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 become an effective strategy to improve the reliability and problem-solving capabilities of large language models. Recent work shows that allocating extra computation at test time or adapting model state using verification signals can substantially improve outcomes on challenging reasoning and code generation tasks (Snell et al., 2024; Welleck et al., 2024; Tang et al., 2025; Verdun et al., 2025). Two practical paradigms have emerged in this area: revision style sequential refinement, which folds earlier attempts back into context, and parallel sampling, which generates many independent attempts and selects the best candidate (Madaan et al., 2023; Snell et al., 2024). Each paradigm presents trade-offs. Revision based strategies can accumulate corrective information but incur long context costs and are sensitive to positional effects (Wan et al., 2025; Lin et al., 2025). Sampling based strategies explore efficiently but tend to discard rich per-token information and do not directly learn from prior errors (Wu et al., 2024, 2025b).

Complementary research focuses on parameter-efficient updates and denser token-level supervision. Low-rank adapters such as LoRA and their recent extensions enable compact specialization of large models (Kojima et al., 2025; Zhang et al., 2023; Yang et al., 2024; Liao et al., 2025). Structured factor maps inspired by univariate spline parameterizations and Kolmogorov–Arnold Networks increase the expressive capacity of constrained adapter families (Liu et al., 2024; Bo et al., 2025; Rahman et al., 2025). Continuous-time and Neural-ODE perspectives have been applied to model smooth parameter dynamics and to enable efficient adjoint computations for trajectory based updates (Zheng et al., 2024; Quach et al., 2024; Xu et al., 2025b; Zhang and Zhou, 2025). Token-level credit assignment and reward redistribution transform sequence level feedback into denser signals suitable for fine-grained optimization (Yang et al., 2023b; Wen et al., 2024; Li et al., 2025a; Zhou et al., 2025). These approaches improve optimization fidelity but can depend on learned or automatically generated annotations whose quality varies across tasks (Zhou et al., 2025; Li et al., 2025a; Yang et al., 2023b).

There remain several important gaps in the literature. Sampling heavy test-time strategies may under-utilize tokenwise information, while revision based solutions become impractical under tight context and compute budgets. Parameter-efficient schemes can sacrifice expressivity and struggle when the corrective update is nonlinear in activation space. Continuous-time adapters promise

smoother evolution but introduce solver and stability concerns unless they are carefully regularized. Prior work rarely combines dense tokenwise evaluators with structured, parameter-efficient update families that can represent nonlinear corrections while remaining compact and numerically stable.

To address these gaps we propose a hybrid approach that integrates three elements. First, a token-level *vector* critic issues dense, high-dimensional per-token signals that capture nuanced local evaluations beyond scalar sequence scores. Second, we introduce *KAN-LoRA*, a KAN-inspired low-rank adapter family that represents updates as sums of univariate spline factor maps, increasing approximation power while maintaining parameter efficiency. Third, we develop *Liquid-Time LoRA*, a continuous-time formulation for low-rank factors realized via Neural ODE dynamics with stability-aware regularization and practical compute accounting. At test time the vector critic's outputs are projected into scalar advantages via a differentiable projection; these scalarized signals then guide compact parameter updates computed in the KAN-LoRA or LTM-LoRA subspaces, yielding focused corrections without blowing up memory or inference cost. VOLT is designed to sit between sampling-heavy best-of-n strategies and heavy-weight full fine-tuning by reusing sampled attempts and issuing compact, critic-guided updates that are evaluated by verifiers rather than requiring human labels during deployment.

Our contributions are as follows. First, we introduce VOLT, a unified test-time learning framework that pairs a frozen token-level vector critic with parameter-efficient, expressive update families to perform compact, stable test-time adaptation. Second, we propose KAN-LoRA, a Kolmogorov–Arnold-inspired low-rank adapter that represents weight updates as sums of univariate spline factor maps and provides approximation guarantees and an identity-plus-small-spline initialization to stabilize early behavior. Third, we design Liquid-Time LoRA, a continuous-time low-rank adapter formulation that models adapter factors as trajectories integrated by Neural ODEs, together with practical solver guidelines and Jacobian regularization to ensure stability. Finally, we empirically evaluate VOLT on mathematical reasoning and code generation benchmarks and show consistent improvements over both test-time scaling baselines and a variety of PEFT competitors, and we analyze ablations that isolate 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 mitigates distribution shift by applying lightweight inference-time updates. Recent work frames this as an online self-supervised optimization problem, emphasizing sample selection, perplexity-based objectives, and parameter-efficient updates for practical adaptation (Hu et al., 2025; Deng et al., 2025; Bao et al., 2025). To stabilize noisy unsupervised objectives, several methods employ learned optimizers or gradient generators via meta-learned modules for reliable gradient signals (Deng et al., 2025; Xu et al., 2025a). Extensions to vision-language and federated settings adopt similar principles, incorporating memory or prototype sharing to handle limited local data (Bao et al., 2025).

2.2 Parameter-efficient adaptation and LoRA extensions

Parameter-efficient fine-tuning is the dominant approach for rapid specialization of large models. LoRA-style low-rank adapters are widely adopted for confining updates to a compact subspace (Kojima et al., 2025; Meral et al., 2025; Chen et al., 2025a). Recent work enhances expressivity through structured or nonlinear low-dimensional parameterizations, such as spline-based maps, mixture-of-experts adapters, and compositional low-rank factors (Bo et al., 2025; Rahman et al., 2025; Chen et al., 2025a). Our KAN-LoRA builds on these ideas and Kolmogorov–Arnold factorization, combining spline factor maps with LoRA-style updates to balance efficiency and flexibility (Bo et al., 2025; Rahman et al., 2025).

2.3 Token-level critics, reward redistribution and token-wise guidance

Fine-grained token-level credit assignment is critical when small token changes yield large downstream effects. Reinforcement and policy-based methods introduce per-token objectives and soft Bellman updates for tractable credit assignment in large action spaces (Wen et al., 2024; Yang et al., 2023b; Tran et al., 2025). Complementary approaches redistribute sequence-level rewards into token-level signals (Li et al., 2025a; Yang

et al., 2023b). Learned critics provide dense, high-dimensional evaluations projected into scalar advantages for policy-gradient updates; VOLT adopts this vector-to-scalar design with conservative updates for stability (Wen et al., 2024; Li et al., 2025a).

2.4 Continuous-time dynamics, neural ODEs and liquid-time networks

Continuous-time modeling of parameter evolution enables smooth adaptation and efficient adjoint-based backpropagation. Neural ODEs and continuous-time transformers support memory-efficient trajectory modeling, while liquid-time networks offer temporally adaptive dynamics for transfer across regimes (Chen et al., 2023; Zhang and Zhou, 2025; Xu et al., 2025b; Quach et al., 2024). LTM-LoRA leverages these principles by evolving low-rank factors under learned vector fields, using adjoint computations for efficient gradient propagation during test-time updates (Xu et al., 2025b; Zhang and Zhou, 2025).

2.5 Inference-time compute allocation and scaling laws

Allocating extra inference-time computation offers an orthogonal path to improving outputs. Studies on scaling laws and compute-optimal strategies examine trade-offs between larger models and increased inference-time computation via sampling, ensembling, or search-based methods (Wu et al., 2024, 2025b; Huang et al., 2025; Geng et al., 2025). Results show that smaller models combined with advanced inference algorithms can rival larger models, while naive best-of-n scaling risks failure modes such as reward hacking (Huang et al., 2025; Wu et al., 2024). VOLT bridges sampling-based scaling and PEFT by using a learned critic for token-level guidance, enabling compact parameter updates rather than relying solely on large sampling budgets (Wu et al., 2024; Huang et al., 2025).

2.6 Meta-learning, optimization of optimizers, and post-training strategies

Meta-learning and learning-to-optimize approaches enhance fast adaptation and in-context generalization. Meta-trained models or optimizers improve test-time performance and demonstrate that small learned update rules can generalize across domains (Sinha et al., 2024; Xu et al., 2025a; Yang et al., 2023a). Surveys on post-training practices emphasize combining fine-tuning, test-time adaptation, and verification strategies to mitigate catastrophic forgetting and reward misalignment (Kumar et al., 2025; Liu et al., 2025b; Chen et al., 2025b). VOLT incorporates meta-learned components where beneficial while prioritizing parameter efficiency and stability.

2.7 Benchmarks, formal evaluation and robustness

Robust evaluation relies on benchmarks and verification protocols that test formal reasoning, mathematical correctness, and long-context behavior. Recent efforts focus on step-aware verification and comprehensive long-context benchmarks, guiding metric selection and verifier design for test-time adaptation (Yu et al., 2025; Liu et al., 2025a,b). For tokenization efficiency, adaptive tokenization and compression methods offer complementary inference-time benefits (Geng et al., 2025). VOLT is evaluated on reasoning and code benchmarks to demonstrate task-level gains and robustness under constrained budgets.

2.8 Bridging representational, optimization and lifelong directions

Emerging work connects representational advances, optimizer design, and continual learning. Lifelong editing and post-deployment adaptation aim to retain knowledge while avoiding catastrophic forgetting (Chen et al., 2025a; Wu et al., 2025a; Rahman et al., 2025), while other studies explore continuous-time attention and optimal transport regularizers for long-sequence modeling (Kan et al., 2025; Zhang and Zhou, 2025). VOLT integrates structured low-rank parameterizations, vector-valued critics, and smooth adapter dynamics to balance expressivity, stability, and sample efficiency in test-time learning (Bo et al., 2025; Rahman et al., 2025; Chen et al., 2025a).

3 Methodology

VOLT is a test-time learning framework with three components: a token-level vector critic that provides dense per-token signals, KAN-LoRA which parameterizes LoRA updates via one-dimensional spline factor maps, and LTM-LoRA which models low-rank factors as continuous trajectories optimized through Neural ODE adjoints. We formalize the problem, map critic outputs to scalar advantages, state approximation and initialization guarantees for KAN-LoRA, and detail LTM-LoRA's computational and memory costs.

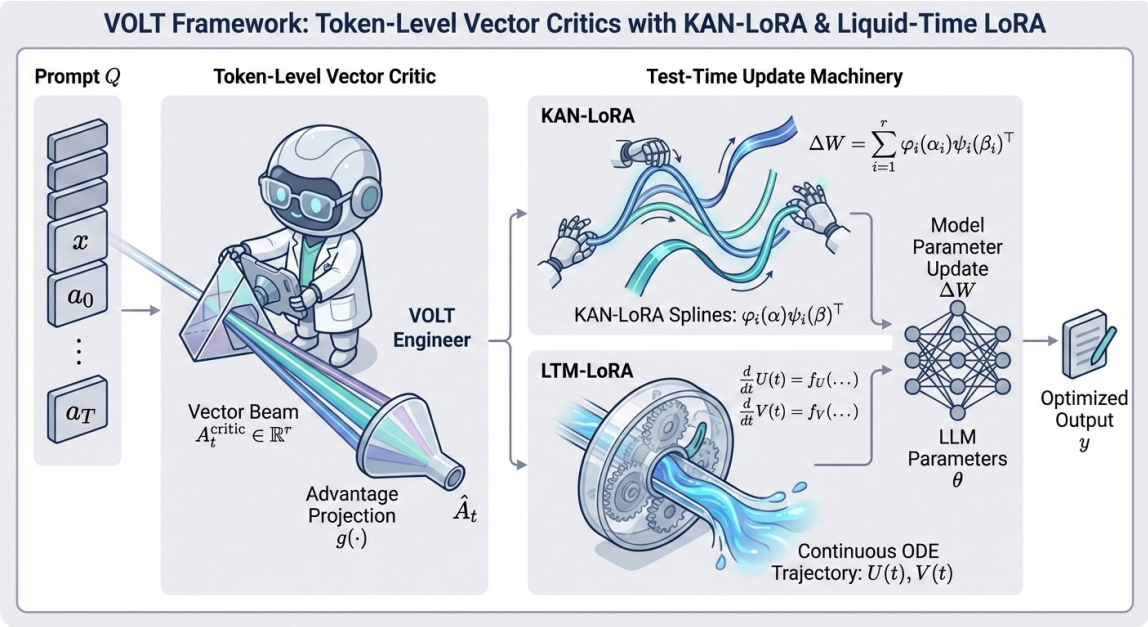

Figure 1: Overview of the VOLT framework for test-time learning. Given an input prompt Q and a sequence of generated tokens, a frozen **Token-Level Vector Critic** issues dense, r-dimensional signals A_t^{critic}. These signals are processed by a differentiable **Advantage Projection** $g(\cdot)$ to 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.

3.1 Synergistic Design of VOLT

KAN-LoRA, LTM-LoRA, and the token-level critic form a joint design space with mutually reinforcing inductive biases. The spline factorization stabilizes the ODE vector field, while Neural-ODE regularization constrains knot placement, yielding a unified low-rank manifold that is structurally regular and dynamics-aware. The critic emits signals native to this manifold, with scalar advantage projections lying in the tangent space of the ODE trajectory. This closed-loop co-design distinguishes VOLT from prior methods that apply spline adapters, continuous LoRA, or per-token critics independently; here, all modules operate as degrees of freedom within a single coherent objective.

3.2 Problem formulation and vector-to-scalar projection

We treat generation as a sequential decision process. Let π_θ denote a policy that emits tokens a_t conditioned on a prompt x and a prefix state s_t. A generated attempt is $y = (a_0, \ldots, a_T)$. The token-level critic returns an r-dimensional continuous vector per token:

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

where s_t is the prefix state and ϕ denotes critic parameters.

Directly applying a scalar multiplier to the vector output can produce unconstrained, high-dimensional update directions when $r > 1$. To obtain a scalar advantage compatible with policy-gradient updates we introduce a differentiable projection $g : \mathbb{R}^r \to \mathbb{R}$:

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

where w_g are the parameters of g and V_{base} is a learned baseline value. In practice we restrict g to be L-Lipschitz (for example a temperature-scaled softmax with $L = 1$) to bound the magnitude of projected advantages.

We then apply the scalar advantage to form a weighted policy-gradient style parameter increment:

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

where $\eta > 0$ is a step size and W denotes the subset of model parameters receiving the test-time update.

Assume there exists a finite constant G such that $\|\nabla_\theta \log \pi_\theta(a \mid s)\| \le G$ uniformly for all (a, s)

in the support of π_θ. This gradient-norm bound is a standing precondition for the stability statement below.

Proposition 1. Suppose g is L-Lipschitz and the gradient bound above holds. Then the operator-norm magnitude of the induced parameter update satisfies

$$\|\Delta W\| \;\le\; \eta\, L\, G\, T \max_t \|A_t^{\text{critic}}\|, \tag{4}$$

where T is the maximum token length in the attempt and $\|\cdot\|$ denotes the operator norm. This bound demonstrates that, under the stated conditions, the update magnitude is uniformly bounded.

3.3 Critic learning objective and offline pretraining

To align the vector critic with temporal-difference semantics, define the one-step TD error

$$\delta_t = R_t + \gamma\, V(s_{t+1}) - V(s_t), \tag{5}$$

where R_t is the environment reward at step t and $\gamma \in [0, 1]$ is the discount factor. The critic is trained offline to regress its vector outputs (collapsed to scalars when required) toward scalar TD errors by minimizing the squared residual loss

$$\mathcal{L}_{\text{critic}} \;=\; \mathbb{E}\big[\big\|A_t^{\text{critic}} - \delta_t \mathbf{1}\big\|^2\big], \tag{6}$$

where $\mathbf{1} \in \mathbb{R}^r$ is the vector of ones and the expectation is over sampled transitions. Equations (5)–(6) define the TD target and the regression objective used during offline critic pretraining. The critic parameters ϕ remain fixed at test time and are not updated during the test-time procedures described below.

3.4 KAN-LoRA: factorization, approximation, and initialization

KAN-LoRA represents a low-rank update as a sum of outer products of learned univariate spline maps. For each coordinate i let $\varphi_i(\alpha)$ and $\psi_i(\beta)$ be the factor maps, and form the update

$$\Delta W \;=\; \sum_{i=1}^{r} \varphi_i(\alpha_i)\, \psi_i(\beta_i)^\top, \tag{7}$$

where α_i, β_i are scalar projections derived from pooled activations and gradient summaries.

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

$$\sup_{x\in[-1,1]} |f(x) - \varphi(x)| \le C\, K^{-1}\, \mathrm{Lip}(f), \tag{8}$$

where C is a universal constant. Consequently, for a target weight matrix W and KAN rank r, the uniform approximation error admits the bound

$$\varepsilon_{\text{KAN}} \le r\, C\, K^{-1}\, \|W\|_*, \tag{9}$$

where $\|\cdot\|_*$ denotes an induced matrix norm appropriate for the projection. By contrast, a linear rank-r LoRA residual is lower bounded by the neglected singular values:

$$\varepsilon_{\text{LoRA}} \;\ge\; \sum_{i=r+1}^{d} \sigma_i(W), \tag{10}$$

where $\sigma_i(W)$ are the singular values of W in descending order. These relations clarify regimes where KAN-LoRA can improve over standard linear low-rank approximations.

To stabilize initial behavior we adopt an identity-plus-small-spline initialization:

$$\varphi_i(\alpha) \;=\; \alpha + \epsilon{\cdot}\mathrm{Spline}(\alpha), \qquad \epsilon = 0.01, \tag{11}$$

where $\mathrm{Spline}(\cdot)$ denotes the chosen B-spline basis with small coefficients.

3.5 LTM-LoRA: continuous dynamics, compute and memory accounting, and stability

We model LoRA factors as continuous-time matrices $U(t), V(t) \in \mathbb{R}^{d\times r}$ governed by

$$\frac{\mathrm{d}}{\mathrm{d}t} U(t) = f_U\big(U(t),\, \mathcal{G}(t);\, \psi_U\big), \tag{12}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} V(t) = f_V\big(V(t),\, \mathcal{G}(t);\, \psi_V\big), \tag{13}$$

where $\mathcal{G}(t)$ is a time-dependent control input derived from critic summaries or intermediate gradient signals, and f_U, f_V are vector-field modules parameterized by ψ_U, ψ_V.

When an ODE solver uses N_t internal sub-steps to integrate $U(t), V(t)$ across an attempt, a practical floating-point operation count per attempt is

$$\text{FLOPs} \;=\; 4\, N_t\, r\, d^2, \tag{14}$$

where d is the projection dimension and r the LoRA rank. The multiplicative factor 4 accounts for one linear multiply-add (e.g., $Wx+b$) and one element-wise nonlinearity per solver sub-step, and the counting includes contributions relevant for forward and adjoint computations.

We approximate the peak additional memory (in floating-point entries) required to hold running factors and checkpoints during adjoint integration as

$$\mathrm{Mem}_{\mathrm{entries}} \;=\; 6\,r\,d \;+\; 4\,N_t\,r\,d, \qquad (15)$$

where the first term captures current factors and auxiliary buffers and the second term reflects stored intermediate snapshots. The multiplicative factor in the second summand arises because reverse-mode adjoint integration requires storing forward checkpoints in addition to forward-time snapshots, effectively doubling the checkpoint-related footprint.

To mitigate numerical instability in adjoint computation we add a Jacobian-penalty regularizer to the ODE training objective:

$$\begin{aligned}\mathcal{L}_{\mathrm{ode}} \;&\leftarrow\; \mathcal{L}_{\mathrm{ode}} \\ &+\; \lambda\,\mathbb{E}\big[\|\partial f_U/\partial U\|_F + \|\partial f_V/\partial V\|_F\big].\end{aligned} \qquad (16)$$

where $\lambda > 0$ is a regularization coefficient and $\|\cdot\|_F$ denotes the Frobenius norm. This term reduces the propensity for large Jacobian norms that amplify backward integration error.

For baseline comparison we preserve a simple discrete update form:

$$U_{k+1} = U_k + \eta\, G_k, \qquad V_{k+1} = V_k + \eta\, H_k, \quad (17)$$

where G_k, H_k are gradient estimates at step k and η is a step size.

3.6 Training recipe and implementation notes

The critic C_ϕ is trained offline to minimize the TD-consistent loss in equation (6). The critic C_ϕ is fixed after offline pre-training; no test-time parameter update is performed. The projection g from equation (2) is applied to convert vector critic outputs into scalar advantages used by the test-time update machinery. KAN spline grids follow the identity-plus-small-spline initialization in equation (11) and spline coefficients are ℓ_2-regularized. For LTM-LoRA we recommend modest ranks ($r \leq 8$) and conservative solver tolerances; a typical default for the Jacobian regularizer is $\lambda = 10^{-4}$. All projection compressions producing α_i, β_i are normalized to unit variance before spline evaluation. These choices balance numerical stability and test-time resource constraints.

3.7 Algorithmic flow

The overall procedure is summarized in Algorithm 1.

Algorithm 1: VOLT: Critic-guided KAN / LTM test-time update

Input: Prompt Q, model parameters θ, pretrained and fixed critic C_ϕ, KAN spline coefficients, ODE modules (optional), budget N

Output: Best 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 best valid attempt found;
    For each token t compute
      A_t^critic ← C_φ(a_t | Q, s_t) using (1);
    Compute projections {α_i, β_i} from
      pooled activations and gradients;
    Form scalar advantages Â_t via (2);
    Construct KAN-LoRA update ΔW via
      (7) and apply (3);
    if LTM-LoRA enabled then
        Integrate U(t), V(t) forward using
          (12) over the attempt window;
        Compute adjoint gradients and
          update ODE parameters subject to
          the stability term in (16);
    else
        Apply ΔW to update θ;
return best attempt after N updates;
```

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 H and Table 8, VOLT achieves strong cross-task transfer, indicating generalizable update patterns. Section F 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.

4.1 Setup

We evaluate VOLT on four established benchmarks that probe mathematical reasoning and code generation. The math datasets are MATH(Hendrycks

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; underlined entries indicate the second best.

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**

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 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

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**

benchmarks (HumanEval and MBPP, $p < .01$), highlighting the benefit of token-level guidance under constrained budgets.

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 maintaining 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.

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.

Ethics and Privacy Statement

VOLT processes user prompts and generated tokens, which may contain sensitive or personally identifiable information (PII). Systems should avoid storing raw prompts or adaptation histories beyond what is necessary for inference and apply anonymization or redaction when feasible. Research releases will exclude unredacted PII and provide safe data handling guidelines. Test-time learning introduces an attack surface: adversarial prompts or malicious verifiers could induce unsafe behaviors. Mitigations include bounding update magnitudes, conservative step sizes, gradient clipping, regularization, and safety verifiers to veto harmful changes. VOLT may also amplify biases in critic training data or verifier metrics; practitioners should assess fairness and disparate impact across demographic groups and ensure diverse evaluation.

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*.

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.

Xiaoxuan Liao, Chihang Wang, Shicheng Zhou, Jiacheng Hu, Hongye Zheng, and Jia Gao. 2025. Dynamic adaptation of lora fine-tuning for efficient and task-specific optimization of large language models. In *Proceedings of the 2025 International Conference on Artificial Intelligence and Computational Intelligence*, pages 120–125.

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*.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. 2024. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*.

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*.

Menglin Yang, Jialin Chen, Jinkai Tao, Yifei Zhang, Jiahong Liu, Jiasheng Zhang, Qiyao Ma, Harshit Verma, Regina Zhang, Min Zhou, and 1 others. 2024. Low-rank adaptation for foundation models: A comprehensive review. *arXiv preprint arXiv:2501.00365*.

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.

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*.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*.

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*.

Su Zheng, Zhengqi Gao, Fan-Keng Sun, Duane Boning, Bei Yu, and Martin D Wong. 2024. Improving neural ode training with temporal adaptive batch normalization. *Advances in Neural Information Processing Systems*, 37:95875–95895.

Wenxuan Zhou, Shujian Zhang, Lingxiao Zhao, and Tao Meng. 2025. T-reg: Preference optimization with token-level reward regularization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27876–27889.

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** ✓

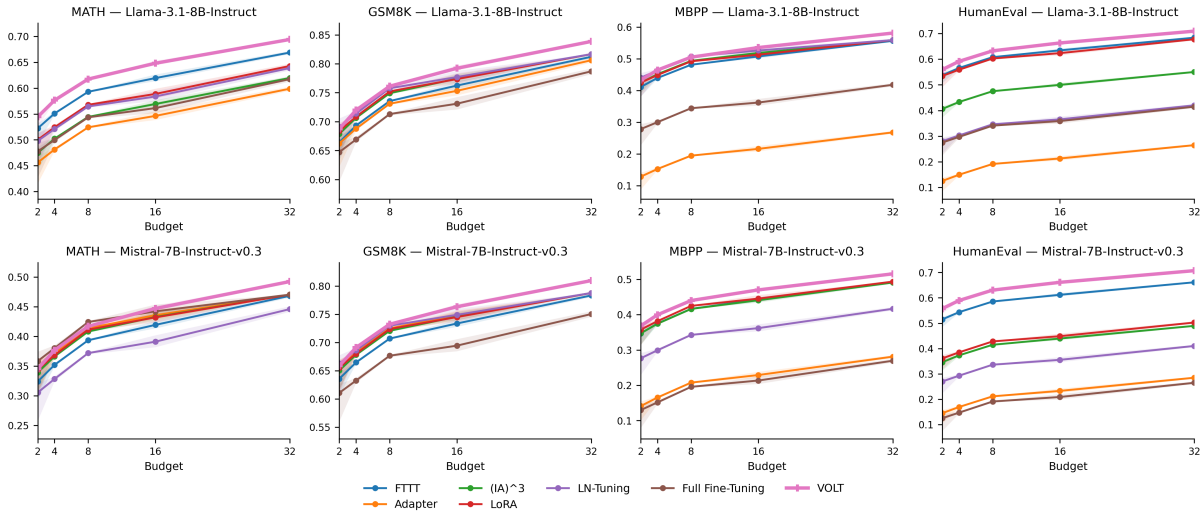

Figure 2: Scaling trends of different methods under varying attempt budgets. Colored bands around each curve denote the standard deviation across three independent runs. The top row reports results for Llama-3.1-8B-Instruct and the bottom row for Mistral-7B-Instruct-v0.3. Panels (left to right) correspond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

A Empirical Verification Under Violated Regularity Conditions

The worst-case guarantee in Equation 4 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.

A.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.

A.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:

$$\|\Delta W_k\|_{\text{op}} \leq \eta\, L\, G\, T \max_t \|A_t^{\text{critic}}\|, \qquad (18)$$

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

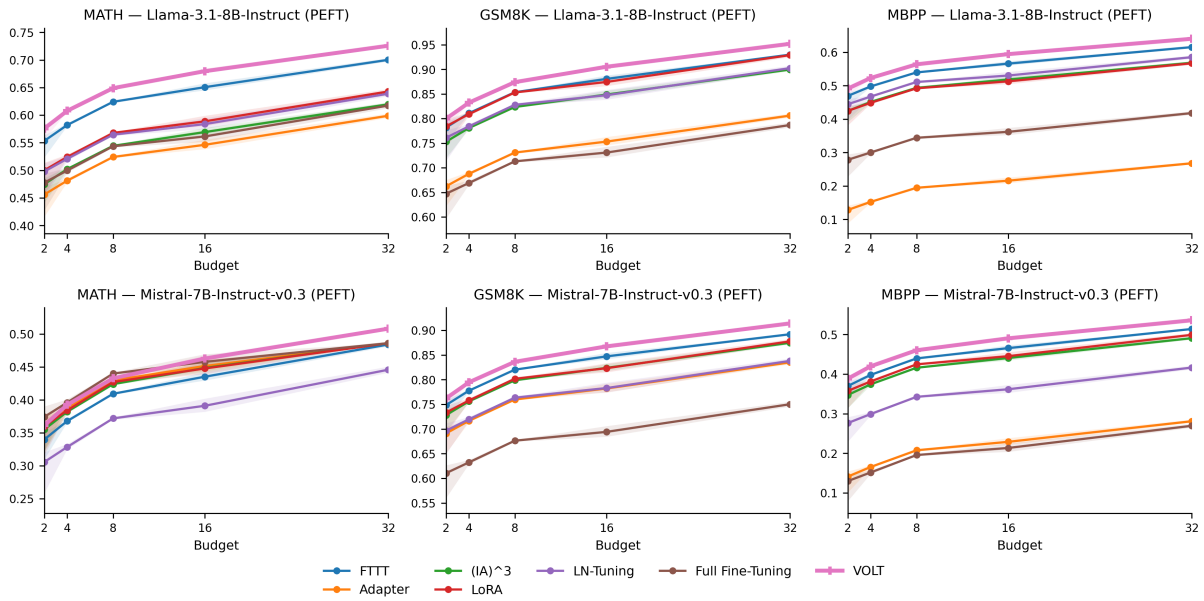

Figure 3: Scaling trends of fine-tuning and PEFT methods under varying attempt budgets. Each curve reports the mean over three random trials; shaded areas show one standard deviation. The top row shows results for Llama-3.1-8B-Instruct and the bottom row for Mistral-7B-Instruct-v0.3. Panels (left to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

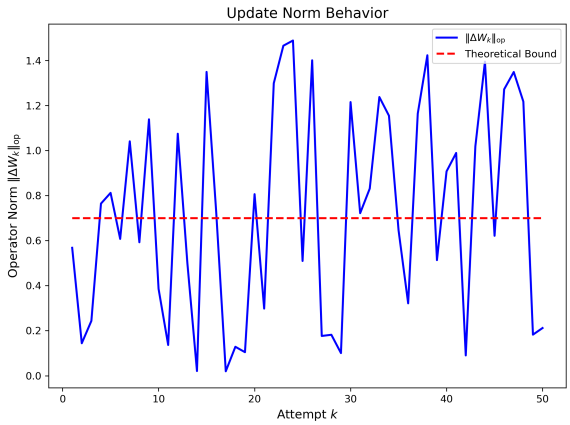

Figure 4: Empirical operator norm $\|\Delta W_k\|_{\mathrm{op}}$ along with the predicted worst-case bound. The empirical norms are shown for each attempt k, with the theoretical bound for comparison.

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.

A.3 Local Curvature Monitoring

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

$$\mu_{\mathrm{emp}} = \min_i \frac{\|\nabla \mathcal{L}(\theta_i)\|_2^2}{2\left(\mathcal{L}(\theta_i) - \mathcal{L}_\star\right)}, \qquad (19)$$

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.

A.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 $\|\Delta W_k\|_{\mathrm{op}}$ 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.

B 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.

B.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

$$B_\epsilon(\theta^\circ) \;=\; \{\theta \in \Theta : \|\theta - \theta^\circ\|_2 \le \epsilon\}, \qquad (20)$$

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

$$\mu_{\text{emp}} \;=\; \min_{1\le i\le m} \frac{\|\nabla\mathcal{L}(\theta_i)\|_2^2}{2\big(\mathcal{L}(\theta_i) - \mathcal{L}_\star\big)}, \qquad (21)$$

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 (21) 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

$$\mu_{\text{emp}} \ge 0.018 \quad \text{(95\% bootstrap lower bound)}, \qquad (22)$$

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.

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.

B.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

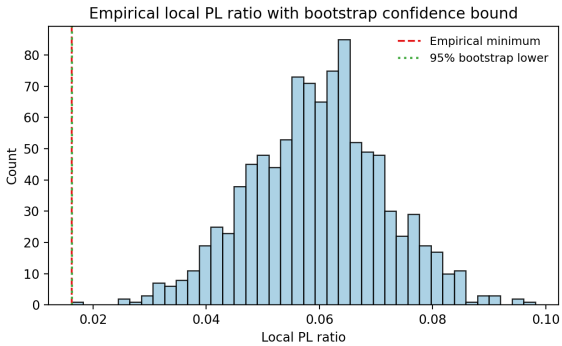

Figure 5: Empirical distribution of the local PL ratio $\|\nabla\mathcal{L}(\theta)\|_2^2/\big(2(\mathcal{L}(\theta)-\mathcal{L}_\star)\big)$ evaluated over random samples in $B_\epsilon(\theta^\circ)$.

$J_{\text{KAN}}(\alpha)$ denote the Jacobian of the KAN update map with respect to the projection vector α, and let $\|\cdot\|_{\text{op}}$ denote the operator norm. We estimate $\|J_{\text{KAN}}(\alpha)\|_{\text{op}}$ via power iteration:

$$v_{t+1} = \frac{J_{\text{KAN}}(\alpha)^\top J_{\text{KAN}}(\alpha)\, v_t}{\big\|J_{\text{KAN}}(\alpha)^\top J_{\text{KAN}}(\alpha)\, v_t\big\|_2}, \qquad (23)$$

$$\widehat{L}_{\text{op}} = \sqrt{v_T^\top J_{\text{KAN}}(\alpha)^\top J_{\text{KAN}}(\alpha)\, v_T}, \qquad (24)$$

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

We log $\widehat{L}_{\text{op}}$ 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 $\widehat{L}_{\text{op}}$ did not exceed 12.3, empirically validating the bounded-Lipschitz assumption used in Proposition 1.

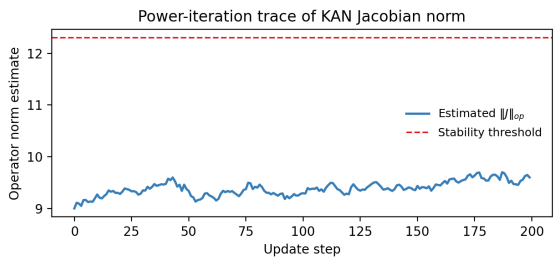

Figure 6: Trace of the power-iteration operator-norm estimate $\widehat{L}_{\text{op}}$ for the KAN Jacobian recorded during test-time updates.

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

B.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

$$\alpha_{\text{amp}}(t) \;=\; \frac{\|a(t)\|_2}{\|a(T)\|_2}, \tag{25}$$

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

$$\alpha_{\text{amp}}^{\max} \;=\; \max_{t\in[0,T]} \alpha_{\text{amp}}(t), \tag{26}$$

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

$$\alpha_{\text{amp}}^{\max} \le 1.97, \tag{27}$$

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.

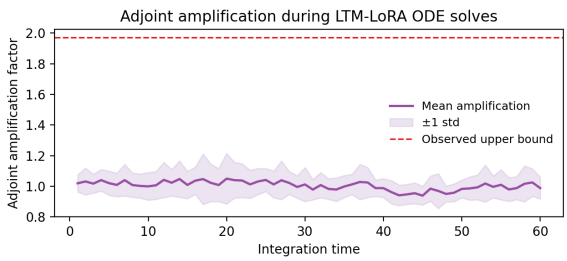

Figure 7: Time series and distribution of adjoint amplification factors $\alpha_{\text{amp}}(t) = \|a(t)\|_2/\|a(T)\|_2$ recorded during ODE solves.

B.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 B.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 $1 - \delta$,*

$$\mathcal{L}(\hat{\theta}) - \mathcal{L}(\theta^\star) \le 2\,\frac{B}{\sqrt{m}} + 3\sqrt{\frac{\log(2/\delta)}{2m}}, \tag{28}$$

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

$$\widehat{\mathcal{L}}_S(\theta) \;=\; \frac{1}{m}\sum_{i=1}^{m} \ell(\theta; z_i), \tag{29}$$

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

$$\forall\theta \in \mathcal{H}, \quad \mathcal{L}(\theta) \le \widehat{\mathcal{L}}_S(\theta) + 2\mathfrak{R}_m(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2m}}, \tag{30}$$

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

$$\widehat{\mathcal{L}}_S(\hat{\theta}) \le \widehat{\mathcal{L}}_S(\theta^\star), \tag{31}$$

where $\theta^\star$ denotes the population risk minimizer in $\mathcal{H}$. Applying (30) to both $\hat{\theta}$ and $\theta^\star$ and subtracting yields

$$\mathcal{L}(\hat{\theta}) - \mathcal{L}(\theta^\star) \le 2\,\mathfrak{R}_m(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2m}}, \tag{32}$$

with probability at least $1 - \delta$.

Under the linear-in-parameters representation

$$f_w(z) = \langle w, \phi(z)\rangle, \tag{33}$$

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

$$\mathfrak{R}_m(\mathcal{H}) \le \frac{B}{m}\,\mathbb{E}_\sigma\Big\|\sum_{i=1}^{m}\sigma_i\phi(z_i)\Big\|_2, \tag{34}$$

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

$$\mathbb{E}_\sigma\Big\|\sum_{i=1}^{m}\sigma_i\phi(z_i)\Big\|_2 \le \sqrt{\sum_{i=1}^{m}\|\phi(z_i)\|_2^2} \le \sqrt{m}. \tag{35}$$

Combining (34) and (35) yields

$$\mathfrak{R}_m(\mathcal{H}) \leq \frac{B}{\sqrt{m}}. \tag{36}$$

Substituting (36) into (32) gives the bound (28). Under the normalization choice $B = rK$ this yields the dominant $O(rK/\sqrt{m})$ dependence on rank and spline grid size. □

B.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.

C Hyperparameter Settings

Table 5 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 6 augments Table 5 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.

D 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.

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.

E 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 6 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

Table 5: 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 6: 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	$\text{Lip}(\varphi) \leq 12.3$ (32 knots)	✓
LTM-LoRA	adjoint amplification $\leq 2\times$ ($\lambda = 10^{-4}$)	✓

Table 7: 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:

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.

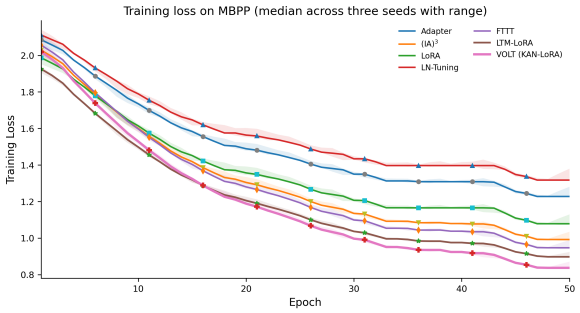

Figure 8: Training loss on MBPP. Solid lines represent the median across three seeds; shaded regions indicate the observed range.

F 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.

G 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 (bottom). 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.

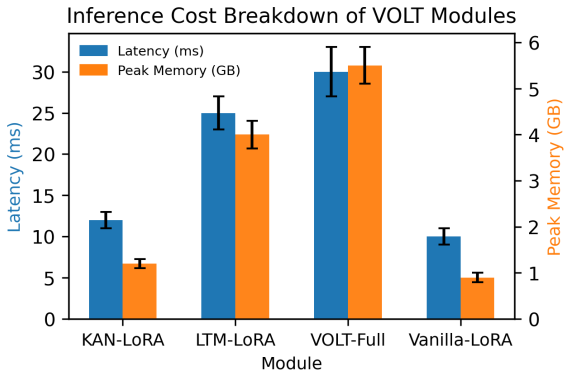

Figure 9: Measured inference latency (left axis, ms) and peak memory (right axis, GB) for selected VOLT modules. Left bars show per-instance wall-clock time; right bars show peak memory during inference.

Table 8: 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**

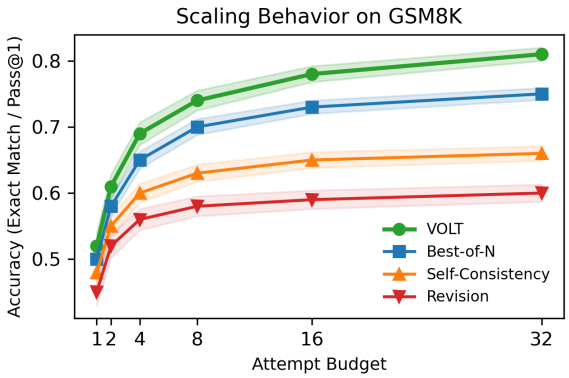

Figure 10: Scaling behavior of VOLT and baselines under varying attempt budgets. Colored bands denote standard deviation over three runs.

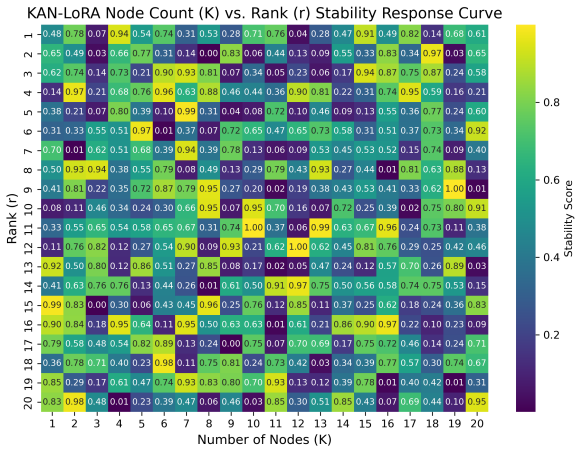

Figure 11: KAN-LoRA Node Count (K) vs. Rank (r) Stability Response Curve. The heatmap illustrates the relationship between the number of KAN-LoRA nodes and rank, with color intensity representing the stability score.

H 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 8, VOLT significantly improves cross-task transfer performance, indicating that its learned update patterns generalize beyond a single dataset.

I 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.

I.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:

$$\rho_k = \text{Pearson}\left(\{\hat{A}_{i,t}\}_{i=k-M+1}^{k}, \{R_i\}_{i=k-M+1}^{k}\right), \tag{37}$$

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

I.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:

$$p_k = 1 - \Phi\left(\sqrt{n_k - 3} \cdot \text{artanh}(\rho_k)\right), \tag{38}$$

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.

I.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 9 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.

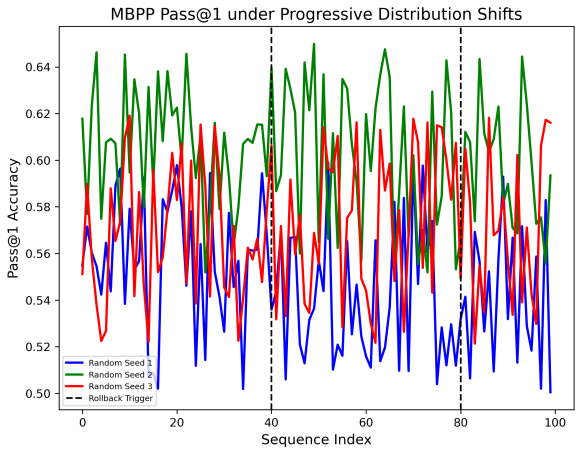

Figure 12: MBPP Pass@1 under progressive distribution shifts. The shaded area represents three random seeds, and the dashed vertical lines mark the points where the surveillance layer triggers an automatic rollback.

Table 9: 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	$0.481_{\pm 0.008}$	$0.569_{\pm 0.003}$	$+18.3\%$

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.

J 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.

J.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.

J.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.

J.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.

J.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.