LEARNING WHILE GENERATING IN A SINGLE FORWARD PASS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deployed LLMs constantly receive feedback from users, verifiers, and task outcomes. Test-time adaptation uses this feedback so that each interaction improves the next one. Today, this learning needs a second pass after feedback arrives, which either reruns the model over the response or keeps all of its activations until then. We observe that the model computation in this second pass does not depend on the feedback, so it can be done in the forward pass that generates the response. Based on this observation, we propose **One-Pass Forward-Trace Adaptation (OFTA)**, which learns while generating in a single forward pass. As the model generates each token, OFTA also records how the log-probability of the response changes along a random parameter direction. This design brings three benefits. First, it removes the second pass. When feedback arrives, OFTA turns the record into an update with a few simple operations, without running the adapting model again, using backpropagation, or keeping activations. Second, the record works for immediate and delayed feedback, and one record supports many feedback losses. Third, moving the computation into generation costs nothing in update quality. Theoretically, OFTA gives the same directional derivative as a separate pass after feedback, so it keeps the unbiased estimate of forward-gradient methods. Experiments on Qwen3 series models show that, in the tested settings and counting all extra work, OFTA lowers request time, applies updates almost as soon as feedback arrives, uses much less memory than backpropagation, and learns as well as a separate pass. In simpler terms, OFTA lets a model learn from a response in the same pass that writes it.

1 INTRODUCTION

Deployed large language models (LLMs) constantly receive signals that could improve their future behavior. Users flag poor responses, verifiers check generated solutions, and environments return outcomes after actions. Test-time adaptation (TTA) uses such feedback during deployment, so that one interaction improves the ones that follow (Akyürek et al., 2025; Hu et al., 2025; Zweiger et al., 2025). For TTA to work in practice, learning from an interaction should cost little more than serving.

Today, learning from an interaction takes a second pass through the model. Backpropagation, which parameter-efficient methods such as LoRA (Hu et al., 2022) still use, must keep the activations of the whole response until feedback arrives, or recompute them afterward (Chen et al., 2016). Zeroth-order methods such as MeZO (Malladi et al., 2023) rerun the model at perturbed parameters over the full prompt and response. Forward-gradient methods avoid the backward pass (Silver et al., 2022; Baydin et al., 2022; Panchal et al., 2024). They still need the loss before they start. So all of these methods need a second pass once the feedback arrives. This pass delays the update and repeats work that generation has just done. This leads to our central question.

> *Can the forward pass that generates a response also do the computation needed to learn from it?*

Our answer is yes. How response statistics, such as the sequence log-probability, change with the parameters depends only on the model and the response. The feedback only decides how much weight each change receives. By the chain rule, the gradient of the loss is the product of these two factors. The factor from the feedback needs no model computation. The factor that measures how the

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

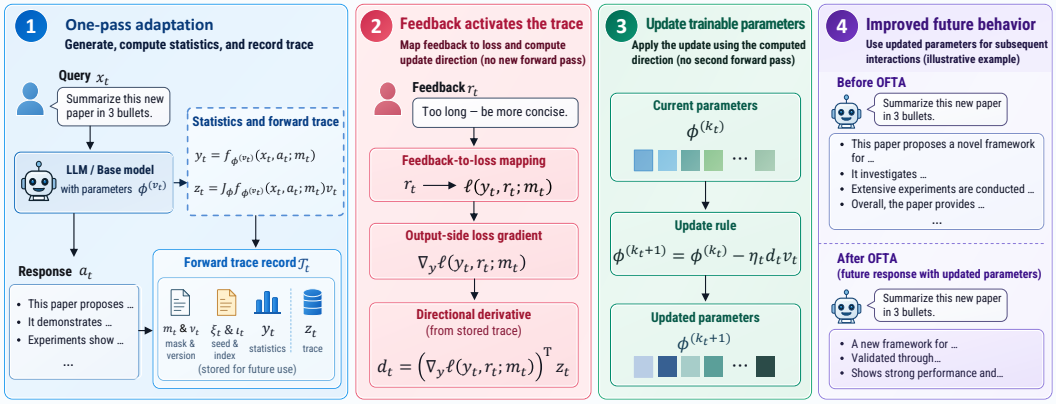

Figure 1: **Learning while generating in a single forward pass.** (1) While the model generates the response a_t to the query x_t, OFTA also computes the response statistics y_t (e.g., the sequence log-probability) and their forward trace z_t along a random direction v_t. OFTA stores them in the trace record $\mathcal{T}_t$. (2) When feedback r_t arrives, OFTA maps it to a loss. It combines the loss gradient $\nabla_y \ell$ with the stored trace to get d_t. This step needs no new forward pass. (3) OFTA updates the trainable parameters along v_t with step size $\eta_t d_t$. (4) Later requests use the updated parameters (illustrative example). Reported costs include the extra computation and memory used during generation.

response statistics change does not need the feedback. Forward-mode differentiation computes this second factor token by token, in the same order as generation. So this factor fits inside the forward pass that generates the response.

In this work, we propose **One-Pass Forward-Trace Adaptation (OFTA)**, which learns while generating in a single forward pass (Figure 1). At a high level, OFTA splits learning into two steps. During generation, OFTA computes how the response statistics change along a random parameter direction and stores the result as a compact trace. When feedback arrives, OFTA combines the trace with the loss derivative and updates the parameters with a few simple operations. It does not run the adapting model again, does not use backpropagation, and does not keep activations. The trace stays exact no matter how long the feedback takes. One trace also supports many feedback losses, such as REINFORCE, GRPO-style group objectives, and DPO-style losses on recorded responses (Section 3). OFTA does not introduce a new gradient estimator. It keeps the forward gradient and moves all of its model computation into the generation pass.

We support this design with theory. Theoretically, we prove that the trace computed during generation gives exactly the same directional derivative as a separate pass after feedback (Section 5). OFTA therefore keeps the unbiasedness of forward gradients. Under on-policy reward feedback, its expected update follows the expected-reward gradient. We also bound the error when other updates happen before the feedback arrives. This bound depends only on how far the parameters move.

We also support it with experiments on Qwen3 series models, counting all extra work done during generation (Section 6). The traces match independent backpropagation on all 61 tested responses. For responses of 32 to 256 tokens, OFTA cuts request time by 32–37% compared with backpropagation and zeroth-order replay. It applies each update 0.68 s after verifier feedback on average, which is 3 to 5 times sooner than the baselines. With short outputs at batch size 16, it uses over 40% less memory than backpropagation. Learning only from verifier feedback, OFTA corrects 60 of 100 questions that the frozen model had failed. This matches a baseline that computes the same update in a separate pass after feedback. In simpler terms, a model can learn from a response in the same pass that writes it.

In summary, our contributions are as follows.

- We show that learning from feedback does not need a second pass, because its model computation does not depend on the feedback.
- We propose OFTA, which learns while generating in a single forward pass and supports immediate or delayed feedback and many feedback losses.

- We prove that OFTA gives the exact directional derivative. We show experimentally that OFTA lowers request time and memory cost, and learns as well as a separate pass.

2 Related Work

Test-time adaptation updates a model on incoming test data (Sun et al., 2020; Wang et al., 2021). For LLMs, recent work learns from test-time context, self-generated data, or rewards during reasoning (Akyürek et al., 2025; Hu et al., 2025; Zweiger et al., 2025; Tandon et al., 2025; Zuo et al., 2025). These works decide what the model learns from. OFTA changes how the learning is computed, so it can serve their objectives when they fit its interface (Section 3).

Parameter-efficient methods choose which parameters to train, and zeroth-order methods estimate gradients without backpropagation (Hu et al., 2022; Malladi et al., 2023; Zhang et al., 2024; Liu et al., 2025). OFTA works with any chosen parameter space. It also avoids the reruns that zeroth-order methods need after feedback. Forward gradients, including versions for LLM fine-tuning (Baydin et al., 2022; Silver et al., 2022; Xu et al., 2024; Panchal et al., 2024), compute directional derivatives without backpropagation. Eligibility traces (Tallec & Ollivier, 2018; Bellec et al., 2020) compute sensitivities online and combine them with learning signals that arrive later. OFTA builds on both ideas. Our contribution is to make them work for LLM generation with delayed feedback. OFTA computes directional derivatives of declared response statistics during generation, including through the prompt prefill and the KV cache. It combines them with the loss derivative when feedback arrives (Proposition 5.1 and Section 6.4). Appendix A covers further work.

3 Problem Setup: Learning within Generation

For prompt x_t, interaction t generates a response $a_t = (a_{t,1}, \ldots, a_{t,T_t})$ and later receives feedback r_t. The trainable parameters are $\phi \in \mathbb{R}^p$, with versions $\phi^{(j)}$. The response is generated with the *generation-time parameters* $\phi^{(\nu_t)}$. Its update is applied to the *current parameters* $\phi^{(k_t)}$, with $k_t \geq \nu_t$, and produces version $k_t + 1$.

OFTA must not revisit the response to learn from it. So the loss can depend on the parameters only through values computed during generation. Before generation, we declare which statistics $y_t \in \mathbb{R}^{d_y}$ of the response to compute. We also declare a loss family ℓ that is differentiable in these statistics. The response and its feedback are not known in advance. The generation metadata m_t^{gen} can be extended to m_t with extra loss inputs, such as fixed reference scores.

$$\begin{aligned} y_t &= f_{\phi^{(\nu_t)}}(x_t, a_t; m_t^{\text{gen}}), \\ L_t(\phi) &= \ell(f_\phi(x_t, a_t; m_t^{\text{gen}}), r_t; m_t). \end{aligned} \tag{1}$$

Tokens, stopping positions, and reference values are held fixed when we differentiate.

Most LLM feedback objectives fit this interface, because they depend on the parameters only through log-probabilities of generated responses. On-policy REINFORCE, GRPO-style group-normalized advantages (Shao et al., 2024), and DPO- or KTO-style losses on recorded responses (Rafailov et al., 2023; Ethayarajh et al., 2024) use sequence scores as y_t. Only feedback that needs the model to score a sequence it did not generate, such as a user-written correction, requires extra model computation. Our experiments evaluate four representative objectives (Appendix B.5).

The *directional trace* z_t is the derivative of y_t along a direction v_t. The *trace record* $\mathcal{T}_t$ stores z_t together with y_t, a direction seed, and data that links the record to its feedback (Eq. (3)). The *historical derivative* is the directional derivative of the loss at the generation-time parameters. *Without rerunning* means no further pass through the adapting model.

OFTA has four design goals, and each one removes a cost named in Section 1. (P1) *One pass.* OFTA computes the trace in the same prefill and decoding steps that produce the response. (P2) *No backpropagation.* OFTA keeps no activations for a backward pass. (P3) *Update without replay.* OFTA forms the update from the record, the feedback, and the extra loss inputs alone. (P4) *Delay tolerance.* The record stays exact until feedback arrives (Section 5). One pass does not mean one forward call or equal FLOPs. We count all extra work.

Algorithm 1 OFTA update for one interaction

Require: Declared statistics f, loss family ℓ, step size η_t, fixed coordinate layout and direction rule.

1: Sample a seed ξ_t and a direction $v_t \sim \mathcal{N}(0, I_p)$ at the generation-time parameters $\phi^{(\nu_t)}$.
2: Generate a_t. In the same pass, accumulate y_t and $z_t = J_\phi f_{\phi^{(\nu_t)}} v_t$ through the prefill and KV cache.
3: Deliver a_t, keep only $\mathcal{T}_t = (y_t, z_t, \xi_t, \nu_t, m_t^{\text{gen}}, \iota_t)$, and release the caches.
4: When feedback r_t arrives, compute $u_t = \nabla_y \ell(y_t, r_t; m_t)$ and $d_t = u_t^\top z_t$.
5: Regenerate v_t from ξ_t and update $\phi^{(k_t+1)} \leftarrow \phi^{(k_t)} - \eta_t d_t v_t$.

4 One-Pass Forward-Trace Adaptation

Why this design. Each part of OFTA follows from one goal. OFTA must form the update without a post-feedback pass through the adapting model. (i) The trace must be computed before the feedback exists. So the loss can depend on the parameters only through statistics declared in advance (Section 3). (ii) The derivatives of these statistics must be computed in generation order, before any loss is known. Forward-mode differentiation does exactly this. (iii) Forward mode gives one Jacobian-vector product per direction. A random direction keeps the trace at d_y numbers, while the full Jacobian needs $d_y \times p$ numbers. The random direction still gives an unbiased estimate (Theorem 5.2). Any direction law whose conditional second moment is the identity works. (iv) Later tokens reuse keys and values that depend on the parameters. So tangents must also flow through the KV cache. Our detached-cache controls detect when these tangents are missing (Section 6.3). (v) The parameters and the direction stay fixed for each response. So the trace is the derivative of one function at one point. A seed is enough to recover the direction. (vi) Updates that happen before the feedback arrives make the derivative historical. Section 5 bounds this staleness. Algorithm 1 gives the full procedure.

4.1 Computing the Trace during Generation

At the generation-time parameters $\phi^{(\nu_t)}$, OFTA draws $v_t \sim \mathcal{N}(0, I_p)$ from a seed ξ_t. Both stay fixed for the whole response. The trainable coordinates can be full weights, LoRA factors, heads, subsets, or any differentiable reparameterization. Their layout and the direction rule stay fixed until feedback arrives. LoRA is not required for exactness. Our experiments use LoRA adapters, single-layer parameter subsets, and normalization parameters (Appendices D.3 and D.5). During prefill and cached decoding, each operation computes its output together with its tangent. Sampled tokens are held fixed. This gives

$$y_t = f_{\phi^{(\nu_t)}}(x_t, a_t; m_t^{\text{gen}}), \qquad z_t = J_\phi f_{\phi^{(\nu_t)}}(x_t, a_t; m_t^{\text{gen}}) v_t. \tag{2}$$

The tangent computation reuses the weights and cached states of the normal (primal) computation, so both run in the same decoding step. Its overhead depends on the implementation, and we measure it in Section 6.4. The construction is not specific to transformers. It only needs a differentiable recurrence over reused state that depends on the parameters (Appendix F), such as the KV cache or the state of a recurrent or state-space model. After the response is delivered, OFTA releases the caches and keeps only the trace record

$$\mathcal{T}_t = (y_t, z_t, \xi_t, \nu_t, m_t^{\text{gen}}, \iota_t). \tag{3}$$

Here ι_t links the record to its feedback and extra inputs. The seed regenerates v_t, so old parameters need not be stored. Responses sampled for one prompt share the parameters and directions, so a group loss can combine their traces. With K directions, each direction adds its own tangent computation and cache, and a group of N responses keeps $O(N(1+K))$ numbers.

4.2 Forming the Update when Feedback Arrives

When the feedback and any extra inputs are ready, OFTA computes $u_t = \nabla_y \ell(y_t, r_t; m_t)$. It then takes the inner product with the stored trace.

$$d_t = u_t^\top z_t = D_{v_t} L_t(\phi^{(\nu_t)}), \tag{4}$$

$$\phi^{(k_t+1)} = \phi^{(k_t)} - \eta_t d_t v_t. \tag{5}$$

The historical derivative thus updates the current parameters. The cost is computing u_t, one inner product, regenerating v_t, and writing the parameters. With K directions, $\widehat{g}_t = K^{-1} \sum_j (u_t^\top z_{t,j}) v_{t,j}$ replaces $d_t v_t$. The estimate can also feed stateful optimizers. Appendix B.3 derives the update for APEX, and Appendix F.3 treats regularizers.

In our protocols, which records are kept, how many are stored, and which ones receive feedback do not depend on the sampled direction (Appendix E.2). Section 5 explains why this keeps the update unbiased. Our request times count all work, including tangent computation during generation. OFTA is faster whenever this overhead is smaller than the pass it removes, which holds at every tested length from 32 to 256 tokens (Section 6.4). Appendix C.2 also describes pre-feedback JVP variants, which start a separate pass after delivery and before feedback.

5 Theoretical Analysis

OFTA changes when the derivative is computed. It computes the derivative during generation, before the loss is known. It applies the derivative after feedback, possibly after other updates. This raises three questions, and each result below answers one of them. Question 1 asks whether the derivative computed during generation is the right one. Question 2 asks whether an update along one random direction follows the gradient. Question 3 asks how much error delayed feedback adds. Throughout, L_t is the realized loss in Eq. (1), with tokens, stopping, metadata, random scorer state, and reference values fixed. The dimension $p = \dim(\phi)$ counts the coordinates that receive tangents and updates. Proofs appear in Appendix F.

Question 1. Is the trace exact? OFTA computes z_t before the loss exists, and later tokens reuse cached states that depend on the parameters. So it is not obvious that $u_t^\top z_t$ equals the derivative that a separate pass after feedback would compute. If the two differed, OFTA would only approximate the forward gradient. Proposition 5.1 shows that they are equal.

Proposition 5.1 (Generation-time trace exactness)**.** *Assume the declared statistics are differentiable and can be computed from the executed generation graph. Keep $\phi^{(\nu_t)}$ and v_t fixed throughout prefill and decoding, and propagate tangents through every later state that depends on the parameters. Then, without rounding errors, the computed pair satisfies Eq.* (2)*. For every supported loss,*

$$\nabla_y \ell(y_t, r_t; m_t)^\top z_t = \nabla_\phi L_t(\phi^{(\nu_t)})^\top v_t. \tag{6}$$

A separately implemented scorer is a valid reference only if its value and Jacobian at the generation-time parameters equal those of the executed fixed-response program.

The key condition is that tangents flow through every reused state, including the KV cache. This condition is necessary. Without cache tangents, the derivative misses how the parameters affect later tokens through earlier keys and values, and our controls in Section 6.3 detect this error. The last sentence of the proposition states when an independent implementation is a valid check, which Section 6.3 uses. The feedback time does not appear in the proof, so the result holds for immediate and delayed feedback alike.

Question 2. Is the update unbiased? Proposition 5.1 gives the exact derivative along one direction v_t. This is only one component of the full gradient. To use OFTA as a stochastic gradient method, the update $d_t v_t$ must follow the full gradient on average. Theorem 5.2 shows that it does.

Theorem 5.2 (Historical estimator unbiasedness)**.** *Let $\mathcal{C}_t$ determine the realized objective and the generation-time parameters, so that $g_t = \nabla_\phi L_t(\phi^{(\nu_t)})$ is $\mathcal{C}_t$-measurable. Let $\widehat{g}_t = (g_t^\top v_t) v_t$. If $\mathbb{E}[v_t v_t^\top \mid \mathcal{C}_t] = I_p$, then*

$$\mathbb{E}[\widehat{g}_t \mid \mathcal{C}_t] = g_t. \tag{7}$$

This is the standard forward-gradient guarantee (Baydin et al., 2022; Silver et al., 2022). It applies to OFTA because Proposition 5.1 makes d_t exact. An inexact trace would add an error to every update. The moment condition is also necessary in practice. If the choice of which records receive updates depends on the direction, for example by dropping records with large tangents, the estimate becomes biased (Appendix E.2). OFTA therefore keeps this choice independent of the direction (Section 4.2). Under on-policy reward feedback, the expected update is a policy-gradient step on the expected

reward (Corollary F.2 in Appendix F.5). The price of using one direction is variance. Averaging K independent Gaussian directions gives mean-squared error $(p+1)\|g_t\|^2/K$ (Corollary F.3), so more directions trade computation for lower error (Appendix D.5).

Question 3. How much error does delayed feedback add? Feedback may arrive after other updates. OFTA then applies $d_t^{\text{hist}} = d_t$, the derivative at the generation-time parameters $\phi^{(\nu_t)}$, to the current parameters $\phi^{(k_t)}$. We need to know how far it is from the derivative at the current parameters, $d_t^{\text{fresh}} = \nabla L_t(\phi^{(k_t)})^\top v_t$. Proposition 5.3 bounds this gap.

Proposition 5.3 (Drift-dependent historical–fresh discrepancy)**.** *If ∇L_t is $L_{\nabla,t}$-Lipschitz on the segment between $\phi^{(\nu_t)}$ and $\phi^{(k_t)}$, then*

$$|d_t^{\text{hist}} - d_t^{\text{fresh}}| \le L_{\nabla,t}\|\phi^{(k_t)} - \phi^{(\nu_t)}\|\,\|v_t\|. \tag{8}$$

With the same learning rate η_t, the two SGD steps differ by at most $\eta_t L_{\nabla,t}\|\phi^{(k_t)} - \phi^{(\nu_t)}\|\,\|v_t\|^2$.

The bound depends only on how far the parameters move between generation and update. If no other update happens, the gap is zero for any waiting time. In our sequential experiments, each update happens before the next interaction, so $k_t = \nu_t$ and the derivative is current.

6 Experiments

We first check that OFTA's update rule learns well under known supervision (Section 6.2). We then ask three research questions. Research Question 1 asks whether generation computes the exact trace (Section 6.3). Research Question 2 asks what learning during generation saves (Section 6.4). Research Question 3 asks whether learning from generated responses improves future answers (Section 6.5). Appendix D.5 gives ablations on the number of directions and the adaptation space.

6.1 Experimental Setup

Unless noted, generation studies use Qwen3-1.7B with FP32 LoRA on all linear layers ($r = 8$, $\alpha = 64$, zero dropout, 8,716,288 coordinates). The known-supervision study uses fused scalar-loss JVPs on Qwen3-1.7B and Qwen3-8B. Appendix D.4 specifies the corrective-QA study.

APEX (Accuracy Projection from Verifier-Labeled Experience, Appendix B.3) is a group loss for binary verifier feedback. For the N responses sampled for a question, it compares each response's log-probability ratio to a frozen reference model. A logistic loss then favors verified responses and penalizes failed ones. We chose APEX because it depends on the parameters only through the N recorded sequence log-probabilities. So it tests a coupled objective whose feedback arrives after generation.

BP replay and MeZO replay score the response again after feedback. Post-feedback JVP computes the same derivative as OFTA in a separate pass after feedback. It uses the same response, parameters, objective, and direction. With a matched backend, it isolates the effect of where the derivative is computed (Appendix C.2). In fixed-response comparisons, all methods share the model, parameter space, objective, parameters, tokens, masks, directions, precision, rewards, and reference scores. They differ only in where the derivative is computed. In sequential runs, each method generates with its own updated parameters.

Each APEX update uses one question, $N = 8$ responses, and $K = 1$. Each method adapts on the first 50 questions in a fixed order. We evaluate it with greedy decoding at the prespecified update 50, on 100 separate questions. We select questions that the frozen model answers wrongly with greedy decoding, and for which 4 to 6 of its 8 sampled answers are correct (Appendix B.1). We then test the frozen model on the selected questions again. Its measured greedy accuracy is 0%, which confirms the selection. These questions come from the GSM8K training split and do not overlap with the adaptation questions.

6.2 Task Quality under Controlled Adaptation

Under known supervision, OFTA trains successfully and matches MeZO in task quality on both model sizes (Table 1). On Qwen3-1.7B, OFTA wins on six tasks, ties on one, and loses on three by at

most 1.30 points. On Qwen3-8B, OFTA wins on eight tasks, ties on one, and loses on one by 1.03 points. OFTA and MeZO use the same 20,000-update budget. BP uses exact gradients with 2,000 updates, serves as a reference, and stays stronger on most tasks. The following sections test the effect of computing derivatives during generation.

Table 1: Task quality (%) under known supervision on Qwen3-1.7B and Qwen3-8B, averaged over seeds 0–2. OFTA and MeZO use 20,000 directional updates, and BP uses 2,000. Bold marks the better of OFTA and MeZO. Numbers in parentheses are differences from MeZO.

Task	SST-2	BoolQ	CB	RTE	WiC	COPA	ReCoRD	SQuAD	DROP	MultiRC
Metrics	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	F1	F1	Acc.
Qwen3-1.7B										
OFTA	**91.63** (+1.15)	**80.60** (+1.20)	**87.50** (+0.00)	**81.59** (+0.36)	59.40 (-0.32)	**77.00** (+1.00)	65.10 (-1.30)	86.11 (-0.78)	**42.19** (+0.55)	**82.40** (+0.10)
MeZO	90.48	79.40	**87.50**	81.23	**59.72**	76.00	**66.40**	**86.89**	41.64	82.30
BP	92.66 (+2.18)	82.20 (+2.80)	96.43 (+8.93)	85.20 (+3.97)	68.81 (+9.09)	72.00 (-4.00)	65.10 (-1.30)	87.97 (+1.08)	54.91 (+13.27)	84.40 (+2.10)
Qwen3-8B										
OFTA	91.20 (-1.03)	**83.40** (+0.60)	**88.00** (+1.60)	**89.20** (+0.00)	**74.20** (+0.40)	**90.00** (+0.40)	**67.80** (+0.20)	**89.57** (+1.09)	**46.75** (+0.85)	**83.60** (+0.80)
MeZO	**92.23**	82.80	86.40	**89.20**	73.80	89.60	67.60	88.48	45.90	82.80
BP	93.13 (+0.90)	83.50 (+0.70)	96.50 (+10.10)	89.80 (+0.60)	80.31 (+6.51)	90.00 (+0.40)	68.00 (+0.40)	90.17 (+1.69)	60.27 (+14.37)	86.40 (+3.60)

6.3 Research Question 1. Does Generation Compute the Exact Trace?

Yes. We compare derivatives computed during generation with independent cached JVPs and backpropagation projections. The comparison covers 19 parameter and input points, 61 responses, and six actual updates (Appendix D.1). We use the directional criterion

$$|d_{\text{generation}} - d_{\text{reference}}| \leq 10^{-5} + 2 \times 10^{-4} |d_{\text{reference}}|. \tag{9}$$

Losses and parameter updates have their own error budgets. All **61/61** cached-BP comparisons pass, with no sign disagreement (Figure 2). The primary checks pass at all 19 points, together with 30 APEX group checks. The six resulting APEX updates match an independent reconstruction.

The checks also catch errors. We inject wrong directions, zero derivatives, detached KV-cache tangents, and missing group candidates. The checks detect all 53 informative response controls and all six group controls (Appendix D.1). One more single-token cache control leaves the derivative unchanged, so it carries no information. A separate teacher-forced graph computes the derivative in a different way, and it passes **60/61** conditioned checks (Appendix D.2).

6.4 Research Question 2. What Does Learning during Generation Save?

With a 204-token prompt, OFTA is 35.0–37.1% faster than BP replay and 31.6–35.6% faster than MeZO replay at every tested response length from 32 to 256 tokens (Figure 4). A matched test isolates the effect of where the derivative is computed. In a single-token setting with one trainable matrix ($p = 8{,}192$), OFTA and post-feedback JVP share the implementation, loss, initialization, direction, and update rule. They produce identical updates in all 96 requests. OFTA lowers the median request time by 44.0–45.1% (Appendix D.3). So computing the trace during generation costs less than the pass it replaces. On responses of about seven tokens, a teacher-forced backend is faster than our eager token-wise backend. On the same token-wise backend, OFTA remains about 16% faster than post-feedback JVP (Appendix D.6).

On APEX, the feedback simulates a user who rates each response as good or bad, a common form of feedback on chat and coding platforms. OFTA applies each update 0.68 s after this feedback on average, 3.0–5.1× sooner than the baselines (Figure 3a). After feedback, OFTA only needs a few

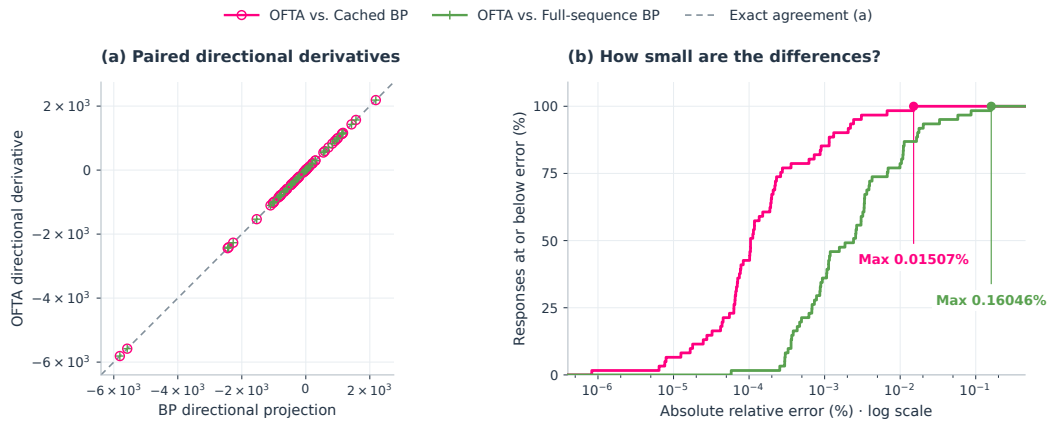

Figure 2: **Derivatives computed during generation match independent backpropagation** on the same 61 responses at 19 fixed points. (a) OFTA directional derivatives against cached and full-sequence (teacher-forced) BP projections. The dashed line marks equality. (b) CDF of the relative error $100\,|d_{\text{gen}} - d_{\text{BP}}|/|d_{\text{BP}}|$ on a log scale. The medians are 0.0001% and 0.0024%, and the maxima are 0.015% and 0.16%.

output-side operations, direction regeneration, and parameter writes. The baselines run the model over all eight responses again.

At batch size 16, OFTA's median peak memory for 1- and 4-token outputs is 43.1% and 41.9% below BP. It is 7.0% and 9.5% above MeZO (Figure 3b). These full-step measurements use short, replicated inputs (Appendix D.7).

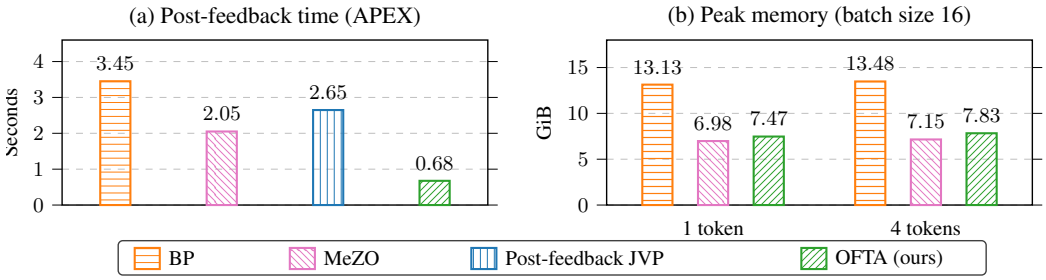

Figure 3: **Post-feedback time and memory on separate workloads.** (a) Mean APEX time from the verifier result to the applied update, including reference work (50 seed-211 requests per method, Appendix B.2). (b) Median peak GPU allocation of a complete step at $B = 16$, with replicated inputs and 1- or 4-token outputs (18 processes and 108 measurements, Appendix D.7).

6.5 Research Question 3. Does Learning from Feedback Improve Future Answers?

Yes. OFTA learns only from verifier feedback on its own responses. After 50 updates, it corrects 60 of the 100 evaluation questions that the frozen model had failed (Figure 5). Post-feedback JVP corrects 61, and BP replay corrects 67.

OFTA learns as well as post-feedback JVP. This is expected, because the two methods compute the same update from the same response and parameters. Controlled comparisons confirm this exactly for 96 of 96, 960 of 960, and 8 of 8 updates (Appendices D.3, D.4 and E.3). In the online runs, each method generates its own responses, so the two trajectories can drift apart. The one-point gap reflects this drift. It does not reflect a difference in the update rule.

BP replay uses exact gradients and serves as a reference. Its larger gain per update matches the gap between forward gradients and backpropagation in Table 1. This gap is due to the gradient estimator. It is not due to where the gradient is computed. We report the prespecified endpoint at update 50.

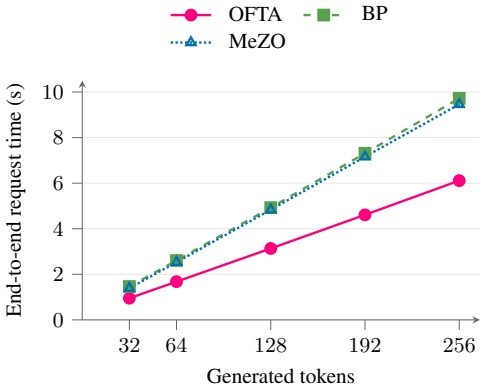

Figure 4: End-to-end request time for different output lengths (204-token prompt). Time includes generation with tangents, feedback, and one completed update. Setup is excluded. Points are medians of three process medians, and bars show the range across processes. Parameters are reset for each request.

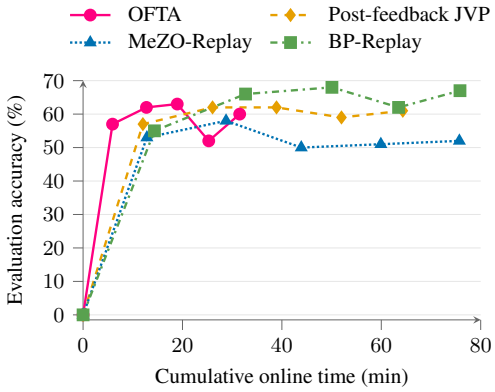

Figure 5: Accuracy against online time (APEX, checkpoints every 10 updates). We report greedy accuracy on 100 previously exposed questions from the GSM8K training split that the frozen model failed. They do not overlap with the adaptation questions. Online time sums each method's request-to-update intervals and excludes setup and evaluation. BP replay uses same-condition reruns (Appendix B.1). The other methods use seed 211. Lines connect measured checkpoints.

7 Limitations and Conclusion

Limitations. OFTA covers feedback that can be written as a loss on statistics recorded during generation, which includes the common objectives in Section 3. Feedback that asks the model to score new text, such as a user-written correction, needs extra model computation on that text. Like all forward-gradient methods, OFTA estimates the gradient along random directions, so each update is noisier than a backpropagation step. Averaging more directions reduces this noise (Section 5). OFTA saves the most time on responses of moderate or long length, which are typical in chat and coding. For very short responses, replay can be faster (Section 6.4). Our online study uses simulated good or bad feedback on GSM8K. Evaluating OFTA with real user feedback in larger deployments is a natural next step.

Conclusion. We showed that an LLM can learn while generating in a single forward pass. OFTA computes the needed derivatives in the same pass that generates the response. When feedback arrives, immediately or later, OFTA forms the update from the stored trace without running the adapting model again. Theoretically, this update uses the exact derivative at the generation-time parameters and stays unbiased. In our experiments, OFTA trains as well as zeroth-order methods on both Qwen3 model sizes. It lowers request time on the tested response lengths, applies updates within a second of feedback on average, and uses much less memory than backpropagation. It also learns from good or bad feedback as well as a separate pass that computes the same update. More broadly, if a feedback objective depends on the model only through values computed during generation, learning from that feedback does not need to revisit the response.

AI use statement

AI tools were used to aid and polish writing, support retrieval and discovery of related work, and draft sections of this paper. Writing assistance covered organization, clarity, terminology, and figure

captions; retrieval assistance identified potentially relevant literature; drafting assistance produced text for author revision. These uses concern manuscript preparation and do not replace verification of cited sources, mathematical arguments, or empirical evidence. Responsibility for the final content rests with the authors.

Ethics statement

Our experiments use public benchmarks (GLUE, SuperGLUE, SQuAD, DROP, and GSM8K) and simulated feedback. They involve no human subjects or private user data. Any method that updates a deployed model from feedback carries risks. Malicious or wrong feedback can corrupt updates, stored records can contain sensitive content, and parameter drift can degrade earlier behavior. Deployments should use consented and minimized data, authenticated feedback, update audits, drift monitoring, and a tested rollback path.

Reproducibility statement

Section 4 and Algorithm 1 define OFTA. Appendix F states all assumptions and gives complete proofs of Proposition 5.1, Theorem 5.2, and Proposition 5.3. Appendix B specifies the APEX objective, the GSM8K question selection, and the request clocks. Appendix C.1 lists the model revision, trainable coordinates, data partitions, seeds, and optimizer settings for every study (Table 2). Appendices D and E describe the numerical checks and the delayed-feedback protocols. Anonymized code for all experiments is included in the supplementary material.

References

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=asgBo3FNdg.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018. URL http://jmlr.org/papers/v18/17-468.html.

Atilim Gunes Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation, 2022. URL https://arxiv.org/abs/2202.08587.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11:3625, 2020. doi: 10.1038/s41467-020-17236-y.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016. URL https://arxiv.org/abs/1604.06174.

Yiming Chen, Yuan Zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. Enhancing zeroth-order fine-tuning for language models with low-rank structures. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9BiVepgmWW.

Sizhe Dang, Yangyang Guo, Yanjun Zhao, Xiaodong Zheng, Guang Dai, Ivor Tsang, and Haishan Ye. FZOO: Fast zeroth-order optimizer for fine-tuning large language models towards Adam-scale speed. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=NMlF3YjS8E.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OUIFPHEgJU.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*,

pp. 12634–12651. PMLR, 2024. URL https://proceedings.mlr.press/v235/ethayarajh24a.html.

Yujie Feng, Xujia Wang, Zexin Lu, Shenghong Fu, Guangyuan Shi, Yongxin Xu, Yasha Wang, Philip S. Yu, Xu Chu, and Xiao-Ming Wu. Recurrent knowledge identification and fusion for language model continual learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27396–27413, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1328. URL https://aclanthology.org/2025.acl-long.1328/.

Louis Fournier, Stephane Rivaud, Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Can forward gradient match backpropagation? In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10249–10264. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/fournier23a.html.

Tanmay Gautam, Youngsuk Park, Hao Zhou, Parameswaran Raman, and Wooseok Ha. Variance-reduced zeroth-order methods for fine-tuning language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=VHO4nE7v41.

Wentao Guo, Jikai Long, Yimeng Zeng, Zirui Liu, Xinyu Yang, Yide Ran, Jacob R. Gardner, Osbert Bastani, Christopher De Sa, Xiaodong Yu, Beidi Chen, and Zhaozhuo Xu. Zeroth-order fine-tuning of LLMs with transferable static sparsity. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=myYzr50xBh.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jinwu Hu, Zitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-time learning for large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=iCYbIaGKSR.

L. Isserlis. On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables. *Biometrika*, 12(1/2):134–139, 1918. ISSN 00063444, 14643510. URL http://www.jstor.org/stable/2331932.

Jiyeon Kim, Hyunji Lee, Dylan Zhou, Sue Hyun Park, Seunghyun Yoon, Trung Bui, Franck Dernoncourt, Sungmin Cha, and Minjoon Seo. Can large language models keep up? Benchmarking online adaptation to continual knowledge streams. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42240–42272, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1956. URL https://aclanthology.org/2026.acl-long.1956/.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNfLdxr3A.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1950–1965, 2022. URL https://openreview.net/forum?id=rBCvMG-JsPd.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24bn.html.

Yong Liu, Zirui Zhu, Chaoyu Gong, Minhao Cheng, Cho-Jui Hsieh, and Yang You. Sparse MeZO: Less parameters for better performance in zeroth-order LLM fine-tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Tjw0ACu3NL.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Vota6rFhBQ.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=6ZBHIEtdP4.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16888–16905. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/niu22a.html.

Yann Ollivier, Corentin Tallec, and Guillaume Charpiat. Training recurrent networks online without backtracking, 2015. URL https://arxiv.org/abs/1507.07680.

Kunjal Panchal, Nisarg Parikh, Sunav Choudhary, Lijun Zhang, Yuriy Brun, and Hui Guan. Thinking forward: Memory-efficient federated finetuning of language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=dGQtja9X2C.

Peter Phan, Dhruv Agarwal, Kavitha Srinivas, Horst Samulowitz, Pavan Kapanipathi, and Andrew McCallum. MiGrATe: Mixed-policy GRPO for adaptation at test-time, 2025. URL https://arxiv.org/abs/2508.08641.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward

model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JxpBP1JM15-.

Tao Ren, Zishi Zhang, Jinyang Jiang, Guanghao Li, Zeliang Zhang, Mingqian Feng, and Yijie Peng. FLOPS: Forward learning with OPtimal sampling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=z1nSpA2dAW.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28, pp. 3528–3536. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/de03beffeed9da5f3639a621bcab5dd4-Paper.pdf.

Sifeng Shang, Jiayi Zhou, Chenyu Lin, Minxian Li, and Kaiyang Zhou. Fine-tuning quantized neural networks with zeroth-order optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mGeeRFToaW.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

David Silver, Anirudh Goyal, Ivo Danihelka, Matteo Hessel, and Hado van Hasselt. Learning by directional gradient descent. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=5i7lJLuhTm.

Chenyang Song, Xu Han, Zheni Zeng, Kuai Li, Chen Chen, Zhiyuan Liu, Maosong Sun, and Tao Yang. ConPET: Continual parameter-efficient tuning for large language models, 2023. URL https://arxiv.org/abs/2309.14763.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Corentin Tallec and Yann Ollivier. Unbiased online recurrent optimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJQDjk-0b.

Qitao Tan, Jun Liu, Zheng Zhan, Caiwen Ding, Yanzhi Wang, Xiaolong Ma, Jaewoo Lee, Jin Lu, and Geng Yuan. Harmony in divergence: Towards fast, accurate, and memory-efficient zeroth-order LLM fine-tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Rx6m16By6l.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context, 2025. URL https://arxiv.org/abs/2512.23675.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7191–7201, 2022. doi: 10.1109/CVPR52688.2022.00706.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/.

Mengwei Xu, Dongqi Cai, Yaozong Wu, Xiang Li, and Shangguang Wang. FwdLLM: Efficient federated finetuning of large language models with perturbed inferences. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pp. 579–596, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-41-0. URL https://www.usenix.org/conference/atc24/presentation/xu-mengwei.

Yifan Yang, Kai Zhen, Ershad Banijamali, Athanasios Mouchtaris, and Zheng Zhang. AdaZeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 977–995, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.56. URL https://aclanthology.org/2024.emnlp-main.56/.

Wanhao Yu, Ziyan Wang, Zheng Wang, Abeer Matar Almalky, Yihang Zuo, Shuteng Niu, Sen Lin, Adnan Siraj Rakin, Deliang Fan, and Li Yang. Dominant-layer ZO: A single layer dominates zeroth-order fine-tuning of LLMs, 2026. URL https://arxiv.org/abs/2606.05516.

Ziming Yu, Pan Zhou, Sike Wang, Jia Li, Mi Tian, and Hua Huang. Zeroth-order fine-tuning of LLMs in random subspaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4475–4485, October 2025.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

Qingyang Zhang, Xinke Kong, Haitao Wu, Qinghua Hu, Minghao Wu, Baosong Yang, Yu Cheng, Yun Luo, Ganqu Cui, and Changqing Zhang. TEMPO: Scaling test-time training for large reasoning models, 2026. URL https://arxiv.org/abs/2604.19295.

Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D. Lee, Wotao Yin, Mingyi Hong, Zhangyang Wang, Sijia Liu, and Tianlong Chen. Revisiting zeroth-order optimization for memory-efficient LLM fine-tuning: A benchmark. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=THPjMr2r0S.

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor Tsang. Second-order fine-tuning without pain for LLMs: A hessian informed zeroth-order optimizer. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=bEqI61iBue.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=VuVhgEiu20.

Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=JsNUE84Hxi.

A Extended Related Work

Test-time and continual adaptation. Test-time training, entropy minimization, efficient sample selection and continual adaptation study learning after deployment (Sun et al., 2020; Wang et al., 2021; Niu et al., 2022; Wang et al., 2022). For LLMs, O-LoRA, ConPET and Recurrent-KIF address sequential knowledge retention (Wang et al., 2023; Song et al., 2023; Feng et al., 2025); test-time training, TLM and SEAL study updates from test inputs or self-generated training signals (Akyürek et al., 2025; Hu et al., 2025; Zweiger et al., 2025). OAKS evaluates online adaptation to evolving knowledge (Kim et al., 2026). End-to-End Test-Time Training and MiGrATe further motivate comparing test-time learning mechanisms by their training conditions, objective and available feedback (Tandon et al., 2025; Phan et al., 2025). TTRL and TEMPO develop reasoning-time reward and policy-refinement procedures (Zuo et al., 2025; Zhang et al., 2026). OFTA addresses the placement of sensitivity acquisition inside response generation, complementing work on feedback design and continual-learning strategies.

Trainable spaces. Adapters, prefixes, soft prompts and activation scaling restrict the parameters changed during adaptation (Houlsby et al., 2019; Li & Liang, 2021; Lester et al., 2021; Liu et al., 2022). LoRA and its variants additionally change rank allocation, backbone precision, parameter sharing or initialization (Hu et al., 2022; Zhang et al., 2023; Dettmers et al., 2023; Kopiczko et al., 2024; Liu et al., 2024; Meng et al., 2024). These choices determine the geometry and cost of learning. OFTA uses LoRA in its LLM experiments and formulates sensitivity acquisition in a declared differentiable coordinate space, allowing the execution mechanism to accommodate different parameterizations.

Zeroth-order optimization. MeZO estimates a directional derivative with two perturbed loss evaluations (Malladi et al., 2023); broader LLM studies examine ZO algorithms, parameterizations and forward-gradient alternatives (Zhang et al., 2024). Variance reduction and low-dimensional structure are studied by MeZO-SVRG, AdaZeta, LOZO, static sparsity and SubZero (Gautam et al., 2024; Yang et al., 2024; Chen et al., 2025; Guo et al., 2025; Yu et al., 2025). Sparse MeZO and Dominant-Layer ZO also target which coordinates should receive updates (Liu et al., 2025; Yu et al., 2026). HiZOO, DiZO, FZOO and QZO address other optimization or execution costs (Zhao et al., 2025; Tan et al., 2025; Dang et al., 2026; Shang et al., 2026). For the same realized loss, two-sided finite differences use $[L(\phi+\mu v) - L(\phi-\mu v)]/(2\mu)$. One-sided reuse additionally requires the same parameters, response, objective and random state for the baseline value. Preparing perturbed outputs before feedback moves their acquisition cost and retained state into the pre-feedback phase. OFTA obtains the exact directional sensitivity of the declared differentiable program through forward AD during generation.

Forward gradients and online sensitivities. Directional descent and forward-gradient learning predate OFTA (Silver et al., 2022; Baydin et al., 2022). Local losses, informed directions and query allocation address their variance (Ren et al., 2023; Fournier et al., 2023; Ren et al., 2025); FwdLLM and Spry study backward-free or forward-AD LLM adaptation (Xu et al., 2024; Panchal et al., 2024). OFTA integrates this directional learning rule into serving by separating the feedback-independent trace $J_\phi f_\phi v$ from the feedback-dependent coefficient $\nabla_y \ell$ and acquiring the former inside the serving traversal. Pre-feedback JVP starts a separate pass after generation; post-feedback JVP starts it after feedback. The pre-feedback controls include full-prefix and cached implementations. Historical replay uses the acquisition parameters and Fresh Replay uses the application parameters. These distinctions define the execution controls in Appendix C.2.

Eligibility traces and online sensitivities provide a conceptual precedent for combining earlier sensitivity with a later learning signal (Bellec et al., 2020; Ollivier et al., 2015; Tallec & Ollivier, 2018). OFTA specializes this separation to objective-facing directional records from autoregressive generation, with explicit acquisition, retention, feedback readiness and versioned application.

B APEX OBJECTIVE AND FEEDBACK INTERFACE

APEX (*Accuracy Projection from Verifier-Labeled Experience*) is the response-level objective used in the verifier-feedback experiments. It combines response scores, binary verifier outcomes and a fixed reference policy, providing a concrete interface for OFTA's acquisition and activation mechanism.

B.1 MAIN GSM8K CORRECTION PROTOCOL

GSM-CORR studies correction of questions on which a frozen Qwen3-1.7B model fails greedily but produces successful sampled answers. The candidate source is the 7,473-question GSM8K training split. We exclude 291 source IDs used for tuning (64), monitoring (192), warmup (3), and prior adaptation (32). We also exclude questions or effective tokenized prompts matching these reserved examples and retain only the first occurrence of each remaining effective prompt in the fixed candidate order.

Screening uses the original LoRA bank, FP32 with TF32 disabled, and the chat template with thinking disabled. Prompts are left-truncated to 512 tokens; responses end at EOS or 512 generated tokens. Each question receives a greedy response. For a greedy failure, eight responses are sampled with temperature 1, top-$p = 1$, top-$k = 0$ and fixed question-specific decoding seeds, then scored by the boxed-numeric verifier. Eligibility requires a greedy failure and four to six correct sampled responses. A complete scan of 7,182 candidates yields 252 eligible questions.

Eligible questions are ordered by sampled correctness count (6, then 5, then 4), with ties resolved by SHA256 of the fixed screen identifier, selection seed 20260922 and source index. The first 200 questions are alternately assigned to adaptation and evaluation, then separately ordered by a role-specific SHA256 key. Each role contains 100 questions: 43 with six correct samples, 36 with five, and 21 with four.

The comparison uses a common 50-interaction budget: each method consumes the first 50 questions in the fixed adaptation order and publishes one APEX update per question. All methods share the endpoint at update 50 and the evaluation schedule at updates 0, 10, 20, 30, 40 and 50. Each checkpoint is evaluated by greedy final-answer accuracy on the 100 evaluation questions. The update-0 evaluation tests the frozen model again on these questions. Its measured accuracy is 0%, which confirms the screening result. The original four trajectories, one per method, use master seed 211. Each online event generates eight new candidates under that method's current parameters, so $B = 1$ counts the question group, $N = 8$ its candidates, and $K = 1$ the direction used by directional methods.

In Figure 5, the BP-Replay curve comes from same-condition reruns. The other three curves and the shared initial checkpoint use seed 211.

The run uses one H100 80GB per worker, FP32 with TF32 disabled, directional-SGD learning rate 3×10^{-6}, BP AdamW rate 10^{-5}, and MeZO radius 10^{-4}. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$ and zero weight decay.

RQ1-S19 independently examines derivatives and parameter updates at 19 points with 61 responses and six updates (Appendix D.1). Together, the studies connect generation-time sensitivity acquisition to sequential adaptation.

B.2 APEX REQUEST CLOCKS

Let $t_0, t_{\text{delivery}}, t_{\text{feedback}}, t_{\text{publication}}$ denote request start, response delivery, feedback arrival and update publication. We distinguish

$$T_{\text{delivery}} = t_{\text{delivery}} - t_0, \qquad T_{\text{post}} = t_{\text{publication}} - t_{\text{feedback}}, \qquad T_{\text{complete}} = t_{\text{publication}} - t_0.$$

Thus $T_{\text{complete}} = T_{\text{delivery}} + (t_{\text{feedback}} - t_{\text{delivery}}) + T_{\text{post}}$. Tangent propagation is charged to OFTA delivery. Activation, update and publication are charged after feedback. Figure 3a reports arithmetic means of T_{post} over each method's 50 requests in the original seed-211 runs. In the order BP replay, MeZO replay, post-feedback JVP and OFTA, mean T_{delivery} is 87.55, 88.80, 74.54 and 37.13 s, and mean T_{complete} is 91.00, 90.86, 77.20 and 37.81 s. Generation backends differ, so these timings do not isolate derivative placement. The BP-Replay curve in Figure 5 uses same-condition reruns (Appendix B.1).

B.3 Objective and Scoring Convention

A simple reward-score example. Record $q_t = \log \pi_{\phi^{(\nu_t)}}(a_t \mid x_t)$ and $z_t = D_{v_t} q_t$. For $\ell(q, r) = -rq$, a later reward gives $d_t = -r_t z_t$. At the generation-time parameters, a negative reward moves that response's score downward to first order when $z_t \neq 0$. The unchanged record can wait for delayed feedback without changing this historical derivative. The following APEX loss is a separate, coupled objective using sequence scores and auxiliary reference values.

For a group $G_t = (x_t, \{(a_{t,i}, b_{t,i})\}_{i=1}^{N_t})$, let $b_{t,i} \in \{0, 1\}$ be the verifier outcome and define

$$q_{t,i} = \log \pi_{\phi^{(\nu_t)}}(a_{t,i} \mid x_t), \qquad q_{t,i}^{\mathrm{ref}} = \log \pi_{\mathrm{ref}}(a_{t,i} \mid x_t).$$

Each score is the sum of token log-probabilities. It includes EOS when sampled and ends at the final emitted token when the token budget is reached. The reference checkpoint is fixed. With $s_{t,i} = 2b_{t,i} - 1$ and $\sigma(u) = 1/(1 + \exp(-u))$, the response-mean objective is

$$\begin{aligned} \log \widehat{Z}_t &= \log \left[\frac{1}{N_t} \sum_{j=1}^{N_t} \exp(b_{t,j}/\beta) \right], \\ h_{t,i} &= \beta(q_{t,i} - q_{t,i}^{\mathrm{ref}}) + \beta \log \widehat{Z}_t, \\ \mathcal{L}_t^{\mathrm{APEX}} &= \frac{1}{N_t} \sum_{i=1}^{N_t} \mathrm{softplus}(-s_{t,i} h_{t,i}). \end{aligned} \tag{10}$$

The APEX configuration fixes $\beta = 1$, separately from sampling temperature 1 and top-$p = 1$. Raw binary rewards enter the partition estimate; signed labels enter the logistic term. Reference scores are constant under differentiation.

OFTA records $y_t = (q_{t,1}, \ldots, q_{t,N_t})$ and $z_t = (D_{v_t} q_{t,1}, \ldots, D_{v_t} q_{t,N_t})$. Once inputs are ready,

$$d_t = \sum_{i=1}^{N_t} \underbrace{-\frac{\beta}{N_t} s_{t,i} \sigma(-s_{t,i} h_{t,i})}_{\partial \mathcal{L}_t^{\mathrm{APEX}} / \partial q_{t,i}} D_{v_t} q_{t,i}.$$

Once feedback arrives, activation consists of this output-side contraction and the directional parameter update.

B.4 Acquisition and Group Activation

The implemented group engine generates eight candidates sequentially at one unchanged acquisition snapshot, with a common Gaussian direction per lane and separately fixed decoding seeds. Each candidate stops at natural EOS or the 512-token cap. Natural groups retain every response, including all-positive, all-negative, invalid and truncated cases. Each group contains the original eight candidates, with their observed reward mix. Controlled 4P+4N groups are separate diagnostics. The full group activates when all feedback in the group has arrived; summing equivalent per-candidate contractions avoids materializing a full Jacobian.

A retained record is $\mathcal{T}_t = (y_t, \{z_{t,j}, \xi_{t,j}\}_{j=1}^{K}, \nu_t, m_t^{\mathrm{gen}}, \iota_t)$. Its metadata fixes scoring, stopping, coordinates and feedback association. Transient generation caches can be released after required records are detached. A candidate bank is published after activation; in the sequential protocol the next request starts afterward.

The A22 implementation fixes Qwen3-1.7B, and a detached copy of the original LoRA bank serves as the reference policy. After delivery, an independent CPU verifier simulates a user who rates each response as good or bad, a common form of feedback on chat and coding platforms. The verifier compares each final answer with the known GSM8K answer, so the binary rating is exact. The verifier's completion defines the feedback timestamp. OFTA then forms the update from the stored traces with output-side operations, direction regeneration, and parameter writes.

B.5 A Common Objective across Execution Methods

OFTA collects scores and traces during original generation. BP-Replay generates normally, then scores the fixed candidates and backpropagates APEX after feedback. MeZO-Replay evaluates

the same group at positive and negative perturbations without regenerating responses or changing rewards/reference scores. Post-feedback JVP in Figures 5 and 3a starts after feedback. The sequential protocol holds the acquisition bank fixed during all eight candidate generations and loss evaluation, then publishes the update.

Interface scope beyond APEX. Sequence-score forms of on-policy REINFORCE, group-normalized advantage losses as in GRPO, and DPO/KTO-style losses on recorded responses can reuse the same response-score trace (Shao et al., 2024; Rafailov et al., 2023; Ethayarajh et al., 2024). Token-level clipping and KL terms are supported by declaring per-token statistics together with fixed old-policy and reference scores. KTO-style offsets are supported when they are computed from recorded responses.

Natural-language feedback must first be mapped to r_t or loss coefficients; trace formation does not interpret it. Declared statistics can include token/span scores, temperature-scaled scores or next-token distributions at executed positions. Scoring a user-written correction along unexecuted prefixes requires additional adapting-model computation.

APEX, first-position corrective cross-entropy, reference-free score losses and length-mean F1-weighted losses are separate objectives. The configuration index below binds each result to its actual objective. Known-supervision scalar-loss JVP studies evaluate the directional learning rule with supervision available at differentiation time.

C Implementation, Protocols, and Reproducibility

C.1 Configuration-to-Result Index

Table 2 identifies the model, objective and trainable space for each study. The configuration IDs are used consistently below.

Table 2: Configuration identities. B counts examples/groups, N candidates per group, and K parameter-space directions.

ID	Model, objective, coordinates and measurement scope
RQ1-S19	Qwen3-1.7B, all-linear $r8/\alpha64$; 19 points, 61 responses and six actual APEX updates.
GSM-CORR	Same model/space; APEX, 50 updates and 100 evaluation questions, $B = 1, N = 8, K = 1$; seed 211 except the same-condition BP-Replay reruns.
A22	Qwen3-1.7B, $p = 8{,}716{,}288$; APEX interface, GSM8K role partition, cap 512; validation LR 10^{-6}.
L-LENGTH	Qwen3-1.7B, all-linear $r8/\alpha64$; response-score loss $(1/2 - r)q$, prompt 204, lengths 32–256, three processes.
L-CE	Qwen3-1.7B, all-linear LoRA; first-position CE, 320 adaptation/128 fresh QA contexts, cap 16, $B = K = 1$.
L-PLACE	Qwen3-1.7B, single-token CE; last value-projection B ($p = 8{,}192$).
R-EAGER	Qwen3-1.7B, all-linear $r8/\alpha64$; length-mean F1-weighted score, 32 natural systems inputs, cap 16, $B = N = K = 1$.
H-20K	Qwen3-1.7B; known-supervision task-quality study with 20k directional versus 2k BP budgets.
H-20K-8B	Qwen3-8B; same as H-20K except for the model.

H-20K-8B changes only the model from Qwen3-1.7B to Qwen3-8B. All other training settings, including the tasks, seeds, update budgets, and optimizer settings, are the same as in H-20K.

For A22 and R-EAGER, all-linear includes both LoRA factors on attention Q/K/V/O and MLP gate/up/down projections in 28 layers. Canonical named-tensor order defines Gaussian coordinates, and each run restores the saved initial adapter bank. Model revision is `70d244cc86ccca08cf5af4e1e306ecf908b1ad5e`. A22 partitions GSM8K into 1024 adaptation, 64 tuning, 192 monitoring, 3 warmup and 1319 final questions, using a decimal-number verifier. R-EAGER draws its 32 systems inputs from a pool of 556 SQuAD contexts, excluding 1836 previously used contexts. Selection uses public-input hashes and prompt length at most 512. L-CE additionally requires single-token answers and uses its own final evaluation set.

A22 and R-EAGER execute weights, activations, tangents, caches and score accumulation in FP32, without TF32 or autocast; BP projection/norm reductions use FP64. Eager forward AD uses functional parameter substitution and complete cached prefill/decode sensitivity, with last-position logits and ShapeLinear rules. The workstation environment uses PyTorch 2.8.0+cu128 and a 48 GB workstation GPU. Studies on H100 are identified explicitly.

L-CE selects among six rates per method on development data, then fixes directional SGD 3×10^{-7}, BP AdamW 10^{-5} and FD radius 10^{-4}.

C.2 Execution Placement and Information Order

The sequential protocol releases verifier feedback after response delivery, activates or differentiates the loss, publishes the update, then starts the next request. Replay scores the actual generated response using either cached token-wise or whole-sequence evaluation. OFTA computes the directional trace during generation. Pre-feedback JVP starts a separate pass after generation but before feedback; post-feedback JVP starts it after feedback. Archived configuration IDs label the latter `separated` in GSM-CORR. Historical replay reconstructs both the acquisition bank and original interaction. Fresh Replay keeps the same realized response/feedback but evaluates at the application bank.

In L-PLACE, the recorded ID `separated` denotes post-feedback JVP. Pre-feedback full-prefix JVP begins full-prefix sensitivity after generation and before the synchronous feedback call; pre-feedback cached JVP reuses frozen-prefix features and differentiates the trainable tail. Cached BP/MeZO similarly reuse features; prepaid MeZO prepares two-sided perturbed logits, with radius 10^{-3}. All preparation is charged. Retained BP keeps the differentiable generation graph and its activations. Pre-feedback BP may store a full statistic gradient or immediately project it onto v and retain only that scalar. The projected variant performs backpropagation during acquisition and retains the resulting scalar directional derivative; a vector interface requires the corresponding statistic derivatives.

Fixed-state cost measurements reset the bank and match response, direction, objective and reduction. Sequential learning carries each published update into the next request, allowing responses and rewards to evolve under each method. Directions, decoding, request order and feedback scheduling use separate declared random streams; Appendix E.2 describes their information flow. Coupled candidates share a snapshot and direction per lane, with N candidates forming one group counted by B.

C.3 Clocks, Resources, and Statistical Units

Clocks use monotonic `perf_counter_ns` with CUDA synchronization at measured boundaries. Warm request intervals cover the declared online work, including preparation performed before feedback. Loading, warmup, reset and graph capture are recorded separately. End-to-end intervals are measured directly; model calls, JVPs, backward calls and phase times describe the work within those intervals. The APEX accounting is specified in Appendix B.2.

Resource accounting distinguishes model and optimizer state, gradients, KV caches, workspace, retained records, reference banks and pending parameter versions. Reported allocator peaks measure framework-managed device memory. With equal precision and layout, one primal KV cache and K full tangent caches occupy $(1 + K)$ times the primal KV storage. The statistical unit is specified for each study: for example, three process repeats over 32 inputs provide repeated measurements of 32 questions. Item-bootstrap intervals resample evaluation items conditional on the observed seeds and common evaluation set.

D Numerical Validation and Supplementary Results

We first compare generation-acquired derivatives with independent automatic-differentiation references, then examine how execution path, objective, direction count and adaptation space affect accuracy and cost.

D.1 Cached Derivatives and APEX Updates

The RQ1-S19 study evaluates 19 parameter/input configurations comprising 61 responses: six natural-generation groups (48 responses), eight length-boundary configurations, and five difficult-input configurations. Direction master seeds 19 and 23 supply four and two natural groups, respectively. Boundary prompt/response lengths are 8/1, 64/16, 256/128 and 512/512, each evaluated at the initial bank and after three updates. Every derivative comparison fixes parameters, inputs, realized tokens and direction.

The cached-graph directional criterion is

$$|d_{\text{generation}} - d_{\text{reference}}| \leq 10^{-5} + 2 \times 10^{-4} |d_{\text{reference}}|. \tag{11}$$

The absolute term controls near-zero discrepancies; the relative term scales the budget with the reference derivative. All 61 generation-to-cached-BP comparisons satisfy this criterion, and all 19 configurations pass their primary checks. The largest reference-relative discrepancy is 0.0150672%, with absolute error 0.001254818 and allowed error 0.001675633. Primal losses and parameter updates use their separate acceptance budgets.

Table 3: Six RQ1-S19 APEX updates. Generation-acquired loss derivatives agree with independent cached-BP projections within the directional criterion.

Master / event	Generation derivative	BP projection	Absolute error
19 / 0	8.220407869	8.220451419	4.355×10^{-5}
19 / 1	−47.440442872	−47.440483474	4.060×10^{-5}
19 / 2	−13.844724557	−13.844815307	9.075×10^{-5}
19 / 3	26.614205869	26.614229424	2.356×10^{-5}
23 / 0	44.548969876	44.549019920	5.004×10^{-5}
23 / 1	1.269619014	1.269579917	3.910×10^{-5}

The six natural groups satisfy all 30 loss/directional checks. Reconstructing directional SGD from each saved scalar exactly reproduces the published update; candidates formed from independent BP projections also satisfy the parameter-error budgets. Table 3 connects derivative agreement to the six executed updates.

Controls on 18 responses test zero derivatives, wrong directions and detached caches. All 53 informative response-level controls are detected; the remaining single-token detached-cache control leaves its derivative unchanged. All six group controls that contract only the first candidate are detected. These checks test the computation's use of the chosen direction, cache sensitivities and all candidates in the group objective.

A full-sequence teacher-forced BP reference evaluates a different computation graph. Generation-to-teacher comparisons pass 56/61 checks under the criterion above and 60/61 under the conditioned criterion defined below. These criteria scale their budgets differently and provide complementary numerical diagnostics.

D.2 Computation Path and Precision

Table 4 gives independent derivative and update checks for the length, corrective-QA, placement and eager full-response studies discussed below. These studies use the reference-relative limit $10^{-5} + 10^{-4}|d_{\text{BP}}|$ and, where specified, the conditioned limit $10^{-5} + 10^{-4} \max(|d_{\text{BP}}|, \|g_{\text{BP}}\|)$. The conditioned scale accounts for the full BP gradient when its projection onto the sampled direction is small. All listed paths have zero sign disagreements.

L-PLACE uses 32 distinct inputs, each repeated in three processes. The L-CE final-bank checks evaluate two inputs at each of 12 banks. Loss scale, direction, step size and trainable space determine the derivative and parameter-error scales in each row.

D.3 Response Length and Update Placement

The length study (L-LENGTH), shown in Figure 4, uses a common sampled trajectory with a 204-token prompt, response lengths 32–256, native FP32, all-linear LoRA and $B = K = 1$ for directional

Table 4: Numerical checks for the measured computation paths. e_d is absolute derivative error and e_ϕ is the L2 difference between directional-SGD candidates. Strict counts use the 10^{-4} reference-relative coefficient specified in this subsection.

Path	Strict	Conditioned	Max. e_d	Max. e_ϕ
L-LENGTH	15/18	18/18	3.03991×10^{-2}	8.97456×10^{-2}
L-CE, states	15/15	15/15	3.22443×10^{-3}	2.97168×10^{-6}
L-CE, final banks	24/24	24/24	1.39374×10^{-3}	1.50121×10^{-6}
L-PLACE	32/32	32/32	1.28805×10^{-5}	1.16290×10^{-5}
R-EAGER	96/96	96/96	5.95934×10^{-4}	1.94733×10^{-6}

methods. All responses stop at the cap and receive reward zero. The objective is $(1/2 - r)q$, and the parameter bank is reset before each request. The warm request clock includes update publication; 108 requests are measured. BP uses AdamW and the directional methods use SGD. Numerical checks appear in Table 4. OFTA takes 0.950–6.112 s, with aggregate-median request-time reductions of 35.0–37.1% relative to BP and 31.6–35.6% relative to MeZO under these implementations.

The single-token placement study (L-PLACE) fixes the loss, initial bank, direction and update rule across 32 requests and three process repeats. OFTA and post-feedback JVP agree on all 96 losses, directional scalars and published parameter banks. Preparation is included in the warm request clock; loading, reset and warmup are measured outside it. OFTA request medians are 49.229–50.441 ms, compared with 89.666–90.116 ms for post-feedback JVP, a reduction of 44.03–45.10%.

D.4 Corrective-QA Learning

The corrective-QA study (L-CE) tests adaptation with a post-response first-position cross-entropy loss. It uses 320 adaptation events, natural EOS with a response cap of 16 tokens, and complete greedy-answer evaluation on 128 fresh eligible contexts. From initial F1 of 61.344%, OFTA direction masters 19, 23 and 29 reach 69.239%, 67.235% and 69.743%, respectively; BP reaches 84.766%.

The predeclared success rule requires a mean gain of at least two percentage points, at most a two-point loss for any direction, and nondecreasing mean exact match. Both the all-linear branch and the late-layer single-token branch meet this rule. For the natural-response branch, 10,000 item-bootstrap resamples give a mean-gain interval of [2.26,12.70] percentage points over the evaluated directions and contexts. An independently scored JVP implementation matches all 960 OFTA updates across the three directions. MeZO reaches the same final F1 with different parameter banks. This study demonstrates learning through the scalar-loss JVP rule under corrective cross-entropy supervision.

D.5 Direction Count and Trainable Space

The estimator study measures how direction count changes error at a fixed objective and parameter bank. It uses a synthetic teacher-forced input/response of 64/16 tokens, including EOS, with $L = -q$ and native FP32 model computation. For each of seven trainable spaces, 32 independent replicates sample eight directions; estimates for smaller K use nested prefixes. All 1,792 physical JVP draws enter the estimates, and 1,769 of them also pass the strict directional tolerance. Vector reductions use FP64, and each direction uses a separate model traversal.

Mean squared error decreases with K in every tested space, consistent with Corollary F.3. The dimension p includes LoRA-A coordinates whose initial gradients are zero, since random directions still vary along those coordinates. Changing rank or target modules changes both model capacity and gradient geometry; averaging more directions reduces estimation error within each fixed space.

D.6 Full-Response Objectives and Replay Cost

The eager full-response study (R-EAGER) uses

$$L = (1/2 - \mathrm{F1}(a)) \frac{1}{|a|} \sum_s \log \pi(a_s \mid x, a_{<s}),$$

Table 5: Fixed-objective mean squared estimator error divided by $(p+1)\|g_{\mathrm{BP}}\|^2$. All sampled directions contribute to the estimates. Strict counts use the 10^{-4} reference-relative criterion.

Space	p	Strict /256	$K=1$	$K=2$	$K=4$	$K=8$
All-linear $r=4$	4,358,144	253	.6267	.3402	.1949	.1078
All-linear $r=8$	8,716,288	251	.9933	.5446	.2616	.1143
All-linear $r=16$	17,432,576	254	1.3104	.5090	.2583	.1252
QKV $r=4$	1,146,880	255	.9642	.4637	.2386	.1340
QKV $r=8$	2,293,760	252	.9759	.4283	.2154	.1126
QKV $r=16$	4,587,520	254	1.1378	.5435	.2555	.1228
First input RMSNorm	2,048	250	.6959	.4402	.2243	.1160

including EOS. Its 32 inputs produce responses averaging 7.09 tokens; 90/96 process/input runs stop at EOS. Across three processes, OFTA request medians are 1243.732–1261.200 ms. Teacher-forced post-feedback JVP takes 462.408–487.545 ms, BP takes 381.496–395.869 ms, and MeZO takes 339.047–351.532 ms. Token-wise cached post-feedback JVP takes 1481.763–1513.971 ms and reproduces all 96 OFTA updates; independent BP projections pass 96/96 checks. On this shared token-wise backend, OFTA is about 16% faster than post-feedback JVP. The teacher-forced controls use a different backend that processes the whole short response at once.

D.7 Peak memory of one TTA step

We measure peak GPU memory from prefill and generation through the applied parameter update on an H100 80 GB, using Qwen3-1.7B with FP32 all-linear LoRA ($r = 8$, $\alpha = 64$; 8,716,288 trainable coordinates) and two prompts of 58 and 113 tokens. We record `torch.cuda.max_memory_allocated` after resetting peak allocator statistics immediately before each step and reading the peak after parameter publication. Generation, ordinary/tangent caches, feedback, gradient or perturbation buffers, update construction and publication are included; warmup, numerical diagnostics and CPU validation snapshots are excluded. Allocated memory is not allocator-reserved capacity or device-wide driver/context occupancy; $1\,\mathrm{GiB} = 2^{30}$ bytes.

Batch size 16 physically repeats one input 16 times in a single model batch, not serial calls or gradient accumulation. Methods share the initial bank and native-greedy fixed response tokens at lengths 1 and 4. The loss is the batch mean of $(0.5-b)\sum_s \log \pi(a_s \mid x, a_{<s})$, with synthetic binary feedback released after delivery and $K=1$ for OFTA/MeZO. Every step applies an update, then resets to the initial bank; this is a repeated one-step execution measurement, not an online learning trajectory. BP uses full LoRA-gradient SGD, without checkpointing or offload. MeZO materializes its direction and out-of-place perturbed LoRA banks; it is not a minimum-memory in-place implementation.

For each batch size, the three methods shown use 18 independent processes and 108 measured steps: two output lengths, three methods, three process repeats, and two prompts with three measured steps each. Each process also has one warmup per prompt. The batch-1 rows reuse the corresponding 18 processes and 108 measurements from the earlier confirmation study; the batch-16 extension adds 18 processes and 108 measurements, for 36 processes and 216 measured steps across the displayed comparison. Corresponding methods share the physical GPU within each length/repeat group. For each process/prompt, we take the maximum of its three step peaks, then report the median of six process/prompt maxima per method and length. Prompts and steps within a process are not independent process repeats. Percentages use unrounded byte-level medians.

At batch size 1, the three methods' median total peaks span 6.605–6.973 GiB, with resident memory dominating. At batch size 16, OFTA's extra allocation is 0.964/1.324 GiB for 1-/4-token outputs, versus MeZO's 0.474/0.648 GiB; Table 6 retains all values.

Resident memory is the actual allocated memory immediately before the request, not model weights alone. It includes the loaded model and common GPU reset bank: 6.506810 GiB for OFTA/MeZO and 6.538060 GiB for BP. The reported 7.0%/9.5% excess over MeZO compares medians; individual matched observations range from 5.00% to 11.84%.

Table 6: Median peak allocated GPU memory (GiB) of one complete TTA step. "Extra" is the step peak minus its measured pre-request resident allocated memory, aggregated with the same maximum-then-median rule.

Batch	Output tokens	Total peak			Extra over resident		
		OFTA	MeZO	BP	OFTA	MeZO	BP
1	1	6.610	6.605	6.951	0.104	0.098	0.413
1	4	6.624	6.605	6.973	0.117	0.098	0.435
16	1	7.471	6.981	13.134	0.964	0.474	6.596
16	4	7.831	7.155	13.480	1.324	0.648	6.942

All predeclared numerical and update-publication checks pass. These results concern short, repeated-input batches and the stated implementations, not long-context memory, learning quality or runtime speed.

E Execution Semantics and Delayed Feedback

The following protocols examine matched execution, direction-independent generation and the recovery of historical derivatives after feedback delays.

E.1 Matched-Comparison Protocol

Matched comparisons fix the parameter bank, realized tokens, masks, objective, direction, reference scores and publication rule. The response-score workload varies response length; the APEX workload uses $B = 1, N = 8, K = 1$. For each workload, same-path post-feedback JVP performs primal delivery followed by cached joint replay after feedback. A teacher-forced post-feedback JVP control instead evaluates the complete fixed response with a scalar-loss JVP after feedback. Loss, derivative and published-bank checks connect each implementation to the shared objective and update.

This design separates the cost of acquisition placement on a matched cached path from the cost of choosing a different differentiation backend. The own-policy measurements in Appendix B.2 measure delivery, feedback-to-publication and complete-request intervals with each method's generated responses and backend.

E.2 Direction Independence and Information Flow

The direction-independence study fixes one initial FP32 Qwen3-1.7B all-linear LoRA bank and one question, with $B = 1, N = 8, K = 1$, response cap 512, binary APEX $\beta = 1$, and candidate-only directional SGD at rate 10^{-6}. Decoding seeds and request order are fixed independently of direction. Four natural-generation runs comprise native primal generation, joint generation with master seeds 19 and 23, and a repeat of master 19. This produces 32 responses in eight paired decoding slots. The parameter bank remains fixed throughout candidate evaluation.

Tokens, EOS decisions, rewards and reference scores agree across all four runs. The joint runs have bitwise-equal primal scores and losses. Changing direction changes the tangent and update candidate; repeating master 19 reproduces the direction, tangent, scalar and candidate bitwise. Every joint candidate agrees with independent SGD reconstruction. The native run provides the primal-generation comparison.

A two-request metadata queue replay uses a predetermined schedule and capacity one. The direction-blind rule makes identical decisions after changing direction; the deliberately direction-dependent eviction rule changes its decision and is detected. These observations verify the implemented separation between primal generation, direction-specific sensitivities and the specified queue rule. In the reported protocols, admission, capacity and feedback eligibility are direction-independent.

E.3 Delay, Historical Derivatives, and Live Publication

The delay study fixes an eight-response APEX group comprising 2,211 tokens, together with its masks, feedback and reference scores. It uses FP32 all-linear LoRA with rank 8, $\alpha = 64$, and directional-SGD rate 10^{-6}. Replay evaluates the fixed responses. Six anchored computations compare three methods in two regimes: an unchanged acquisition/application bank, and a fixed drift from the initial bank to the saved step-10 bank. The latter has parameter L2 distance 7.798316 and version gap 10.

Table 7: Anchored directional derivatives. Store-Tangent and Historical Replay agree bitwise before rounding.

Regime	Store-Tangent	Historical Replay	Fresh Replay
Wait-only	214.189701	214.189701	214.189701
Fixed drift	214.189701	214.189701	−469.584538

Historical Replay reconstructs the acquisition bank and interaction, then restores the application bank before updating. Store-Tangent and Historical Replay recover identical derivatives and publish identical candidates in both regimes; all three methods agree with an unchanged bank. With fixed drift, Fresh Replay evaluates the current parameters, producing a candidate at L2 distance 2.017732 from the historical candidate. Independent SGD reconstruction validates all six publications. Wrong-bank controls are detected in the drifted regime, where acquisition and application banks differ; the drifted Fresh stale-candidate control is also detected. These results connect the trace-recovery identity to actual parameter updates at a later bank.

The live-stream study runs one process per method, each with two waves of four serially acquired fixed-token groups. Within each wave, updates are applied FIFO; the second wave acquires at the bank published after the first four updates. Feedback becomes eligible at delivery+T and after verifier completion, where $T = 28.409737222$ s is the prescribed feedback delay. All 24 updates satisfy the numerical and publication checks. Store-Tangent and Historical Replay match all eight published-bank transitions. Fresh Replay first diverges at request 1 (zero-based) and subsequently follows its own parameter history.

The stream clock includes the wave barrier, online snapshots and archival work, with cold setup and post-stream AD validation outside the clock. Post-feedback work includes activation and publication. Every process retains four full acquisition-bank snapshots (133 MiB): Historical Replay uses them for reconstruction, and Store-Tangent and Fresh Replay retain them for update validation. Their storage and capture time are included in the measured execution.

Table 8: Observed stream horizons under two waves of serial acquisition and FIFO publication, with one process per method.

Method	Horizon (s)	Published	Queue peak	Unfinished
Store-Tangent	480.347	8	4	0
Historical Replay	641.760	8	4	0
Fresh Replay	633.783	8	4	0

Store-Tangent has the shortest observed stream horizon.

F Theory Proofs

We establish how generation-time sensitivity propagation and feedback-time contraction recover a historical directional derivative. We then connect this derivative to established forward-gradient estimators, quantify the roles of direction count and trainable coordinates, and bound the effect of parameter drift before application.

F.1 Notation, Realized Objectives, and Assumptions

Interaction t is generated and traced at trainable coordinates $\phi^{(\nu_t)} \in \mathbb{R}^p$, and its update is applied at $\phi^{(k_t)}$. The coordinates may describe all parameters, a subset, or a differentiable reparameterization. Write $a_t = (a_{t,1}, \ldots, a_{t,T_t})$ for the realized response, x_t for the prompt, r_t for feedback, and m_t^{gen} for generation-time metadata, and m_t for its extension with declared auxiliary inputs acquired later. The statistic y_t and its tangent z_t belong to $\mathbb{R}^{d_y}$; the direction v_t and estimator $\widehat{g}_t$ belong to $\mathbb{R}^p$. All vector norms are Euclidean, and matrix norms are induced operator norms unless stated otherwise.

For a realized interaction, the differentiated function is

$$L_t(\phi) = \ell\big(f_\phi(x_t, a_t; m_t^{\mathrm{gen}}), r_t; m_t\big). \tag{12}$$

Here $x_t, a_t, r_t, m_t^{\mathrm{gen}}, m_t$ are fixed under differentiation, including token choices, EOS, and the realized stopping position. Additional random choices specifying the executed scorer are fixed and included in its metadata or conditioning information. Fixed reference scores have zero derivative and retain the same numerical values across parameter-point comparisons. Generation produces tokens on the primal path; the corresponding scoring program treats those realized tokens as fixed inputs. Section F.5 connects this realized-sample derivative to differentiation of a parameter-dependent response distribution through the score-function identity (Schulman et al., 2015).

A1: Fixed coordinate identification. The coordinate dimension, order, masks, parameter sharing, and reparameterization map remain fixed between acquisition and application of $-\eta dv$.

A2: Fixed acquisition point and direction. One parameter version $\phi^{(\nu_t)}$ and one direction v_t are used throughout a response's prefill and generation. For multiple directions, each tangent lane has its own fixed direction. Each acquisition uses an immutable parameter snapshot through completion.

A3: Differentiable realized program. The declared scorer is differentiable in the chosen coordinates at acquisition, and ℓ is differentiable in its recorded statistics. The states and elementary operations in the realized program admit the chain rules used below. Smoothness along an acquisition–application segment is an additional assumption used only for the staleness bound.

A4: Complete sensitivity propagation. Every parameter-dependent state used downstream carries its correct tangent, including prompt-prefill state and reused cache entries.

A5: Declared, generation-realizable interface. The loss family and statistic-acquisition rule are fixed before generation begins. Acquisition uses the information available during generation. Response-dependent lengths, indices, and association information may be realized along the primal path. A single generation traversal supports statistics accumulated or read out from its executed operations and states. A separately implemented declared scorer must match both the value and Jacobian of the executed fixed-response program at acquisition; equality of the differentiable functions on a neighborhood is sufficient.

A6: Ideal arithmetic. Exactness refers to the specified real-valued differentiable program. Section F.10 bounds the effect of perturbations in the retained statistics and tangents.

A7: Separately stated probability assumptions. Trace exactness holds pathwise for each v_t. Conditional unbiasedness assumes a conditional second-moment identity, and the Gaussian covariance formulas assume a conditional Gaussian law. All expectations used below are assumed to exist.

A8: Update scope. Update-discrepancy statements compare the SGD-type steps $-\eta_t d_t v_t$ at the same learning rate.

OFTA supplies the historical directional derivative through generation-time acquisition and retained-trace activation. Applying the established forward-gradient estimator then gives its standard conditional unbiasedness (Baydin et al., 2022).

F.2 Generation-Time Tangent Propagation

The realized autoregressive program. In this subsection, abbreviate the generation-time metadata m_t^{gen} by m. Suppress t, fix $\phi_* = \phi^{(\nu_t)}$, and let $v = v_t$. Define $D_v F(\phi_*) = \frac{d}{d\epsilon} F(\phi_* + \epsilon v)\big|_{\epsilon=0}$. Let c_s contain all model state reused downstream, including KV entries, and let w_s be a statistic accumulator. A generation-realizable scorer can be represented by

$$c_0 = P_\phi(x), \qquad w_0 = U_\phi(c_0; m), \tag{13a}$$

$$h_s = H_{\phi,s}(c_{s-1}), \qquad a_s = \mathrm{Decode}(h_s, \omega_s), \tag{13b}$$

$$c_s = C_{\phi,s}(c_{s-1}, a_s), \tag{13c}$$

$$w_s = A_{\phi,s}(w_{s-1}, c_{s-1}, h_s, c_s, a_s; m), \tag{13d}$$

$$y = F_\phi(w_T, c_T; m). \tag{13e}$$

The decoder specifies the realized primal tokens. For differentiation, its outputs a_s are clamped, so scoring at nearby parameter points uses the same realized response. Subscripts s permit step-dependent shapes and operations. After fixing T and the discrete execution choices, this is a finite differentiable program.

The recursion follows the executed generation graph. When generation stops immediately after emitting a_T, take $C_{\phi,T}$ to be the identity, with $c_T = c_{T-1}$, and form the readout from the available state and logits. Equivalently, omit this terminal transition.

Forward recurrences. All partial derivatives below are evaluated at the primal arguments of Eq. (13) with $\phi = \phi_*$. Explicit parameter partial derivatives hold the displayed state arguments fixed. For $A_{\phi,s}$, ∂_- and ∂_+ denote derivatives with respect to its incoming- and outgoing-state arguments, respectively. Initialize

$$\dot{c}_0 = \partial_\phi P_\phi(x) v, \tag{14}$$

$$\dot{w}_0 = \partial_\phi U_\phi(c_0; m) v + \partial_c U_\phi(c_0; m) \dot{c}_0. \tag{15}$$

At each realized token step, propagate

$$\dot{h}_s = \partial_\phi H_{\phi,s} v + \partial_c H_{\phi,s} \dot{c}_{s-1}, \tag{16a}$$

$$\dot{c}_s = \partial_\phi C_{\phi,s} v + \partial_c C_{\phi,s} \dot{c}_{s-1}, \tag{16b}$$

$$\begin{aligned} \dot{w}_s = {} & \partial_\phi A_{\phi,s} v + \partial_w A_{\phi,s} \dot{w}_{s-1} + \partial_- A_{\phi,s} \dot{c}_{s-1} \\ & + \partial_h A_{\phi,s} \dot{h}_s + \partial_+ A_{\phi,s} \dot{c}_s. \end{aligned} \tag{16c}$$

Finally, set

$$z = \partial_\phi F_\phi v + \partial_w F_\phi \dot{w}_T + \partial_c F_\phi \dot{c}_T. \tag{17}$$

The clamped tokens have $D_v a_s = 0$. These recurrences specialize forward-mode chain-rule propagation to the autoregressive state and accumulator structure (Baydin et al., 2018).

Generation-time part of Proposition 5.1. For every sufficiently small ϵ, execute the clamped-token program at $\phi_* + \epsilon v$, and denote its states by $c_s(\epsilon), h_s(\epsilon), w_s(\epsilon)$. At $\epsilon = 0$, its primal states equal the generation states: prefill is identical, and each subsequent step consumes the same realized token. Thus its terminal output is the statistic of the executed fixed-response scorer.

At initialization, differentiating $P_{\phi_*+\epsilon v}(x)$ gives $c_0'(0) = \dot{c}_0$. Differentiating $U_{\phi_*+\epsilon v}(c_0(\epsilon); m)$ gives $w_0'(0) = \dot{w}_0$, including the prefill-state contribution.

Suppose $c_{s-1}'(0) = \dot{c}_{s-1}$ and $w_{s-1}'(0) = \dot{w}_{s-1}$. Differentiating $H_{\phi_*+\epsilon v,s}(c_{s-1}(\epsilon))$ gives Eq. (16a), so $h_s'(0) = \dot{h}_s$. Since a_s is fixed, differentiating $C_{\phi_*+\epsilon v,s}(c_{s-1}(\epsilon), a_s)$ gives Eq. (16b), so $c_s'(0) = \dot{c}_s$. Differentiating the accumulator with respect to its explicit parameter argument and each of its four differentiable input arguments gives Eq. (16c), so $w_s'(0) = \dot{w}_s$. This completes the induction.

At the fixed realized stopping position T, differentiating the terminal readout gives Eq. (17). Consequently, for the executed fixed-response scorer f^{gen},

$$y = f^{\mathrm{gen}}_{\phi_*}(x, a; m), \qquad z = J_\phi f^{\mathrm{gen}}_{\phi_*}(x, a; m) v. \tag{18}$$

If the declared scorer is this program, this proves the acquisition claim. If the declared scorer f is implemented separately, equality of f^{gen} and f on a neighborhood of ϕ_* implies equality of their values and Jacobians at ϕ_*. Alternatively, value and Jacobian equivalence may be established directly. Under either matching condition, Eq. (18) holds with f in place of f^{gen}. □

Cached versus full-sequence evaluation. A sufficient setting for the matching condition is an exact causal model whose cached and full implementations use identical tokens, masks, positions, parameter sharing, mathematical operations, model mode, and fixed random choices. Assume each stored prefix state is precisely the state computed for that prefix by full evaluation. Prefill establishes this property initially. At the next token, causal masking makes every relevant prefix representation independent of future tokens. The new-token computation therefore receives the same prefix states as full evaluation; layer-by-layer induction gives the same new representations and logits. Appending their exact cache entries preserves the prefix-state property. Token induction then establishes the same logits and statistic for every parameter point in a differentiable neighborhood, proving the required function identity. The matching condition concerns both values and derivatives: for example, $f^{\mathrm{gen}}(\phi) = \phi$ and $f^{\mathrm{full}}(\phi) = 2\phi$ have equal values at $\phi = 0$ and respective derivatives 1 and 2.

Sequence log-probability. Suppose the declared statistic is the raw-policy score

$$q_\phi(x, a) = \sum_{s=1}^{T} \log \pi_\phi(a_s \mid x, a_{<s}), \qquad \pi_\phi(\cdot \mid x, a_{<s}) = \mathrm{softmax}(h_s). \tag{19}$$

Writing $p_s = \mathrm{softmax}(h_s)$ and $\lambda_s = h_{s,a_s} - \log \sum_j \exp(h_{s,j})$, differentiation gives

$$D_v \lambda_s = \dot{h}_{s,a_s} - \frac{\sum_j \exp(h_{s,j}) \dot{h}_{s,j}}{\sum_j \exp(h_{s,j})} \tag{20}$$

$$= (e_{a_s} - p_s)^\top \dot{h}_s. \tag{21}$$

Hence the online accumulator is

$$q_0 = 0, \qquad \dot{q}_0 = 0, \tag{22}$$

$$q_s = q_{s-1} + \lambda_s, \qquad \dot{q}_s = \dot{q}_{s-1} + (e_{a_s} - p_s)^\top \dot{h}_s, \tag{23}$$

and, at termination,

$$(q_T, \dot{q}_T) = \left(q_{\phi_*}(x, a), D_v q_\phi(x, a)\big|_{\phi_*} \right). \tag{24}$$

The logits producing the terminal token also score it. A declared statistic may use a specified subset of token scores, including a convention that omits EOS. For the complete-trajectory reward bridge in Section F.5, score every sampled token, including EOS when sampled, and end the trajectory at the actual token-budget cutoff otherwise.

If the score instead uses fixed temperature $\tau > 0$, its per-token tangent is

$$D_v \log \mathrm{softmax}(h_s/\tau)_{a_s} = \frac{1}{\tau} \big(e_{a_s} - \mathrm{softmax}(h_s/\tau)\big)^\top \dot{h}_s. \tag{25}$$

A raw-policy scorer uses $\mathrm{softmax}(h_s)$ even when sampling uses a separate temperature; a transformed-policy scorer uses its declared normalized law. For fixed realized $T > 0$, the tangent of q_ϕ/T is $D_v q_\phi/T$. Parameter-dependent temperatures, normalization factors, and differentiable logit processors contribute their chain-rule terms.

Sensitivity through reused state. For $c = \phi x$ and $y = \phi c$, the derivative is $D_v y = cv + \phi D_v c = 2\phi x v$: the cached-state contribution supplies $\phi x v$ of the total. Prefill and cache tangents preserve this contribution throughout generation. A state with zero downstream derivative contribution may carry a zero tangent; parameter independence is a sufficient condition.

The construction therefore acquires the declared statistics and their exact tangents during one autoregressive generation traversal, including its prefill and token-level computations.

F.3 Proof of Exact Historical Trace Activation

Activation part of Proposition 5.1. Define the historical Jacobian and output derivative by

$$J_t = J_\phi f_{\phi^{(\nu_t)}}(x_t, a_t; m_t^{\mathrm{gen}}), \qquad u_t = \nabla_y \ell(y_t, r_t; m_t). \tag{26}$$

Since all parameter dependence of Eq. (12) is through f_ϕ, componentwise differentiation gives

$$\frac{\partial L_t}{\partial \phi_j}(\phi^{(\nu_t)}) = \sum_{i=1}^{d_y} \frac{\partial \ell}{\partial y_i}(y_t, r_t; m_t)(J_t)_{ij}, \qquad \nabla_\phi L_t(\phi^{(\nu_t)}) = J_t^\top u_t. \tag{27}$$

The generation-time proof has already established $z_t = J_t v_t$. Therefore

$$D_{v_t} L_t(\phi^{(\nu_t)}) = v_t^\top J_t^\top u_t \tag{28}$$

$$= u_t^\top J_t v_t \tag{29}$$

$$= u_t^\top z_t = d_t. \tag{30}$$

An unchanged record and fixed objective interface therefore recover the same historical derivative at any feedback arrival time, including after subsequent parameter updates. □

Equivalence to a fused scalar-loss JVP. With the same parameter point, realized response, feedback, metadata, and fixed stochastic state, exact forward-mode differentiation of the scalar composite L_t returns $D_{v_t} L_t(\phi^{(\nu_t)})$. Equation (30) therefore proves equality with the stored-trace contraction. If the reference uses a separately implemented scorer, the value-and-Jacobian matching condition in the generation-time proof applies. This establishes the common derivative computed by stored-trace activation and fused scalar-loss differentiation.

Additional parameter dependence. If $L_t^{\mathrm{total}}(\phi) = L_t(\phi) + R(\phi)$, then

$$D_{v_t} L_t^{\mathrm{total}}(\phi^{(\nu_t)}) = u_t^\top z_t + \nabla R(\phi^{(\nu_t)})^\top v_t. \tag{31}$$

To recover the historical total derivative, the second term must also be available at the historical point, for example by recording it during acquisition or including the regularizer in the declared statistic. For $R(\phi) = \lambda \|\phi\|^2/2$, retain the scalar $\lambda(\phi^{(\nu_t)})^\top v_t$. More generally, an explicit ϕ argument in the outer loss contributes its partial derivative evaluated at acquisition. A direct regularizer $R(\phi)$ contributes $D_{v_t} R(\phi^{(\nu_t)})$ at the generation-time parameters; current-parameter weight decay is a separate convention.

F.4 Conditional Unbiasedness and Gaussian Moments

Direction count trades estimation error against computation without changing the trace-to-update interface. For K directions on the same realized response,

$$d_{t,j} = u_t^\top z_{t,j}, \qquad \widehat{g}_t = \frac{1}{K} \sum_{j=1}^{K} d_{t,j} v_{t,j}. \tag{32}$$

Lanes may share primal generation, but tangent arithmetic and caches are additional costs. The covariance and direction-sampling details follow here; Appendix F.6 gives the K-direction and minibatch decomposition.

Conditioning information. Let $\mathcal{C}_t$ be a sigma-field containing the acquisition coordinates, prompt, realized response, feedback, metadata, and all additional state needed to determine the realized objective and its historical gradient

$$g_t = \nabla_\phi L_t(\phi^{(\nu_t)}). \tag{33}$$

Thus g_t is $\mathcal{C}_t$-measurable. The conditional-moment assumption below is imposed on this specified objective-determining sigma-field.

Proof of Theorem 5.2. Assume

$$\mathbb{E}[v_t v_t^\top \mid \mathcal{C}_t] = I_p. \tag{34}$$

Historical exactness gives $\widehat{g}_t = d_t v_t = v_t v_t^\top g_t$. Conditional measurability of g_t therefore yields

$$\mathbb{E}[\widehat{g}_t \mid \mathcal{C}_t] = \mathbb{E}[v_t v_t^\top g_t \mid \mathcal{C}_t] \tag{35}$$

$$= \mathbb{E}[v_t v_t^\top \mid \mathcal{C}_t] g_t \tag{36}$$

$$= g_t. \tag{37}$$

The second-moment identity in Eq. (34) suffices for every gradient in the chosen coordinates. For a particular g_t, the weaker condition $\mathbb{E}[v_t v_t^\top \mid \mathcal{C}_t] g_t = g_t$ suffices. □

A sufficient execution condition. Let $\mathcal{H}_t$ denote state determined before sampling a fresh direction, including $\phi^{(\nu_t)}$, x_t, and the coordinate rule. Suppose $v_t \mid \mathcal{H}_t \sim \mathcal{N}(0, I_p)$. Let Ω_t collect the external randomness that determines primal generation, feedback, and objective construction. Assume v_t and Ω_t are conditionally independent given $\mathcal{H}_t$, and that $\mathcal{C}_t \subseteq \mathcal{H}_t \vee \sigma(\Omega_t)$. Thus the realized response, feedback, and objective construction are determined by the external randomness and pre-direction state.

For every bounded measurable function ψ, conditional independence and the tower property give

$$\mathbb{E}[\psi(v_t) \mid \mathcal{C}_t] = \mathbb{E}[\mathbb{E}[\psi(v_t) \mid \mathcal{H}_t, \Omega_t] \mid \mathcal{C}_t] \tag{38}$$

$$= \int \psi(v)\, d\mathcal{N}(0, I_p)(v). \tag{39}$$

Thus $v_t \mid \mathcal{C}_t \sim \mathcal{N}(0, I_p)$, which implies Eq. (34) and connects the probabilistic assumption to direction-independent generation and feedback.

Selection and direction reuse. If an activation event S_t is used, unbiasedness among activated records requires $\mathbb{E}[v_t v_t^\top \mid \mathcal{C}_t, S_t = 1] = I_p$, or the corresponding identity on g_t. Selection conditionally independent of v_t given $\mathcal{C}_t$ preserves the direction law within the selected interaction population. The conditional second moment also gives a direct description of direction-dependent selection. For $p = 1$, $v \sim \mathcal{N}(0, 1)$, and $c > 0$, integration by parts gives

$$\mathbb{E}[v^2 \mid |v| > c] = \frac{\int_c^\infty u^2 \varphi_{\mathrm{N}}(u)\, du}{1 - \Phi_{\mathrm{N}}(c)} \tag{40}$$

$$= 1 + \frac{c\varphi_{\mathrm{N}}(c)}{1 - \Phi_{\mathrm{N}}(c)} > 1, \tag{41}$$

where φ_{N} and Φ_{N} are the standard normal density and distribution function. Taking $f_\phi = \phi$ and $\ell(y) = y$ makes $z = v$ and $g = 1$, so selecting large $|z|$ gives an activated estimator with mean greater than the gradient $g = 1$.

Direction reuse can likewise change the conditional law. For $L(\phi) = \phi^2/2$, $\phi_0 = 1$, $\eta > 0$, and a reused scalar Gaussian v, the first step gives $\phi_1 = 1 - \eta v^2$. A second estimator is $\widehat{g}_1 = \phi_1 v^2$, hence $\mathbb{E}[\widehat{g}_1 \mid \phi_1] = \phi_1(1 - \phi_1)/\eta$. Both examples show why the estimator assumption is stated conditionally on the objective and its evaluation point.

Proposition F.1 (Gaussian estimator moments)**.** *Under the conditioning and gradient definition of Theorem 5.2, let $\widehat{g}_t = (g_t^\top v_t)v_t$. If $v_t \mid \mathcal{C}_t \sim \mathcal{N}(0, I_p)$, then*

$$\mathrm{Cov}(\widehat{g}_t \mid \mathcal{C}_t) = \|g_t\|^2 I_p + g_t g_t^\top, \tag{42}$$

$$\mathbb{E}[\|\widehat{g}_t - g_t\|^2 \mid \mathcal{C}_t] = (p + 1)\|g_t\|^2. \tag{43}$$

Proof of Proposition F.1. Assume the stronger law $v_t \mid \mathcal{C}_t \sim \mathcal{N}(0, I_p)$. Work conditionally and abbreviate $g = g_t$, $v = v_t$, and $\widehat{g} = (g^\top v)v$. All expectations in this proof are conditional on $\mathcal{C}_t$. Componentwise,

$$\widehat{g}_i = \sum_{a=1}^{p} g_a v_a v_i. \tag{44}$$

The fourth Gaussian moment is the three-pair identity of Isserlis (Isserlis, 1918):

$$\mathbb{E}[v_a v_b v_i v_j] = \delta_{ab}\delta_{ij} + \delta_{ai}\delta_{bj} + \delta_{aj}\delta_{bi}. \tag{45}$$

For independent standard normal coordinates, this identity can also be checked directly: an index occurring an odd number of times gives zero; two distinct repeated pairs give 1; and four equal indices give $\mathbb{E}[v_i^4] = 3$. Expanding the estimator's uncentered second moment now gives

$$\mathbb{E}[\widehat{g}_i \widehat{g}_j] = \sum_{a,b} g_a g_b \mathbb{E}[v_a v_b v_i v_j] \tag{46}$$

$$= \delta_{ij} \sum_a g_a^2 + g_i g_j + g_j g_i \tag{47}$$

$$= \|g\|^2 \delta_{ij} + 2 g_i g_j. \tag{48}$$

Therefore

$$\mathbb{E}[\widehat{g}\widehat{g}^\top] = \|g\|^2 I_p + 2gg^\top, \tag{49}$$

$$\mathrm{Cov}(\widehat{g}) = \mathbb{E}[\widehat{g}\widehat{g}^\top] - gg^\top = \|g\|^2 I_p + gg^\top. \tag{50}$$

Taking the trace of the covariance and of the uncentered second moment, respectively, yields

$$\mathbb{E}\|\widehat{g} - g\|^2 = (p+1)\|g\|^2, \tag{51}$$

$$\mathbb{E}\|\widehat{g}\|^2 = (p+2)\|g\|^2. \tag{52}$$

The distinction between these two traces accounts for the different dimension factors. □

From direction averaging to response averaging. The result above recovers a fixed realized gradient by averaging directions. A specified data law and valid interchange of differentiation and expectation connect this to a population gradient. For a parameter-dependent response law, differentiation also acts on its probability. Under fixed-support and differentiation-under-the-sum conditions,

$$\nabla_\phi \mathbb{E}_{a\sim\mu_\phi}[\Lambda_\phi(a)] = \mathbb{E}_{a\sim\mu_\phi}[\nabla_\phi \Lambda_\phi(a) + \Lambda_\phi(a)\nabla_\phi \log \mu_\phi(a)]\,. \tag{53}$$

Differentiating each product $\mu_\phi(a)\Lambda_\phi(a)$ gives this decomposition into pathwise and sampling-law contributions (Schulman et al., 2015). The following specialization uses the score-function contribution to connect trace activation to an expected-reward gradient.

F.5 Learning-Objective Bridge for On-Policy Rewards

Corollary F.2 connects an OFTA-supported reward-score interface to the expected-reward gradient through the score-function identity (Schulman et al., 2015). Fix the prompt x and acquisition point ϕ. Write $\bar{r}(x,a) = \mathbb{E}[r \mid x, a]$ and $J_x(\phi) = \mathbb{E}_{a\sim\pi_\phi(\cdot|x)}[\bar{r}(x,a)]$. The reward kernel is independent of ϕ conditional on (x,a). Assume a locally fixed support, differentiability of its positive probabilities, integrability of the estimator, and valid interchange of derivatives with the sum or integral. A finite horizon with EOS or fixed-budget truncation included in the trajectory definition is one possible setting; the score must be the likelihood of that actual law.

Corollary F.2 (Learning from on-policy outcome feedback)**.** *Fix the generation-time parameters ϕ and the prompt x. Let $a \sim \pi_\phi(\cdot \mid x)$, where π_ϕ is the normalized law of the executed response, including sampling transformations and stopping. Assume the following. The reward law given (x,a) has no explicit dependence on ϕ. Scoring uses the log-probability of this same law, which is differentiable with locally fixed support near ϕ. Differentiation and expectation can be interchanged. The conditional direction moment of Theorem 5.2 holds. Then, for $L(\phi; a, r) = -r \log \pi_\phi(a \mid x)$ with (a,r) held fixed,*

$$\mathbb{E}_{a,r,v}[\widehat{g} \mid x, \phi] = -\nabla_\phi \mathbb{E}_{a,r}[r \mid x, \phi]. \tag{54}$$

Proof of Corollary F.2. The realized surrogate is $L(\phi; a, r) = -r \log \pi_\phi(a \mid x)$, with response and reward held fixed when differentiated. By generation exactness and the conditional second-moment assumption,

$$\mathbb{E}_v[\widehat{g} \mid x, \phi, a, r] = -r\nabla_\phi \log \pi_\phi(a \mid x). \tag{55}$$

The tower property and the likelihood-ratio identity then give

$$\mathbb{E}_{a,r,v}[\widehat{g} \mid x, \phi] = -\sum_a \pi_\phi(a \mid x)\bar{r}(x,a)\nabla_\phi \log \pi_\phi(a \mid x) \tag{56}$$

$$= -\sum_a \bar{r}(x,a)\nabla_\phi \pi_\phi(a \mid x) \tag{57}$$

$$= -\nabla_\phi J_x(\phi). \tag{58}$$

For continuous actions, replace the sum by the corresponding integral. This is the score-function construction discussed by Schulman et al. (2015). Direction averaging contributes the exact sample-surrogate gradient; averaging the matched-policy responses connects that gradient to the expected outcome. □

The result averages the derivative of the realized surrogate with (a, r) clamped, evaluated at the acquisition parameters. Its matched-policy score includes the actual normalized sampling transformations and termination law. This supplies an expected-reward interpretation for the reward-score interface; APEX uses the coupled objective specified in Appendix B. Section F.8 quantifies the derivative discrepancy when an update is applied at a later parameter version.

F.6 Multiple Directions and Minibatch Decomposition

Corollary F.3 (Independent Gaussian directions)**.** *For a common realized objective with gradient g and directions conditionally independent and standard Gaussian given its objective information $\mathcal{C}$, let $\widehat{g}_K = K^{-1}\sum_{j=1}^{K}(g^\top v_j)v_j$. Then*

$$\mathbb{E}[\widehat{g}_K \mid \mathcal{C}] = g, \qquad \mathbb{E}[\|\widehat{g}_K - g\|^2 \mid \mathcal{C}] = \frac{p+1}{K}\|g\|^2. \tag{59}$$

Proof of Corollary F.3. Throughout this proof only, all expectations and covariances are conditional on $\mathcal{C}$. Condition on the realized objective and parameter point, so its gradient g is fixed. Let $v_1, \ldots, v_K$ be conditionally independent $\mathcal{N}(0, I_p)$ directions and define

$$\widehat{g}_K = \frac{1}{K}\sum_{j=1}^{K}(g^\top v_j)v_j, \qquad e_j = (g^\top v_j)v_j - g. \tag{60}$$

Each e_j has conditional mean zero. For $i \neq j$, conditional independence gives $\mathbb{E}[e_i^\top e_j] = \mathbb{E}[e_i]^\top\mathbb{E}[e_j] = 0$. Consequently,

$$\mathbb{E}[\widehat{g}_K] = g, \tag{61}$$

$$\mathbb{E}\|\widehat{g}_K - g\|^2 = \frac{1}{K^2}\left(\sum_j \mathbb{E}\|e_j\|^2 + \sum_{i\neq j}\mathbb{E}[e_i^\top e_j]\right) \tag{62}$$

$$= \frac{p+1}{K}\|g\|^2. \tag{63}$$

The same expansion for the covariance matrix and the centered norm identity give

$$\mathrm{Cov}(\widehat{g}_K) = \frac{1}{K}\left(\|g\|^2 I_p + gg^\top\right), \tag{64}$$

$$\mathbb{E}\|\widehat{g}_K\|^2 = \left(1 + \frac{p+1}{K}\right)\|g\|^2. \tag{65}$$

□

Averaging K independent directions reduces the mean-squared error by a factor of K for the same response, objective, and parameter point. Multiple tangent lanes can share primal computation, with computation and storage for each tangent. Antithetic directions yield identical estimator samples, since $(g^\top(-v))(-v) = (g^\top v)v$.

Minibatch and shared-direction variance decomposition. At one fixed parameter point, suppose the sample gradients $g_1, \ldots, g_B$ are independent with $\mathbb{E}[g_i] = g$ and $\mathbb{E}\|g_i - g\|^2 = \sigma^2$. Define $\bar{g}_B = B^{-1}\sum_i g_i$. Expanding the square and using centered cross terms gives

$$\mathbb{E}\|\bar{g}_B - g\|^2 = \frac{1}{B^2}\left(\sum_i \mathbb{E}\|g_i - g\|^2 + \sum_{i\neq j}\mathbb{E}[(g_i - g)^\top(g_j - g)]\right) \tag{66}$$

$$= \frac{\sigma^2}{B}, \tag{67}$$

$$\mathbb{E}\|\bar{g}_B\|^2 = \|g\|^2 + \frac{\sigma^2}{B}. \tag{68}$$

The corresponding zero-cross-term conditions also suffice for this $1/B$ scaling.

Draw K independent Gaussian directions independently of the batch, with each direction shared by every example. The resulting estimator is

$$\widehat{g}_{B,K} = \frac{1}{K}\sum_{j=1}^{K}(\bar{g}_B^\top v_j)v_j = \frac{1}{BK}\sum_{i=1}^{B}\sum_{j=1}^{K}(g_i^\top v_j)v_j. \tag{69}$$

Let $\mathcal{B} = \sigma(g_1, \ldots, g_B)$. Applying the same-gradient result conditional on $\mathcal{B}$ gives

$$\mathbb{E}[\widehat{g}_{B,K} - \bar{g}_B \mid \mathcal{B}] = 0, \tag{70}$$

$$\mathbb{E}[\|\widehat{g}_{B,K} - \bar{g}_B\|^2 \mid \mathcal{B}] = \frac{p+1}{K}\|\bar{g}_B\|^2. \tag{71}$$

The error decomposition is

$$\widehat{g}_{B,K} - g = (\widehat{g}_{B,K} - \bar{g}_B) + (\bar{g}_B - g). \tag{72}$$

Its cross term vanishes since

$$\mathbb{E}[(\widehat{g}_{B,K} - \bar{g}_B)^\top(\bar{g}_B - g)] = \mathbb{E}\big[\mathbb{E}[\widehat{g}_{B,K} - \bar{g}_B \mid \mathcal{B}]^\top(\bar{g}_B - g)\big] \tag{73}$$

$$= 0. \tag{74}$$

Taking expectations of the two squared terms therefore gives

$$\mathbb{E}\|\widehat{g}_{B,K} - g\|^2 = \frac{\sigma^2}{B} + \frac{p+1}{K}\left(\|g\|^2 + \frac{\sigma^2}{B}\right), \tag{75}$$

□

Interpreting the decomposition. Equation (75) separates sample variability from directional-estimation variability at a fixed parameter point. For a coupled loss such as APEX, one sample gradient represents the entire group, with the independence assumptions imposed between groups. For shared prompts, the same argument can be applied conditionally on the prompt when the corresponding conditional independence and moment assumptions hold.

For any batch mean with mean g and variance $\tau_B^2 = \mathbb{E}\|\bar{g}_B - g\|^2$, conditioning on the batch gives $\tau_B^2 + (p+1)(\|g\|^2 + \tau_B^2)/K$; τ_B^2 incorporates any sample correlation. With $\sigma^2 = 0$, the shared-direction estimator still has error $(p+1)\|g\|^2/K$, independently of B. Thus minibatch size and direction count control distinct components of error, and the chosen coordinate dimension directly enters the forward-gradient variance (Fournier et al., 2023).

F.7 Trainable Coordinates, Reparameterizations, and Masks

Let full model parameters be $\theta = h(\phi) \in \mathbb{R}^D$, and write the corresponding fixed-response loss as $\mathcal{L}(\theta)$, with $L(\phi) = \mathcal{L}(h(\phi))$. For differentiable h, the derivative of the composite statistic and the coordinate gradient are

$$D_v[f_{h(\phi)}] = J_\theta f_{h(\phi)}\, J_\phi h(\phi) v, \tag{76}$$

$$\nabla_\phi L(\phi) = J_\phi h(\phi)^\top \nabla_\theta \mathcal{L}(h(\phi)). \tag{77}$$

Indeed, $h(\phi+\epsilon v) = h(\phi)+\epsilon J_\phi h(\phi)v + o(|\epsilon|)$, and substitution into the differentiability expansion of f gives the first identity. Applying the scalar chain rule to $\mathcal{L}$ gives the second. Thus trace acquisition, contraction, and conditional estimation apply in the selected trainable coordinates.

The induced model-parameter update. Consider a coordinate gradient evaluated and applied at the same parameter point, conditional on objective information $\mathcal{C}$ satisfying the estimator assumptions above. Set $J_h = J_\phi h(\phi)$ and make a coordinate step $\delta\phi = -\eta\widehat{g}_\phi$. Differentiability alone gives

$$h(\phi + \delta\phi) - h(\phi) = J_h\delta\phi + r(\delta\phi), \qquad \frac{\|r(\delta\phi)\|}{\|\delta\phi\|} \longrightarrow 0 \quad \text{as } \delta\phi \to 0. \tag{78}$$

The conditional expectation of the linear term is

$$\mathbb{E}[J_h \delta\phi \mid \mathcal{C}] = -\eta J_h \nabla_\phi L = -\eta J_h J_h^\top \nabla_\theta \mathcal{L}. \tag{79}$$

The map $J_h J_h^\top$ describes the effect of the coordinates on the expected linearized step. With orthonormal columns of J_h, it is the orthogonal projector onto their column space. For $h(\phi) = c\phi$ with fixed scalar c, the expected weight-space step is exactly $-\eta c^2 \nabla_\theta \mathcal{L}$.

If J_h is M_h-Lipschitz along the step segment, the integral remainder satisfies

$$r(\delta\phi) = \int_0^1 \big[J_h(\phi + s\delta\phi) - J_h(\phi)\big]\delta\phi \, ds, \tag{80}$$

$$\|r(\delta\phi)\| \le \frac{M_h}{2}\|\delta\phi\|^2. \tag{81}$$

For a random step, this is a pathwise bound. If the same M_h holds on every realized step segment, then the Gaussian K-direction second moment also gives

$$\|\mathbb{E}[r(-\eta\widehat{g}_\phi) \mid \mathcal{C}]\| \le \frac{M_h \eta^2}{2}\left(1 + \frac{p+1}{K}\right)\|\nabla_\phi L\|^2. \tag{82}$$

This bound quantifies the nonlinear remainder under a common Lipschitz constant across the realized step segments.

For the LoRA parameterization $W = W_0 + \gamma BA$ with fixed γ, the coordinate-map tangent is $\dot{W} = \gamma(\dot{B}A + B\dot{A})$ (Hu et al., 2022). This provides the corresponding instance of the general coordinate construction.

Masks and subspaces. Let $P_\mathcal{S} \in \mathbb{R}^{D\times D}$ be an orthogonal projector, so $P_\mathcal{S}^\top = P_\mathcal{S}$ and $P_\mathcal{S}^2 = P_\mathcal{S}$. For an ambient gradient g_θ and a Gaussian $\zeta \sim \mathcal{N}(0, I_D)$ independent of the fixed objective, set $v_\theta = P_\mathcal{S}\zeta$. Then

$$\mathbb{E}[v_\theta v_\theta^\top] = P_\mathcal{S}\mathbb{E}[\zeta\zeta^\top]P_\mathcal{S}^\top = P_\mathcal{S}, \tag{83}$$

$$\mathbb{E}[(g_\theta^\top v_\theta)v_\theta] = P_\mathcal{S} g_\theta. \tag{84}$$

Thus the estimator targets the projected gradient. A general linear mask or map M instead induces second moment $MM^\top$. An orthonormal basis for $\mathcal{S}$ defines isotropic reduced coordinates in which the earlier dimension formulas apply.

F.8 Historical–Fresh Discrepancy

Fix the same prompt, realized response, feedback, metadata, scoring convention, numerical reference values, and direction in both comparisons. Define

$$d_t^{\mathrm{hist}} = \nabla L_t(\phi^{(\nu_t)})^\top v_t, \qquad d_t^{\mathrm{fresh}} = \nabla L_t(\phi^{(k_t)})^\top v_t. \tag{85}$$

Fresh differentiation evaluates the same realized objective and recorded response at the application parameters. Fresh Replay implements this comparator by reevaluating the recorded interaction; stored-trace activation evaluates its historical derivative.

Proposition 5.3 in Section 5 bounds the gap between these two derivatives. The corresponding directional estimators satisfy

$$\|d_t^{\mathrm{hist}} v_t - d_t^{\mathrm{fresh}} v_t\| \le L_{\nabla,t}\|\phi^{(k_t)} - \phi^{(\nu_t)}\| \, \|v_t\|^2. \tag{86}$$

Proof of Proposition 5.3. Suppose ∇L_t is $L_{\nabla,t}$-Lipschitz on the line segment joining $\phi^{(\nu_t)}$ and $\phi^{(k_t)}$. Put

$$b_t = \nabla L_t(\phi^{(\nu_t)}) - \nabla L_t(\phi^{(k_t)}), \qquad \Delta_t = \|\phi^{(k_t)} - \phi^{(\nu_t)}\|. \tag{87}$$

Cauchy–Schwarz and the stated smoothness give

$$|d_t^{\mathrm{hist}} - d_t^{\mathrm{fresh}}| = |b_t^\top v_t| \tag{88}$$

$$\le \|b_t\| \, \|v_t\| \tag{89}$$

$$\le L_{\nabla,t}\Delta_t\|v_t\|. \tag{90}$$

Multiplication by the same direction gives

$$\|d_t^{\mathrm{hist}} v_t - d_t^{\mathrm{fresh}} v_t\| = |d_t^{\mathrm{hist}} - d_t^{\mathrm{fresh}}| \, \|v_t\| \tag{91}$$

$$\leq L_{\nabla,t} \Delta_t \|v_t\|^2. \tag{92}$$

For two steps from the same current point with the same learning rate $\eta_t \geq 0$,

$$\|(-\eta_t d_t^{\mathrm{hist}} v_t) - (-\eta_t d_t^{\mathrm{fresh}} v_t)\| \leq \eta_t L_{\nabla,t} \Delta_t \|v_t\|^2. \tag{93}$$

These pathwise inequalities hold for each realized direction and intervening parameter trajectory. □

Parameter drift over the delay. The bound identifies parameter displacement as the quantity controlling historical–fresh discrepancy. At unchanged parameters the discrepancy is zero for any waiting time. The triangle inequality gives

$$\Delta_t \leq \sum_{j=\nu_t}^{k_t-1} \|\phi^{(j+1)} - \phi^{(j)}\| \tag{94}$$

so bounds on the intervening step sizes translate an update-count gap into a displacement bound.

Conditional mean-square refinements. If, additionally, $\mathbb{E}[v_t v_t^\top \mid b_t] = I_p$, then

$$\mathbb{E}[(b_t^\top v_t)^2 \mid b_t] = b_t^\top \mathbb{E}[v_t v_t^\top \mid b_t] b_t = \|b_t\|^2. \tag{95}$$

Under the stronger condition $v_t \mid b_t \sim \mathcal{N}(0, I_p)$, $\mathbb{E}[\|(b_t^\top v_t) v_t\|^2 \mid b_t] = (p+2)\|b_t\|^2$. These refinements explicitly condition on the gradient displacement: for example, $b_t = v_t$ instead gives $\mathbb{E}[(b_t^\top v_t)^2 \mid b_t] = \|b_t\|^4$. The pathwise result in Proposition 5.3 applies directly to direction-dependent parameter trajectories.

F.9 Objective-Relative Record Sufficiency

Suppose every supported future loss depends on model parameters only through the declared $y = f_\phi(x, a; m^{\mathrm{gen}})$, and $\nabla_y \ell(y, r; m)$ is computable from the retained record, released feedback, and declared auxiliary inputs. Auxiliary acquisition may have its own model and waiting costs; sufficiency concerns activation once those inputs are available. For K directions, retain

$$\mathcal{T}_t = \big(y_t, z_{t,1}, \ldots, z_{t,K}, \xi_{t,1}, \ldots, \xi_{t,K}, \nu_t, m_t^{\mathrm{gen}}, \iota_t\big), \tag{96}$$

where ι_t contains whatever response or association information the objective and feedback interface actually require. Let $\mathscr{R}$ denote the fixed direction-reconstruction rule, including coordinate layout, generator, and numerical conventions, so that $v_{t,j} = \mathscr{R}(\xi_{t,j})$.

Sufficiency for the declared interface. Given $\mathcal{T}_t$, released r_t, and the declared auxiliary inputs extending m_t^{gen} to m_t, first compute $u_t = \nabla_y \ell(y_t, r_t; m_t)$. For each lane, compute $d_{t,j} = u_t^\top z_{t,j}$ and reconstruct $v_{t,j}$ using $\mathscr{R}$. The activation identity proves that each $d_{t,j}$ is the historical directional derivative. Thus $d_{t,j} v_{t,j}$, or their average, is constructible without historical model reevaluation. The retained record therefore supports every loss in the declared family through feedback-time contraction and direction reconstruction. □

Matching the record to the loss. Both the statistic and its tangent enter activation. For $f_\phi = \phi$ and $\ell(y, r) = (y - r)^2/2$, retaining $y = \phi$ alongside $z = v$ gives $d = (\phi - r)v$. The declared statistic also determines the supported loss family. For example, $y = \phi_1$ and $z = v_1$ agree at the historical points $(0, 0)$ and $(0, 1)$ for the same direction, whereas the objective $\phi_2^2/2$ has respective directional derivatives 0 and v_2. Supporting this additional objective requires its own statistic and tangent.

The direction seed, reconstruction rule, version identifier, and response-association information have distinct roles in the record: reconstructing a direction, identifying its acquisition point, and associating the retained statistics with feedback. The probability results use the stated conditional direction law; the Gaussian formulas apply to the ideal Gaussian sampling model.

The output-tensor part of one record has $d_y(1 + K)$ scalar entries, hence size $O(d_y(1 + K))$ at fixed precision. For N scalar response scores this is $O(N(1+K))$. For a coupled group loss, each lane's N

score tangents must correspond to one common direction and parameter point to represent the group's JVP; u_t must include all output-side coupling derivatives. Parameter-dependent group normalizers contribute their derivatives unless the objective explicitly defines them with stop-gradient.

The complete retained record also includes the direction seeds and reconstruction rule, response and feedback-association information, reference metadata, and version identifiers. Generation working memory includes primal and tangent caches; application accesses and updates the trainable parameters. System-wide pending storage sums retained records and auxiliary state across outstanding interactions. These quantities provide separate accounting units for acquisition, retention, and activation.

F.10 Finite-Precision Trace Perturbations

Let $\widetilde{y} = y + e_y$ and $\widetilde{z} = z + e_z$, with fixed feedback and metadata. Write $u(y) = \nabla_y \ell(y, r; m)$ and assume u is L_{out}-Lipschitz on a neighborhood containing the segment from y to $\widetilde{y}$. Then

$$u(\widetilde{y})^\top \widetilde{z} - u(y)^\top z = u(y)^\top e_z + [u(\widetilde{y}) - u(y)]^\top (z + e_z), \tag{97}$$

$$\left| u(\widetilde{y})^\top \widetilde{z} - u(y)^\top z \right| \le \|u(y)\| \, \|e_z\| + L_{\mathrm{out}} \|e_y\| (\|z\| + \|e_z\|). \tag{98}$$

If the computed outer gradient is $u(\widetilde{y}) + e_u$ and the final contraction has scalar arithmetic error e_{ctr}, the right-hand side additionally contains $\|e_u\| \, \|\widetilde{z}\| + |e_{\mathrm{ctr}}|$. With exact direction reconstruction, the corresponding estimator-error bound is multiplied by $\|v\|$. The bound separates tangent error, statistic error, outer-gradient error, and contraction error in the activated derivative. Together with the acquisition and activation identities, it connects numerical accuracy of the retained record to accuracy of the historical directional update.