INFORMATION-THEORETIC CORRECTIVE EXAMPLE SELECTION (IT-CES): CALIBRATED IN-CONTEXT EXAMPLE SELECTION FOR AUTOMATED CODE CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

In corrective in-context learning for automated code repair, the effectiveness of Large Language Model (LLM) self-correction depends substantially on the quality of retrieved demonstrations. Standard similarity-based selection methods, such as Top-K retrieval, often select examples independently and ignore correlations among candidates, leading to redundant contexts with diminishing marginal utility. We propose Information-Theoretic Corrective Example Selection (IT-CES), a theory-motivated framework for selecting a compact set of corrective demonstrations that jointly cover complementary repair patterns for the current failure. IT-CES represents each correction instance using a diagnostic tuple that combines the problem, the buggy solution, and the observed failure signal, and applies a calibrated log-determinant selection objective to choose examples that are both relevant to the current failure and complementary to one another. The calibration step mitigates the concentration of dense-retrieval similarity scores, improving the contrast of the diagnostic kernel used by the log-determinant objective. Across three code-repair benchmarks, IT-CES consistently improves Pass@1 repair accuracy over representative similarity-based and diversity-aware retrieval methods across the evaluated context budgets, generators, and dense retrievers. We further report step-by-step component ablations, paired exact statistical tests, intrinsic retrieval diagnostics, and multi-round dynamic repair evaluations. These results support the effectiveness of calibrated diagnostic example selection within the tested repair settings. The implementation is uploaded as supplementary material.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated exceptional capabilities in complex tasks such as code generation and logical reasoning (Chen et al., 2021; Wei et al., 2022). Despite these advances, they remain susceptible to hallucinations and subtle logical errors, which significantly limits their reliability in autonomous workflows (Ji et al., 2023). Self-correction—the ability of a model to identify and repair its own mistakes—has emerged as an important challenge in contemporary LLM research (Madaan et al., 2023; Shinn et al., 2023), particularly for code correction tasks. Since updating model parameters is often computationally prohibitive in real-time settings and task-specific fine-tuning data is expensive to obtain, prompt optimization has become the primary mechanism for steering model behavior at inference time (Zhou et al., 2022). By precluding direct parameter updates, prompt optimization shifts the burden of behavioral adaptation onto the input context itself, providing a flexible and lightweight means of enabling self-correction.

Within the landscape of prompt-based correction, existing strategies range from generic instructional prompts to structured reasoning frameworks. However, approaches that rely primarily on the model's intrinsic knowledge often prove insufficient for debugging. While zero-shot prompting and Chain-of-Thought (CoT) reasoning can resolve simple ambiguities, their effectiveness frequently plateaus on complex logical errors, where the model lacks a clear internal representation of the underlying failure mode (Olausson et al., 2024). Recent studies indicate that Corrective In-Context Learning (ICL)—which augments the prompt with paired examples of buggy code and correspond-

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.

ing fixes—offers a promising alternative for inference-time code repair (Rubin et al., 2022; Liu et al., 2022). By grounding inference in concrete repair demonstrations, Corrective ICL enables the model to emulate the repair logic exhibited by retrieved examples, rather than relying solely on its parametric knowledge.

Limitations: Diagnostic Redundancy in Corrective ICL. Crucially, the effectiveness of Corrective ICL hinges on the information content of the selected demonstrations. Most existing approaches rely on dense retrieval methods that select the Top-K examples most similar to the user's query. However, this similarity-maximization paradigm can suffer from a key limitation in corrective ICL: diagnostic redundancy. Dense embeddings are designed to capture high-level semantic similarity, but they often fail to resolve fine-grained diagnostic cues that distinguish distinct failure mechanisms (Xu et al., 2025). Consequently, selecting demonstrations based solely on embedding proximity can produce a diagnostically redundant subset. Retrieved examples may share superficial characteristics—such as overlapping variable names or control structures—while encoding near-duplicate repair patterns, rather than covering complementary failure mechanisms. This diagnostic redundancy leads to inefficient use of the prompt budget: additional examples provide diminishing marginal utility and may introduce weakly aligned cues that distract the model. As a result, the context window may be populated with repetitive instances instead of complementary diagnostic evidence useful for effective correction (Zhao et al., 2025).

Challenges and Our Proposal. Addressing this mismatch is non-trivial. A natural extension is to incorporate diversity-aware retrieval methods such as Maximal Marginal Relevance (MMR) (Carbonell & Goldstein, 1998), which balances query similarity against redundancy. In corrective ICL, however, the relevant diversity is over corrective mechanisms conditioned on the observed failure. This motivates an uncertainty-reduction view informed by analyses of ICL as implicit Bayesian inference (Xie et al., 2021) and classical information theory (Shannon, 1948). We operationalize it through a log-determinant surrogate over diagnostic representations; the information-theoretic language motivates the design rather than asserting formal equivalence to repair utility.

Our Approach: Information-Theoretic Corrective Example Selection (IT-CES). Motivated by this diagnostic coverage view, we propose *IT-CES*, a calibrated in-context example selection framework for automated code correction (illustrated in Figure 1). The workflow operates as a post-hoc refinement mechanism initiated by a verified failure. First, treating the diagnostic tuple of *(Problem, Buggy Code, Failure Signal)* as a composite query, we use a dense retriever to aggregate a broad *candidate pool* of historical repair scenarios that are semantically relevant to the current failure. Second, to distill this candidate pool into a compact corrective prompt, IT-CES uses a calibrated quality–diversity log-determinant surrogate inspired by Determinantal Point Processes (DPPs) (Kulesza et al., 2012). The calibration step addresses the concentration of dense-retrieval similarities, while the determinant scores a set jointly so that relevance and complementarity are considered together. The information-theoretic and Gaussian-process views serve as geometric motivation only; the empirical contribution is the calibration of diagnostic quality and candidate support for corrective ICL.

Contributions. (1) We formulate corrective example selection as a diagnostic, set-level selection problem conditioned on the tuple $(x, y_{\mathrm{buggy}}, \delta)$, capturing the problem, the erroneous solution, and the observed failure signal. (2) We introduce *IT-CES*, a training-free quality–diversity selector that calibrates diagnostic relevance and candidate support before log-determinant subset selection. (3) We evaluate IT-CES on three code-repair benchmarks and provide a unified full-baseline comparison, component ablations, paired exact tests, and protocol diagnostics. The results show higher observed Pass@1 under the reported budgets, with gains that vary by dataset and prompt budget.

2 Related Work

Our research intersects self-correction in Large Language Models (LLMs), example selection for In-Context Learning (ICL), and recent attempts to connect ICL theory with practical correction. We briefly review these lines of work and position IT-CES as a diagnostic example-selection method for corrective code repair.

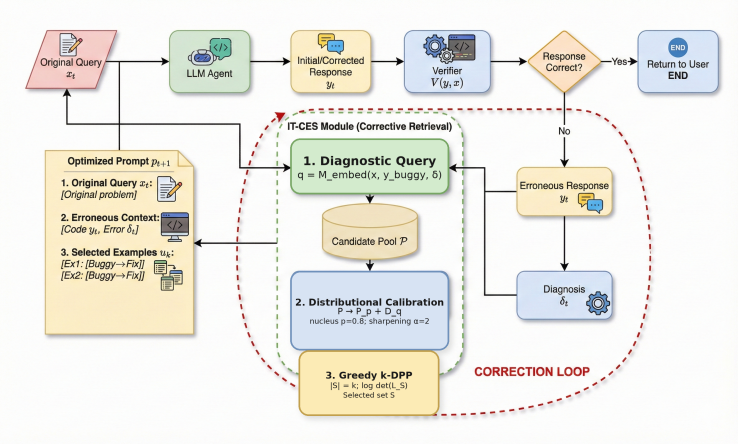

Figure 1: *IT-CES workflow.* A verifier diagnoses a failed response, the diagnostic tuple retrieves a candidate pool, and calibrated log-determinant selection builds a corrective prompt.

Self-Correction in Large Language Models. LLM self-correction methods can broadly be divided into training-time approaches that internalize correction ability into model parameters and post-hoc approaches that refine an initial output at inference time (Pan et al., 2023). Training-time methods such as SCoRe improve correction ability through reinforcement learning but require additional training and model-generated data (Kumar et al., 2024). Post-hoc methods, including iterative refinement and tool-assisted correction, are more flexible but often depend on high-quality external feedback, such as verifier signals or code execution traces (Madaan et al., 2023; Kamoi et al., 2024). Without such guidance, intrinsic self-correction can be unreliable: models may fail to recognize errors in their own outputs even when they can identify similar errors in user-provided text (Huang et al., 2023; Tsui, 2025). These observations motivate inference-time correction methods that augment the prompt with external diagnostic evidence, including retrieved repair demonstrations.

In-Context Learning and Example Selection. ICL example selection is often viewed as reducing uncertainty over a latent task distribution (Jeon et al., 2024). In practice, most methods rely on similarity-based retrieval, such as k-NN or dense Top-K selection, which selects demonstrations with the highest embedding similarity to the query (Guo et al., 2003). Diversity-aware reranking methods such as Maximal Marginal Relevance (MMR) improve on pure relevance ranking by balancing query similarity against redundancy among selected examples (Carbonell & Goldstein, 1998). More structured set-level objectives, including Determinantal Point Processes (DPPs) (Kulesza et al., 2012) and recent log-determinant ICL selectors such as KITE (Singh et al., 2025), further formalize selection as a relevance–diversity trade-off. In corrective code repair, however, semantic relevance and generic diversity are not always aligned with diagnostic usefulness: examples should cover complementary repair patterns conditioned on the buggy solution and failure signal. This motivates IT-CES to represent each correction instance as a diagnostic tuple $(x, y_{\text{buggy}}, \delta)$ and to calibrate the resulting diagnostic kernel before log-determinant selection.

The Intersection: Bridging Theory and Practice. Recent theory connects self-correction and ICL by showing how contextual examples containing feedback-augmented tuples, such as (x, y, r), can implicitly improve a model's generation policy (Wang et al., 2024). At the same time, practical Corrective ICL (CICL) studies show that simply presenting $(y_{\text{buggy}}, y_{\text{fixed}})$ pairs can be insufficient or even harmful when the retrieved examples introduce ambiguity rather than guidance (Sanz-Guerrero & von der Wense, 2025). This suggests a gap between the theoretical promise of correction through contextual examples and the practical challenge of selecting examples that are diagnostically useful. IT-CES addresses this gap by moving from static input-only retrieval toward diagnostic retrieval conditioned on the problem, buggy solution, and observed failure signal.

3 METHODOLOGY

3.1 PROBLEM FORMULATION: ERROR CORRECTION VIA PROMPT OPTIMIZATION

We study corrective in-context learning for automated code repair. Given a programming query x, an LLM first produces an erroneous response y_{buggy}. A diagnoser D evaluates this response, identifies a failure, and produces a structured diagnosis δ (e.g., error type, traceback, compiler message, or failed test case). Our objective is to construct a corrective prompt p that guides the LLM to generate a revised response y with a higher likelihood of passing the verifier.

To formalize this, let $\mathcal{P} = \{(x^{(i)}, y^{(i)}_{\text{buggy}}, \delta^{(i)}, y^{(i)}_{\text{fixed}})\}$ be a candidate pool of historical repair records, where each record contains a problem, a buggy solution, an observed failure signal, and a corresponding fix. We select a fixed-budget subset of demonstrations $S \subseteq \mathcal{P}$ with $|S| = k$. The corrective prompt p is formed by augmenting the original query and diagnostic context with this selected subset:

$$p = \mathcal{T}(x, y_{\text{buggy}}, \delta, S),$$

where $\mathcal{T}$ is a prompt template function. Let $\Pi(y \mid p)$ denote the LLM's generation policy and $V(y, x)$ be a verifier function returning a correctness score (e.g., pass/fail or a scalar in $[0, 1]$). Ideally, we would like to select the utility-optimal subset S^*_{utility} that maximizes the expected verification score of the revised generation:

$$S^*_{\text{utility}} = \arg \max_{S \subseteq \mathcal{P},\, |S|=k} \mathbb{E}_{y \sim \Pi(\cdot \mid \mathcal{T}(x, y_{\text{buggy}}, \delta, S))}[\, V(y, x)\,]. \tag{1}$$

Challenges. Solving Eq. 1 directly poses three practical challenges. First, *Combinatorial Complexity*: The search space of possible subsets $\binom{|\mathcal{P}|}{k}$ grows exponentially with the pool size, making exhaustive enumeration impossible. Second, *Inference Intractability*: Evaluating the objective function requires executing the LLM and the verifier for every candidate subset. Since LLM inference is computationally expensive and the verifier (e.g., running test cases) is time-consuming, computing the true expected verification score $\mathbb{E}[V]$ during inference is prohibitively costly. Third, *Semantic-Diagnostic Mismatch*: As discussed in Section 1, retrieval based only on semantic similarity to x can be insufficient because it does not directly condition on the diagnostic context $(y_{\text{buggy}}, \delta)$. It may retrieve examples that are topically relevant but diagnostically redundant or weakly aligned with the current failure.

These challenges imply that we cannot directly optimize Eq. 1 at inference time. Instead, we seek an efficient *surrogate objective* that is intended to (1) reflect factors relevant to the verification score, (2) be computationally cheap to evaluate, and (3) incorporate diagnostic information from the current failure. This motivates the information-theoretic selection objective introduced in the following section.

3.2 MOTIVATION: FROM UTILITY TO GEOMETRIC COVERAGE

Motivation: Uncertainty Reduction as a Proxy. Let $c = (x, y_{\text{buggy}}, \delta)$ denote the diagnostic context for the current correction instance, and let q denote its linearized diagnostic query representation. We write the utility objective as

$$f(S) \equiv \mathbb{E}_{y \sim \Pi(\cdot \mid \mathcal{T}(c, S))}[V(y, x)]\,.$$

Directly optimizing $f(S)$ is intractable because it requires repeated LLM inference and verification. To obtain a tractable selection rule, we use an information-theoretic uncertainty-reduction motivation. Our working hypothesis is that useful demonstrations reduce uncertainty about the latent failure mechanism underlying the current failure. A GP-inspired model for local dependencies in the diagnostic representation space then motivates a closed-form geometric surrogate; it is not treated as a literal model of the LLM. This motivates our framework design:

$$\max_S f(S) \xrightarrow{\text{surrogate}} \max_S \text{IG}(\theta; S \mid q) \xrightarrow{\text{geometric surrogate}} \max_S \log\det(L_S + \epsilon I)\,.$$

We now detail the motivation for each step.

Step 1: An uncertainty-reduction interpretation. Following prior analyses of In-Context Learning (ICL) as *Implicit Bayesian Inference* (Xie et al., 2021; Garg et al., 2022), we use a Bayesian lens to motivate the selection objective. This is an interpretive bridge for designing a retrieval surrogate, not a derivation or bound relating the surrogate to repair accuracy. Let $\theta \in \Theta$ denote the latent failure mechanism relevant to the current correction attempt. Conditioning on the diagnostic context c and demonstrations S induces an implicit posterior over this mechanism, denoted $p(\theta \mid c, S)$. Under a binary pass/fail verifier, the utility objective can be viewed as the posterior predictive probability of generating an accepted solution:

$$f(S) = \Pr(y \in \mathcal{Y}^*(x) \mid c, S)\,,$$

where $\mathcal{Y}^*(x)$ denotes the set of accepted solutions for problem x. A detailed posterior-predictive derivation is provided in Appendix A.1.

The key implication is that the selected subset S affects the utility through the posterior over the latent failure mechanism. However, the mechanism-conditioned success probability is not available during retrieval, because evaluating it would again require LLM generation and verification. We therefore replace direct utility maximization with a utility-agnostic uncertainty-reduction proxy: selecting demonstrations that provide information about the latent failure mechanism conditioned on the diagnostic query. Formally, we use conditional Information Gain,

$$\mathrm{IG}(\theta; S \mid q) = H(\theta \mid q) - H(\theta \mid q, S),$$

where q is the linearized diagnostic query representation of c. This gives the surrogate relation

$$\max_S f(S) \xrightarrow{\text{surrogate}} \max_S \mathrm{IG}(\theta; S \mid q).$$

This reframes the intractable black-box utility objective as an experimental-design-inspired selection problem: choosing S to reduce uncertainty about the latent failure mechanism relevant to the current diagnostic query.

Step 2: A geometric surrogate inspired by GP design. The information-gain objective $\mathrm{IG}(\theta; S \mid q)$ is still intractable because computing mutual information in high-dimensional function spaces is difficult (Belghazi et al., 2018).

To obtain a closed-form surrogate, we use a GP-inspired modeling assumption for local dependencies in the diagnostic representation space. We model a latent diagnostic response function f_θ with a *Gaussian Process (GP)* prior (Williams & Rasmussen, 2006). Each demonstration is treated as a noisy observation of this latent function in the diagnostic representation space, with observation noise $\epsilon \sim \mathcal{N}(0, \lambda)$. The GP view is used as a modeling surrogate rather than a claim of exact model specification.

Under this GP surrogate, the Information Gain of observing a subset S is typically formulated as (Srinivas et al., 2010):

$$\mathrm{IG}(S) = \frac{1}{2} \log\det\big(I + \lambda^{-1} K_S^{\mathrm{GP}}\big)\,.$$

However, standard IG reduces *global* uncertainty. In correction, our goal is to maximize information specifically relevant to the current failure. We therefore retain the query as a quality signal and use a quality–diversity kernel for the implemented geometric surrogate. Let $d_i \in \mathcal{P}_p$ denote a candidate record after support selection, let s_i be its raw query–candidate cosine similarity, and let C_{ij} be candidate–candidate cosine similarity. Here $\mathcal{P}_p \subseteq \mathcal{P}$ is the calibrated support defined in Section 3.3.

$$q_i = \left(\max\left\{\frac{s_i}{\max_{j\in\mathcal{P}_p} s_j}, \epsilon\right\}\right)^{\alpha}, \qquad G_{ij} = \frac{C_{ij}+1}{2}, \qquad L = D_q G D_q + \epsilon I, \tag{2}$$

where $D_q = \mathrm{diag}(q_i : i \in \mathcal{P}_p)$, s_i is the raw query–candidate cosine similarity, $\alpha = 2.0$, and $\epsilon = 10^{-8}$ is the numerical floor used in the implementation. With a fixed prompt budget, the implemented subset objective is the fixed-cardinality k-DPP form $\hat{S} = \arg\max_{S\subseteq\mathcal{P}_p, |S|=k} \log\det(L_S)$. Since $L_{ii} = q_i^2 + \epsilon \le 1 + \epsilon$, this fixed-cardinality constraint ensures that every budget position is filled; each greedy step uses the positive conditional Schur complement of the current selected block. The query therefore affects the quality weights, while the Gram geometry rewards complementary

candidates. This construction is the operational surrogate used by IT-CES; it is not an assertion that information gain equals correction accuracy.

The GP view supplies a motivation for using log-determinant volume to represent local coverage, rather than a literal probabilistic model of LLM behavior. The candidate Gram matrix G is PSD when it is formed from the shifted construction above, and congruence by D_q preserves PSD. Under this PSD construction, adding ϵI gives a positive-definite matrix and the log-determinant objective has the usual diminishing-returns interpretation. The nonlinear calibration and candidate filtering steps are practical pre-conditioning operations; no unconditional PSD, submodularity, or $(1 - 1/e)$ guarantee is claimed for every transformed matrix.

Summary of the Motivation. The above argument motivates corrective demonstration selection as uncertainty reduction over latent failure mechanisms and uses a GP-inspired approximation to obtain a tractable quality–diversity log-determinant objective. The resulting construction is a theory-motivated selection surrogate: it encourages diagnostic coverage and complementarity while remaining computationally feasible. No formal equivalence, upper or lower bound, or monotone correspondence between information gain and correction accuracy is claimed.

3.3 THE IT-CES ALGORITHM

IT-CES is an algorithmic instantiation of the surrogate objective in Eq. 2. It consists of two steps: calibrating query-side quality scores to address score concentration in dense code retrieval, and greedily optimizing the resulting quality–diversity objective.

Step 1: Kernel Construction via Distributional Calibration. The surrogate objective in Eq. 2 requires a diagnostic representation that captures the correction state. Unlike standard retrieval settings that often use only the problem statement x, we construct a composite *diagnostic query* q by linearizing the diagnostic tuple $(x, y_{\text{buggy}}, \delta)$. Each candidate record d_i is represented by its corresponding diagnostic context. We encode these records into dense vectors so that the quality score reflects similarity in the diagnostic repair space. The frozen Jina encoder provides useful semantic embeddings, but raw cosine similarities can concentrate in a narrow high-similarity regime (e.g., $[0.85, 1.0]$). In this regime, the quality weights provide limited contrast between redundant examples and complementary repair patterns.

To address this score concentration, we introduce a *Distributional Calibration* mechanism as a local pre-conditioning step. We convert query–candidate relevance scores into a local temperature-scaled softmax within the candidate pool:

$$z_i = \frac{s_i - \mu_{\mathcal{P}}}{\sigma_{\mathcal{P}}}, \qquad \pi_i = \frac{\exp(z_i)}{\sum_{\ell \in \mathcal{P}} \exp(z_\ell)} = \frac{\exp(s_i/\sigma_{\mathcal{P}})}{\sum_{\ell \in \mathcal{P}} \exp(s_\ell/\sigma_{\mathcal{P}})}. \tag{3}$$

Here $\mu_{\mathcal{P}}$ and $\sigma_{\mathcal{P}} > 0$ are the empirical mean and standard deviation of query–candidate cosine scores in the current top-200 pool. The additive mean cancels in the softmax, while the pool dispersion acts as a local temperature: the entropy of π adapts to score concentration. The probabilities define support only and do not replace the raw cosine scores in Eq. 2. We sort candidates by decreasing π_i and let $\mathcal{P}_p$ be the smallest prefix whose cumulative mass is at least $p = 0.8$. We enlarge this prefix until $|\mathcal{P}_p| \geq k$ and retain the highest-similarity candidate as an anchor. The kernel and greedy selection are then computed only on $\mathcal{P}_p$ (with the registered candidate limit applied after support formation). Thus nucleus filtering changes the candidate support, while q_i uses max-normalized raw cosine within that support.

The candidate–candidate matrix is then formed from the normalized embedding matrix X as $C = XX^\top$, with $G = (C + \mathbf{1}\mathbf{1}^\top)/2$. Thus G is PSD, and the quality–diversity kernel $L = D_q G D_q + \epsilon I$ is positive definite for $\epsilon > 0$. We use the resulting kernel in conditional Schur-complement updates. The calibration parameters $\alpha = 2.0$ and $p = 0.8$ are fixed before evaluation. The numerical floor is $\epsilon = 10^{-8}$; the implementation does not introduce additional temperature or lower-CDF constants.

The calibration is intentionally applied to the query-side quality scores rather than to arbitrary pairwise entries. This distinction keeps the candidate geometry interpretable: the shifted Gram matrix represents the relationships among demonstrations, while D_q controls their relevance to the current failure. For numerical safeguards, the resulting matrix can additionally be projected onto the PSD

cone by clipping negative eigenvalues. The evaluated construction uses the explicit Gram form above and a small ϵ for conditioning; the jitter is a numerical stabilizer and is not treated as a proof that an arbitrary indefinite matrix is PSD.

Step 2: Solving the Objective via Submodular Optimization. With the quality–diversity kernel constructed, we solve the subset selection problem greedily. Although Eq. 2 avoids repeated LLM inference, exact maximization remains combinatorial. Log-determinant objectives are closely connected to submodular information objectives in experimental design and subset selection (Fujishige, 2005; Krause et al., 2008). The relevant intuition is *diminishing returns*: once a repair pattern is represented in the prompt, another similar example contributes less than one that adds complementary diagnostic information. A formal statement under these PSD assumptions is provided in Appendix A.2.

For this PSD construction, the classical result of Nemhauser et al. (1978) gives the usual $(1 - 1/e)$ approximation for the monotone information objective $\log\det(I + \lambda^{-1} L_S)$ under a cardinality constraint. This conditional guarantee is distinct from the fixed-cardinality objective implemented here: directly maximizing $\log\det(L_S)$ with $|S| = k$ is the standard k-DPP greedy volume-maximization view (Kulesza et al., 2012). Neither statement extends automatically to arbitrary calibrated matrices. Starting from $S_0 = \emptyset$, IT-CES selects the candidate with the largest conditional Schur-complement gain:

$$i_r^* = \arg \max_{j \in \mathcal{P}_p \setminus S_{r-1}} \log \max \left(L_{jj} - L_{jS_{r-1}} L_{S_{r-1}S_{r-1}}^{-1} L_{S_{r-1}j}, \epsilon \right).$$

After selecting i_r^*, we update $S_r = S_{r-1} \cup \{i_r^*\}$ and continue for $r = 1, \ldots, k$. Cholesky rank-one updates avoid repeated full matrix inversion while preserving the same selected-set objective.

The marginal gain measures how much candidate j expands the diagnostic volume beyond the examples already selected. A candidate that is highly relevant but nearly redundant with S_{r-1} has a small Schur-complement gain, whereas a relevant candidate that contributes a complementary repair direction retains a larger gain. This gives the algorithm an explicit relevance–complementarity trade-off while avoiding repeated LLM generation and verification during retrieval.

Computational Efficiency. Naively applying kernel methods to the full candidate pool can incur cubic costs in $|\mathcal{P}|$. IT-CES avoids full eigendecomposition or full-kernel inversion by using greedy maximization with Cholesky rank-one updates. For a fixed prompt budget k, sorting the broad pool and forming the calibrated support cost $O(|\mathcal{P}| \log |\mathcal{P}|)$, while greedy selection on the support costs $O(|\mathcal{P}_p|\, k^3)$. The total is $O(|\mathcal{P}| \log |\mathcal{P}| + |\mathcal{P}_p|\, k^3)$; in our protocol $|\mathcal{P}| = 200$ and the mean support size is 41.8. A detailed complexity analysis is provided in Appendix A.3.

4 Experiment

4.1 Experimental Setup

Datasets and Task Construction. We evaluate IT-CES on three benchmark subsets derived from distinct sources in the broader *CodeContests* aggregation (Li et al., 2022). Each repair instance pairs an erroneous submission (syntactically valid but failing hidden tests) with its accepted solution. The diagnostic signal δ is obtained by executing the buggy program and recording the traceback or first failed test. HackerEarth contains practical data-structure, array, and simulation tasks (HackerEarth Inc., 2012); AIZU emphasizes graph, number-theoretic, and dynamic-programming reasoning under strict I/O constraints (University of Aizu, 2004); and Desc2Code contains longer descriptions with a larger gap between semantic similarity and the repair operation (Husain et al., 2019). This three-way construction keeps practical, algorithmically rigorous, and verbose-description regimes visible in one fixed protocol. Dataset counts and filtering details are given in Appendix B.

Corpus Construction and Evaluation Protocol. For every source, we index an execution-verified retrieval corpus totaling $36{,}500$ historical repair candidates (12,400 HackerEarth, 8,900 AIZU, and 15,200 Desc2Code; Table 4), strictly disjoint from evaluation problems at the problem-ID level. We form a corrective Hard Set by excluding instances solved by an x-only zero-shot generation from the Qwen3-Coder-Next backbone. This filtering yields 230 unique zero-shot failures: a disjoint validation cohort ($N_{\text{val}} = 60$, 20 per source) and the unified evaluation cohort

Table 1: *Primary comparison on the unified $N = 170$ Hard Set.* Each entry is passes (Pass@1%). The retrieval corpus contains 36,500 execution-verified candidates; Δ is the percentage-point gain of IT-CES over the strongest retrieval baseline in the row. Symmetric validation tuning gives MMR $38/170$ and KITE $39/170$ at $k = 5$ (Section 4.3).

Dataset	Budget	Top-K	MMR	BM25	KITE	IT-CES	Δ
Overall (170)	$k = 3$	35 (20.6)	41 (24.1)	37 (21.8)	40 (23.5)	**49 (28.8)**	**+4.7 pp**
	$k = 5$	32 (18.8)	36 (21.2)	35 (20.6)	38 (22.4)	**48 (28.2)**	**+5.9 pp**
HackerEarth (53)	$k = 3$	11 (20.8)	14 (26.4)	12 (22.6)	13 (24.5)	**15 (28.3)**	+1.9 pp
	$k = 5$	11 (20.8)	12 (22.6)	11 (20.8)	12 (22.6)	**14 (26.4)**	+3.8 pp
AIZU (45)	$k = 3$	8 (17.8)	10 (22.2)	8 (17.8)	10 (22.2)	**14 (31.1)**	+8.9 pp
	$k = 5$	7 (15.6)	12 (26.7)	8 (17.8)	11 (24.4)	**14 (31.1)**	+4.4 pp
Desc2Code (72)	$k = 3$	16 (22.2)	17 (23.6)	17 (23.6)	17 (23.6)	**20 (27.8)**	+4.2 pp
	$k = 5$	14 (19.4)	12 (16.7)	16 (22.2)	15 (20.8)	**20 (27.8)**	+5.6 pp

($N = 170$ **Hard Set**: 53 HackerEarth, 45 AIZU, 72 Desc2Code). For each retained problem, y_{buggy} is the fixed erroneous held-out submission paired with its accepted solution for verification; the x-only filtering step determines membership in the Hard Set and does not replace that submission with a newly generated program. We additionally evaluate diagnostic self-repair at $k = 0$, which supplies $(x, y_{\text{buggy}}, \delta)$ but no retrieved demonstrations; this is a reference condition and is not an ICL selector. The fixed $N = 170$ cohort is comparable in scale to complete canonical code-generation test suites such as HumanEval (164 problems) and the official CodeContests test split (165 problems) (Chen et al., 2021; Li et al., 2022); unlike a mixed-difficulty test split, every retained query is an active repair case. Keeping every selector paired on exactly the same problems enables exact paired McNemar tests to use each problem as its own control. This is a fixed, cross-source comparison cohort whose conclusions are scoped to the tested protocol.

For each query, dense retrieval first forms a broad top-200 candidate pool $\mathcal{P}$ from the corresponding source corpus. Top-K, MMR, KITE, uncalibrated DPP, and IT-CES all select their k demonstrations from this same pool; IT-CES subsequently constructs its calibrated support $\mathcal{P}_p \subseteq \mathcal{P}$. BM25 is evaluated on the source-local lexical index described in Appendix C.

Baselines. Top-K selects the k highest query–candidate cosine scores; MMR balances relevance and redundancy with $\gamma = 0.5$ (Carbonell & Goldstein, 1998). BM25 is a lexical comparator and KITE a generic log-determinant selector on the same eligible view; implementations and validation sweeps are in Appendix C.

Implementation and Metrics. The unified protocol uses Qwen3-Coder-Next with the frozen jina-code-embeddings-1.5b retriever. IT-CES uses $(\alpha, p) = (2.0, 0.8)$ selected on a disjoint validation cohort of $N_{\text{val}} = 60$ queries (20 per source). We report first-attempt Pass@1 correction accuracy. The fixed Analogical CoT prompt compares the target diagnostic tuple with retrieved buggy–fixed transformations, then asks for an analysis, repair plan, and complete program; this reduces gains from copying repeated snippets alone. Transfer configurations are reported separately in Section 4.3; the full prompt template appears in Appendix C.5. The average five-reference prompt uses approximately 2,400 tokens.

4.2 Main Experiment

Table 1 reports the complete unified panel. Every method receives the same held-out query tuple, generator, checker, and reference budget; the dense selectors share the same top-200 candidate pool, while BM25 uses the source-local lexical index described above. The table focuses on retrieval-based ICL selectors on the fixed *Hard Set*; the hard-set construction and its zero-shot filtering criterion are defined in Setup. We emphasize that every reported ICL selector is evaluated on these same 170 zero-shot-failure queries, so the denominator is fixed before retrieval and is not supplemented by an easier $k = 0$ pool.

IT-CES has the highest observed correction rate at both reported budgets: $49/170$ (28.8%) at $k = 3$ and $48/170$ (28.2%) at $k = 5$. Exact paired tests and Holm corrections are reported in Appendix D.1; source-level rows show the positive direction across all three regimes.

Table 2: *Transferability checks across generator–encoder configurations on the unified $N = 170$ Hard Set ($k = 3$).* Each entry reports passes (Pass@1%). Δ is the percentage-point gain of IT-CES over the best-performing baseline in each row.

Generator	Encoder	Top-K	MMR	BM25	KITE	**IT-CES**	Δ
Qwen3-Coder-Next (main)	jina-code-1.5b	35 (20.6)	41 (24.1)	37 (21.8)	40 (23.5)	**49 (28.8)**	**+4.7 pp**
Qwen2.5-Coder-14B	CodeBERT-base	33 (19.4)	34 (20.0)	31 (18.2)	35 (20.6)	**42 (24.7)**	**+4.1 pp**
Qwen3-Coder-Plus	CodeBERT-base	67 (39.4)	68 (40.0)	64 (37.6)	70 (41.2)	**78 (45.9)**	**+4.7 pp**
DeepSeek-Coder-33B	UniXcoder-base	38 (22.4)	40 (23.5)	36 (21.2)	41 (24.1)	**47 (27.6)**	**+3.5 pp**

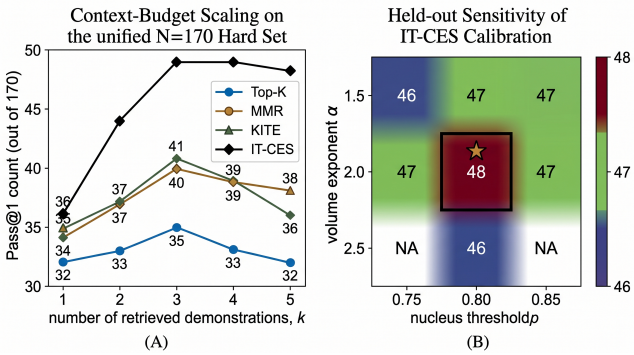

Figure 2: **Budget scaling and calibration sensitivity.** The left panel reports Pass@1 counts for $k = 1$–5 on the unified $N = 170$ Hard Set; the right panel reports the evaluated (α, p) grid, with the selected setting highlighted.

4.3 Transferability, Mechanism, and Sensitivity Analysis

Table 2 reports three alternative generator–encoder pairs. IT-CES leads all rows by +4.1–+4.7 percentage points across generator and encoder changes.

Figure 2 shows the held-out choice $(\alpha, p) = (2.0, 0.8)$ and the rise from 36/170 at $k = 1$ to 49/170 at $k = 3$, remaining 48–49/170 through $k = 5$. C1 raises Top-K from 32/170 to 34/170 and C2 raises MMR from 36/170 to 42/170; full IT-CES reaches 48/170, six queries above C2. BM25 and KITE remain complete baselines because they do not define component ablations.

Table 3: *Component progression and calibrated controls* on the unified $N = 170$ Hard Set at $k = 5$. C1/C2 apply the IT-CES calibration to Top-K/MMR; Red(S) and Noise diagnostics are reported in Appendix C.3.

Stage	Query view	Selector / calibration	Passes	Rate (%)
M1	Code-only	Top-K	26/170	15.3
M2	Diagnostic	Top-K	32/170	18.8
C1	Diagnostic	Calibrated Top-K	34/170	20.0
M3	Diagnostic	MMR ($\gamma = .5$)	36/170	21.2
C2	Diagnostic	Calibrated MMR	42/170	24.7
M4	Diagnostic	Uncalibrated DPP	36/170	21.2
M5	Diagnostic	DPP + nucleus	40/170	23.5
M6	Diagnostic	DPP + sharpening	43/170	25.3
M7	Diagnostic	Full IT-CES	**48/170**	**28.2**

5 Conclusion

We introduce IT-CES, a theory-motivated corrective ICL selector using a diagnostic tuple and calibrated log-determinant objective. On the unified $N = 170$ Hard Set it reaches $49/170$ at $k = 3$ and $48/170$ at $k = 5$, versus $31/170$ for diagnostic self-repair (paired $p = 0.000121$ and 0.000221). Ablations, full-cohort transfer, and dynamic repair support the result within the tested settings; broader encoders remain future work.

Our conclusions are scoped to the tested generator–encoder configurations and fixed Hard Set. Broader models and datasets remain future work.

AI USE STATEMENT

In this work, we used generative AI tools for drafting preliminary sections of the manuscript. We have not used generative AI tools for research ideation or execution, generating synthetic datasets, or proving mathematical claims. Additionally, we used generative AI tools for aiding and polishing English writing, as well as literature retrieval and discovery. We have reviewed all AI-assisted work: all drafted paragraphs were heavily revised, fact-checked, and reorganized by the authors, and all retrieved references were manually verified against primary literature. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 335–336, 1998.

Mark Chen, Jerry Tworek, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, John Schulman, Ilya Sutskever, Wojciech Zaremba, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Satoru Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022.

Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 986–996. Springer, 2003.

HackerEarth Inc. HackerEarth. `https://www.hackerearth.com/`, 2012. Accessed: 2026-01-28.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019.

Hong Jun Jeon, Jason D Lee, Qi Lei, and Benjamin Van Roy. An information-theoretic analysis of in-context learning. In *International Conference on Machine Learning*, 2024.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

Jina AI. jina-code-embeddings-1.5b. Hugging Face model card, 2024. URL `https://huggingface.co/jinaai/jina-code-embeddings-1.5b`.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(2):235–284, 2008.

Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, 2024.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd workshop on knowledge extraction and integration for deep learning architectures*, pp. 100–114, 2022.

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

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Theo X Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Demystifying gpt self-repair for code generation. In *International Conference on Learning Representations*, 2024.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*, 2023.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2655–2671, 2022.

Mario Sanz-Guerrero and Katharina von der Wense. Corrective in-context learning: Evaluating self-correction in large language models. *arXiv preprint arXiv:2503.16022*, 2025.

Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Vaibhav Singh, Soumya Suvra Ghosal, Kapu Nirmal Joshua, Soumyabrata Pal, and Sayak Ray Chowdhury. Kite: Kernelized and information theoretic exemplars for in-context learning. *arXiv preprint arXiv:2509.15676*, 2025.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

Ken Tsui. Self-correction bench: Uncovering and addressing the self-correction blind spot in large language models. *arXiv preprint arXiv:2507.02778*, 2025.

University of Aizu. Aizu Online Judge. `https://onlinejudge.u-aizu.ac.jp/`, 2004. Accessed: 2026-01-28.

Yifei Wang, Yuyang Wu, Zeming Wei, Stefanie Jegelka, and Yisen Wang. A theoretical understanding of self-correction through in-context alignment. *Advances in Neural Information Processing Systems*, 37:89869–89912, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2021.

Liyan Xu, Zhenlin Su, Mo Yu, Jiangnan Li, Fandong Meng, and Jie Zhou. Dense retrievers can fail on simple queries: Revealing the granularity dilemma of embeddings. *arXiv preprint arXiv:2506.08592*, 2025.

Yicong Zhao, Shisong Chen, Jiacheng Zhang, and Zhixu Li. Recode: Improving llm-based code repair with fine-grained retrieval-augmented generation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 4368–4378, 2025.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*, 2022.

A Additional Methodological Details

A.1 Posterior-Predictive View of the Utility Surrogate

Under the Bayesian lens used in Section 3, the LLM's prediction can be viewed through a posterior predictive distribution:

$$p(y \mid c, S) = \int_{\Theta} p(y \mid c, \theta)\, p(\theta \mid c, S)\, d\theta.$$

The corresponding utility objective is

$$f(S) = \mathbb{E}_{y \sim p(\cdot \mid c, S)}[V(y, x)].$$

For the pass/fail evaluation used in code repair, we model the verifier as an indicator over the set of accepted solutions $\mathcal{Y}^*(x)$:

$$V(y, x) = \mathbb{I}[y \in \mathcal{Y}^*(x)]\,.$$

Thus,

$$f(S) = \mathbb{E}_{y \sim p(\cdot \mid c, S)}[\mathbb{I}[y \in \mathcal{Y}^*(x)]] = \Pr(y \in \mathcal{Y}^*(x) \mid c, S)\,.$$

This probability can be written as an expectation over the latent failure mechanism:

$$\Pr(y \in \mathcal{Y}^*(x) \mid c, S) = \mathbb{E}_{\theta \sim p(\cdot \mid c, S)}[\Pr(y \in \mathcal{Y}^*(x) \mid c, \theta)]\,.$$

This expression shows that the subset S enters the Bayesian view by changing the posterior $p(\theta \mid c, S)$. Since the mechanism-conditioned success term is not available during retrieval, IT-CES uses conditional information gain as a tractable uncertainty-reduction surrogate rather than directly optimizing this utility.

A.2 Submodular Greedy Justification

A set function G is submodular if it satisfies diminishing returns: for any $A \subseteq B \subset \mathcal{P}$ and any $s \notin B$,

$$G(A \cup \{s\}) - G(A) \geq G(B \cup \{s\}) - G(B).$$

For the monotone information objective $\log\det(I + \lambda^{-1} L_S)$ under a cardinality constraint, the greedy algorithm achieves a $(1 - 1/e)$ approximation guarantee (Nemhauser et al., 1978). Noisy log-determinant information objectives with PSD kernels are a standard example in GP experimental design and subset selection (Krause et al., 2008). The fixed-cardinality objective used by IT-CES, $\log\det(L_S)$ with $|S| = k$, is instead the usual k-DPP greedy volume-maximization view (Kulesza et al., 2012). In IT-CES, the calibrated kernel is used as a practical diagnostic surrogate; these conditional interpretations do not provide an unconditional guarantee for every transformed matrix.

A.3 Computational Complexity Analysis

A critical consideration for any framework relying on Gaussian Processes (GP) or Determinantal Point Processes (DPP) is computational feasibility. Standard GP regression or exact DPP sampling typically entails a prohibitive $O(|\mathcal{P}|^3)$ computational cost due to the inversion or eigendecomposition of the full kernel matrix $L_{\mathcal{P}} \in \mathbb{R}^{|\mathcal{P}|\times|\mathcal{P}|}$, where $|\mathcal{P}|$ denotes the broad dense-retrieval pool.

IT-CES avoids applying cubic-time operations to the full candidate pool. We use the GP view as a modeling surrogate to motivate the log-determinant objective, rather than performing posterior inference over the entire pool. Consequently, the computational complexity is governed by the greedy maximization algorithm, which we analyze below.

The greedy algorithm performs exactly k iterations to select a subset of size k, where k is the prompt budget. After calibration, the scan is restricted to the nucleus support $\mathcal{P}_p \subseteq \mathcal{P}$. In greedy iteration $r \in [1, k]$, the algorithm scans all remaining candidates $j \in \mathcal{P}_p \setminus S_{r-1}$ to identify the element maximizing the marginal gain:

$$\Delta F(j) = \log\det(L_{S_{r-1}\cup\{j\}}) - \log\det(L_{S_{r-1}}).$$

A naive evaluation of this determinant for every candidate would incur a cost of $O(r^3)$ per candidate, leading to a total complexity of $O(|\mathcal{P}_p| \cdot k^4)$.

To avoid recomputing the determinant from scratch, IT-CES employs rank-one Cholesky updates. We maintain the Cholesky decomposition R_{r-1} of the current kernel submatrix, such that $L_{S_{r-1}} = R_{r-1}R_{r-1}^\top$. When evaluating a new candidate j, the new kernel matrix is a block matrix:

$$L_{S_{r-1}\cup\{j\}} = \begin{pmatrix} L_{S_{r-1}} & \mathbf{v} \\ \mathbf{v}^\top & c \end{pmatrix},$$

where $\mathbf{v}$ is the vector of similarities between candidate j and the selected set S_{r-1}, and $c = L_{jj}$. The new Cholesky factor can be computed by appending a row to R_{r-1}, reducing the per-candidate update cost to $O(r^2)$.

Consequently, the cost per greedy iteration r involves scanning $|\mathcal{P}_p|$ candidates with an $O(r^2)$ update for each. Summing over k iterations gives:

$$\text{Total Complexity} \approx \sum_{r=1}^{k} |\mathcal{P}_p| \cdot O(r^2) = O(|\mathcal{P}_p| \cdot k^3).$$

The initial sort and support construction add $O(|\mathcal{P}|\log|\mathcal{P}|)$, so the total is $O(|\mathcal{P}|\log|\mathcal{P}| + |\mathcal{P}_p| \cdot k^3)$. Since the budget k is small in our experiments (typically $k \leq 5$), the selection cost scales effectively linearly with the broad pool and avoids the cubic bottleneck of full-kernel methods.

B Dataset Details and Curation Protocol

In this section, we provide additional details on the three benchmarks, the hard-subset curation process, and the retrieval corpus construction.

B.1 Benchmark Characteristics

Our evaluation leverages three distinct sources to test different benchmark settings. Table 4 summarizes the key statistics for each subset.

HackerEarth (Industrial Complexity). Sourced from an online programming assessment platform, this dataset contains practical programming tasks.

- *Content:* Problems cover a wide range of practical scenarios, including data structure manipulation, array operations, and simulation tasks.
- *Style:* Problem descriptions often include narrative elements (storytelling) mixed with technical specifications, introducing a moderate level of ambiguity typical in software engineering requirements.
- *Difficulty:* Balanced. The test set includes both standard implementation tasks and complex optimization problems.

AIZU (Algorithmic Rigor). Sourced from the Aizu Online Judge (AOJ), widely used for computer science education in Japan.

- *Content:* Heavily skewed towards fundamental computer science domains: Graph Theory (shortest path, spanning trees), Number Theory (primality test, GCD), and Dynamic Programming.
- *Style:* Highly formalized. Descriptions are concise, mathematically precise, and enforce strict Input/Output formatting.
- *Challenge:* The primary difficulty lies in *logical correctness* and *edge-case handling* rather than semantic understanding. This subset tests the model's ability to reason precisely when the requirements are clear.

Desc2Code (Linguistic Complexity). A subset included to evaluate retrieval under longer, natural-language-heavy problem descriptions.

- *Content:* Diverse programming tasks but selected based on description length and complexity.
- *Style:* Verbose and natural-language-heavy. The average description length is significantly higher than AIZU (see Table 4).
- *Challenge:* The "Semantic Gap." The retriever must distill the core intent from a noisy, long-context description. This mimics the "long-tail" queries in semantic code search where Jina diagnostic embeddings often degrade.

Table 4: *Dataset Statistics.* We report the number of unique problems in the Hard Subset, the size of the retrieval corpus constructed for each benchmark, and the average token length of problem descriptions.

Benchmark	*Hard Subset Size* (Unique Problems)	*Retrieval Corpus Size* (Candidates)	*Avg. Desc. Length* (Words)	*Avg. Code Length* (LOC)
HackerEarth	53	12,400	245.2	38.4
AIZU	45	8,900	112.5	45.1
Desc2Code	72	15,200	480.6	41.2
Total	*170*	*36,500*	-	-

B.2 Curation of the "Hard Subset"

To ensure our metrics reflect *corrective* capability rather than rote memory, we employed a strict zero-shot filtering pipeline:

1. *Initial Pool:* We started with the raw test splits from the respective sources.
2. *Zero-Shot Evaluation:* We gave only the problem description x to `Qwen3-Coder-Next`, without a buggy program, failure signal, or retrieved context.
3. *Filtering:* Any problem correctly solved (Pass@1) in this x-only generation setting was *excluded*.
4. *Result:* The remaining 170 unique problems constitute our "Hard Subset." We additionally evaluate diagnostic self-repair at $k = 0$, supplying $(x, y_{\text{buggy}}, \delta)$ and execution feedback without retrieved demonstrations; it is a reference condition rather than a retrieval selector.

Note on Statistical Testing: In the main paper, paired exact McNemar tests are computed separately for each context budget. For each budget, the paired test is performed over the 170 unique hard-subset problems, avoiding the pooling of $k = 3$ and $k = 5$ observations.

B.3 Retrieval Corpus Construction

For each benchmark, we constructed a dense retrieval corpus to study diagnostic redundancy and low-confidence distractors in dense retrieval:

- *Source:* We aggregated all submissions from the training splits of the corresponding platforms.
- *Clustering Strategy:* Unlike standard RAG setups that often assume a single "Gold Document," real-world repositories contain multiple valid solutions. We preserved this distribution. *Strict problem-level isolation:* the Retrieval Corpus and the 170 held-out test problems have zero overlap, $\mathcal{P}_{\text{corpus}} \cap \mathcal{P}_{\text{test}} = \emptyset$, at the problem-ID level. Within the corpus, historical buggy-to-fixed trajectories form clusters around recurring repair mechanisms; distractor clusters come from unrelated problems. No accepted solution for a held-out test problem is placed in the retrieval corpus.
- *Density:* The average corpus size exceeds 10,000 candidates per domain (Table 4), providing a large search space in which dense retrievers can return high-similarity false positives, making calibration relevant.
- *Diagnostic Pre-computation*: To align the retrieval space with the query tuple (x, y, δ), we performed an offline execution of all historical buggy submissions in the corpus against their respective test suites. For *Runtime Errors*, we extracted the standard error traceback (e.g., `IndexError`). For *Logical Errors* (Wrong Answer), we captured the input/output pair of the first failed test case. These signals form the canonical definition of δ used in our experiments.
- *Candidate Linearization*: Unlike standard retrieval that indexes only code, each candidate record $d_i \in \mathcal{P}$ is indexed as a *linearized diagnostic tuple* $D_i = (x^{(i)}, y^{(i)}_{buggy}, \delta^{(i)})$. This allows the dense retriever to compute similarity based on the full diagnostic context rather than just surface-level code syntax.

The candidate–candidate matrix uses the same observable diagnostic tuple and excludes $y^{(i)}_{\text{fixed}}$. This avoids query-time leakage because a target fix is unavailable when retrieval is performed; consequently, the selected-set diversity is an observable diagnostic-space proxy. Similarity over the repair transformation itself is outside the present evaluation.

C Implementation Details

In this section, we provide the specific configurations for the backbone models, the exact prompt templates used for our Analogical Chain-of-Thought (CoT) strategy, and the hyperparameter search space to ensure full reproducibility of our results.

C.1 Model Configurations

Generator (Qwen3-Coder-Next). We utilize `Qwen3-Coder-Next` as the backbone solver. All generations are performed using the `vLLM` library for efficient inference. To reduce sampling-related variability and ensure fair comparison, we use *Greedy Decoding*:

- *Temperature:* 0.0
- *Top-P:* 1.0 (Disabled)
- *Max New Tokens:* 2,048 (Sufficient for full solution generation)
- *Repetition Penalty:* 1.0 (No penalty, trusting the model's internal logits)

Retriever (jina-code-embeddings-1.5b). We use the frozen model `jinaai/jina-code-embeddings-1.5b` (Jina AI, 2024) at revision `39aeb4fb9b60f930934c78ae5d749a46287c248a`. The implementation in `scripts/code_embedding_encoder.py` uses last-token pooling, producing 1536-dimensional vectors, followed by L_2 normalization. The model contract allows up to 32768 tokens. The 512-token limit used in the Desc2Code diagnostic is an experimental structured allocation ($256/128/128$ for problem, buggy code, and failure signal), not an encoder architecture limit.

Transfer configuration details. The transfer table in Section 4.3 uses Qwen2.5-Coder-14B with CodeBERT-base, Qwen3-Coder-Plus with CodeBERT-base, and DeepSeek-Coder-33B with UniXcoder-base on the unified $N = 170$ Hard Set at $k = 3$. These rows use the fixed $(\alpha, p) = (2.0, 0.8)$ calibration, the same buggy–fixed repair transformation, analysis–plan–code prompt structure, and greedy decoding at temperature 0.0 within each targeted configuration. They are separate checks and are not pooled with the unified Qwen3-Coder-Next–Jina statistics.

Retrieval Hyperparameters. The unified protocol uses a registered broad candidate stage of 200 dense candidates where configured, followed by source-local selection. Top-K, MMR, and IT-CES operate on the same dense top-200 view; BM25 uses a source-local lexical index rather than reranking another method's selected examples. Each selector's eligible view is frozen before evaluation. BM25 is implemented with `rank_bm25.BM25Okapi`, using $k_1 = 1.5$ and $b = 0.75$ on the source-local lexical view. KITE follows the official Algorithm 1 formulation with a linear kernel, $\beta = 0.02$, and diversity weight $\lambda = 0.5$. Thus KITE is a generic log-determinant selector on the same eligible view, whereas IT-CES changes the query representation and applies the calibrated support and quality weighting. The nucleus support size is query dependent rather than a fixed hyperparameter: it is bounded below by k and above by the registered local pool size. Across the unified $N = 170$ protocol with $|\mathcal{P}| = 200$ and $p = 0.8$, the mean support size is 41.8 ± 14.9, the median is 39.0, the interquartile range is $[28, 53]$, and the observed range is $[12, 84]$. These values characterize the query-dependent support used by the selector.

Token Budget Analysis. Despite the inclusion of diagnostic tuples, the prompt length remains efficient. On average, a $k = 5$ shot prompt with CoT reasoning consumes approximately *2,400 tokens* (including the problem description and 5 reference pairs). Given Qwen3-Coder-Next's context window of 16k tokens, this occupies approximately 15% of the available budget, leaving sufficient room for generation under the model's context window.

Structured encoder allocation on Desc2Code. For the long-description subset, the 512-token encoder input is allocated deterministically as 256 tokens for the problem description, 128 for the buggy program, and 128 for the observed failure signal. Under the unified Qwen3-Coder-Next–jina-code-embeddings-1.5b protocol, this allocation gives 61.1% Hit@5 and 27.8% Pass@1 for IT-CES on Desc2Code, compared with 19.4% Pass@1 for Top-K, an 8.4-point gain. Hit@5 is the fraction of queries for which at least one selected candidate matches the pre-defined repair-pattern signature; it is reported only as a retrieval diagnostic. This positive result is reported as an input-budget diagnostic for the unified protocol.

C.2 Validation-set Selection and Baseline Sweeps

Hyperparameters were selected on a held-out validation cohort of 60 problems (20 from each source), with no problem-level overlap with the retrieval corpus or the 170-problem test set. The selected IT-CES setting is $(\alpha, p) = (2.0, 0.8)$, reaching $18/60$ on validation and $48/170$ at $k = 5$ on test. Nearby IT-CES settings reach $46/170$ for $(1.5, 0.75)$, $47/170$ for $(1.5, 0.80)$, $47/170$ for $(2.0, 0.75)$, $47/170$ for $(2.0, 0.85)$, and $46/170$ for $(2.5, 0.80)$. These values document local stability around the selected setting and are not used to claim universal hyperparameter optimality.

For protocol symmetry, the validation sweep also evaluated MMR over $\gamma \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and KITE over $\lambda \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The validation-selected settings were $\gamma = 0.7$ and $\lambda = 0.3$; their corresponding test outcomes were $38/170$ and $39/170$, respectively. IT-CES remains at $48/170$ under the same $k = 5$ budget.

C.3 Intrinsic Retrieval Diagnostics

For each corpus candidate, the execution-derived repair-pattern signature is obtained from its recorded buggy-to-fixed transition and verifier trace. Cov@5 counts distinct signatures among the five selected candidates, and Hit@5 is the fraction of queries for which at least one selected candidate matches the query signature. Noise is the fraction of selected candidates whose raw query similarity satisfies $s_i < \mu_{\mathcal{P}}$, where $\mu_{\mathcal{P}}$ is computed over that query's broad top-200 pool; this definition is independent of the nucleus support. The signatures are used only for offline diagnosis and do not enter selection. On the unified $N = 170$ cohort at $k = 5$, the full IT-CES selector has mean

pairwise selected-set cosine redundancy 0.812, covers 3.42 distinct signatures out of five selected references on average, reaches overall diagnostic $\mathrm{Hit@5} = 63.5\%$, and has Noise 2.4%. Its reranking latency is 3.4 ms/query. These diagnostics characterize the selected set and are not treated as independent proof of downstream accuracy.

Table 5: *Effective nucleus support cardinality* $|\mathcal{P}_p|$ at $p = 0.8$ ($|\mathcal{P}| = 200$).

Source	Mean ± SD	Median	IQR	Range	Pruned
HackerEarth	43.2 ± 14.6	41	[31,54]	[14,78]	78.4%
AIZU	36.5 ± 12.1	35	[26,46]	[12,65]	81.8%
Desc2Code	44.1 ± 16.2	42	[30,56]	[16,84]	78.0%
Overall	41.8 ± 14.9	39	[28,53]	[12,84]	79.1%

Table 6: *Intrinsic retrieval diagnostics* on the unified $N = 170$ cohort at $k = 5$.

Selector	Red(S) ↓	Cov@5 ↑	Hit@5 ↑	Noise ↓	Latency (ms) ↓
Diagnostic Top-K	.934	1.68	41.8%	8.2%	.9
Calibrated Top-K	.928	–	–	3.5%	–
MMR ($\gamma = .5$)	.805	2.45	49.4%	28.5%	1.8
Calibrated MMR	.818	–	–	3.8%	–
KITE ($\lambda = .5$)	.828	2.64	52.4%	19.8%	4.1
Uncalibrated DPP	.891	2.19	48.2%	22.1%	3.6
DPP + nucleus	.874	2.58	54.7%	3.1%	2.9
DPP + sharpening	.824	3.05	58.2%	11.4%	3.6
IT-CES	**.812**	**3.42**	**63.5%**	**2.4%**	**3.4**

C.4 Validated Context-budget Trace

Across context budgets $k \in \{1, 2, 3, 4, 5\}$, IT-CES achieves $36/170$, $44/170$, $49/170$, $49/170$, and $48/170$, respectively. The increase from one to three references is consistent with complementary examples contributing beyond a single nearest neighbour; the $k = 3$ and $k = 5$ values are the same primary-test entries reported in Table 1.

The complete budget trajectory is shown in Figure 2.

C.5 The Corrective Prompt Template

To ensure the model performs logic-based repair rather than rote copying, we employ an *Analogical Chain-of-Thought (CoT)* prompt. Crucially, unlike standard code generation templates, this prompt explicitly structures the input as a *Diagnostic Tuple* (x, y_{buggy}, δ), and presents retrieved references as *repair transformations* rather than static snippets.

The exact schema used in our experiments is shown below:

System Instruction

You are an expert software engineer specializing in debugging. You will be given a specific coding problem, a piece of buggy code, and a set of *Reference Repair Examples* retrieved from a database.

Input Context

RETRIEVED REPAIR PATTERNS: The following examples show how similar logical errors were fixed in the past.
<Reference_1>
Buggy Snippet:
{REF_BUGGY_CODE_1}
Fixed Snippet:
{REF_FIXED_CODE_1}
</Reference_1>
...
<Reference_k>
...
</Reference_k>
TARGET PROBLEM:
Problem Description:
{PROBLEM_DESCRIPTION}
Target Buggy Code:
{TARGET_BUGGY_CODE}
Diagnostic Info / Failure Signal:
{ERROR_MESSAGE_OR_TEST_FAILURE}

Task Instruction

Please fix the Buggy Code in the Target Problem by applying the repair logic found in the References.

1. *Analyze*: Briefly compare the `Target Buggy Code` with the `Reference Repair Patterns`. Identify the shared logical flaw.
2. *Plan*: Synthesize a step-by-step repair plan.
3. *Code*: Write the final complete solution in Python3.

Response:

C.6 Hyperparameter Search

For the IT-CES algorithm, we selected hyperparameters via a grid search on a held-out validation set separate from the test set.

- *Volume Exponent (α):* Search space $\{1.0, 2.0, 3.0\}$. We selected $\alpha = 2.0$ as it provided the best trade-off between score differentiation and numerical stability.
- *Nucleus Threshold (p):* Search space $\{0.5, 0.6, \ldots, 0.95\}$. We selected $p = 0.8$ for the dense jina-code-embeddings-1.5b space; lower values (e.g., $p = 0.5$) pruned the support too aggressively and removed useful complementary repair signals, whereas higher values (e.g., $p = 0.95$) retained too many low-confidence tail candidates.
- *Trade-off (γ) for MMR:* The main table uses 0.5; the symmetric validation sweep and selected $\gamma = 0.7$ test result are reported in Appendix C.2.

C.7 Compute Resources

All local inference experiments were run on a workstation with a single NVIDIA RTX 4090 GPU using vLLM. The main Qwen3-Coder-Next experiments used one 24GB GPU. A full evaluation pass over the 170 hard-subset problems required approximately 2–3 GPU-hours for each retrieval method and prompt budget. The full main-table evaluation over all methods and budgets therefore required roughly 12–18 GPU-hours of local inference, excluding preprocessing and retrieval-index construction. Retrieval embedding computation and IT-CES reranking were lightweight compared with LLM generation and were run on the same workstation.

D SUPPLEMENTARY RESULTS CONSISTENT WITH THE MAIN PROTOCOL

The primary supplementary results in this section use the unified Qwen3-Coder-Next generator and jina-code-embeddings-1.5b retriever, matching the main text. The transfer table in Section 4.3 records three separate generator–encoder checks; these targeted rows are not pooled with the unified protocol.

D.1 PAIRED STATISTICS FOR THE UNIFIED HARD SET

Table 7: *Exact paired McNemar statistics for the unified $N = 170$ Hard Set.*

k	Comparison	Wins	Losses	Ties (a/d)	p_{raw}	Holm mult.	Holm p
3	IT-CES vs. $k = 0$	20	2	29/119	.000121	–	–
5	IT-CES vs. $k = 0$	19	2	29/120	.000221	–	–
3	Top-K	17	3	32/118	.002577	$\times 4$	**.0103**
3	BM25	15	3	34/118	.007538	$\times 3$	**.0226**
3	KITE	11	2	38/119	.022461	$\times 2$	**.0449**
3	MMR	10	2	39/119	.038574	$\times 1$	**.0386**
5	Top-K	19	3	29/119	.000855	$\times 4$	**.0034**
5	BM25	16	3	32/119	.004425	$\times 3$	**.0133**
5	MMR	15	3	33/119	.007538	$\times 2$	**.0151**
5	KITE	12	2	36/120	.012939	$\times 1$	**.0129**

D.2 COMPLETE COMPONENT ABLATION

The unified ablation uses the same M1–M7 mapping as Table 3: M1 is code-only Top-K; M2 adds the diagnostic tuple; M3 replaces Top-K with MMR; M4 uses the uncalibrated DPP; M5 adds nucleus support; M6 adds quality sharpening; and M7 is the full IT-CES configuration. The corresponding downstream counts are reported once in the main table to avoid duplicating the same result panel in the appendix.

D.3 POSITIVE SELECTION TRACE

For the "1427_C" runtime failure, IT-CES selected 607_A, 183_B, 1203_F1, 1249_D2, and 30_C, after which the generated repair passed verification. The selected records span complementary repair transformations (indexing, control flow, state handling, and boundary checks), illustrating the set-level behavior targeted by the determinant objective.

D.4 MULTI-ROUND DYNAMIC REPAIR

The multi-round protocol keeps the same 170-problem Hard Set and verifier; counts are cumulative, with each problem counted at its first passing round. Static IT-CES reuses first-round references, whereas Dynamic Top-K and Dynamic IT-CES recompute retrieval from (x, y_t, δ_t) after each failed round.

Table 8: *Cumulative multi-round repair counts* on the unified Hard Set.

Strategy	Round 1	Round 2	Round 3
Zero-shot Reflexion	31/170	38/170	42/170
Static IT-CES	49/170	55/170	58/170
Dynamic Top-K	35/170	44/170	49/170
Dynamic IT-CES	**49/170**	**62/170**	**71/170**

The dataset-level Dynamic IT-CES trajectory is HackerEarth $15/53 \rightarrow 20/53 \rightarrow 23/53$, AIZU $14/45 \rightarrow 17/45 \rightarrow 19/45$, and Desc2Code $20/72 \rightarrow 25/72 \rightarrow 29/72$; the corresponding Dynamic Top-K counts are $11/53 \rightarrow 14/53 \rightarrow 16/53$, $8/45 \rightarrow 11/45 \rightarrow 13/45$, and $16/72 \rightarrow 19/72 \rightarrow 20/72$.