When Gradients Collide: Failure Modes of Multi-Objective Prompt Optimization for LLM Judges

Parth Darshan[♢*]
[♢]IIT Jodhpur
b22cs040@iitj.ac.in

Abhishek Divekar[♠†*]
[♠]Independent
adivekar@utexas.edu

Abstract

Customizing an LLM judge to a specific task or domain involves optimizing its prompt across multiple evaluation criteria simultaneously. Textual gradient methods automate this for a single judge criterion, however they produce natural-language critiques, not numerical vectors. Thus, the conflict-resolution toolkit of multi-task learning (PCGrad, MGDA) doesn't apply to the multi-objective textual gradient setting. We test five decomposition modes of textual gradient optimizers by varying how much cross-task information the loss, gradient and optimizer LLMs share. In 6 of 10 configurations on SUMMEVAL, we observe that optimization never improves over the initial prompt (Table 1). Gradient specificity drops by 59% (from 9.0 to 3.7) when the gradient LLM processes multiple criteria jointly Table 4). Separately, we observe that naively combining per-task instructions into a single prompt degrades Spearman's ρ by -5.3%. These results identify two separable failure modes: optimization-time *gradient dilution* and inference-time *instruction interference*, which together constrain the design space for multi-objective judge optimization using textual feedback.

1 Introduction

Modern LLM judges are used to evaluate text along multiple quality dimensions at once. SUMMEVAL (Fabbri et al., 2021) scores summaries on four dimensions simultaneously; MT-Bench (Zheng et al., 2023) spans eight categories from coding to roleplay. Kim et al. (2024) developed Prometheus 2 specifically because existing evaluators "do not possess the ability to evaluate based on *custom evaluation criteria*". Prompt optimization methods like TEXTGRAD (Yüksekgönül et al., 2025), OPRO (Yang et al., 2024), and GEPA (Agrawal et al., 2026) can improve prompts automatically, but they optimize a single objective. Whether they extend to multi-criteria judges, where one prompt must satisfy several evaluation dimensions at once, is unknown.

The core challenge is structural. The conflict-resolution tools of numerical multi-task learning do not apply directly, because textual gradients lack the vector-space structure these methods require. When multi-task deep learning models encounter conflicting task gradients, methods like PCGrad (Yu et al., 2020) and MGDA (Sener and Koltun, 2018) resolve conflicts via projection or constrained optimization. As textual gradients are natural-language strings, they do not have equivalent concepts of magnitude, inner products, or linear subspaces. The instruction "Make the coherence rubric more specific" cannot be numerically projected against "simplify the fluency criteria".

Prior work addresses adjacent problems but not this intersection. For example, intra-task feedback quality has been studied: recently Chu et al. (2026) identify *rule dilution* when heterogeneous error modes are aggregated within a single optimization step, and Melcer et al. (2025) show that the gradient analogy does not accurately explain why Automatic Prompt Optimization (APO) methods succeed. Multi-objective prompt optimization also exists: Jafari et al. (2024) find that MGDA underperforms volume-based alternatives for discrete prompt tokens. Multi-task judge *training* is studied extensively (Whitehouse et al., 2026; Yang et al., 2026; Wang et al., 2025), but these methods update weights, not prompts. To our knowledge, no prior work studies multi-criteria *prompt* optimization for LLM judges.

Concretely, we comprehensively test all five decomposition modes for textual gradient optimization (SINGLE-TASK, SSS, SSC, SCC, CCC) that encode whether each pipeline stage (loss, gradient, optimizer) processes tasks separately (S) or combined (C). We also introduce two process-level

[*]Equal contribution. [†]Research lead; senior and corresponding author.

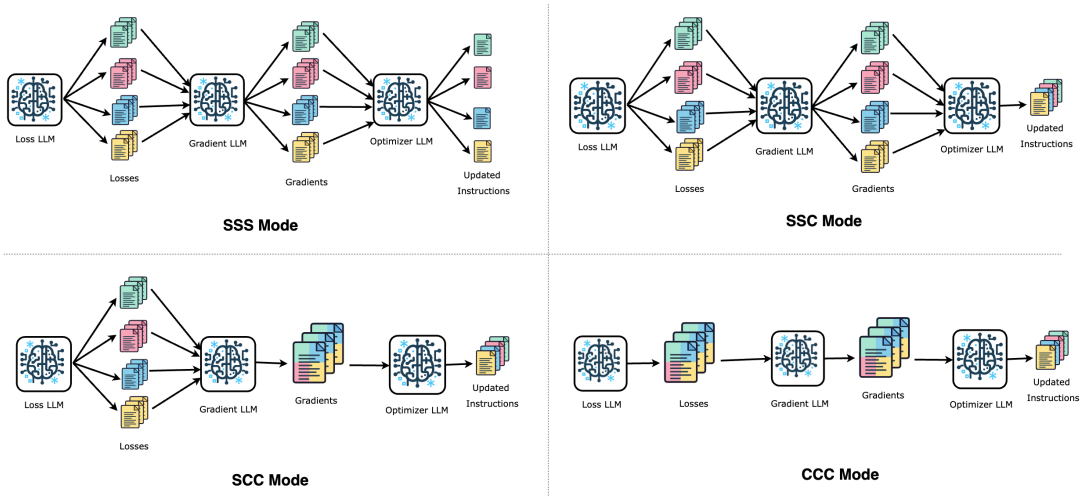

Figure 1: Overview of the optimization pipeline. Each step consists of four stages: (1) the task model predicts scores, (2) the loss LLM critiques predictions against ground truth, (3) the gradient LLM converts critiques into instruction edits, and (4) the optimizer LLM rewrites the prompt. The decomposition mode determines whether each stage operates per-task (Separate) or over all tasks jointly (Combined). Only the per-task instruction text is updated; the prompt skeleton and output format remain frozen throughout optimization.

diagnostics: *gradient specificity* (how targeted each gradient is to a single task) and *feedback adherence* (whether the optimizer follows the gradient). We evaluate on SUMMEVAL (Fabbri et al., 2021) with four criteria, using two validation settings, and $N = 3$ runs of 12 steps each.

Our results reveal consistent structure in how and why multi-criteria prompt optimization stagnates. In 6 of 10 configurations, optimization never exceeds the initial generic prompt (§4, Table 1); only SINGLE-TASK with `val=mae` improves ($+0.031$ Spearman at step 2). Our diagnostics localize this bottleneck: gradient specificity drops by 59% (9.0→3.7) when the gradient LLM processes all tasks jointly §5.1, Table 4). Feedback adherence remains high (7.8–8.8), so the optimizer incorporates the gradient's suggestions regardless of their specificity. A separate oracle experiment shows that even independently optimal per-task instructions *degrade* by -0.053 Spearman when combined into one prompt (§5.2, Table 2).

These results identify two separable failure modes: optimization-time *gradient dilution* and inference-time *instruction interference*. Our contributions are as follows: (i) We present an empirical study of multi-criteria textual gradient optimization for LLM judges across five decomposition modes (§3–4, Table 1); (ii) Two process-level diagnostics (*gradient specificity* and *feedback adherence*) that localize the bottleneck to gradient quality, not optimizer compliance (§5.1, Figure 3, Table 3). (iii) An oracle-instruction experiment which isolates inference-time *instruction interference* as a failure mode distinct from optimization-time dilution (§5.2, Table 2).

2 Related Work

Textual Gradient Methods. Early methods in Prompt optimization used scalar signals: OPRO (Yang et al., 2024) rewrites prompts by conditioning on (prompt, score) histories, and APE (Zhou et al., 2023) generates candidates from demonstrations and selects the highest-scoring variant. PROTEGI (Pryzant et al., 2023) replaced scalar signals with *textual gradients*: natural-language critiques that guide prompt rewrites. TEXTGRAD (Yüksekgönül et al., 2025) extended this to multi-component computation graphs, propagating critiques through LLM pipelines in analogy to backpropagation.

Subsequent work has questioned the gradient analogy. GPO (Tang et al., 2025) decomposed textual optimization into two factors (update direction and update method), drawing analogies to gradient, momentum, and learning rate, but found that adding a reflection step *hurts* performance. Melcer et al. (2025) showed empirically that the gradient decomposition (the chain-rule structure central to the analogy) does not consistently improve prompt optimization, and that the gradient metaphor does not accurately explain why APO methods succeed.

	MAE validation				No validation			
Mode	Initial	Best† (step)	Δ	HVI	Initial	Best† (step)	Δ	HVI
SINGLE-TASK	0.274	**0.305**† (2)	**+0.031**	—	0.269	**0.284**† (5)	**+0.015**	—
SSS	0.284	0.284 (0)	+0.000	2.749	0.283	0.283 (0)	+0.000	2.867
SSC	0.289	0.289 (0)	+0.000	2.832	0.283	0.291† (2)	+0.007	2.845
SCC	0.282	0.282 (0)	+0.000	2.801	0.282	0.282 (0)	+0.000	2.779
CCC	0.285	0.296 (9)	+0.012	**2.900**	0.287	0.287 (0)	+0.000	**2.983**

Table 1: Main results on SUMMEVAL. *avg* ρ is the task-averaged Spearman correlation (mean over $N = 3$ runs). † marks the best avg ρ within each validation setting. Δ is the improvement from the initial prompt to the best step. HVI is the hypervolume indicator at step 12 (higher = more Pareto-diverse prompts). We evaluate five decomposition modes (SSS through CCC) plus an independent single-task baseline (SINGLE-TASK), covering the full spectrum from separate to joint optimization. In 6 of 10 configurations, the initial generic prompt ("*Rate from 1 to 5*") is never exceeded.

GEPA (Agrawal et al., 2026) evolves programs via reflective mutation with Pareto selection. To our knowledge, all methods above optimize a single objective and none provide a mechanism to observe or control how per-task feedback interacts during optimization.

Multi-Objective Prompt Optimization. Multi-objective prompt optimization is a nascent area with two distinct approaches. Population-based methods maintain a Pareto front of candidate prompts: MOPO (Menchaca Resendiz and Klinger, 2025) applies NSGA-II (Deb et al., 2002) with LLM-based mutation to affective text generation, and ParetoPrompt (Zhao et al., 2025) decomposes objectives into scalarized subproblems. MORL-Prompt (Jafari et al., 2024) adapts multi-objective reinforcement learning to discrete prompt tokens but found that MGDA (Sener and Koltun, 2018) underperforms the simpler product-of-rewards at balancing competing objectives. Evolutionary approaches include EVOPROMPT (Guo et al., 2024) for single-objective and Baumann and Kramer (2024) for multi-objective sentiment balancing. These methods operate at the *candidate-selection* level: they choose which prompts to keep from a population. To our knowledge, none of the above studies how per-task feedback interacts *within* a single textual gradient trajectory, which is the setting we investigate.

LLM-as-a-Judge Evaluation. MT-Bench and Chatbot Arena (Zheng et al., 2023) established the paradigm of LLM-based evaluation where human annotation is expensive. Subsequent works use a fixed prompt to improve judge quality through tuning weights on rubric-grounded data (Kim et al., 2024), reinforcement learning (Whitehouse et al., 2026), debiasing (Yang et al., 2026), and preference optimization (Wang et al., 2025). A parallel line of work optimizes the evaluation prompt itself. CARO (Chu et al., 2026) identifies *rule dilution* when heterogeneous error modes are aggregated into a single optimization step. RRD (Shen et al., 2026) recursively decomposes rubrics to improve coverage and remove redundancy. MPO (Sharma and Henley, 2026) applies section-local textual gradients to individual prompt components (role, context, constraints) independently. MAPGD (Han et al., 2025) coordinates multiple gradient agents, using semantic similarity to resolve conflicting edits.

To our knowledge, all of these operate on a single evaluation objective, or decompose along axes other than evaluation criteria. None jointly optimizes a judge prompt across multiple criteria while preserving per-task gradient observability (i.e., the ability to trace how each criterion's feedback shaped each rewrite), the setting we study.

3 Experimental Setup

Dataset and task. We evaluate on SUMMEVAL (Fabbri et al., 2021), a summarization meta-evaluation benchmark. The original dataset contains 100 source news articles, each paired with summaries from 16 different summarization systems (1,600 pairs total), with expert annotations from 3 annotators per pair.

We use the expert annotations directly (not crowd-sourced), parse each annotator's scores, and randomly split with seed 42 into 160 pairs for training (used as optimization batches) and 480 pairs for held-out evaluation. Each pair is scored on four dimensions (*fluency*, *relevance*, *coherence*, *consistency*) on a 1–5 Likert scale. These four dimensions are the tasks in our multi-task optimization setting:

the judge prompt must produce accurate scores across all four simultaneously.

We report Spearman rank correlation (ρ) between predicted and human scores as the primary metric, following prior work on LLM-based evaluation. Unless stated otherwise, all reported results are *task-averaged Spearman* $\bar{\rho}$, the arithmetic mean of per-task ρ values across the four dimensions.

TextGrad pipeline. We implement a 4-stage optimization loop (Figure 1). At each step, a **task model** (Qwen3-8B, Yang et al. (2025)) predicts dimension scores for a batch of examples using the current prompt. A **loss LLM** (Qwen3-235B-A22B; $T = 0.3$) then critiques the predictions by comparing them to ground-truth annotations and produces a natural-language loss for each example. A **gradient LLM** (Qwen3-235B-A22B; $T = 0.3$) aggregates the per-example losses into a textual gradient: a structured set of instruction-level edit suggestions. An **optimizer LLM** (Qwen3-235B-A22B; $T = 0.7$) rewrites the prompt's per-task instructions based on the gradient. The higher optimizer temperature encourages diverse rewrites; the lower loss and gradient temperatures promote consistent critiques. Only the per-task instruction text is modified; the prompt skeleton (role preamble, output format, few-shot examples) is frozen throughout (see Appendix A for the full initial prompt and an example of optimized instructions). This design isolates the effect of instruction wording from confounds introduced by structural prompt changes.

Decomposition modes. We parameterize multi-task interaction via a 3-letter *decomposition code*. Each letter denotes whether a stage operates in **S**eparate (per-task) or **C**ombined (all-tasks-joint) mode, applied to the loss, gradient, and optimizer stages respectively. Four multi-task modes span the design space. **SSS** operates all three stages independently per task. **SSC** computes loss and gradient per task, but the optimizer receives all four gradients simultaneously and rewrites the prompt in a single call. **SCC** computes only the loss per task; the gradient LLM receives critiques from all four tasks and produces a unified set of edits. **CCC** operates all stages jointly. We also include a **Single** baseline in which each task is optimized in a completely independent run.

The key architectural boundary lies between SSC and SCC. In SSC, the gradient LLM sees one task at a time. In SCC, it must reconcile feedback from all four tasks into a coherent edit plan.

Validation. We evaluate two validation strategies. Under **val=mae**, a candidate prompt is accepted only if its mean absolute error on a held-out validation set (100 examples) doesn't exceed that of the current prompt. This acts as a monotonic filter that prevents regression. Under **val=none**, every candidate is accepted unconditionally; we can observe the optimization trajectory without gating. For each configuration (decomposition mode $\times$ validation strategy), we run 3 independent trials with different random seeds, each for 12 optimization steps. We report the mean and standard deviation of task-averaged Spearman $\bar{\rho}$ across the 3 runs.

4 Results

Table 1 reports task-averaged Spearman ρ for each decomposition mode and validation configuration, averaged over $N = 3$ runs. In 6 of 10 mode $\times$ validation configurations, the best prompt is the initial generic prompt ("*Rate from 1 to 5*"): optimization either fails to improve or actively degrades performance. The only multi-task mode that improves is CCC with MAE validation, which achieves a modest $+0.012$ Spearman gain over 12 steps (Table 1, row CCC, Δ column). This gain is not statistically significant at $N = 3$. Only the single-task baseline with MAE validation achieves meaningful improvement: $+0.031$ Spearman at step 2 (Table 1, **bold**). This confirms that the TEXTGRAD pipeline can improve individual-task prompts when gradient signal isn't contaminated by cross-task information.

The trajectory plots in Figure 2 reveal the dynamics behind these aggregates. Without a validation gate (`val=none`, bottom row), SSC drops from 0.283 to 0.184 by step 7; SCC shows comparable degradation. The optimizer proposes changes that improve the training-batch loss but harm held-out generalization. MAE validation partially mitigates this by rejecting updates that increase validation error. With the validation gate active (top row), trajectories flatten rather than decline: the gate prevents catastrophic degradation but can't compensate for uninformative gradients.

Performance degrades roughly along the spectrum Single $>$ SSS $>$ SSC $>$ SCC $>$ CCC in the `val=none` condition. This is consistent with the hypothesis that increasing cross-task coupling amplifies gradient dilution. The pattern is non-monotonic under MAE validation, however: CCC slightly outperforms SSC. Full coupling may occasionally produce complementary gradients that

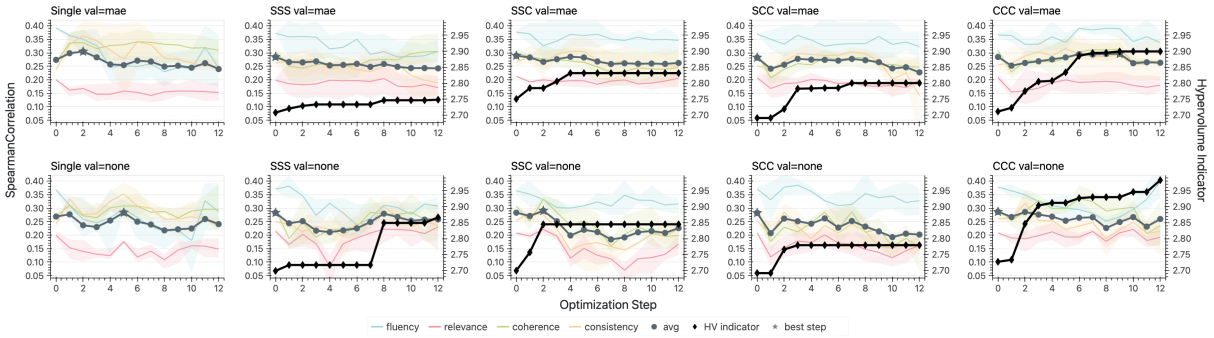

Figure 2: Per-task Spearman ρ over optimization steps on SUMMEVAL, averaged over $N = 3$ runs (shaded bands show min–max). Columns: five decomposition modes (Single through CCC). **Top row:** MAE validation. **Bottom row:** `val=none`. Gray line: task-averaged ρ; stars mark best step. Black diamonds (right axis, multi-task modes only): accumulated hypervolume indicator. Without validation, multi-task modes degrade; with it, they plateau. Only Single improves.

survive the validation gate. The process-level diagnostics in §5 disentangle these effects.

Despite stagnation in average Spearman, the hypervolume indicator (HVI) shows a contrasting trend (see also Appendix B). For CCC, HVI grows from 2.712 at step 0 to 2.900 at step 12 (+6.9%; Table 1, HVI column). The optimizer discovers diverse specialist prompts that expand the Pareto front, even when no single prompt dominates the initialization on all four tasks. Menchaca Resendiz and Klinger (2025) report a similar pattern: multi-objective prompt optimization expands the Pareto front at modest per-objective cost. In our setting, however, the per-task degradation is substantially larger when coupling is high.

5 Analysis

The results in §4 establish *that* multi-task textual gradient optimization stagnates; this section investigates *why*. We identify two separable failure modes: optimization-time *gradient dilution* (§5.1) and inference-time *instruction interference* (§5.2).

5.1 Gradient Specificity: The Dilution Cliff

To measure gradient quality directly, we evaluate each textual gradient for *task-focus*: the degree to which its improvement suggestions target a single evaluation criterion rather than offering generic advice. An LLM evaluator (Claude Sonnet 4.6, Anthropic (2026)) rates each gradient on a 1–10 scale. A score of 10 means the gradient addresses exactly one task's rubric; a score of 1 means it's so generic it could apply to any task (see Appendix G for the evaluation prompt). We evaluate all gradients from steps 1–12 across all five modes and three seeds. This diagnostic tests the cross-task analogue of Chu et al.'s rule-dilution hypothesis. Concretely, we ask: does combining multiple evaluation criteria in a single gradient call dilute task-specific signal?

We observe a sharp transition (Figure 3). Per-task modes (Single, SSS, SSC), where each gradient LLM call processes exactly one task, achieve a mean specificity of 9.0 ($\pm$0.3; Appendix Table 4, top rows). All-task modes (SCC, CCC), where the gradient LLM processes all four tasks in a single call, drop to 3.7 ($\pm$0.5; Appendix Table 4, bottom rows). That's a 59% reduction with zero overlap between the two groups.

We also measure *feedback adherence* (using the same Claude Sonnet 4.6 evaluator): the degree to which the optimizer LLM's proposed prompt edit follows the gradient's suggestions. Adherence is uniformly high across all modes (7.8–8.8 on the same 10-point scale; Table 3 in Appendix C). These measurements indicate that the bottleneck is gradient quality rather than optimizer compliance: the optimizer incorporates the gradient's suggestions even when those suggestions are generic rather than task-specific.

This gradient dilution extends Chu et al.'s within-task finding to the cross-task setting. Chu et al. (2026) show that aggregating heterogeneous error modes in a single optimization step degrades rubric accuracy within educational grading; we observe an analogous effect when multiple task gradients are combined in a single gradient call, degrading per-task optimization signal. A per-task breakdown (Appendix Table 4) reveals that consistency is most diluted (specificity 2.4 in CCC), while coherence

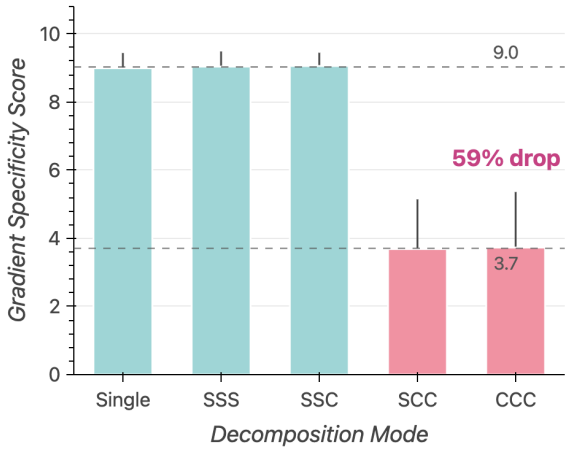

Figure 3: Gradient specificity (1–10 scale, higher is more task-focused) by decomposition mode. Per-task modes (Single, SSS, SSC) produce highly specific gradients (mean 9.0). All-task modes (SCC, CCC) drop to 3.7, a 59% reduction. Error bars: ± 1 std across seeds and steps.

retains moderate focus (5.1). This likely occurs because coherence rubrics share surface-level vocabulary with the generic "writing quality" feedback the gradient LLM defaults to under multi-task load.

To confirm the cliff is structural when moving from per-task to all-task gradients, we change the gradient LLM backbone to DeepSeek v4 Pro, and observe the same drop from SSC to SCC specificity (Appendix D).

5.2 Inference-Time Instruction Interference

Gradient dilution explains why SCC and CCC fail to optimize effectively, but it doesn't explain why SSS and SSC also stagnate. These per-task modes produce highly specific gradients (9.0) and high-adherence prompt updates (8.8), yet still fail to improve over the initialization. To diagnose this, we design an oracle cherry-pick experiment that isolates the *inference-time* component of multi-task failure. For each task, we select the single best instruction across all single-task optimization runs (the instruction with the highest held-out Spearman ρ for that task), then combine these four oracle-optimal instructions into one multi-task prompt and evaluate on the full test set.

Oracle-optimal instructions *degrade* when combined (Table 2). The best combination (`val=mae`, selected by OB1 metric) achieves 0.232 average Spearman, which is 0.053 below the initial generic baseline of 0.285. The worst case (`val=none`, selected by Spearman; Table 5 in Appendix F) drops to 0.120, with fluency collapsing to near-zero correlation. These instructions each individually outperform the baseline on their respective tasks. Combining them produces performance strictly worse than the generic initialization.

Method	fl	rel	coh	con	avg
Initial (generic)	.366	.208	.308	.256	.285
Single composited	.350	.168	.338	.363	**.305**
Cherry-pick (Spear.)	.303	.257	.215	.105	.220
Cherry-pick (OB1)	.322	.186	.225	.195	.232

Table 2: Oracle cherry-pick experiment (w. MAE validation). For each task, the single-task instruction with the highest test-set metric is selected and combined into one multi-task prompt. Despite oracle selection, combined instructions degrade performance below the initial generic baseline, demonstrating inference-time instruction interference.

The primary mechanism appears to be instruction-length asymmetry. Optimization produces over-specified rubrics for some tasks (the fluency instruction expands to $\sim$800 words with detailed scoring anchors) while leaving others under-specified (the relevance instruction remains $\sim$4 words). When packed into a single prompt, verbose instructions receive disproportionate attention relative to brief ones at inference time.

This finding strengthens a result from Shen et al. (2026), who observe that naive rubric construction degrades GPT-4o preference-judgment accuracy by 13 points on JudgeBench. The degradation we observe is larger (-0.053 to -0.163 Spearman) and occurs with oracle-selected rather than naively constructed instructions. Shen et al.'s result shows that *bad* rubrics hurt. Ours shows that individually *good* rubrics can hurt when combined. This implies that instruction interference isn't resolvable by improving per-task optimization alone.

6 Conclusion

Multi-criteria textual gradient optimization for LLM judges exhibits two bottlenecks that our decomposition study and process-level diagnostics expose. These bottlenecks are systematic: they correspond to distinct pipeline stages and affect different decomposition modes: (i) Gradient dilution operates at optimization time. When the gradient LLM must reconcile feedback from multiple criteria in a single call, its suggestions lose task-specificity (59% drop; Appendix Table 4). The optimizer propagates the low-specificity signal through to the final prompt; (ii) Instruction interference op-

erates at inference time. Independently optimized per-task instructions degrade when combined into one prompt (Table 2) because instruction-length asymmetry causes verbose rubrics to receive disproportionate attention relative to brief ones.

For practitioners customizing judges to domain-specific criteria (Kim et al., 2024), these results indicate that multi-criteria prompt optimization via textual gradients requires architectural changes before it can work reliably.

Addressing either failure mode alone isn't sufficient. Per-task decomposition at inference time (separate judge calls per criterion) eliminates interference but multiplies cost. Conflict-aware gradient resolution adapted from numerical multi-task learning (Yu et al., 2020; Liu et al., 2021) could address dilution if textual gradients can be meaningfully embedded and projected. Our diagnostics (gradient specificity and feedback adherence) provide the measurement tools to evaluate either approach.

7 Future Work

Our findings open several concrete directions to broader research on customized LLM evaluation.

Synthetic task generation for aligned criteria. A new direction is to *synthesize the criteria themselves* rather than treat them as fixed. Retrieval-augmented (Divekar and Durrett, 2024) and correlated decoding-time synthesis (Kowshik et al., 2024) can produce sets of criteria that are semantically diverse yet complementary in coverage. If the synthesized tasks are mutually aligned by construction, the combined gradient call should suffer less semantic drift, raising gradient compliance and potentially mitigating both failure modes at their source.

Statistically reliable LLM diagnostics with PPI. Our diagnostics are themselves LLM-judged, introducing an evaluator-bias confound. Prediction-Powered Inference (Angelopoulos et al., 2023) combines a small human-judged set with a large LLM-judged set to produce provably unbiased estimates; the hierarchical PPI extension of Divekar and Majumder (2026) is directly applicable here, since annotations are per-gradient but the quantity of interest is the per-mode mean. This would allow us to report the specificity gap as a confidence interval and scale diagnostics to hundreds of runs without scaling human annotation linearly.

Mitigations. Our diagnostics motivate concrete mitigations. For *gradient dilution*, a specificity-aware router could fallback to per-task gradient LLM when a multi-task LLM specificity drops below threshold, capturing CCC's hypervolume without losing task-focus. For *instruction interference*, we propose two avenues: (i) next-token attention masking for per-criterion output generation (eliminating interference as no cost), and (ii) length-aware instruction synthesis that normalizes rubric length during optimization so no single criterion dominates the attention budget.

Limitations

Our experiments are scoped to SUMMEVAL as it provides expert human annotations on four clearly separable criteria. Other benchmarks would validate our identified failure modes: BRIGHTER (Muhammad et al., 2025) tests whether dilution scales with task count and crosses language boundaries; ASAP++ (Mathias and Bhattacharyya, 2018) (per-trait essay-grading rubrics) tests whether the SCC to CCC gradient specificity cliff transfers beyond summarization; EMSCAD (Vidros et al., 2017), GitBugs (Patil et al., 2026), test heterogeneous classification criteria rather than ordinal quality scales.

Other prompt optimization paradigms (OPRO, GEPA, evolutionary methods, population-based approaches) may exhibit different multi-task dynamics, though our diagnostics are algorithm-agnostic and can be applied to any textual gradient approach.

Our sample size ($N = 3$ runs) limits statistical power, and we restrict our claims to effect sizes that are robust at this sample size (e.g., the 59% specificity drop and -0.053 cherry-pick degradation). Finally, gradient specificity and feedback adherence are scored by an LLM evaluator, which introduces a potential confound; we report the evaluation prompts in Appendix G for reproducibility.

Ethics Statement

Optimized judge prompts inherit biases present in the underlying LLMs and in the human annotations used for evaluation; practitioners should audit optimized prompts before deploying them in sensitive evaluation contexts (e.g., content moderation or hiring). Our code and diagnostics will be released under an open-source license to support reproducibility.

References

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Daniel Klein, Matei Zaharia, and Omar Khattab. 2026. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*.

Anastasios N. Angelopoulos, Stephen Bates, Clara Fannjiang, Michael I. Jordan, and Tijana Zrnic. 2023. Prediction-powered inference. *Science*, 382(6671):669–674.

Anthropic. 2026. Introducing claude sonnet 4.6. Anthropic Blog.

Jill Baumann and Oliver Kramer. 2024. Evolutionary multi-objective optimization of large language model prompts for balancing sentiments. In *Applications of Evolutionary Computation (EvoApplications)*, pages 212–224. Springer.

Yucheng Chu, Hang Li, Kaiqi Yang, Yasemin Copur-Gencturk, Joseph Krajcik, Namsoo Shin, and Jiliang Tang. 2026. Confusion-aware rubric optimization for LLM-based automated grading. *arXiv preprint arXiv:2603.00451*.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Abhishek Divekar and Greg Durrett. 2024. Synthesizrr: Generating diverse datasets with retrieval augmentation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 19200–19227. Association for Computational Linguistics.

Abhishek Divekar and Anirban Majumder. 2026. PRECISE: Reducing the bias of LLM evaluations using prediction-powered ranking estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence (IAAI Track)*, volume 40, pages 39929–39938.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*.

Yichen Han, Bojun Liu, Zhengpeng Zhou, Guanyu Liu, Zeng Zhang, Yang Yang, Wenli Wang, Isaac Shi, Yunyan, Lewei He, and Tianyu Shi. 2025. MAPGD: Multi-agent prompt gradient descent for collaborative prompt optimization. In *NeurIPS 2025 Workshop on Scaling Environments for Agents*.

Yasaman Jafari, Dheeraj Mekala, Rose Yu, and Taylor Berg-Kirkpatrick. 2024. MORL-prompt: An empirical analysis of multi-objective reinforcement learning for discrete prompt optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9878–9889, Miami, Florida, USA. Association for Computational Linguistics.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353.

Suhas S. Kowshik, Abhishek Divekar, and Vijit Malik. 2024. CorrSynth: A correlated sampling method for diverse dataset generation from LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 16076–16095. Association for Computational Linguistics.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, pages 18878–18890.

Sandeep Mathias and Pushpak Bhattacharyya. 2018. ASAP++: enriching the ASAP automated essay grading dataset with essay attribute scores. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Daniel Melcer, Qi Chen, Wen-Hao Chiang, Shweta Garg, Pranav Garg, and Christian Bock. 2025. Textual gradients are a flawed metaphor for automatic prompt optimization. *arXiv preprint arXiv:2512.13598*.

Yarik Menchaca Resendiz and Roman Klinger. 2025. MOPO: Multi-objective prompt optimization for affective text generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5588–5606, Abu Dhabi, UAE. Association for Computational Linguistics.

Shamsuddeen Hassan Muhammad, Nedjma Ousidhoum, Idris Abdulmumin, Jan Philip Wahle, Terry Ruas, Meriem Beloucif, Christine de Kock, Nirmal Surange, Daniela Teodorescu, Ibrahim Said Ahmad, David Ifeoluwa Adelani, Alham Fikri Aji, Felermino D. M. A. Ali, Ilseyar Alimova, Vladimir Araujo, Nikolay Babakov, Naomi Baes, Ana-Maria Bucur,

Andiswa Bukula, and 29 others. 2025. BRIGHTER: BRIdging the gap in human-annotated textual emotion recognition datasets for 28 languages. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8895–8916, Vienna, Austria. Association for Computational Linguistics.

Avinash Patil, Siru Tao, and Aryan Jadon. 2026. Gitbugs: Bug reports for duplicate detection, retrieval augmented generation, triage, and more. *Preprint*, arXiv:2504.09651.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Prith Sharma and Austin Z. Henley. 2026. Modular prompt optimization: Optimizing structured prompts with section-local textual gradients. *arXiv preprint arXiv:2601.04055*.

William F. Shen, Xinchi Qiu, Chenxi Whitehouse, Lisa Alazraki, Shashwat Goel, Francesco Barbieri, Timon Willi, Akhil Mathur, and Ilias Leontiadis. 2026. Rethinking rubric generation for improving LLM judge and reward modeling for open-ended tasks. *arXiv preprint arXiv:2602.05125*.

Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. 2025. Unleashing the potential of large language models as prompt optimizers: Analogical analysis with gradient-based model optimizers. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence*.

Sokratis Vidros, Constantinos Kolias, Georgios Kambourakis, and Leman Akoglu. 2017. Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. *Future Internet*, 9(1):6.

Peifeng Wang, Austin Xu, Yilun Zhou, Caiming Xiong, and Shafiq Joty. 2025. Direct judgement preference optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1979–2009.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2026. J1: Incentivizing thinking in LLM-as-a-judge via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Bo Yang, Lanfei Feng, Yunkui Chen, Yu Zhang, Xiao Xu, and Shijian Li. 2026. FairJudge: An adaptive, debiased, and consistent LLM-as-a-judge. *arXiv preprint arXiv:2602.06625*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pages 12028–12068.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc.

Mert Yüksekgönül, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616.

Guang Zhao, Byung-Jun Yoon, Gilchan Park, Shantenu Jha, Shinjae Yoo, and Xiaoning Qian. 2025. Pareto prompt optimization. In *The Thirteenth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

A Multi-Criteria Evaluation Prompt Template (SUMMEVAL)

Our multi-criteria prompt has two parts: a **frozen skeleton** (evaluation directive, output format) and **mutable per-task instructions** (the "Instructions" section). Only the per-task instructions are updated during optimization; the skeleton remains fixed throughout all 12 steps.

Initial prompt template

```
You are a careful, calibrated evaluator.
Your goal is to produce an accurate evaluation
by following the Instructions below.

## Task
Evaluate the Summary given the Source Text
using the Instructions below.
1. Consider every strength and flaw you find
   when making your evaluation.
2. Based on the number and severity of the
   strengths and flaws, assign a value.
Use the Instructions below to perform your
evaluation. Output a JSON with the requested
scores. Do NOT include reasoning or
explanations.

## Output format (follow this EXACTLY):
{
  "fluency": 1|2|3|4|5,
  "relevance": 1|2|3|4|5,
  "coherence": 1|2|3|4|5,
  "consistency": 1|2|3|4|5
}

## Instructions:
- fluency: Rate from 1 to 5.
- relevance: Rate from 1 to 5.
- coherence: Rate from 1 to 5.
- consistency: Rate from 1 to 5.
```

After optimization, the per-task instructions expand substantially. Below is an example from a SCC `val=mae` run after step 1, showing how the optimizer rewrites the generic one-line instructions into detailed rubrics:

Example optimized instructions (step 1)

```
## Instructions:
- fluency: Fluency: Rate 1-5 based on
  grammatical correctness, sentence flow,
  and ease of parsing. A score of 5 reflects
  natural, effortless readability even with
  minor syntactic informality. Do not penalize
  for stylistic choices that do not impede
  comprehension.
- relevance: Relevance: Rate 1-5 based on how
  completely the summary addresses the core
  informational needs implied by the source
  text. A score of 5 means all key facts,
  outcomes, and implications are included with
  no deviation. Relevance penalizes both
  omission (missing key facts) and commission
  (adding unsupported claims).
- coherence: Coherence: Rate 1-5 based on
  logical flow, temporal sequence, and
  referential stability. A score of 5 means
  events unfold in a cause-effect or
  chronological order, with clear antecedents
  for all pronouns and noun phrases.
- consistency: Consistency: Rate 1-5 based on
  whether the summary contradicts any claim
  in the source text. A score of 5 requires
  no factual or inferential contradictions.
  A single major inconsistency reduces the
  score to 1.
```

B HVI Trajectories

The hypervolume indicator trajectories are shown as black diamonds (right y-axis) in Figure 2 in the main paper. CCC achieves the highest HVI growth under both validation settings (2.900 for `val=mae`, 2.983 for `val=none`; Table 1).

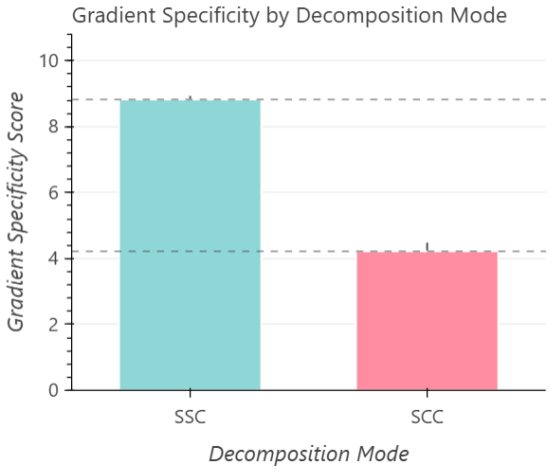

Figure 4: Gradient specificity for SSC vs. SCC after swapping the gradient LLM to DeepSeek-V4-Pro while keeping the other stages on Qwen. Bar values: SSC 8.82 ± 0.10, SCC 4.22 ± 0.26. Overall drop: 52%.

C Feedback Adherence Details

Feedback adherence measures whether the optimizer LLM incorporates the textual gradient into its instruction edits. Across all modes and validation settings, adherence is uniformly high (7.8–8.8 on a 10-point scale), confirming that the optimizer faithfully implements whatever gradient it receives. The bottleneck is gradient quality, not optimizer compliance.

Mode	val=mae	val=none
Single	8.70 ± 0.47	8.53 ± 0.60
SSS	8.84 ± 0.44	8.72 ± 0.54
SSC	7.94 ± 0.80	7.83 ± 0.78
SCC	8.08 ± 1.03	7.76 ± 1.49
CCC	7.90 ± 1.62	8.01 ± 1.28

Table 3: Feedback adherence scores (1–10 scale, mean $\pm$ std over all tasks, runs, and steps). All modes achieve high adherence, ruling out optimizer non-compliance as an explanation for multi-task optimization failure.

D Gradient Specificity Under Gradient-Model Swap

We swap the gradient LLM backbone to `DeepSeek-V4-Pro` (DeepSeek-AI, 2026) while keeping the task, loss, and optimizer LLMs in the Qwen family and holding the rest of the pipeline fixed (8 optimization steps). As shown in Figure 4, SSC remains high while SCC drops sharply (**52%**), indicating that gradient dilution persists under a cross-family gradient model.

E Per-Task Gradient Specificity

Table 4 shows gradient specificity broken down by task for the combined-gradient modes (SCC, CCC). Averaged across SCC and CCC, consistency is the most diluted dimension (specificity 2.5), while coherence retains moderate focus (5.0). This suggests that the gradient LLM's attention is not uniformly distributed across tasks when processing them simultaneously.

Mode	fl	rel	coh	con
Per-task gradient modes (specificity ≈ 9.0):				
Single	8.9	8.9	9.1	9.0
SSS	9.0	9.0	9.1	9.0
SSC	9.0	9.1	9.1	9.0
All-task gradient modes (specificity ≈ 3.7):				
SCC	3.0	4.3	4.8	2.6
CCC	3.2	4.3	5.1	2.4

Table 4: Gradient specificity by task (mean over both val settings, all runs and steps). Per-task modes maintain uniformly high specificity. In all-task modes, consistency is most diluted and coherence retains moderate focus.

F Cherry-Pick Experiment: All Variants

Table 5 shows all six cherry-pick variants (3 selection metrics $\times$ 2 validation settings). All variants degrade below the initial generic baseline, confirming that inference-time instruction interference is robust across metric choices.

Val	Selection	avg ρ
Initial generic baseline		0.283
mae	Spearman	0.220
mae	OB1	0.232
mae	MAE	0.231
none	Spearman	0.120
none	OB1	0.200
none	MAE	0.172

Table 5: All cherry-pick variants. Every oracle combination degrades below the initial generic baseline ($\bar{\rho} = 0.283$). The worst case (val=none, Spearman selection) drops to 0.120 with fluency at zero correlation.

G Diagnostic Evaluation Prompts

Below we include the prompts used for task-level diagnostics. Both diagnostics are evaluated post-hoc by Claude Sonnet 4.6.

G.1 Gradient Specificity Evaluator

The following prompt rates each textual gradient on a 1–10 scale for task-specificity:

Gradient specificity evaluator prompt

```
You are measuring how focused a piece of
textual feedback (called a "gradient") is on
a specific evaluation task, versus being
diluted with generic advice or advice that
belongs to other tasks.

The target task is "{task}". The possible
tasks are: fluency, relevance, coherence,
consistency.

Rate from 1 to 10 how well this gradient
focuses on the "{task}" task.
1 = completely generic or mostly addresses
    other tasks.
10 = laser-focused on "{task}" with concrete,
     task-specific fixes.

## The Gradient
{gradient_text}

Respond with ONLY a single integer from 1
to 10. No explanation.
```

G.2 Feedback Adherence Evaluator

The following prompt measures how well the optimizer incorporated each gradient into its instruction edit:

Feedback adherence evaluator prompt

```
You are evaluating whether revisions to
task-specific instructions correctly
addressed the gradient (suggested changes).

The instructions are for an LLM judge that
evaluates the "{task}" task. The Gradient
may contain suggestions about multiple
tasks; consider only suggestions pertaining
to "{task}".

Rate from 1 to 10 how well the New
Instructions address the Gradient for
"{task}".
1 = completely ignores/contradicts.
10 = precisely addresses every point while
     preserving what worked.

## Old Instructions
{old_instruction}

## New Instructions
{new_instruction}

## Gradient (Suggested Changes)
{gradient_text}

Respond with ONLY a single integer from 1
to 10. No explanation.
```