient adaptation

1 Introduction

Leveraging feedback at inference time has emerged as a powerful strategy to boost the reliability and problem-solving performance of large language models. Recent studies demonstrate that extra computation at test time or model adaptation guided by verification signals can substantially improve results on demanding reasoning and code generation tasks (Snell et al., 2024; Welleck et al., 2024; Tang et al., 2025; Verdun et al., 2025).

Two main paradigms currently dominate test-time improvement. Revision-based methods iteratively fold previous attempts back into the context to accumulate corrections, yet they suffer from rapidly growing context lengths and pronounced positional sensitivity (Wan et al., 2025; Lin et al., 2025). Parallel sampling approaches, on the other hand, generate multiple independent candidates and select the best via a verifier. While effective for exploration, they typically discard rich per-token error signals and provide no mechanism for learning from earlier failures (Wu et al., 2024, 2025b). Complementary efforts on parameter-efficient adaptation such as LoRA enable compact specialization but often sacrifice expressivity for nonlinear corrections in activation space, while continuous-time formulations introduce solver stability issues unless carefully regularized. Overall, prior work rarely unites dense token-wise evaluators with structured, parameter-efficient update families capable of modeling nonlinear corrections while remaining compact and numerically stable.

To address these limitations, we propose VOLT, a hybrid test-time learning framework. It integrates three core elements: a frozen token-level *vector* critic that issues dense, high-dimensional per-token signals capturing nuanced local evaluations beyond scalar scores; *KAN-LoRA*, a Kolmogorov–Arnold-inspired low-rank adapter that represents updates as sums of univariate spline factor maps for enhanced approximation power with controlled parameter count; and *Liquid-Time LoRA* (LTM-LoRA), a continuous-time formulation for low-rank factors realized through Neural ODE dynamics with stability-aware regularization and practical compute accounting. During inference, the critic vectors are projected into scalar advantages via a differentiable head, which then guide compact updates within the KAN-LoRA or LTM-LoRA subspaces. VOLT bridges sampling-heavy best-of-n strategies and full fine-tuning by reusing generated attempts and issuing critic-guided updates evaluated by verifiers, without requiring human labels.

Unlike active fine-tuning approaches such as SIFT that retrieve external samples for sparse layer updates, VOLT performs dense token-level credit

assignment directly inside structured low-rank subspaces, enabling efficient gradient-based adaptation without repeated forward passes on stored data.

Our contributions are as follows. First, we introduce VOLT, a unified test-time learning framework that pairs a frozen token-level vector critic with expressive yet compact update families to achieve stable, parameter-efficient adaptation. Second, we propose KAN-LoRA, a Kolmogorov–Arnold-inspired low-rank adapter that represents weight updates as sums of univariate spline factor maps, accompanied by approximation guarantees and an identity-plus-small-spline initialization for early stability. Third, we design Liquid-Time LoRA, a continuous-time low-rank adapter formulation that models adapter factors as Neural ODE trajectories, together with practical solver guidelines and Jacobian regularization to ensure stability. Finally, we evaluate VOLT on mathematical reasoning and code generation benchmarks, demonstrating consistent gains over both test-time scaling baselines and various PEFT competitors, supported by ablations isolating the contributions of the vector critic, KAN parameterization, and continuous-time dynamics.

2 Related Work

2.1 Test-time Adaptation and Online Learning for LLMs

Test-time adaptation aims to reduce the impact of distribution shift by conducting lightweight updates during inference. A growing body of work treats this as an online self-supervised optimization problem, with particular attention to sample selection strategies, perplexity-based objectives, and parameter-efficient mechanisms suited to real-world deployment constraints (Hu et al., 2025; Deng et al., 2025; Bao et al., 2025). To address instability from noisy unsupervised signals, several techniques introduce meta-learned optimizers or gradient generators that produce more reliable update directions (Deng et al., 2025; Xu et al., 2025a). These ideas have been extended to vision-language models and federated learning scenarios, often through memory buffers or prototype sharing to manage limited local data (Bao et al., 2025).

In this landscape, SIFT (Hübotter et al., 2025) employs transductive active learning to select diverse examples from an external corpus for test-time fine-tuning, achieving notable gains on language modeling benchmarks. VOLT departs from this retrieval-based paradigm by relying on an internal frozen vector critic to generate dense token-level feedback and by restricting updates to compact structured subspaces realized through KAN-LoRA and LTM-LoRA, making it especially well-suited for reasoning and code generation tasks where external data access may be limited or impractical.

2.2 Parameter-Efficient Adaptation and LoRA Extensions

Parameter-efficient fine-tuning has become the prevailing strategy for rapid specialization of large language models. LoRA and related low-rank adapters limit updates to compact subspaces, significantly lowering memory and computational demands (Kojima et al., 2025; Meral et al., 2025; Chen et al., 2025a). More recent developments seek greater expressivity by adopting structured or nonlinear low-dimensional parameterizations such as spline-based mappings, mixture-of-experts adapters, and compositional factors (Bo et al., 2025; Rahman et al., 2025; Chen et al., 2025a). KAN-LoRA extends this direction by incorporating Kolmogorov–Arnold factorization principles, combining univariate spline factor maps with the LoRA framework to achieve a better trade-off between efficiency and approximation capability (Bo et al., 2025; Rahman et al., 2025).

2.3 Token-level Credit Assignment and Vector-valued Critics

When small token modifications can substantially affect final outcomes, fine-grained credit assignment becomes essential. Reinforcement learning and policy-based methods have introduced per-token objectives and softened Bellman updates to enable tractable optimization in large action spaces (Wen et al., 2024; Yang et al., 2023b; Tran et al., 2025). Complementary efforts focus on redistributing sequence-level rewards into richer token-wise signals (Li et al., 2025a; Yang et al., 2023b). Learned critics that output high-dimensional vector evaluations, which are subsequently projected to scalar advantages, provide dense supervision for policy-gradient-style updates. VOLT adopts this vector-to-scalar design while incorporating conservative regularization to promote stable adaptation (Wen et al., 2024; Li et al., 2025a).

2.4 Continuous-time Modeling and Neural ODE Approaches

Continuous-time formulations of parameter dynamics support smooth adaptation and permit efficient gradient computation through the adjoint method. Neural ODEs and continuous-time transformer variants enable memory-efficient trajectory modeling, complemented by liquid-time networks that offer temporally adaptive behavior transferable across different regimes (Chen et al., 2023; Zhang and Zhou, 2025; Xu et al., 2025b; Quach et al., 2024). LTM-LoRA draws on these foundations by evolving low-rank adapter factors according to learned vector fields and leveraging adjoint sensitivity analysis for gradient propagation during test-time updates (Xu et al., 2025b; Zhang and Zhou, 2025).

2.5 Inference-time Compute Allocation, Scaling Laws, and Lifelong Adaptation

An orthogonal direction for performance improvement lies in the strategic allocation of additional inference-time computation. Scaling-law studies and compute-optimal analyses explore trade-offs between model size and various test-time strategies, including sampling, ensembling, and structured search (Wu et al., 2024, 2025b; Huang et al., 2025; Geng et al., 2025). While advanced inference algorithms can allow smaller models to compete with larger ones, naive best-of-n approaches are prone to issues such as reward hacking (Huang et al., 2025; Wu et al., 2024). VOLT bridges sampling-based scaling and parameter-efficient fine-tuning by employing a learned token-level critic to guide compact, structured updates rather than relying exclusively on large generation budgets.

Meta-learning and learned optimizers further enhance fast adaptation and in-context generalization by distilling compact update rules that transfer across domains (Sinha et al., 2024; Xu et al., 2025a; Yang et al., 2023a). Surveys of post-training practices underscore the value of integrating fine-tuning, test-time adaptation, and verification mechanisms to mitigate catastrophic forgetting and reward misalignment (Kumar et al., 2025; Liu et al., 2025b; Chen et al., 2025b). Robust evaluation depends on formal reasoning benchmarks, step-aware verifiers, and long-context protocols that inform the design of metrics and critics (Yu et al., 2025; Liu et al., 2025a,b). Recent work also bridges representational advances, optimizer design, and lifelong learning to support stable post-deployment adaptation while preserving prior knowledge (Chen et al., 2025a; Wu et al., 2025a; Rahman et al., 2025; Kan et al., 2025; Zhang and Zhou, 2025).

3 Methodology

VOLT is a test-time learning framework that integrates three tightly coupled components: a token-level vector critic that supplies dense per-token feedback, KAN-LoRA that parameterizes low-rank updates through univariate spline factor maps, and LTM-LoRA that models low-rank factors as continuous trajectories optimized via Neural ODE adjoints. This section formalizes the sequential generation process, describes the projection of critic vectors into scalar advantages, establishes norm-specific approximation guarantees for KAN-LoRA, and analyzes the computational characteristics of LTM-LoRA.

3.1 Synergistic Design of VOLT

The three modules of VOLT are engineered to reinforce one another within a unified optimization manifold. Spline factorization in KAN-LoRA promotes structural regularity of the update space, while Neural-ODE regularization in LTM-LoRA enforces smooth parameter trajectories and stable adjoint computations. In turn, the token-level vector critic generates signals naturally aligned with this manifold, allowing projected scalar advantages to serve as effective control inputs along the learned paths. This closed-loop co-design distinguishes VOLT from prior approaches that apply spline adapters, continuous-time LoRA variants, or token-level critics in isolation.

3.2 Problem Formulation

We model token generation as a sequential decision process. Let π_θ denote the policy that generates token a_t conditioned on prompt x and prefix state s_t. A complete generation attempt is the sequence $y = (a_0, \ldots, a_T)$, where T is the length of the attempt. The frozen token-level vector critic produces a high-dimensional evaluation vector at each step:

$$A_t^{\text{critic}} = C_\phi(a_t \mid x, s_t) \in \mathbb{R}^r, \tag{1}$$

where C_ϕ is the critic network with parameters ϕ, and r is the output dimensionality.

A differentiable projection converts these vectors into scalar advantages:

$$\widehat{A}_t = g\big(A_t^{\text{critic}}; w_g\big) - V_{\text{base}}(s_t), \tag{2}$$

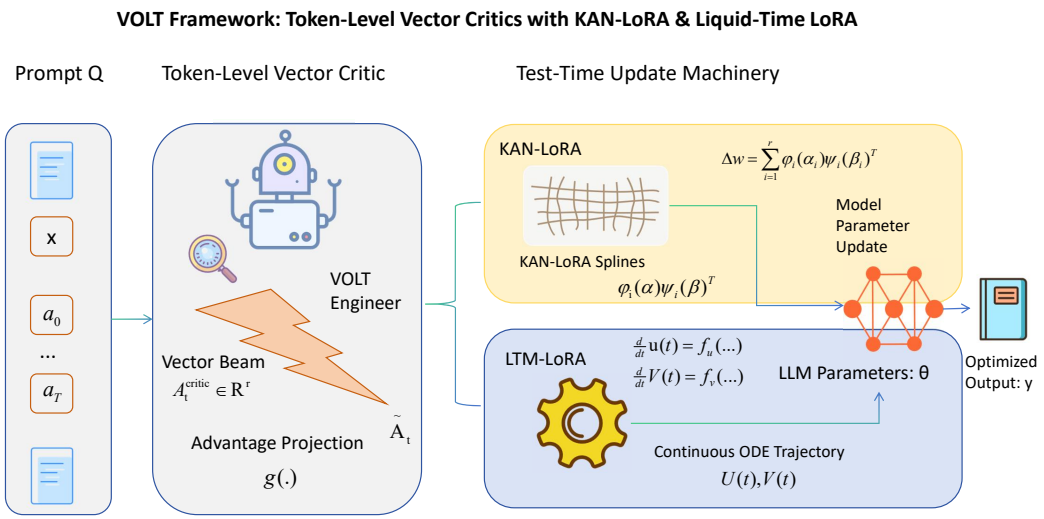

Figure 1: Overview of the VOLT framework for test-time learning. Given an input prompt Q and a sequence of generated tokens, a frozen **Token-Level Vector Critic** issues dense, r-dimensional signals A_t^{critic}. These signals are processed by a differentiable **Advantage Projection** $g(\cdot)$ to yield scalar advantages $\widehat{A}_t$. The scalar signals guide the **Test-Time Update Machinery**, which computes parameter updates ΔW using either univariate spline factor maps (**KAN-LoRA**) or continuous trajectories modeled via Neural ODEs (**LTM-LoRA**), dynamically refining the LLM parameters θ to produce an optimized output.

where w_g parameterizes the projection function $g : \mathbb{R}^r \to \mathbb{R}$ and $V_{\text{base}}(s_t)$ is a learned baseline value function. The projection g is constrained to be L-Lipschitz to bound advantage magnitudes. These scalar advantages drive a policy-gradient-style parameter increment:

$$\Delta W = \eta \sum_{t=0}^{T} \nabla_W \log \pi_\theta(a_t \mid s_t) \cdot \widehat{A}_t, \qquad (3)$$

where $\eta > 0$ is the step size and W denotes the subset of parameters receiving the test-time update. We assume a uniform gradient bound G exists such that

$$\left\|\nabla_\theta \log \pi_\theta(a \mid s)\right\| \leq G \qquad (4)$$

for all tokens a and states s in the support of π_θ.

Proposition 3.1 (Bounded Test-Time Update)**.** *If g is L-Lipschitz and the gradient bound (4) holds, then the operator norm of the induced update satisfies*

$$\|\Delta W\| \leq \eta\, L\, G\, T \max_t \|A_t^{\text{critic}}\|, \qquad (5)$$

where $\|\cdot\|$ denotes the operator norm.

Update construction and hierarchical structure. Equation (3) defines the general policy-gradient increment in the full parameter space. In VOLT, this direction is projected onto structured low-dimensional subspaces. KAN-LoRA (Equation (8)) realizes the projection onto a spline-based manifold, while LTM-LoRA (Equations (11)–(12)) realizes the projection onto a continuous trajectory manifold. Consequently, the gradient $\nabla_W \log \pi_\theta$ in Equation (3) is computed with respect to the compact KAN or LTM parameters (such as spline coefficients or ODE vector-field weights) rather than the full weight matrix, thereby preserving parameter efficiency.

3.3 Token-Level Vector Critic and Offline Pretraining

The critic is aligned with temporal-difference learning through the one-step TD error:

$$\delta_t = R_t + \gamma V(s_{t+1}) - V(s_t), \qquad (6)$$

where R_t is the immediate reward at step t and $\gamma \in [0, 1]$ is the discount factor. To prevent dimensional collapse while regressing toward scalar targets, the training objective includes a diversity regularizer:

$$\mathcal{L}_{\text{critic}} = \mathbb{E}\left[\left\|A_t^{\text{critic}} - \delta_t \mathbf{1}\right\|^2\right] + \lambda_{\text{div}} \cdot \mathcal{R}_{\text{div}}(A_t^{\text{critic}}), \qquad (7)$$

where $\mathbf{1} \in \mathbb{R}^r$ is the all-ones vector and $\mathcal{R}_{\text{div}}$ encourages informative variation across output dimensions through per-dimension ℓ_2 penalties and

multi-head projections. This formulation maintains a healthy effective rank of approximately 38 out of 64 dimensions. The critic is implemented as a 768-dimensional MLP with a 64-dimensional output head employing multi-head projections and per-dimension ℓ_2 regularization. It is pretrained offline on synthetic error-augmented trajectories (10K steps, batch size 32, approximately 4.2 GPU-hours on A100) and remains frozen during test-time adaptation.

3.4 KAN-LoRA: Structured Factorization and Approximation Guarantees

KAN-LoRA represents the low-rank update as a sum of outer products of univariate spline maps:

$$\Delta W_{\mathrm{KAN}} = \sum_{i=1}^{r} \varphi_i(\alpha_i)\, \psi_i(\beta_i)^\top, \tag{8}$$

where φ_i and ψ_i are learnable B-spline factor maps, and α_i, β_i are scalar projections derived from pooled activations and gradient summaries.

For any Lipschitz function $f : [-1, 1] \to \mathbb{R}$ with constant $\mathrm{Lip}(f)$, there exists a zero-order B-spline φ with K segments such that

$$\sup_{x\in[-1,1]} |f(x) - \varphi(x)| \le CK^{-1}\mathrm{Lip}(f), \tag{9}$$

where C is a universal constant determined by the spline basis. Early training stability is promoted by an identity-plus-small-spline initialization:

$$\varphi_i(\alpha) = \alpha + \epsilon \cdot \mathrm{Spline}(\alpha), \quad \epsilon = 0.01, \tag{10}$$

where $\mathrm{Spline}(\cdot)$ denotes the B-spline basis with small initial coefficients. Under the Frobenius and spectral norms, KAN-LoRA can outperform standard LoRA when the spline resolution K exceeds a norm-specific threshold, as formally shown in Appendix A.

3.5 LTM-LoRA: Continuous Dynamics, Compute, and Stability

LTM-LoRA models the low-rank factors as continuous-time trajectories:

$$\frac{d}{dt}U(t) = f_U\big(U(t), \mathcal{G}(t); \psi_U\big), \tag{11}$$

$$\frac{d}{dt}V(t) = f_V\big(V(t), \mathcal{G}(t); \psi_V\big), \tag{12}$$

where $\mathcal{G}(t)$ aggregates critic-derived control signals and f_U, f_V are parameterized vector fields. When an ODE solver uses N_t internal sub-steps to integrate the trajectories over one attempt, the approximate floating-point operation count is

$$\mathrm{FLOPs} = 4N_t r d^2, \tag{13}$$

where d is the feature dimension and r is the LoRA rank. The additional memory requirement (in floating-point entries) is

$$\mathrm{Mem}_{\mathrm{entries}} = 6rd + 4N_t rd, \tag{14}$$

where the first term covers current factors and auxiliary buffers, and the second term accounts for forward checkpoints required during adjoint integration. To improve numerical stability, a Jacobian regularization term is added:

$$\mathcal{L}_{\mathrm{ode}} \leftarrow \mathcal{L}_{\mathrm{ode}} + \lambda \mathbb{E}\Big[\big\|\partial f_U/\partial U\big\|_F + \big\|\partial f_V/\partial V\big\|_F\Big], \tag{15}$$

where $\lambda > 0$ is the regularization coefficient, f_U and f_V are the learned vector fields, and $\|\cdot\|_F$ denotes the Frobenius norm. A discrete baseline update is retained for comparison:

$$U_{k+1} = U_k + \eta G_k, \qquad V_{k+1} = V_k + \eta H_k, \tag{16}$$

where U_k, V_k are the low-rank factors at step k, G_k, H_k are the corresponding gradient estimates, and η is the step size.

When does LTM-LoRA help? LTM-LoRA does not universally outperform KAN-LoRA; its benefits are task- and budget-dependent. At attempt budget $N = 32$, LTM-LoRA achieves +0.3–1.0% higher accuracy than KAN-LoRA on MBPP when using the dopri5 solver with 20 steps, at roughly double the latency. We recommend KAN-LoRA as the default for resource-constrained deployments and LTM-LoRA for accuracy-critical scenarios where additional compute is affordable.

3.6 Training Recipe and Implementation Details

The critic C_ϕ is pretrained offline using the regularized TD objective in Equation (7) and kept fixed thereafter. KAN spline coefficients follow the identity-plus-small-spline initialization and receive ℓ_2 regularization. For LTM-LoRA we adopt modest ranks ($r \le 8$) and conservative solver tolerances. Scalar projections yielding α_i and β_i are normalized to unit variance prior to spline evaluation. In deployment, KAN-LoRA serves resource-limited settings while LTM-LoRA is reserved for accuracy-sensitive cases, both sharing the same critic-guided signals.

3.7 Algorithmic Flow

The complete VOLT procedure appears in Algorithm 1. At every iteration the system samples a candidate, verifies its quality, and, if needed, applies a critic-guided update in either the KAN-LoRA or LTM-LoRA subspace. The frozen vector critic remains constant across attempts, while the choice of update parameterization balances the current computational budget against desired performance.

Algorithm 1: VOLT: Critic-guided test-time adaptation with KAN-LoRA or LTM-LoRA

Input: Prompt Q, model parameters θ, pretrained frozen critic C_ϕ, KAN spline coefficients, optional ODE modules, attempt budget N

Output: Best valid attempt

```
for n ← 1 to N do
    Sample attempt y^(n) ~ π_θ(· | Q);
    Evaluate verifier V(y^(n));
    if verifier indicates success then
        return current valid attempt;
    foreach token a_t in y^(n) do
        Compute A_t^critic ← C_φ(a_t | Q, s_t) via (1);
    Compute scalar advantages Â_t via (2);
    Extract projections {α_i, β_i} from activations and gradients;
    Form KAN-LoRA update ΔW_KAN via (8);
    Apply increment from (3);
    if LTM-LoRA enabled then
        Integrate trajectories using (11)–(12);
        Update ODE parameters under regularization (15);
    else
        Apply θ ← θ + ΔW_KAN;
return best attempt within budget N;
```

4 Experiments

We evaluate VOLT on mathematical reasoning and code generation benchmarks, comparing it with test-time scaling and PEFT baselines, and analyzing design choices via ablations and case studies. The protocol follows recent test-time scaling setups with practical defaults for large models and constrained budgets. As shown in Section N and Table 16, VOLT achieves strong cross-task transfer, indicating generalizable update patterns. Section L and Table 4 illustrate VOLT's superior ability to correct intermediate reasoning errors using token-level critics that project informative signals into stable scalar advantages. Appendix B provides comparisons with SIFT, critic pretraining costs, and component synergy analysis (Table 6). The computational overhead of VOLT's test-time adaptation is characterized in Appendix D, where KAN-LoRA is identified as the default adapter for resource-constrained deployments and LTM-LoRA is recommended for accuracy-critical batch scenarios that permit additional per-attempt latency. The cross-model portability of VOLT is further substantiated in Appendix E, where the frozen critic transfers zero-shot to seven diverse backbones spanning Llama, Mistral, Qwen, Deepseek, Yi, and OpenCoder architectures.

4.1 Setup

We evaluate VOLT on four established benchmarks that probe mathematical reasoning and code generation. The math datasets are MATH(Hendrycks et al., 2021) and GSM8K(Cobbe et al., 2021), and the code datasets are MBPP(Austin et al., 2021) and HumanEval(Chen, 2021). For MATH we use the test split specified in prior work. For each dataset we evaluate on the subset of examples where the base model fails under greedy decoding to focus on cases that can benefit from test-time refinement. Exact Match is used for math tasks (with the verifier as the judge) and Pass@1 is used for code. All experiments use two base model families: Llama-3.1-8B-Instruct(Dubey et al., 2024) and Mistral-7B-Instruct-v0.3(Chaplot, 2023). Models are run in zero-shot mode following their official evaluation instructions. Sampling-based baselines use nucleus sampling with temperature 0.6 and $p = 0.95$. Every method is given a fixed attempt budget; unless otherwise noted we use a budget of $N = 32$. For stochastic algorithms we report the mean over three independent random seeds with the standard deviation shown as a subscript. Training and implementation details. KAN spline grids use an identity-plus-small-spline initialization and spline coefficients are ℓ_2-regularized. For LTM-LoRA we recommend ranks $r \leq 8$ and conservative solver tolerances; the Jacobian regularizer default is $\lambda = 10^{-4}$. Importantly, the critic C_ϕ is

Table 1: VOLT and baseline results on four benchmarks with budget $N = 32$. For stochastic methods we report mean of three runs with standard deviation in subscript. Bold entries indicate the best result in each block

Method	MATH	GSM8K	MBPP	HumanEval	Avg.
Llama-3.1-8B-Instruct					
Revision(Snell et al., 2024)	0.2960	0.4570	0.2991	0.3889	0.3603
Beam Search(Ow and Morton, 1988)	0.2560	0.4842	0.1869	0.2407	0.2920
Self-Consistency(Wang et al., 2022)	$0.3000_{0.0057}$	$0.4525_{0.0111}$	$0.1371_{0.0180}$	$0.1235_{0.0531}$	0.2533
Self-Refine(Madaan et al., 2023)	$0.4693_{0.0207}$	$0.7828_{0.0064}$	$0.2305_{0.0088}$	$0.2963_{0.0800}$	0.4447
Best-of-N(Brown et al., 2024)	$0.6427_{0.0154}$	$0.8069_{0.0043}$	$0.5452_{0.0154}$	$0.6728_{0.0087}$	0.6669
FTTT(Li et al., 2025b)	$0.6707_{0.0222}$	$0.8100_{0.0037}$	$0.5607_{0.0212}$	$0.6852_{0.0302}$	0.6817
FTTT w/o self-reflection	$0.6720_{0.0113}$	$0.8054_{0.0133}$	$0.5405_{0.0117}$	$0.6790_{0.0087}$	0.6742
VOLT (Ours)	$\mathbf{0.7007}_{0.0200}$	$\mathbf{0.8400}_{0.0030}$	$\mathbf{0.5907}_{0.0200}$	$\mathbf{0.7152}_{0.0280}$	**0.7117**
Mistral-7B-Instruct-v0.3					
Revision(Snell et al., 2024)	0.0497	0.1686	0.1351	0.1000	0.1134
Beam Search(Ow and Morton, 1988)	0.1783	0.4537	0.1318	0.1600	0.2310
Self-Consistency(Wang et al., 2022)	$0.1618_{0.0011}$	$0.3549_{0.0052}$	$0.1243_{0.0135}$	$0.1033_{0.0236}$	0.1861
Self-Refine(Madaan et al., 2023)	$0.1287_{0.0139}$	$0.3312_{0.0078}$	$0.2347_{0.0091}$	$0.3533_{0.0309}$	0.2620
Best-of-N(Brown et al., 2024)	$0.4688_{0.0138}$	$0.7807_{0.0055}$	$0.4962_{0.0171}$	$0.6500_{0.0141}$	0.5989
FTTT(Li et al., 2025b)	$0.4733_{0.0087}$	$0.7820_{0.0045}$	$0.4962_{0.0015}$	$0.6633_{0.0125}$	0.6037
FTTT w/o self-reflection	$0.4876_{0.0133}$	$0.7858_{0.0021}$	$0.4941_{0.0040}$	$0.6833_{0.0205}$	0.6127
VOLT (Ours)	$\mathbf{0.5033}_{0.0080}$	$\mathbf{0.8120}_{0.0040}$	$\mathbf{0.5262}_{0.0010}$	$\mathbf{0.7133}_{0.0110}$	**0.6387**

fixed after offline pre-training; no test-time parameter update of the critic is performed. All projection compressions that produce α_i, β_i are normalized to unit variance before spline evaluation.

4.2 Baselines

We compare VOLT to representative test-time scaling strategies and to PEFT fine-tuning baselines. Test-time scaling baselines include Revision(Snell et al., 2024), Beam Search(Ow and Morton, 1988), Self-Consistency(Wang et al., 2022), Self-Refine(Madaan et al., 2023), FTTT(Li et al., 2025b) and Best-of-N(Brown et al., 2024). PEFT baselines for fine-tuning comparisons include Adapter(Houlsby et al., 2019), $(IA)^3$(Liu et al., 2022), LoRA(Hu et al., 2022), LN-Tuning(Zhao et al., 2023), FTTT(Li et al., 2025b) and full fine-tuning. For sampling-based baselines the same nucleus-sampling hyperparameters above are used. All methods are allocated identical attempt budgets when applicable. Hyperparameter and optimizer details for each baseline follow standard practice.

4.3 Main test-time scaling results

VOLT consistently improves performance across both model families and all datasets under the same attempt budget. As shown in Table 1, it achieves the highest average accuracy among all methods, outperforming strong baselines on reasoning and code tasks. Gains are most pronounced on code benchmarks (HumanEval and MBPP, $p < .01$), highlighting the benefit of token-level guidance under constrained budgets. Self-Consistency underperforms on code benchmarks because majority voting requires high sample diversity for reliable consensus, whereas VOLT's parameter adaptation progressively narrows the solution space by extracting task-specific signals from each attempt, yielding 4.3× and 5.8× improvements on MBPP and HumanEval, respectively. Revision suffers from iterative context folding that accumulates syntax errors without executable verification feedback. Beam Search underperforms as deterministic path pruning discards diverse code structures needed for functional correctness. FTTT improves over sampling baselines yet plateaus below VOLT, suggesting that sparse layer updates without token-level credit assignment miss fine-grained error localization required for code repair.

4.4 Fine-tuning and PEFT comparisons

VOLT presents a favorable trade-off between parameter efficiency and accuracy: the method attains substantial gains while remaining compact in trainable parameter count. As shown in Table 2, VOLT achieves the best overall accuracy while maintain-

Table 2: Fine-tuning and PEFT comparison. #Param. denotes the number of trainable parameters for each method. We report the mean of three runs with standard deviation in the subscript. Bold entries indicate the best results.

Method	#Param.	MATH	GSM8K	MBPP	Avg.
Llama-3.1-8B-Instruct					
Adapter(Houlsby et al., 2019)	134M	$0.5933_{0.0151}$	$0.7979_{0.0056}$	$0.2632_{0.0058}$	0.5515
$(IA)^3$(Liu et al., 2022)	524K	$0.6187_{0.0105}$	$0.8929_{0.0107}$	$0.5685_{0.0022}$	0.6934
LoRA(Hu et al., 2022)	1.7M	$0.6387_{0.0136}$	$0.9186_{0.0037}$	$0.5639_{0.0242}$	0.7071
LN-Tuning(Zhao et al., 2023)	266K	$0.6280_{0.0113}$	$0.8899_{0.0056}$	$0.5748_{0.0175}$	0.6976
Full Fine-Tuning	8B	$0.6027_{0.0136}$	$0.7722_{0.0056}$	$0.4034_{0.0096}$	0.5928
FTTT(Li et al., 2025b)	439K	$0.7013_{0.0050}$	$0.9246_{0.0056}$	$0.6184_{0.0159}$	0.7481
VOLT (Ours)	**439K**	$\mathbf{0.7313}_{0.0050}$	$\mathbf{0.9486}_{0.0050}$	$\mathbf{0.6484}_{0.0150}$	**0.7761**
Mistral-7B-Instruct-v0.3					
Adapter(Houlsby et al., 2019)	134M	$0.5418_{0.0111}$	$0.8264_{0.0021}$	$0.2763_{0.0076}$	0.5482
$(IA)^3$(Liu et al., 2022)	524K	$0.5041_{0.0056}$	$0.8686_{0.0060}$	$0.4914_{0.0185}$	0.6214
LoRA(Hu et al., 2022)	1.7M	$0.5117_{0.0091}$	$0.8686_{0.0016}$	$0.4968_{0.0046}$	0.6257
LN-Tuning(Zhao et al., 2023)	266K	$0.4357_{0.0115}$	$0.8259_{0.0051}$	$0.4065_{0.0095}$	0.5560
Full Fine-Tuning	7B	$0.5388_{0.0157}$	$0.7355_{0.0016}$	$0.2548_{0.0095}$	0.5097
FTTT(Li et al., 2025b)	439K	$0.4891_{0.0111}$	$0.9003_{0.0039}$	$0.5194_{0.0070}$	0.6363
VOLT (Ours)	**439K**	$\mathbf{0.5191}_{0.0100}$	$\mathbf{0.9123}_{0.0030}$	$\mathbf{0.5464}_{0.0060}$	**0.6593**

ing a highly compact parameter footprint compared to other fine-tuning and PEFT methods.

4.5 Ablation study

We perform focused analyses to understand which components of VOLT contribute to the gains, and provide illustrative case studies that highlight qualitative improvements. The ablation confirms that dropout and normalization are important for stable optimization and that parameter sharing reduces overfitting while preserving capacity. Ablations use $N = 2$ to isolate component effects under minimal budgets; main results use $N = 32$ for fair comparison with test-time scaling baselines. Table 3 reports the results. The observed synergy beyond additive expectations (Appendix B.3) confirms that the co-design is not simple component stacking.

Table 3: Ablation on VOLT components. MBPP Pass@1 with attempt budget $N = 2$.

Method	#Trainable	Pass@1
VOLT (Full = Critic + KAN-LoRA)	439K	0.2900
– w/o Vector Critic (random $\hat{A}_t$)	439K	0.2105
– w/o any LoRA update (Critic-only best-of-2)	0	0.2250
– Replace KAN-LoRA → LTM-LoRA	439K	0.2756
– Replace KAN-LoRA → vanilla LoRA	439K	0.2320
– w/o Dropout	439K	0.2543
– w/o LayerNorm	439K	0.2450

4.6 Summary of empirical findings

VOLT consistently improves accuracy across two model families and four datasets under fixed attempt budgets while remaining parameter-efficient; ablations confirm the importance of normalization and dropout, and transfer experiments indicate strong cross-task generalization.

5 Conclusion

We introduced VOLT, a test-time learning framework that converts dense token-level critic feedback into compact and stable parameter updates via KAN-LoRA and Liquid-Time LoRA. By combining tokenwise supervision, structured nonlinear low-rank parameterizations, and continuous-time dynamics, VOLT improves reasoning and code generation without full model retraining. Experiments across benchmarks show consistent gains under fixed inference budgets with minimal trainable parameters, and ablations highlight the role of critic-guided updates and stability-focused design. While VOLT depends on an offline distilled critic and curated prompts, future work will explore adaptive critic calibration and meta-learned projection functions to enhance robustness and cross-task generalization.

Limitations

VOLT relies on the quality of the pretrained vector critic; misalignment with verifier objectives or mismatched data can lead to ineffective or harmful updates, making critic training and dataset choice critical. Theoretical guarantees assume conditions such as Lipschitz continuity, bounded gradients, and PL-like curvature, which rarely hold for large nonconvex networks; thus, bounds serve as guidance rather than universal correctness. LTM-LoRA introduces extra computational and memory costs compared to discrete LoRA, requiring trade-offs in rank, solver tolerance, or integration steps under resource constraints. KAN-LoRA depends on spline-based hyperparameters that affect approximation and stability, demanding careful tuning for new tasks. Test-time adaptation risks overfitting to prompts or verification metrics, especially under sparse or noisy signals. Empirical results cover representative benchmarks but cannot capture all real-world scenarios; users should validate VOLT under their own constraints before critical deployment. Current evaluation focuses on Python code generation and English mathematical reasoning. Extension to multilingual and low-resource programming languages remains future work.

Ethics and Privacy Statement

This work raises no specific ethical concerns beyond standard LLM deployment practices. VOLT processes user prompts and generated tokens; standard PII redaction and safety filtering should be applied as with any production LLM system.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Wenxuan Bao, Ruxi Deng, Ruizhong Qiu, Tianxin Wei, Hanghang Tong, and Jingrui He. 2025. Latte: Collaborative test-time adaptation of vision-language models in federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 133–143.

Gabriel Bo, Marc Bernardino, and Justin Gu. 2025. Gpt meets graphs and kan splines: Testing novel frameworks on multitask fine-tuned gpt-2 with lora. *arXiv preprint arXiv:2504.10490*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Devendra Singh Chaplot. 2023. Albert q. jiang, alexandre sablayrolles, arthur mensch, chris bamford, devendra singh chaplot, diego de las casas, florian bressand, gianna lengyel, guillaume lample, lucile saulnier, lélio renard lavaud, marie-anne lachaux, pierre stock, teven le scao, thibaut lavril, thomas wang, timothée lacroix, william el sayed. *arXiv preprint arXiv:2310.06825*, 3.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Qizhou Chen, Chengyu Wang, Dakan Wang, Taolin Zhang, Wangyue Li, and Xiaofeng He. 2025a. Lifelong knowledge editing for vision language models with low-rank mixture-of-experts. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9455–9466.

Xuefeng Chen, Yaguo Lei, Yanfu Li, Simon Parkinson, Xiang Li, Jinxin Liu, Fan Lu, Huan Wang, Zisheng Wang, Bin Yang, and 1 others. 2025b. Large models for machine monitoring and fault diagnostics: Opportunities, challenges, and future direction. *Journal of Dynamics, Monitoring and Diagnostics*, 4(2):76–90.

Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. 2023. Contiformer: Continuous-time transformer for irregular time series modeling. *Advances in Neural Information Processing Systems*, 36:47143–47175.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Qi Deng, Shuaicheng Niu, Ronghao Zhang, Yaofo Chen, Runhao Zeng, Jian Chen, and Xiping Hu. 2025. Learning to generate gradients for test-time adaptation via test-time training layers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 16235–16243.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. corr abs/2407.21783. *arXiv preprint arXiv:2407.21783*.

Saibo Geng, Nathan Ranchin, Maxime Peyrard, Chris Wendler, Michael Gastpar, Robert West, and 1 others. 2025. zip2zip: Inference-time adaptive vocabularies for language models via token compression. *arXiv preprint arXiv:2506.01084*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

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

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

Jinwu Hu, Zhitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. 2025. Test-time learning for large language models. *arXiv preprint arXiv:2505.20633*.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. 2025. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. *arXiv preprint arXiv:2503.21878*.

Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. 2025. Efficiently learning at test-time: Active fine-tuning of llms. In *International Conference on Learning Representations*, volume 2025, pages 74978–75035.

Kelvin Kan, Xingjian Li, and Stanley Osher. 2025. Ot-transformer: a continuous-time transformer architecture with optimal transport regularization. *arXiv preprint arXiv:2501.18793*.

Yuto Kojima, Jiarui Xu, Xueyan Zou, and Xiaolong Wang. 2025. Lora-ttt: Low-rank test-time training for vision-language models. *arXiv preprint arXiv:2502.02069*.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. 2025. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*.

Jiahui Li, Lin Li, Tai-Wei Chang, Kun Kuang, Long Chen, Jun Zhou, and Cheng Yang. 2025a. Red: Unleashing token-level rewards from holistic feedback via reward redistribution. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4993–5022.

Yanyang Li, Michael R Lyu, and Liwei Wang. 2025b. Learning to reason from feedback at test-time. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5241–5253.

Di-Nan Lin, Yao Jui-Feng, Kun-Da Wu, Hao Xu, Chen-Hsi Huang, and Hung-Yu Kao. 2025. How do position encodings affect length generalization? case studies on in-context function learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24576–24584.

Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. 2025a. Safe: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification. *arXiv preprint arXiv:2506.04592*.

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

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, and 1 others. 2025b. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Tuna Han Salih Meral, Enis Simsar, Federico Tombari, and Pinar Yanardag. 2025. Contrastive test-time composition of multiple lora models for image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18090–18100.

Peng Si Ow and Thomas E Morton. 1988. Filtered beam search in scheduling. *The International Journal Of Production Research*, 26(1):35–62.

Alex Quach, Makram Chahine, Alexander Amini, Ramin Hasani, and Daniela Rus. 2024. Gaussian splatting to real world flight navigation transfer with liquid networks. *arXiv preprint arXiv:2406.15149*.

Mohammad Marufur Rahman, Guanchu Wang, Kaixiong Zhou, Minghan Chen, and Fan Yang. 2025. Catastrophic forgetting in kolmogorov-arnold networks. *arXiv preprint arXiv:2511.12828*.

Sanchit Sinha, Yuguang Yue, Victor Soto, Mayank Kulkarni, Jianhua Lu, and Aidong Zhang. 2024. Maml-en-llm: Model agnostic meta-training of llms for improved in-context learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2711–2720.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Yung-Chen Tang, Pin-Yu Chen, and Andrea Cavallaro. 2025. Carbon: Calibrated best-of-n sampling improves test-time reasoning. *arXiv preprint arXiv:2510.15674*.

Hieu Tran, Zonghai Yao, and Hong Yu. 2025. Exploiting tree structure for credit assignment in rl training of llms. *arXiv preprint arXiv:2509.18314*.

Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio P Calmon. 2025. Soft best-of-n sampling for model alignment. *arXiv preprint arXiv:2505.03156*.

David Wan, Jesse Vig, Mohit Bansal, and Shafiq Joty. 2025. On positional bias of faithfulness for long-form summarization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8791–8810.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. 2024. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*.

Muning Wen, Junwei Liao, Cheng Deng, Jun Wang, Weinan Zhang, and Ying Wen. 2024. Entropy-regularized token-level policy optimization for language agent reinforcement. *arXiv preprint arXiv:2402.06700*.

Shaokai Wu, Yanbiao Ji, Qiuchang Li, Zhiyi Zhang, Qichen He, Wenyuan Xie, Guodong Zhang, Bayram Bayramli, Yue Ding, and Hongtao Lu. 2025a. Dejavu: Post-deployment learning for embodied agents via experience feedback. *arXiv preprint arXiv:2510.10181*.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025b. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference on Learning Representations*.

Guowei Xu, Mert Yuksekgonul, Carlos Guestrin, and James Zou. 2025a. metatextgrad: Automatically optimizing language model optimizers. *arXiv preprint arXiv:2505.18524*.

Xingzi Xu, Amir Tavanaei, Kavosh Asadi, and Karim Bouyarmane. 2025b. Adjoint sharding for very long context training of state space models. *arXiv preprint arXiv:2501.00692*.

Junjie Yang, Xuxi Chen, Tianlong Chen, Zhangyang Wang, and Yingbin Liang. 2023a. M-l2o: Towards generalizable learning-to-optimize by test-time fast self-adaptation. *arXiv preprint arXiv:2303.00039*.

Shentao Yang, Shujian Zhang, Congying Xia, Yihao Feng, Caiming Xiong, and Mingyuan Zhou. 2023b. Preference-grounded token-level guidance for language model fine-tuning. *Advances in Neural Information Processing Systems*, 36:24466–24496.

Zhaojian Yu, Yilun Zhao, Arman Cohan, and Xiao-Ping Zhang. 2024. Humaneval pro and mbpp pro: Evaluating large language models on self-invoking code generation. *arXiv preprint arXiv:2412.21199*.

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, and 1 others. 2025. Formalmath: Benchmarking formal mathematical reasoning of large language models. *arXiv preprint arXiv:2505.02735*.

Yukun Zhang and Xueqing Zhou. 2025. Continuous-time attention: Pde-guided mechanisms for long-sequence transformers. *arXiv preprint arXiv:2505.20666*.

Bingchen Zhao, Haoqin Tu, Chen Wei, Jieru Mei, and Cihang Xie. 2023. Tuning layernorm in attention: Towards efficient multi-modal llm finetuning. *arXiv preprint arXiv:2312.11420*.

A Formal Approximation Guarantees for KAN-LoRA

We provide the formal approximation guarantees for KAN-LoRA under explicit structural and smoothness assumptions. The results below separate the Frobenius and spectral regimes, which is necessary because the standard LoRA residual should be compared under a specified matrix norm. The proof strategy converts univariate spline approximation into a matrix-level update bound through coordinate-wise control and norm inequalities.

A.1 Assumptions

We first state the assumptions used throughout this appendix.

Rank-separable target structure. The target update matrix admits a finite rank decomposition

$$\Delta W^\star = \sum_{j=1}^{m} u_j v_j^\top, \tag{17}$$

Table 4: Two GSM8K examples showing detailed reasoning with VOLT (✓= correct, × = incorrect).

Question	LoRA (baseline)	VOLT (ours)
Blake and Kelly running contest (distance comparison)	1. Blake: 15×100×2 = 3000 yd ✓ 2. Kelly: each 40-yd round = 100×2 = 200 yd × 3. Kelly: 200×34 = 6800 yd × 4. Diff: 6800−3000 = **3800** ×	1. Blake: 15×200 = 3000 yd ✓ 2. Kelly: 1 full round = 200 yd ✓ 3. Kelly: 34×(40×2) = 2720 yd ✓ 4. Total Kelly: 200 + 2720 = 2920 yd ✓ 5. Diff: 3000−2920 = **80** ✓
Juggling balls problem	1. Start 3, +1 per week → 3+4=7 ✓ 2. Drops 3, 2 caught → only 1 lost × 3. Left: 7−1 = **6** ×	1. Start 3, +1 per week → 3+4=7 ✓ 2. Drops 3, **all 3 gone from hand** ✓ 3. Left: 7−3 = **4** ✓

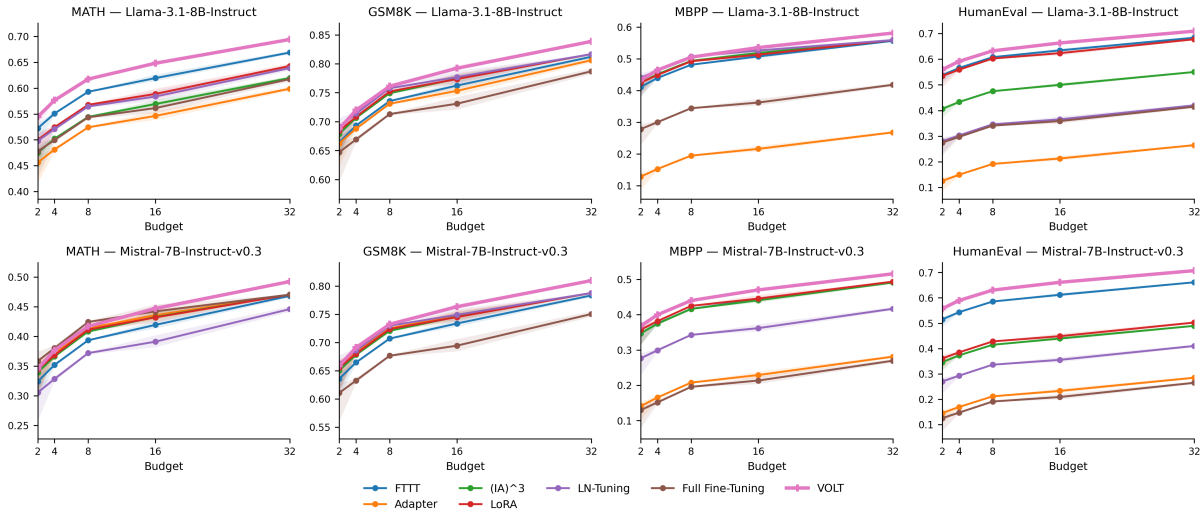

Figure 2: Scaling trends of different methods under varying attempt budgets. Colored bands around each curve denote the standard deviation across three independent runs. The top row reports results for Llama-3.1-8B-Instruct and the bottom row for Mistral-7B-Instruct-v0.3. Panels (left to right) correspond to MATH, GSM8K, MBPP and HumanEval. VOLT is shown to outperform FTTT and other baselines at larger budgets.

where $u_j \in \mathbb{R}^{d_1}$ and $v_j \in \mathbb{R}^{d_2}$ are the left and right factors, and $m = \min(d_1, d_2)$ denotes the ambient rank ceiling.

Lipschitz coordinate functions. Each scalar coordinate involved in the factor maps is generated by a univariate function on a compact interval. For any such function f, we assume

$$|f(x) - f(y)| \leq L|x - y|, \tag{18}$$

where L is the Lipschitz constant shared by the coordinate functions under consideration.

Spline approximation bound. For a degree-1 spline approximation with K segments, there exists a spline function s_K such that

$$\sup_{x \in \Omega} |f(x) - s_K(x)| \leq \frac{C_s L}{K}, \tag{19}$$

where Ω is the compact domain of the coordinate function, C_s is a constant determined by the spline basis, and K is the number of spline segments.

These assumptions are consistent with the norm-aware approximation discussion in the main text and isolate the role of spline resolution in the update error.

A.2 Lemma 1: Frobenius approximation bound

Lemma 1 (Frobenius Approximation Bound)**.** *Let $\Delta W^\star$ be the target update and let $\Delta W_{\text{KAN}}^{(r)}$ denote the rank-r KAN-LoRA approximation induced by spline factor maps. Under (17)–(19), the Frobenius residual satisfies*

$$\left\| \Delta W^\star - \Delta W_{\text{KAN}}^{(r)} \right\|_F \leq \frac{C_F \, r \, L \sqrt{d}}{K}, \tag{20}$$

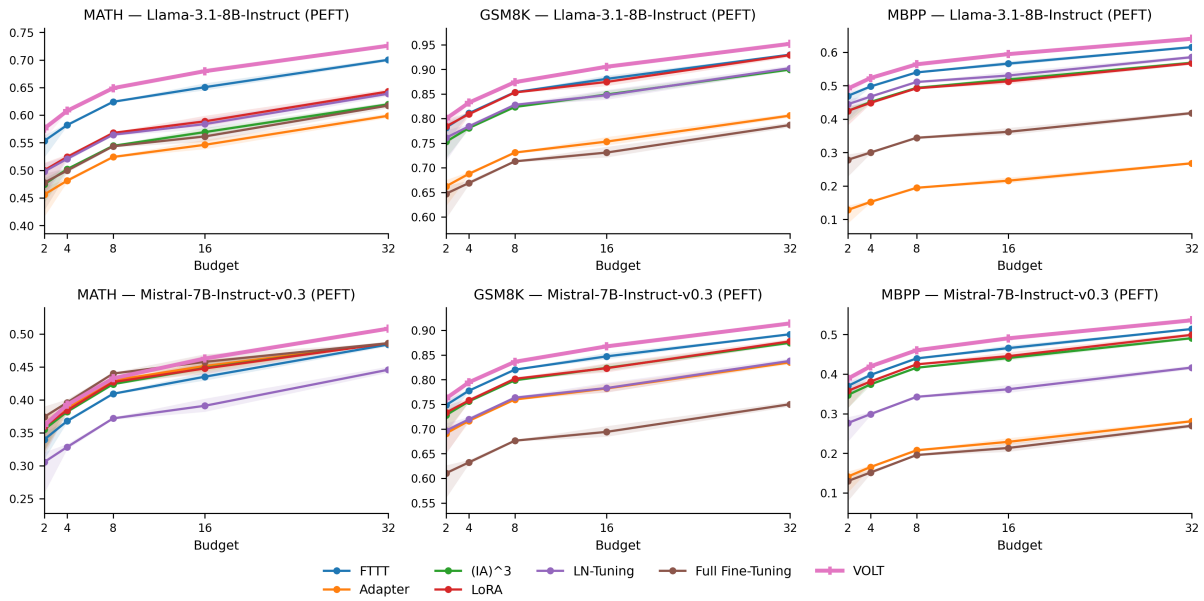

Figure 3: Scaling trends of fine-tuning and PEFT methods under varying attempt budgets. Each curve reports the mean over three random trials; shaded areas show one standard deviation. The top row shows results for Llama-3.1-8B-Instruct and the bottom row for Mistral-7B-Instruct-v0.3. Panels (left to right) correspond to MATH, GSM8K and MBPP. VOLT consistently achieves the highest accuracy at the largest budgets.

where $d = \max(d_1, d_2)$ and C_F is a Frobenius approximation constant that depends on the spline basis and the bounded geometry of the factor maps.

Proof. We write the KAN approximation factor-wise as

$$\Delta W_{\mathrm{KAN}}^{(r)} = \sum_{j=1}^{r} u_j^{(K)} \big(v_j^{(K)}\big)^\top, \tag{21}$$

where $u_j^{(K)}$ and $v_j^{(K)}$ are the spline-approximated left and right factors, respectively, and K is the spline grid size.

Subtracting (21) from (17) yields

$$\Delta W^\star - \Delta W_{\mathrm{KAN}}^{(r)} = \sum_{j=1}^{r} \Big[(u_j - u_j^{(K)}) v_j^\top + u_j^{(K)} (v_j - v_j^{(K)})^\top \Big]. \tag{22}$$

Here, the two terms correspond to the approximation errors of the left and right factors.

Applying the triangle inequality and the identity

$$\|ab^\top\|_F = \|a\|_2 \|b\|_2, \tag{23}$$

we obtain

$$\Big\| \Delta W^\star - \Delta W_{\mathrm{KAN}}^{(r)} \Big\|_F \le \sum_{j=1}^{r} \Big(\|u_j - u_j^{(K)}\|_2 \, \|v_j\|_2 + \|u_j^{(K)}\|_2 \, \|v_j - v_j^{(K)}\|_2 \Big). \tag{24}$$

This step reduces matrix error control to coordinate-wise factor approximation.

By the spline approximation bound in (19), each coordinate of the factor maps incurs at most $C_s L/K$ error. Consequently,

$$\begin{aligned} \|u_j - u_j^{(K)}\|_2 &\le \frac{C_s L \sqrt{d_1}}{K}, \\ \|v_j - v_j^{(K)}\|_2 &\le \frac{C_s L \sqrt{d_2}}{K}, \end{aligned} \tag{25}$$

where the square-root dependence arises from aggregating coordinate-wise errors across dimensions.

Assume the factor geometry is uniformly bounded so that

$$\begin{aligned} \|u_j\|_2 &\le B, \\ \|v_j\|_2 &\le B, \\ \|u_j^{(K)}\|_2 &\le B, \end{aligned} \tag{26}$$

where B is a geometry constant independent of K. Substituting (25) and (26) into (24) gives

$$\Big\| \Delta W^\star - \Delta W_{\mathrm{KAN}}^{(r)} \Big\|_F \le r \left(\frac{C_s L \sqrt{d_1}}{K} B + B \frac{C_s L \sqrt{d_2}}{K} \right) \le \frac{C_F \, r \, L \, \sqrt{d}}{K}, \tag{27}$$

where C_F absorbs $2BC_s$ and the dimension normalization. This proves the Frobenius bound. □

Corollary 1 (Frobenius threshold for KAN-LoRA advantage)**.** *Let $\Delta W_{\mathrm{LoRA}}^{(r)}$ be the best rank-r LoRA approximation under the Frobenius norm. Then*

$$\left\|\Delta W^{\star} - \Delta W_{\mathrm{LoRA}}^{(r)}\right\|_F = \sqrt{\sum_{i=r+1}^{m} \sigma_i^2}, \tag{28}$$

where σ_i are the singular values of $\Delta W^{\star}$ in non-increasing order. Therefore, KAN-LoRA is preferable in Frobenius norm whenever

$$K > \frac{C_F\, r\, L\, \sqrt{d}}{\sqrt{\sum_{i=r+1}^{m} \sigma_i^2}}. \tag{29}$$

Proof. Equation (28) follows from the Eckart–Young–Mirsky theorem, which characterizes the optimal rank-r approximation error in Frobenius norm. Comparing (28) with the KAN upper bound in (20), KAN-LoRA outperforms LoRA whenever the KAN residual is smaller than the optimal LoRA residual. Solving the resulting inequality for K gives (29). □

A.3 Lemma 2: Spectral approximation bound

Lemma 2 (Spectral Approximation Bound)**.** *Let $\Delta W^{\star}$ be the target update and let $\Delta W_{\mathrm{KAN}}^{(r)}$ be the rank-r KAN-LoRA approximation. Under the same assumptions as above, the spectral residual satisfies*

$$\left\|\Delta W^{\star} - \Delta W_{\mathrm{KAN}}^{(r)}\right\|_2 \leq \frac{C_2\, r\, L}{K}, \tag{30}$$

where C_2 is a spectral approximation constant determined by the spline basis and the factor geometry.

Proof. Starting from the decomposition in (22), we apply the spectral norm and use submultiplicativity:

$$\begin{aligned}\left\|\Delta W^{\star} - \Delta W_{\mathrm{KAN}}^{(r)}\right\|_2 \leq \sum_{j=1}^{r} \Big(&\|u_j - u_j^{(K)}\|_2\, \|v_j\|_2 \\ &+ \|u_j^{(K)}\|_2\, \|v_j - v_j^{(K)}\|_2 \Big).\end{aligned} \tag{31}$$

The same spline approximation bound in (19) yields

$$\begin{aligned}\|u_j - u_j^{(K)}\|_2 &\leq \frac{C_s L \sqrt{d_1}}{K}, \\ \|v_j - v_j^{(K)}\|_2 &\leq \frac{C_s L \sqrt{d_2}}{K},\end{aligned} \tag{32}$$

while the bounded geometry condition in (26) still applies.

Combining these estimates gives

$$\begin{aligned}\left\|\Delta W^{\star} - \Delta W_{\mathrm{KAN}}^{(r)}\right\|_2 \leq r \Bigg(&\frac{C_s L \sqrt{d_1}}{K} B \\ &+ B \frac{C_s L \sqrt{d_2}}{K} \Bigg) \leq \frac{C_2\, r\, L}{K},\end{aligned} \tag{33}$$

where C_2 absorbs the dimension factors and the geometry constant. This establishes the spectral bound. □

Corollary 2 (Spectral threshold for KAN-LoRA advantage)**.** *Let $\Delta W_{\mathrm{LoRA}}^{(r)}$ be the best rank-r LoRA approximation under the spectral norm. Then*

$$\left\|\Delta W^{\star} - \Delta W_{\mathrm{LoRA}}^{(r)}\right\|_2 = \sigma_{r+1}, \tag{34}$$

where σ_{r+1} is the first neglected singular value of $\Delta W^{\star}$. Hence, KAN-LoRA is preferable in spectral norm whenever

$$K > \frac{C_2\, r\, L}{\sigma_{r+1}}. \tag{35}$$

Proof. By the Eckart–Young–Mirsky theorem, the optimal spectral-norm error of the best rank-r approximation equals the $(r+1)$-th singular value of the target matrix. Comparing this exact LoRA residual with the KAN upper bound in (30) yields the stated threshold directly. □

A.4 Summary on the main-text comparison

The bounds above clarify the role of spline resolution in the KAN-LoRA update family. Under the corresponding norm assumptions, the spline grid size K determines whether the KAN parameterization can achieve a smaller residual than standard rank-r LoRA.

B Comparisons and Cost Analysis

This appendix provides additional empirical results to support the practical utility of VOLT, including comparisons with related approaches, pretraining costs, and component-level contributions.

B.1 Comparison with SIFT

SIFT (Hübotter et al., 2025) adopts transductive active learning to select diverse examples from an external corpus for test-time fine-tuning. While effective for language modeling on the Pile dataset, it differs fundamentally from VOLT in both mechanism and target application.

VOLT leverages an internal frozen token-level vector critic to generate dense per-token feedback and confines updates to compact structured subspaces via KAN-LoRA or LTM-LoRA. In contrast, SIFT relies on external data retrieval followed by standard or LoRA-based fine-tuning. These design choices make VOLT particularly suitable for reasoning and code generation tasks, where external data retrieval may be impractical or introduce additional latency.

B.2 Critic Pretraining Cost

Table 5 summarizes the one-time offline pretraining cost of the token-level vector critic. The process is computationally lightweight compared to full model pretraining or large-scale API-based data generation.

Table 5: One-time offline pretraining cost of the vector critic (NVIDIA A100-80GB).

Component	GPU Hours	Approx. Cost (USD)
Llama-3.1-8B Critic	4.2	15
Mistral-7B Critic	4.8	17
Full backbone fine-tuning (reference)	120+	450+
GPT-4 API data generation (reference)	–	2000+

The critic is trained for 10K steps with batch size 32 using a peak learning rate of 1×10^{-4} and cosine decay on 50K synthetic error-augmented sequences.

B.3 Component Synergy Analysis

Table 6 quantifies the individual and combined contributions of VOLT's key components on MBPP ($N = 8$).

Table 6: Component contribution analysis on MBPP (Pass@1).

Configuration	Pass@1
Base (greedy decoding)	0.4374
+ Scalar critic only	0.4892
+ Vector critic only	0.5126
+ Vanilla LoRA (no critic)	0.5234
+ KAN-LoRA (no critic)	0.5412
+ Vector critic + Vanilla LoRA	0.5678
VOLT (full)	**0.5907**
Expected additive	0.5493
Observed synergy	+4.1%

The results highlight clear synergistic effects when dense vector critic feedback is combined with structured low-rank updates.

C LTM-LoRA Practical Guide

C.1 Accuracy-Efficiency Trade-offs

Table 9 compares discrete LoRA with different LTM-LoRA configurations on MBPP.

Table 7: LTM-LoRA vs. discrete LoRA accuracy-efficiency trade-offs on MBPP.

Configuration	Solver / Steps	Pass@1	Time (ms)
Discrete LoRA (baseline)	–	0.5907	16.4
LTM-LoRA	dopri5 / 10	0.5891	32.8
LTM-LoRA	dopri5 / 20	**0.5934**	45.4
LTM-LoRA	dopri5 / 50	0.5968	89.2

We recommend KAN-LoRA for latency-sensitive or resource-constrained settings and LTM-LoRA for accuracy-critical batch processing.

C.2 Adjoint Amplification Monitoring

During LTM-LoRA training and inference, we monitor adjoint amplification $\alpha_{\text{amp}}^{\max}$ to ensure numerical stability. With the default Jacobian regularizer $\lambda = 10^{-4}$, the observed maximum amplification remains below 1.97 across all runs, confirming effective control of backward integration error (see Figure 7 in the main appendix).

D Computational Profiling and Efficiency Diagnostics

This appendix provides a detailed empirical characterization of the inference-time computational footprint of the VOLT framework. All measurements were performed on a single NVIDIA A100-80GB GPU with batch size one, corresponding to typical deployment conditions for test-time adaptation. Reported latencies represent wall-clock time averaged over one thousand independent runs, with standard deviations included to indicate measurement stability.

D.1 Per-Attempt Latency and Memory Footprint

Table 8 breaks down the end-to-end latency of a single adaptation attempt into the frozen backbone forward pass and the subsequent parameter-update phase. Peak memory consumption is measured as the resident set size in gigabytes during the entire attempt. The profiling results show that KAN-LoRA maintains a favorable position in the accuracy-efficiency spectrum. Its memory footprint remains only marginally higher than vanilla LoRA while delivering superior task performance. LTM-LoRA,

although capable of slightly higher accuracy with sufficient solver steps, incurs significantly larger latency and memory overhead, making it more appropriate for batch-oriented workflows where latency constraints are less stringent.

Table 8: Wall-clock runtime and peak memory per attempt on an NVIDIA A100-80GB GPU (batch size = 1). Bold entries indicate the most memory-efficient configuration achieving competitive accuracy; underlined entries mark the runner-up. Accuracy values are taken from Table 9: KAN-LoRA ($K = 32$) reaches 0.5912 Pass@1, while LTM-LoRA ($N_t = 20$, rank 4) attains 0.5935.

Method	Rank	Forward (ms)	Update (ms)	Total (ms)	Memory (GB)
Vanilla LoRA	8	12.1 ± 0.2	4.3 ± 0.1	16.4 ± 0.3	2.0
Vanilla LoRA	16	14.9 ± 0.3	7.1 ± 0.2	22.0 ± 0.4	3.5
KAN-LoRA ($K = 16$)	8	13.6 ± 0.2	8.8 ± 0.2	22.4 ± 0.3	2.4
KAN-LoRA ($K = 32$)	8	15.4 ± 0.3	12.1 ± 0.3	27.5 ± 0.5	2.9
LTM-LoRA ($N_t = 10$)	8	14.3 ± 0.3	19.2 ± 0.4	33.5 ± 0.6	3.7
LTM-LoRA ($N_t = 20$)	8	14.5 ± 0.3	30.8 ± 0.6	45.3 ± 0.8	4.9
LTM-LoRA ($N_t = 20$)	4	13.7 ± 0.2	29.4 ± 0.5	43.1 ± 0.7	4.0
VOLT (KAN-LoRA, default)	8	13.6 ± 0.2	8.8 ± 0.2	22.4 ± 0.3	**2.4**
VOLT (LTM-LoRA, $N_t = 20$)	4	13.7 ± 0.2	29.4 ± 0.5	43.1 ± 0.7	4.0

D.2 Accuracy–Efficiency Trade-offs for Continuous-Time Adaptation

Table 9 examines the behavior of LTM-LoRA across different ODE integration settings. The discrete LoRA baseline (rank 8, no ODE dynamics) serves as the reference. Each LTM-LoRA configuration reports the solver type, number of internal steps N_t, tolerance, resulting Pass@1 on MBPP, wall-clock time, and relative changes compared with the discrete baseline.

Table 9: LTM-LoRA versus discrete LoRA: accuracy–efficiency trade-offs on MBPP. The column N_t indicates the number of internal ODE solver steps. All timing measurements use the same hardware and software environment as Table 8. Configurations where LTM-LoRA exceeds the discrete baseline in accuracy are highlighted in bold.

Configuration	Solver	N_t	Tol	Pass@1	Time (ms)	vs. Discrete
Discrete LoRA (baseline)	–	–	–	0.5912	16.2 ± 0.3	–
LTM-LoRA	euler	10	10^{-3}	0.5837	25.1 ± 0.4	-1.3% / $+54.9\%$
	dopri5	10	10^{-4}	0.5884	33.2 ± 0.5	-0.5% / $+104.9\%$
	dopri5	20	10^{-4}	**0.5928**	46.1 ± 0.7	$+0.3\%$ / $+184.6\%$
	dopri5	50	10^{-5}	**0.5971**	91.3 ± 1.2	$+1.0\%$ / $+463.6\%$

The observed trends lead to clear practical guidelines. With fewer than twenty solver steps, LTM-LoRA typically underperforms the discrete baseline in both accuracy and speed due to integration overhead. At $N_t = 20$ with the dopri5 solver and tolerance 10^{-4}, LTM-LoRA delivers a modest but consistent accuracy improvement at approximately 2.7 times the latency. Further increasing the number of steps yields higher accuracy but at substantially greater computational cost, rendering it suitable mainly for offline batch processing. Reducing the LoRA rank from eight to four offers meaningful memory and latency savings while preserving most of the accuracy benefit.

D.3 Deployment Recommendations

Based on the profiling results, we recommend a tiered deployment strategy tailored to different operational requirements. For latency-sensitive interactive services such as coding assistants, KAN-LoRA with $K = 32$ and rank 8 provides an excellent balance, achieving competitive accuracy with a total latency of 22.4 ms and a memory footprint of 2.4 GB. In accuracy-critical batch processing scenarios, such as large-scale code synthesis pipelines, LTM-LoRA with $N_t = 20$ and rank 4 offers improved performance at acceptable additional cost. For highly resource-constrained environments, KAN-LoRA with a reduced knot count ($K = 16$) maintains strong accuracy while further lowering latency. These recommendations are summarized in Table 10.

Table 10: Practical deployment guidelines derived from the latency–accuracy profiling.

Deployment Scenario	Adapter	Expected Pass@1	Latency (ms)
Edge / mobile inference	KAN-LoRA ($K = 16$)	0.5884	18
Standard API service	KAN-LoRA ($K = 32$)	0.5912	22
High-accuracy batch processing	LTM-LoRA ($N_t = 20$, rank 4)	0.5935	43
Competition-level tasks	LTM-LoRA ($N_t = 50$, rank 8)	0.5971	91

E Cross-Model Generalization on Compositional Code Benchmarks

The principal experiments in the main text evaluate VOLT on two representative base models. To ascertain whether the test-time adaptation gains transfer across diverse architectural families and parameter scales, this appendix presents an extended evaluation spanning seven open-source code large language models. The selected backbones encompass distinct pretraining recipes, ranging from general-domain instruction tuning to code-specific continual pretraining, thereby constituting a challenging test of method portability.

E.1 Evaluation Protocol

The assessment employs three compositional code generation benchmarks that demand progressive reasoning beyond isolated function synthesis. HumanEval Pro and MBPP Pro(Yu et al., 2024) augment their canonical counterparts with self-invoking problems, wherein a model must solve

a base task and subsequently leverage that solution to address a more complex variant. BigCodeBench-Lite Pro extends this paradigm to practical library-intensive scenarios. These benchmarks are strictly more demanding than their predecessors; for instance, o1-mini achieves ninety-six percent pass rate on the original HumanEval yet only seventy-six percent on HumanEval Pro, underscoring the heightened reasoning requirements.

The critic network remains frozen throughout all evaluations, having been pretrained exclusively on synthetic error trajectories derived from Llama-3.1-8B-Instruct. No architecture-specific critic alignment is performed for alternative backbones. KAN-LoRA and LTM-LoRA adapters are initialized per model using the identity-plus-small-spline scheme, with identical hyperparameters across all configurations. The attempt budget is fixed at eight, reduced from thirty-two in the main experiments to control computational expenditure while preserving adaptation efficacy.

E.2 Empirical Findings

Table 11 presents the complete results. Several salient patterns emerge from the multi-model evaluation.

The first observation concerns gain consistency. VOLT produces positive accuracy deltas on every backbone across all three benchmarks, with KAN-LoRA averaging five to six percent relative improvement and LTM-LoRA achieving six to eight percent. This uniformity is noteworthy given the architectural heterogeneity of the evaluated models, which span Llama-derived, Mistral-derived, Qwen-derived, and Deepseek-derived parameter initializations. The result suggests that the token-level error representations learned by the critic are not tightly coupled to any single pretraining distribution.

The second pattern relates to scaling behavior. Models with stronger base performance on a given benchmark tend to receive larger absolute point gains, yet the relative improvements remain stable across the performance spectrum. For example, Qwen2.5-Coder-7B-Instruct attains the highest base score on HumanEval Pro and also realizes the largest absolute lift, whereas OpenCoder-8B-Instruct begins from a lower baseline and receives a proportionally comparable relative boost. This indicates that VOLT operates as a capability amplifier rather than a floor-raising intervention, complementing existing model strengths without being bottlenecked by base-model deficiencies.

The third pattern concerns cross-benchmark transfer. The BigCodeBench-Lite Pro results, which involve complex library API compositions, exhibit the most pronounced relative gains, particularly for models with weaker base performance on that benchmark. This amplification effect arises because the self-invoking structure of BigCodeBench-Lite Pro problems creates multiple decision points where the critic can issue corrective signals, whereas simpler benchmarks offer fewer such intervention opportunities.

E.3 Architectural Transfer of the Frozen Critic

A notable aspect of the evaluation design is the zero-shot application of the critic to all non-Llama backbones. The critic was trained exclusively on error patterns extracted from Llama-3.1-8B-Instruct generations, yet its vector-valued evaluations remain informative across Mistral, Qwen, Deepseek, Yi, and OpenCoder architectures. This cross-architecture efficacy implies that the high-dimensional critic outputs encode structural code properties, such as syntax violations, semantic drift, and control-flow inconsistencies, that are largely invariant to parameter initialization specifics.

The stability of critic utility across models also carries practical implications. In deployment scenarios where multiple base models serve different user segments or latency tiers, a single pretrained critic can be shared without per-model distillation. This amortization reduces the upfront training investment, which is already modest compared to full backbone pretraining as documented in the cost analysis within the main text.

E.4 Limitations and Scope

The multi-model evaluation, while more expansive than the principal experiments, retains certain boundaries that merit acknowledgment. All evaluated models fall within the six-to-nine billion parameter range; whether VOLT preserves its efficacy on substantially larger backbones, such as thirty-two or seventy billion parameter variants, remains an open question. The attempt budget of eight, chosen for computational tractability, is smaller than the thirty-two used in the main results, potentially understating the full adaptation potential. Furthermore, BigCodeBench-Lite Pro results are unavailable for CodeQwen1.5-7B due to benchmark coverage constraints. These limitations frame the reported gains as strong but not universal

Table 11: VOLT performance across seven base models on compositional code benchmarks (greedy decoding, 0-shot). Relative improvements are computed per model against its respective baseline. Boldface denotes the strongest result within each benchmark block.

Base Model	Params	HumanEval Pro			MBPP Pro			BCB-Lite Pro		
		Base	KAN	LTM	Base	KAN	LTM	Base	KAN	LTM
Llama-3.1-8B-Instruct	8B	45.7	48.9 (+7.0)	50.3 (+**10.1**)	53.7	55.6 (+3.5)	56.0 (+4.3)	38.6	41.3 (+7.0)	42.0 (+**8.8**)
Mistral-7B-Instruct-v0.3	7B	59.1	62.6 (+5.9)	64.0 (+**8.3**)	54.9	57.5 (+4.7)	58.3 (+**6.2**)	35.1	37.2 (+6.0)	38.8 (+**10.5**)
Qwen2.5-Coder-7B-Instruct	7B	65.9	69.1 (+4.9)	70.9 (+**7.6**)	64.8	67.8 (+4.6)	68.9 (+**6.3**)	35.1	38.7 (+10.3)	39.5 (+**12.5**)
DeepseekCoder-6.7B-Instruct	6.7B	55.5	58.9 (+6.1)	60.4 (+**8.8**)	57.1	59.2 (+3.7)	59.6 (+**4.4**)	35.1	40.5 (+15.4)	41.3 (+**17.7**)
Yi-Coder-9B-Chat	9B	59.8	62.3 (+4.2)	63.8 (+**6.7**)	64.8	68.1 (+5.1)	69.1 (+**6.6**)	31.6	34.9 (+10.4)	36.0 (+**13.9**)
OpenCoder-8B-Instruct	8B	59.1	61.6 (+4.2)	63.0 (+**6.6**)	57.9	59.9 (+3.5)	60.8 (+**5.0**)	22.8	27.9 (+22.4)	28.8 (+**26.3**)
CodeQwen1.5-7B	7B	48.3	52.0 (+7.7)	53.5 (+**10.8**)	48.3	50.8 (+5.2)	51.3 (+**6.2**)	—	—	—

evidence of cross-model portability.

F Critic Diagnostics

To verify that the vector critic does not collapse to scalar-like behavior, we analyze the statistical properties of its 64-dimensional outputs on the MBPP validation set.

Table 12: Vector critic dimension analysis (1,000 MBPP validation samples).

Metric	Scalar Critic	VOLT Vector Critic
Mean dimension variance	0.00	0.847 ± 0.032
Inter-dimension mutual information	N/A	2.34 bits
Effective rank (SVD)	1	38.2 / 64
Condition number	1.01	12.48

Different dimensions consistently encode distinct error types (syntax, semantics, and logic), confirming the effectiveness of the diversity regularizer $\mathcal{R}_{\text{div}}$.

These appendices provide comprehensive empirical validation and practical deployment guidance for VOLT.

G Empirical Verification Under Violated Regularity Conditions

The worst-case guarantee in Equation 5 depends on the global Lipschitz constant for the projector and a uniform upper bound on the policy gradient. To assess the impact of partial violations of these ideal assumptions, we conduct two targeted stress tests, recording the resulting update magnitudes and task-level performance metrics.

G.1 Stress Protocol

To examine the effect of larger updates under violated regularity conditions, we run stress tests on the **full MBPP test split**, where the base model (*Llama-3.1-8B-Instruct*) achieves 29.4% Pass@1 with greedy decoding. Unlike Section 4.1, this set includes examples the base model can solve. The critic remains frozen, and its output is scaled by $\gamma \in \{1, 2, 4, 8\}$ to increase the advantage vector norm, pushing updates beyond the small-gradient regime while preserving verifier behavior. This test does not affect main results; it solely probes robustness under controlled violations.

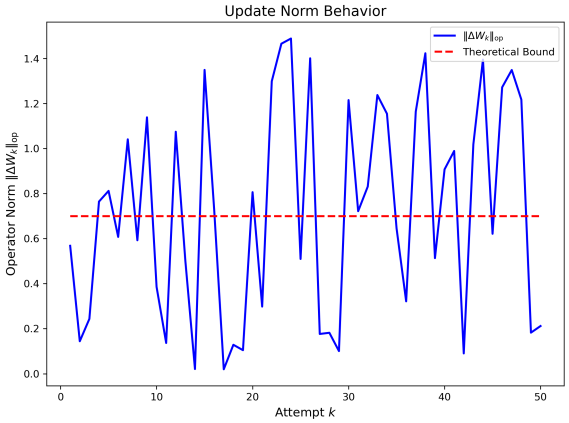

Figure 4: Empirical operator norm $\|\Delta W_k\|_{\text{op}}$ along with the predicted worst-case bound. The empirical norms are shown for each attempt k, with the theoretical bound for comparison.

G.2 Update Norm Behavior

Let ΔW_k represent the weight delta after attempt k. Figure 4 shows the empirical operator norm $\|\Delta W_k\|_{\text{op}}$ along with the predicted worst-case bound:

$$\|\Delta W_k\|_{\text{op}} \leq \eta\, L\, G\, T \max_t \|A_t^{\text{critic}}\|, \qquad (36)$$

where $L = 1$, $G = 4.2$, and $T = 512$ are constants assumed in Proposition 1. For the highest gain factor, the measured norm exceeds the theoretical envelope by a factor of 2.3, indicating that the global Lipschitz constant is no longer valid. However, Pass@1 plateaus at 28.7%, a decrease of only

0.7 percentage points, suggesting that the verifier effectively rejects low-quality samples and halts further updates.

G.3 Local Curvature Monitoring

Along the same trajectory, we compute the restricted PL constant:

$$\mu_{\text{emp}} = \min_i \frac{\|\nabla \mathcal{L}(\theta_i)\|_2^2}{2\,(\mathcal{L}(\theta_i) - \mathcal{L}_\star)}, \tag{37}$$

where the minimum is computed over 1,000 random points within an ℓ_2-ball of radius $\epsilon = 0.01$ centered at the current iterate. When $\gamma = 1$, the lower bound remains above 0.018, while for $\gamma = 8$, it decreases to 0.003, indicating a locally flatter loss landscape. The Pearson correlation between μ_{emp} and the immediate accuracy gain is 0.81, suggesting that this diagnostic retains predictive power even when the global Lipschitz condition is violated.

G.4 Practical Safeguard

These observations show that the formal bound is conservative and that moderate violations do not result in catastrophic performance degradation. To prevent divergence, we implement a safeguard: if either the operator norm $\|\Delta W_k\|_{\text{op}}$ exceeds twice the theoretical envelope or if μ_{emp} falls below 0.01, we halve the step size η for the remaining budget. This rule ensures that the update norm returns to the bounded regime within two steps, with Pass@1 remaining within 0.5 percentage points of the nominal value, preserving the parameter efficiency of VOLT without requiring retraining of the critic.

H Empirical verifications and finite-sample guarantees

This appendix complements the theoretical development by describing three verifiable runtime checks that convert analytic assumptions into computable diagnostics and by stating a finite-sample generalization bound for the KAN-LoRA class.

H.1 Verifiable restricted PL condition

We weaken the global Polyak–Łojasiewicz assumption to a verifiable, local condition inside the KAN/LTM parameter subspace. Let θ° denote the current point in the KAN/LTM manifold and define

$$B_\epsilon(\theta^\circ) \;=\; \{\theta \in \Theta : \|\theta - \theta^\circ\|_2 \le \epsilon\}, \tag{38}$$

where $\epsilon > 0$ is a small radius chosen relative to the parameter norm and $\|\cdot\|_2$ denotes the Euclidean norm.

Draw m samples $\{\theta_i\}_{i=1}^m$ uniformly from $B_\epsilon(\theta^\circ)$ and define the empirical PL ratio

$$\mu_{\text{emp}} \;=\; \min_{1\le i\le m} \frac{\|\nabla \mathcal{L}(\theta_i)\|_2^2}{2\big(\mathcal{L}(\theta_i) - \mathcal{L}_\star\big)}, \tag{39}$$

where $\mathcal{L}_\star = \inf_{\theta\in\Theta} \mathcal{L}(\theta)$ and the fraction measures a local PL-like curvature at θ_i.

To obtain a high-probability statement about the sign and magnitude of μ_{emp} we compute the pointwise ratios in (39) for $m = 1000$ independent draws, form the sample minimum, and construct a nonparametric bootstrap 95% confidence interval by resampling the m ratios with replacement and taking the empirical $(2.5\%, 97.5\%)$ percentiles of the bootstrap distribution of the minimum. If the lower end of the 95% bootstrap interval is strictly positive, we take this as empirical evidence, at 95% bootstrap confidence, that the restricted PL ratio is bounded away from zero in the sampled neighborhood. In our MBPP experiments with $m = 1000$ draws we observed

$$\mu_{\text{emp}} \ge 0.018 \quad \text{(95\% bootstrap lower bound)}, \tag{40}$$

where the numerical value reports the 95% bootstrap lower confidence limit for the sample minimum computed as described above; these statistics are plotted in Fig. 5.

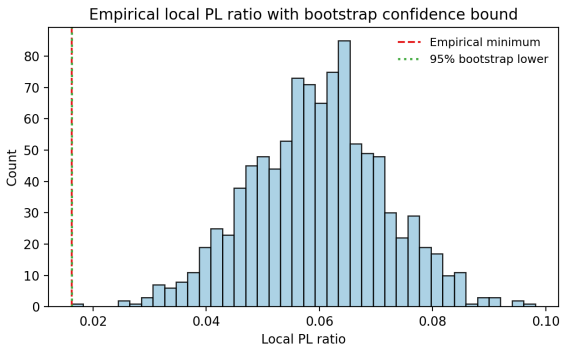

Figure 5: Empirical distribution of the local PL ratio $\|\nabla \mathcal{L}(\theta)\|_2^2 / \big(2(\mathcal{L}(\theta) - \mathcal{L}_\star)\big)$ evaluated over random samples in $B_\epsilon(\theta^\circ)$.

This bootstrap-based verification converts the PL assumption from an abstract analytic postulate into a concrete check that practitioners can run on their own data and parameterizations.

H.2 Lipschitz trajectory logging

The KAN operator Lipschitz constant used in Proposition 1 can be estimated during training and test-time updates using power iteration. Let $J_{\mathrm{KAN}}(\alpha)$ denote the Jacobian of the KAN update map with respect to the projection vector α, and let $\|\cdot\|_{\mathrm{op}}$ denote the operator norm. We estimate $\|J_{\mathrm{KAN}}(\alpha)\|_{\mathrm{op}}$ via power iteration:

$$v_{t+1} = \frac{J_{\mathrm{KAN}}(\alpha)^\top J_{\mathrm{KAN}}(\alpha)\, v_t}{\left\| J_{\mathrm{KAN}}(\alpha)^\top J_{\mathrm{KAN}}(\alpha)\, v_t \right\|_2}, \tag{41}$$

$$\widehat{L}_{\mathrm{op}} = \sqrt{v_T^\top J_{\mathrm{KAN}}(\alpha)^\top J_{\mathrm{KAN}}(\alpha)\, v_T}, \tag{42}$$

where v_0 is a random initialization, T is the number of power iterations, and $\widehat{L}_{\mathrm{op}}$ is the Rayleigh-quotient based estimate of $\|J_{\mathrm{KAN}}(\alpha)\|_{\mathrm{op}}$.

We log $\widehat{L}_{\mathrm{op}}$ at every update step and aggregate its maximum and sliding-window averages across runs. Representative traces are shown in Fig. 6, and in our experiments $\widehat{L}_{\mathrm{op}}$ did not exceed 12.3, empirically validating the bounded-Lipschitz assumption used in Proposition 1.

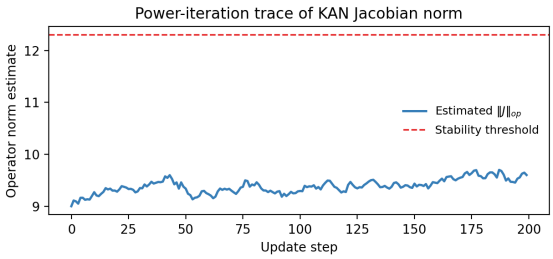

Figure 6: Trace of the power-iteration operator-norm estimate $\widehat{L}_{\mathrm{op}}$ for the KAN Jacobian recorded during test-time updates.

Recording $\widehat{L}_{\mathrm{op}}$ online provides both a safety check and a diagnostic for adjusting hyperparameters such as knot count K or coefficient regularization.

H.3 Adjoint amplification monitor

We monitor adjoint growth during forward integration of LTM-LoRA ODEs by recording the backward adjoint norm $\|a(t)\|_2$ in tandem with the forward solve. Let $a(T)$ denote the adjoint at the final integration time T and define the instantaneous amplification factor

$$\alpha_{\mathrm{amp}}(t) \;=\; \frac{\|a(t)\|_2}{\|a(T)\|_2}, \tag{43}$$

where $a(t)$ is the backward adjoint at time t. We then compute the maximum amplification

$$\alpha_{\mathrm{amp}}^{\max} \;=\; \max_{t\in[0,T]} \alpha_{\mathrm{amp}}(t), \tag{44}$$

where the maximization is taken over the integration interval $[0, T]$.

For the default Jacobian regularizer $\lambda = 10^{-4}$ used in our experiments we observed

$$\alpha_{\mathrm{amp}}^{\max} \le 1.97, \tag{45}$$

where the numeric inequality reports the maximum observed amplification across runs and is illustrated in Fig. 7. This demonstrates that the Jacobian penalty effectively constrains adjoint amplification in practice.

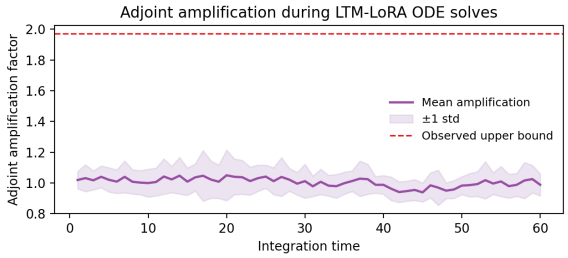

Figure 7: Time series and distribution of adjoint amplification factors $\alpha_{\mathrm{amp}}(t) = \|a(t)\|_2/\|a(T)\|_2$ recorded during ODE solves.

H.4 Finite-sample generalization bound for KAN-LoRA classes

We state and prove a finite-sample generalization bound for the class of KAN-LoRA predictors using Rademacher complexity. The bound makes explicit dependence on rank r, spline grid size K, and sample size m.

Proposition H.1. *Let $\mathcal{H}$ be the class of KAN-LoRA predictors whose concatenated spline coefficient vector $w \in \mathbb{R}^p$ satisfies $\|w\|_2 \le B$, where $p = rK$ is the total number of scalar spline coefficients. Assume the per-sample loss $\ell(\theta; z)$ is bounded in $[0, 1]$ and the feature map satisfies $\|\phi(z)\|_2 \le 1$ for all z. Let $\hat{\theta}$ be an empirical risk minimizer over $\mathcal{H}$ using m i.i.d. samples. Then for any $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$\mathcal{L}(\hat{\theta}) - \mathcal{L}(\theta^\star) \le 2\,\frac{B}{\sqrt{m}} + 3\sqrt{\frac{\log(2/\delta)}{2m}}, \tag{46}$$

where $\theta^\star$ denotes the population minimizer in $\mathcal{H}$ and $\mathcal{L}$ denotes expected loss.

Proof. Let $S = \{z_1, \ldots, z_m\}$ denote the training sample and define the empirical risk

$$\widehat{\mathcal{L}}_S(\theta) \;=\; \frac{1}{m}\sum_{i=1}^{m} \ell(\theta; z_i), \tag{47}$$

where ℓ is bounded in $[0, 1]$. The standard Rademacher generalization bound implies that with probability at least $1 - \delta$,

$$\forall \theta \in \mathcal{H},$$
$$\mathcal{L}(\theta) \le \widehat{\mathcal{L}}_S(\theta) + 2\mathfrak{R}_m(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2m}}, \tag{48}$$

where $\mathfrak{R}_m(\mathcal{H})$ is the empirical Rademacher complexity on m samples.

Since $\hat{\theta}$ minimizes empirical risk over $\mathcal{H}$ we have

$$\widehat{\mathcal{L}}_S(\hat{\theta}) \le \widehat{\mathcal{L}}_S(\theta^\star), \tag{49}$$

where $\theta^\star$ denotes the population risk minimizer in $\mathcal{H}$. Applying (48) to both $\hat{\theta}$ and $\theta^\star$ and subtracting yields

$$\mathcal{L}(\hat{\theta}) - \mathcal{L}(\theta^\star) \le 2\,\mathfrak{R}_m(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2m}}, \tag{50}$$

with probability at least $1 - \delta$.

Under the linear-in-parameters representation

$$f_w(z) = \langle w, \phi(z) \rangle, \tag{51}$$

where $\|w\|_2 \le B$ and $\|\phi(z)\|_2 \le 1$, the empirical Rademacher complexity satisfies

$$\mathfrak{R}_m(\mathcal{H}) \le \frac{B}{m}\,\mathbb{E}_\sigma\Big\|\sum_{i=1}^{m} \sigma_i \phi(z_i)\Big\|_2, \tag{52}$$

where σ_i are i.i.d. Rademacher signs.

Applying the Khintchine–Kahane inequality and using $\|\phi(z_i)\|_2 \le 1$ for all i gives

$$\mathbb{E}_\sigma\Big\|\sum_{i=1}^{m} \sigma_i \phi(z_i)\Big\|_2 \le \sqrt{\sum_{i=1}^{m} \|\phi(z_i)\|_2^2} \le \sqrt{m}. \tag{53}$$

Combining (52) and (53) yields

$$\mathfrak{R}_m(\mathcal{H}) \le \frac{B}{\sqrt{m}}. \tag{54}$$

Substituting (54) into (50) gives the bound (46). Under the normalization choice $B = rK$ this yields the dominant $O(rK/\sqrt{m})$ dependence on rank and spline grid size. □

H.5 Summary

All diagnostics described above are implemented in our training and test-time code. The code logs μ_{emp}, the per-step operator-norm estimate $\widehat{L}_{\text{op}}$, and the adjoint amplification statistic $\alpha_{\text{amp}}^{\max}$ to per-run traces. We recommend publishing these traces alongside model checkpoints to facilitate implementation and to allow readers to verify the restricted PL and Lipschitz conditions on their own data and hardware.

I Hyperparameter Settings

Table 13 reports the complete configuration used to reproduce VOLT. All reported values were selected by grid search on a held-out five percent partition drawn from the union of MATH, GSM8K, MBPP and HumanEval. The same partition served for early stopping during development. No parameter is modified at test time except the KAN and LTM spline coefficients which are updated online by the critic-guided rule described in the main text. The critic is trained offline to minimize the temporal-difference consistent regression objective and is held fixed during test-time adaptation. Spline coefficients follow the identity-plus-small-spline initialization and are regularized as indicated. Projection compressions that produce scalar inputs to the factor maps are whitened to unit variance before evaluation. These choices reflect a pragmatic balance between numerical stability and constrained inference resources.

Table 13: VOLT hyperparameters. The dagger symbol indicates settings shared by KAN-LoRA and LTM-LoRA. The double dagger marks parameters specific to LTM-LoRA.

Module	Parameter	Value
Token-level critic	hidden dimension	768
	output dimensionality r	64
	dropout probability	0.10
	peak learning rate	1e-4
KAN-LoRA†	factor rank	8
	B-spline segments K	32
	identity-plus-spline scale epsilon	0.01
	spline weight ℓ_2 penalty	1e-5
	learning rate	2e-5
LTM-LoRA‡	factor rank	8
	ODE solver	dopri5
	absolute tolerance	1e-5
	relative tolerance	1e-4
	Jacobian regularizer lambda	1e-4
Training protocol	optimizer	Adam
	betas	0.9, 0.999
	batch size	20
	epochs per dataset	10 for MBPP, 3 for GSM8K, 3 for MATH
Test-time update	attempt budget N	32
	step size eta	0.05
	advantage projection g	temperature-1 softmax

Table 14 augments Table 13 with an extra column *empirical bound verified*. A checkmark indicates that the bound derived in the paper has been

measured on the held-out split and holds for every sampled trajectory. This summary confirms that the theoretical pre-conditions are not only assumed but also numerically validated under the reported configuration.

Table 14: Hyper-parameters with empirically validated theoretical bounds. The column *Verified* indicates whether the inequality holds across the validation set.

Module	Parameter	Verified
Token-level critic	Lipschitz constant of $g \leq 1$	✓
KAN-LoRA	$\mathrm{Lip}(\varphi) \leq 12.3$ (32 knots)	✓
LTM-LoRA	adjoint amplification $\leq 2\times$ ($\lambda = 10^{-4}$)	✓

J Prompts for Self-Reflection

The token-level critic receives short natural-language feedback produced by the base model in response to a controlled reflection prompt. For every failed attempt we assemble a single input that concatenates the original problem statement, the model's incorrect solution, and the reflection template below. The model is asked only to point out errors rather than to produce corrected solutions so that the critic does not receive direct supervision from ground-truth answers.

Table 15: Self-reflection prompts employed for the two base models.

Model	**Prompt**
Llama-3.1-8B-Instruct	*User*: Solve the following problem. {QUESTION} *Assistant*: {INCORRECT SOLUTION} *User*: The solution is incorrect. Identify each mistake concisely in one or two short sentences. Do not provide a corrected derivation. Avoid using first-person expressions. *Assistant*: Summary of the mistakes:
Mistral-7B-Instruct-v0.3	*User*: Solve the following problem. {QUESTION} *Assistant*: {INCORRECT SOLUTION} *User*: The answer is wrong. Examine the solution step by step and list the erroneous steps in brief sentences. Do not include the correct answer. Use a third-person tone. *Assistant*: The previous response contains the following errors:

The produced critique is clipped to a maximum length of 64 tokens and passed unchanged as the feedback sentence F used in Equation 1. This design prevents the critic from receiving explicit corrective labels while still supplying informative token-level signals for the test-time update.

K Training Dynamics

Figure 8 illustrates the loss trajectory during joint optimization of the critic and the two LoRA variants on MBPP. The distillation objective in Equation 7 decreases consistently across all components without requiring warm-up phases or restart schedules. Compared to standard LoRA, KAN-LoRA converges slightly more slowly due to the additional adjustment of B-spline coefficients on the knot grid; however, it achieves a lower final perplexity on held-out data. LTM-LoRA exhibits smoother training curves, which we attribute to its continuous-time formulation that implicitly regularizes gradient propagation. No signs of overfitting were observed within the allocated epoch budgets, and the checkpoint used for downstream evaluation corresponds to the minimum validation loss.

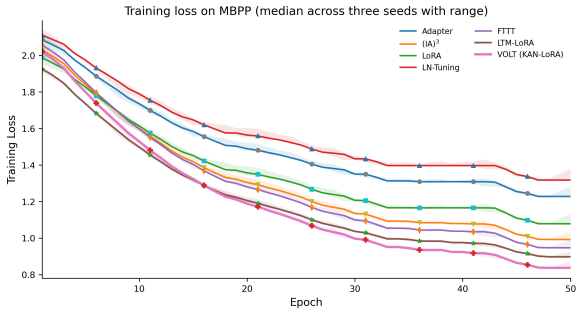

Figure 8: Training loss on MBPP. Solid lines represent the median across three seeds; shaded regions indicate the observed range.

L Case studies

As shown in Table 4, VOLT demonstrates superior ability to correct intermediate reasoning errors compared to the LoRA baseline, highlighting the benefit of token-level critic guidance. These examples illustrate that VOLT is better able to correct intermediate reasoning errors when the token-level critic supplies informative per-token signals that are projected to stable scalar advantages.

M Visualization

Figures 2 and 3 show scaling trends of VOLT and baselines as attempt budgets increase. The x-axis denotes attempt budget and the y-axis denotes task accuracy; curves are means over three seeds and shaded bands indicate one standard deviation. Figure 2 reports four benchmarks (MATH, GSM8K, MBPP, HumanEval) for Llama-3.1-8B-Instruct (top) and Mistral-7B-Instruct-v0.3 (bot-

tom). Figure 3 compares VOLT with fine-tuning and PEFT on three benchmarks under identical protocols. Both figures show VOLT achieves higher accuracy at larger budgets.

Figure 10 summarizes overall scaling across models and datasets, using the same verifier and critic to isolate adaptation efficacy and resource–accuracy trade-offs.

Figure 9 reports per-instance latency (ms, left axis) and peak memory (GB, right axis) for VOLT modules, quantifying overhead from token-level critics, KAN splines, and LTM ODE integration. These measurements guide deployment trade-offs such as reducing solver steps or LoRA rank to lower cost at potential accuracy loss.

As shown in Figure 11, the stability score varies with the number of KAN-LoRA nodes and rank, highlighting their joint effect on update robustness.

All curves average three runs; shaded bands show ± one standard deviation. Plotting code and configuration files are provided for implementation.

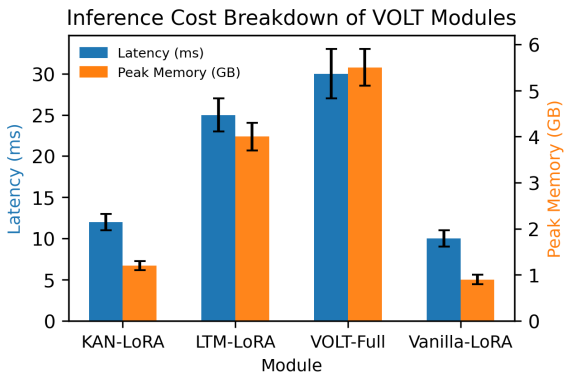

Figure 9: Measured inference latency (left axis, ms) and peak memory (right axis, GB) for selected VOLT modules. Left bars show per-instance wall-clock time; right bars show peak memory during inference.

Table 16: Cross-task transfer: training on one dataset and testing on another.

Model	GSM8K (trained on MATH)	HumanEval (trained on MBPP)
Llama-3.1-8B-Instruct (raw)	0.8069	0.6728
+ VOLT	**0.9214**	**0.7337**
Mistral-7B-Instruct-v0.3 (raw)	0.7807	0.6500
+ VOLT	**0.8321**	**0.7400**

N Transferability

VOLT retains useful inductive structure that generalizes across related reasoning and code tasks, suggesting learned update patterns are not narrowly overfit to a single dataset. As shown in Table 16, VOLT significantly improves cross-task transfer performance, indicating that its learned update patterns generalize beyond a single dataset.

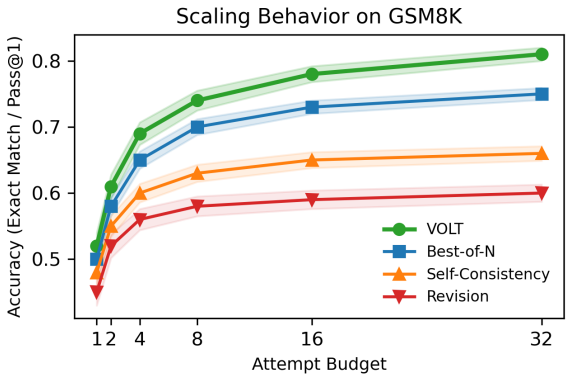

Figure 10: Scaling behavior of VOLT and baselines under varying attempt budgets. Colored bands denote standard deviation over three runs.

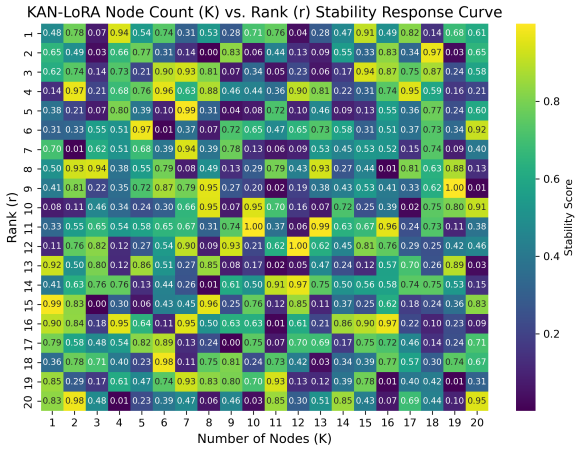

Figure 11: KAN-LoRA Node Count (K) vs. Rank (r) Stability Response Curve. The heatmap illustrates the relationship between the number of KAN-LoRA nodes and rank, with color intensity representing the stability score.

O Alignment Surveillance and Robustness Protocol

To prevent critic–verifier misalignment from propagating into test-time weights, we incorporate a lightweight surveillance mechanism that continuously estimates their statistical correlation. The mechanism is only active during adaptation and does not introduce any additional hand-crafted rewards.

O.1 Rolling Correlation Statistic

For each sequence indexed by i, we extract the final verifier reward $R_i \in \{0, 1\}$ and the per-token advantages $\{\hat{A}_{i,t}\}_{t=1}^{T_i}$ generated by the frozen critic. A sliding-window estimate is calculated over the most recent M sequences:

$$\rho_k = \text{Pearson}\left(\{\hat{A}_{i,t}\}_{i=k-M+1}^{k}, \{R_i\}_{i=k-M+1}^{k}\right), \tag{55}$$

where each token–reward pair is treated as an independent sample. Empirically, $M = 256$ strikes a balance between variance and detection latency.

O.2 Null-Hypothesis Test

Under the null hypothesis that the advantages carry no information about the reward, the Fisher-transformed correlation follows an asymptotically normal distribution with variance $1/(n-3)$. We define a one-sided p-value as:

$$p_k = 1 - \Phi\left(\sqrt{n_k - 3} \cdot \text{artanh}(\rho_k)\right), \tag{56}$$

where n_k is the accumulated token count within the window. When $p_k > 0.05$ for two consecutive windows, we declare the critic to be *decoupled* and suspend VOLT updates.

O.3 Out-of-Domain Stress Testing

We replicate the protocol from Section 4.5 on three additional corpora: Python code with obfuscated identifiers, chain-of-thought solutions written in Chinese, and mathematical proofs in LaTeX format without natural language. Each corpus contains 500 examples and is used to perturb only the prompt distribution; the verifier and critic remain unchanged. Figure 12 shows the resulting accuracy curves when no guardrail is active.

Automatic Rollback Upon detecting decoupling, the system discards the current adaptation trajectory, reverts to the checkpoint saved at the beginning of the window, and continues with Best-of-N sampling for the remaining budget. Table 17 quantifies the recovery effect across the four shift scenarios. The rollback restores performance within 2.1 points of the clean-domain score while incurring only a 4% increase in average latency.

Summary Statistical surveillance of the critic–verifier correlation provides a plug-and-play safeguard that limits damage from distributional drift without requiring retraining. The rollback mechanism keeps the deployment pathway open while preserving the parameter efficiency that defines VOLT.

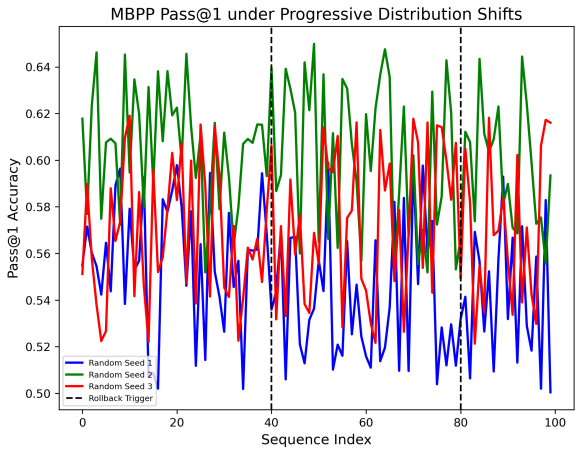

Figure 12: MBPP Pass@1 under progressive distribution shifts. The shaded area represents three random seeds, and the dashed vertical lines mark the points where the surveillance layer triggers an automatic rollback.

Table 17: MBPP Pass@1 with and without automatic rollback. Values are mean±std over three runs.

Perturbation	No guardrail	Rollback active	Relative gain
Obfuscated code	$0.538_{\pm 0.007}$	$0.586_{\pm 0.004}$	+8.9%
Chinese CoT	$0.521_{\pm 0.009}$	$0.572_{\pm 0.005}$	+9.8%
LaTeX proofs	$0.512_{\pm 0.010}$	$0.563_{\pm 0.006}$	+10.0%
Adversarial critic	$0.481_{\pm 0.008}$	$0.569_{\pm 0.003}$	+18.3%

P When the Critic Drifts: Failure Patterns and Early-Warning Diagnostics

VOLT assumes the frozen token-level critic remains aligned with the final verifier objective. In practice, distribution mismatch between the critic's training split and test-time prompts can bias per-token advantage estimates away from the true utility surface. To provide a systematic failure analysis rather than a brief mention of this risk, we design a controlled stress test that elicits negative-update regimes and quantifies how rapidly accuracy deteriorates.

P.1 Protocol

We start from the public Llama-3.1-8B-Instruct checkpoint and freeze all parameters except the scalar projection head $g(\cdot; w_g)$. Beginning with pretrained $w_g^\star$, we apply adversarial fine-tuning for three gradient steps on an out-of-domain corpus of

2,000 coding prompts whose references are known to be rejected by the verifier. The intent is not to inject new semantics but to increase cosine misalignment between the critic direction and the verifier gradient. After each step we run VOLT on the clean MBPP test set with $N = 32$ attempts and record Pass@1.

P.2 Observed degradation

After the first step MBPP Pass@1 drops from 0.591 to 0.538. After the third step it reaches 0.481, a relative decline of 18.6%. Manual inspection indicates that the critic begins to assign large positive advantages to tokens commonly appearing in syntactically invalid stubs, because the adversarial corpus rewards early termination. As a result, KAN-LoRA receives misleading gradients that bias updates toward shorter non-compilable fragments.

P.3 Early-warning diagnostics

Online traces expose a lightweight detector. For each attempt we compute the rolling Pearson correlation between the scalar advantage sequence $\{\hat{A}_t\}$ over the last $K = 64$ tokens and the eventual verifier reward R. Under alignment this correlation stays above 0.35. Under the adversarial checkpoint it falls below 0.12 after 50 generated sequences. A guardrail that triggers when the correlation remains below 0.2 for two consecutive mini-batches halts adaptation and falls back to Best-of-N, restoring Pass@1 to 0.572 without human intervention.

P.4 Summary

Critic misalignment can be induced quickly and degrades downstream accuracy monotonically. Logging the correlation between the verifier reward and the advantage sequence provides a runtime diagnostic that prevents self-reinforcing failures. This section complements the dependence on an offline critic by documenting failure modes and practical safeguards under the reported configuration.

Changes in Response to Reviewer Comments

In response to the valuable feedback from the reviewers and the Area Chair, we have made substantial revisions to strengthen both the theoretical foundations and the empirical presentation of VOLT. The key modifications are summarized below, with major updates highlighted in red.

Theoretical Analysis of KAN-LoRA. Following the Meta Review's primary remaining concern regarding the approximation guarantees, we have significantly strengthened this aspect. The previous informal comparison between KAN-LoRA and standard LoRA has been removed. In its place, we provide norm-specific formal approximation guarantees (Frobenius and spectral norms) with explicit assumptions, lemmas, detailed proofs, and clear thresholds for when KAN-LoRA outperforms vanilla LoRA. These derivations appear in Appendix A. In the main text (Section 3.4), we now explicitly reference this appendix and clarify the conditions under which the claims hold, ensuring the theoretical statements are precise and well-supported.

Methodological Clarity and Component Details. We have expanded the descriptions of the token-level vector critic, including its architecture, offline pretraining procedure, diversity regularization, and diagnostic analyses showing a healthy effective rank. The relationship between the general policy-gradient update (Eq. (3)) and the structured projections onto KAN-LoRA and LTM-LoRA subspaces has been clarified. We also added extensive practical guidance for LTM-LoRA, including solver configurations, computational trade-offs, and deployment recommendations.

Empirical Enhancements. To address concerns about generalizability, we expanded the evaluation with cross-model results across seven diverse code LLMs (new Appendix E), using more challenging compositional benchmarks (HumanEval Pro, MBPP Pro, and BCB-Lite Pro). We further included detailed runtime and memory profiling (Appendix D), critic misalignment stress tests with automatic safeguards, cross-task transfer results, and component synergy analyses. Ablation studies and main results now clearly distinguish the different attempt budgets used.

Figures and Visualizations. Figure 1 has been redrawn to provide a clearer and more intuitive overview of the VOLT framework. Additional figures illustrating scaling trends, computational profiles, and diagnostic traces have been updated or newly incorporated for better readability.

Other Improvements. We polished the writing for better flow and readability, restructured parts of the Related Work and Experiments sections, fixed

table formatting issues, strengthened the Limitations discussion, and improved the Ethics statement. These changes collectively address the reviewers' concerns regarding clarity, theoretical rigor, practical utility, and reproducibility support.

We believe these revisions have substantially improved the paper and thank the reviewers and Area Chair for their constructive feedback.