Text-Twin-Translation (T^3): A Full-Stack Machine Learning Framework for Functional Material-Device Systems Discovery

Technical Appendices and Supplementary Material

A SUPPORTING FIGURES

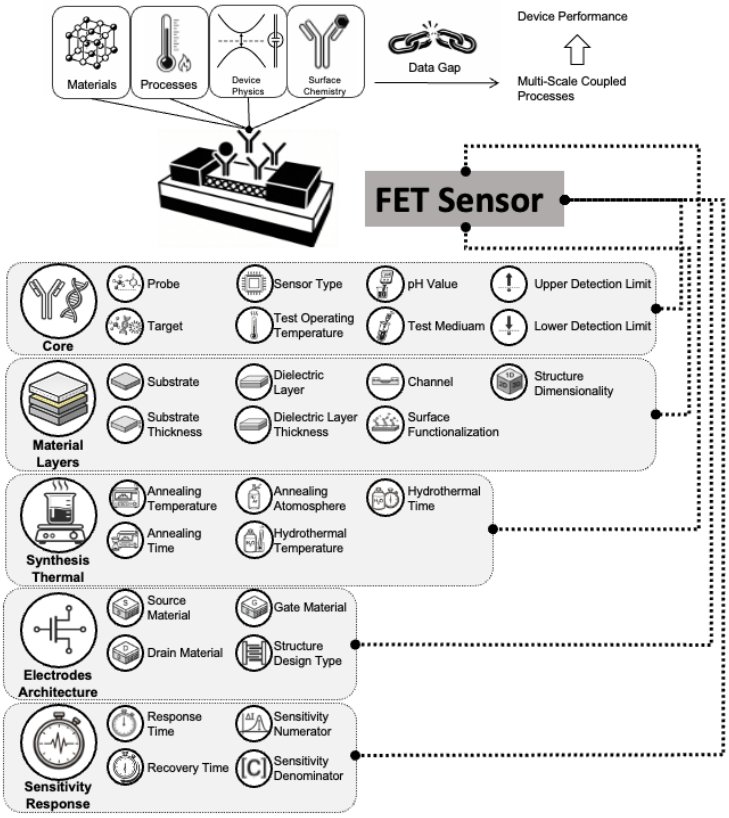

Figure S1: The conceptual gap between traditional "Fragmented View" (optimizing materials or physics in silos) and the reality of "Coupled Systems," separated by the "Data Gap" of unstructured literature. This complexity is instantiated in FET sensors through a high-dimensional ontology mapping the intricate dependencies across synthesis, core chemistry, device architecture, and material layers that jointly determine performance.

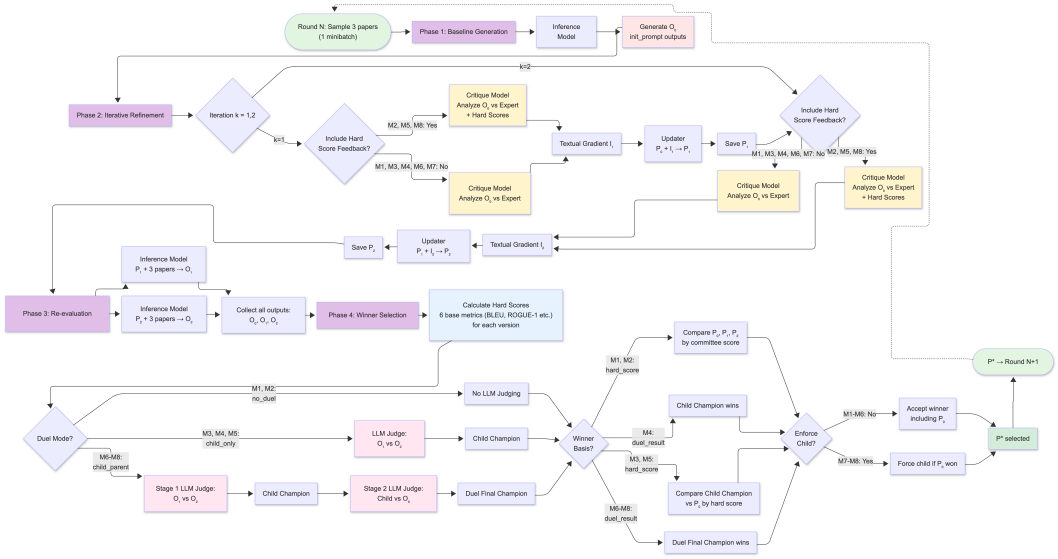

Figure S2: Flowchart schematic of the workflow of different modes: M1-M8 doing automatic prompt optimization.

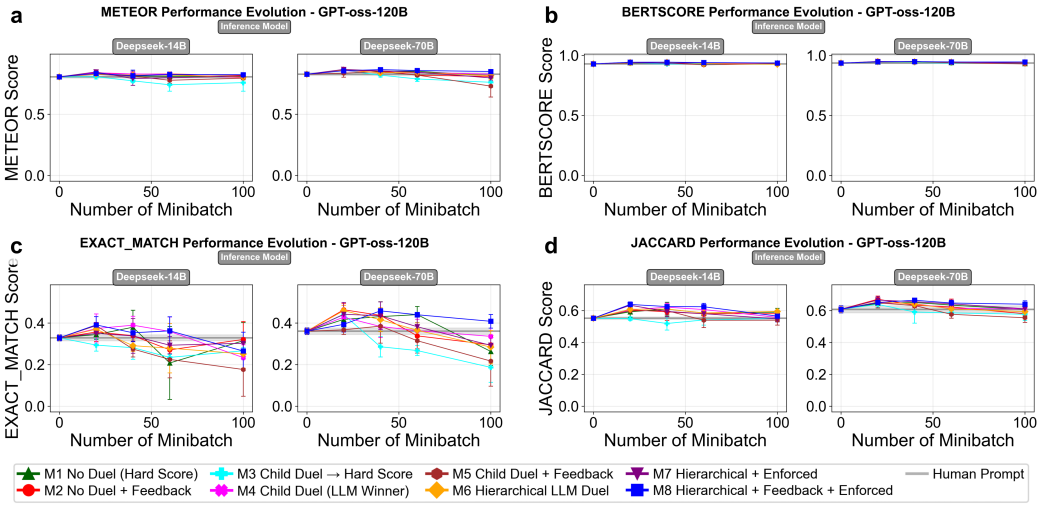

Figure S3: GPT-oss-120B evolution on four additional metrics (METEOR, BERTScore, Exact Match, Jaccard) averaged over 3 folds and inference models Deepseek-14B/70B.

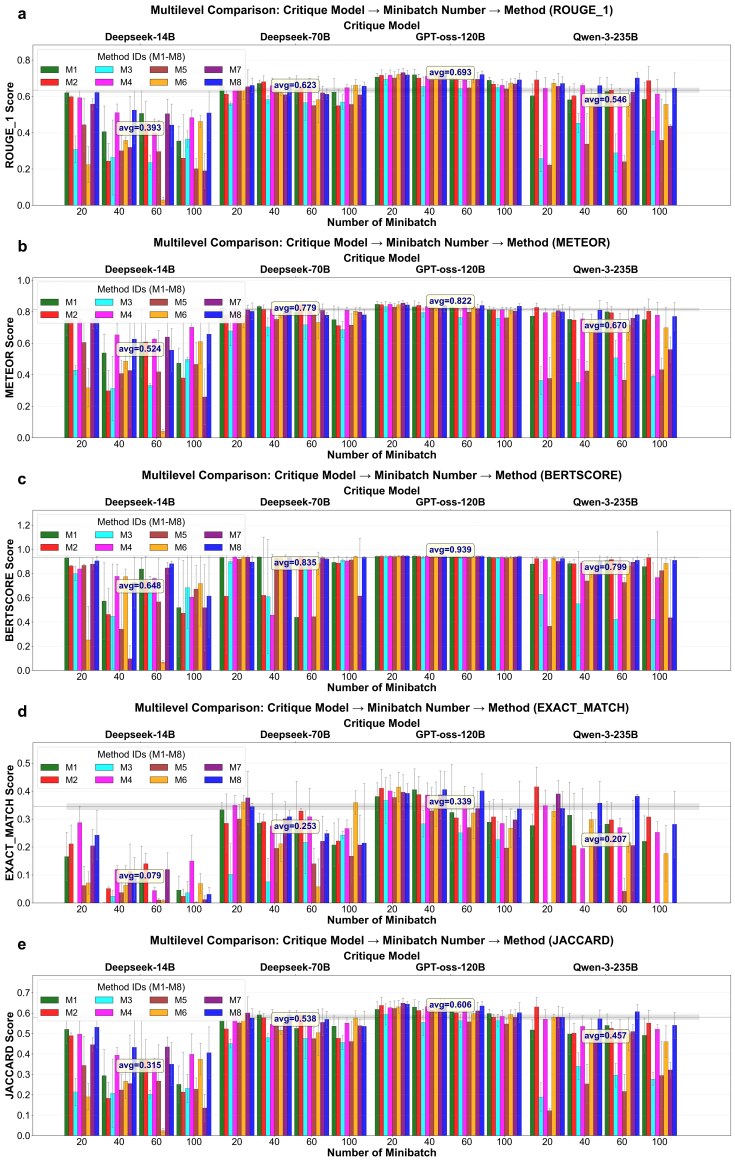

Figure S4: Multilevel barplots (critique → minibatch → method) for ROUGE-1, METEOR, BERTScore, Exact Match, and Jaccard across 3 folds (inference models averaged).

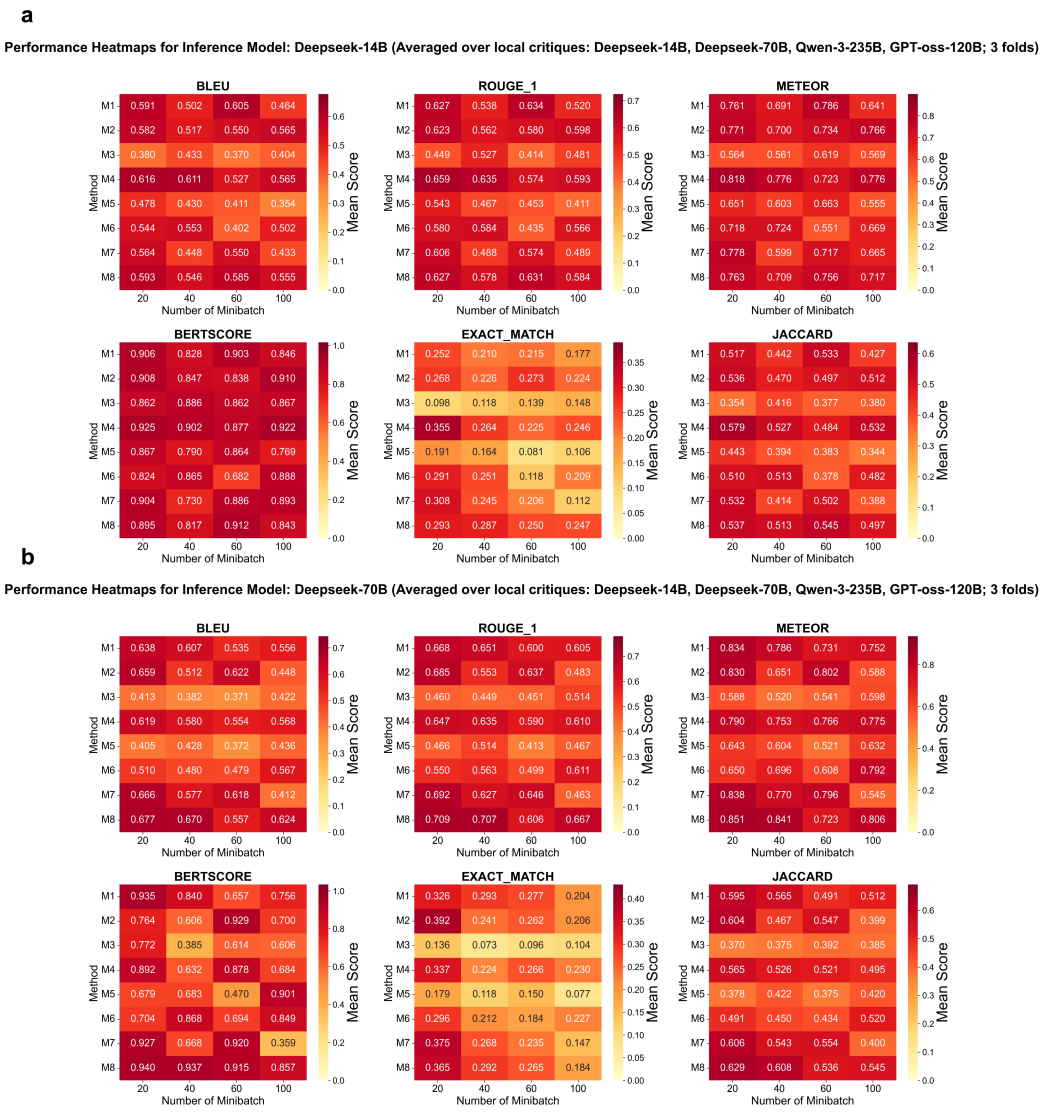

Figure S5: Per-inference-model heatmaps (Deepseek-14B and Deepseek-70B): methods M1–M8 × minibatch sizes (20/40/60/100) across six metrics, averaged over 3 folds.

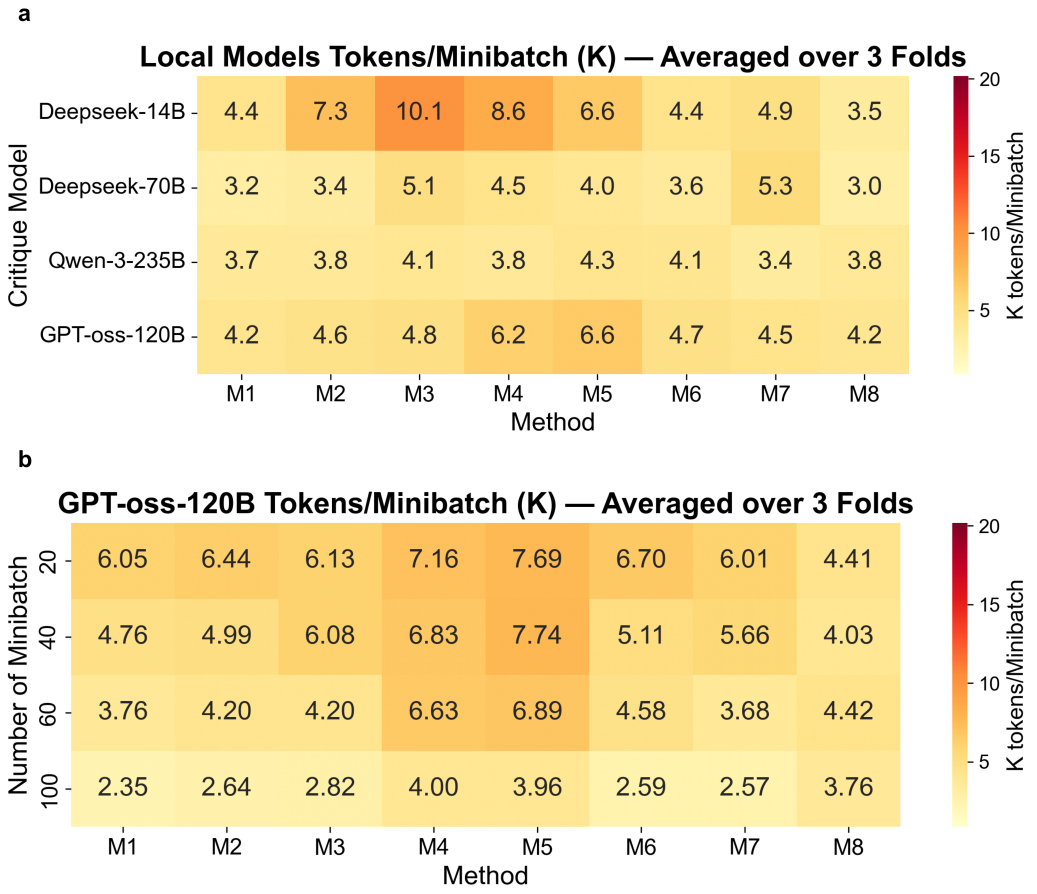

Figure S6: Token usage (K tokens per minibatch) for open-source critiques (top) and GPT-oss-120B (bottom), averaged over inference models and 3 folds.

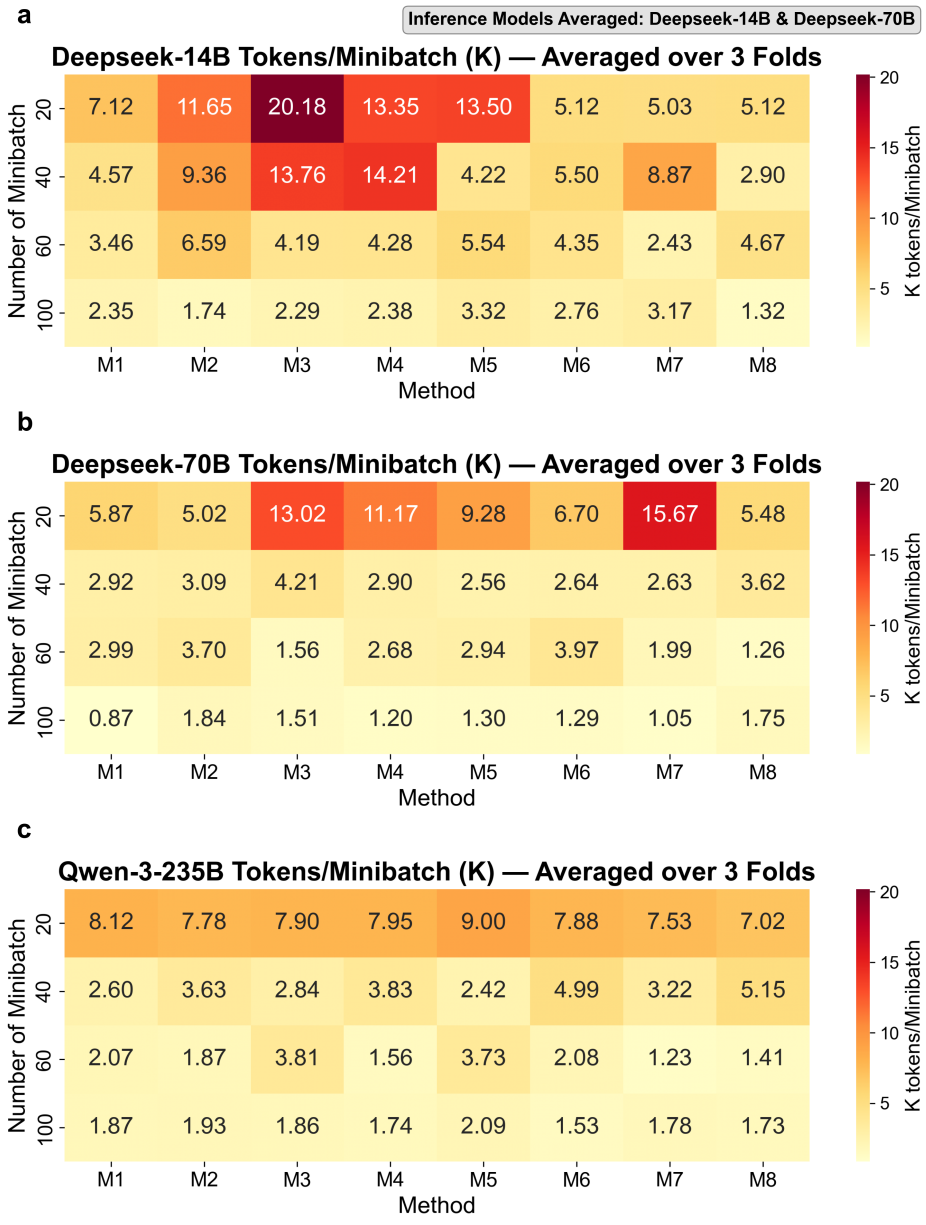

Figure S7: Token usage per minibatch (K tokens) for individual local critiques (Deepseek-14B/70B, Qwen-3-235B), averaged over inference models and 3 folds.

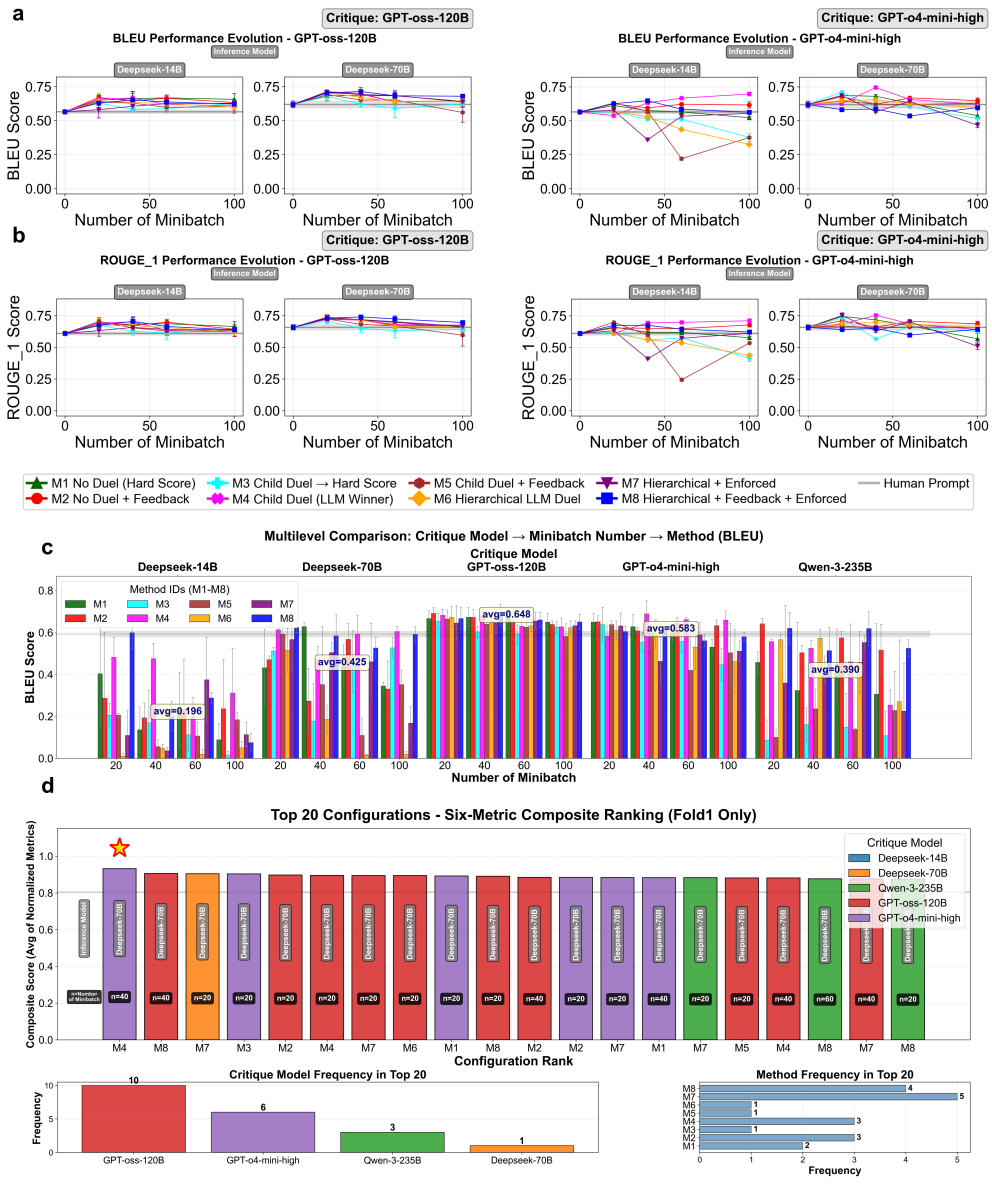

Figure S8: Fold-1 comparison of GPT-oss-120B vs. GPT-o4-mini-high: BLEU/ROUGE evolution, multilevel BLEU barplot, and final recommendation.

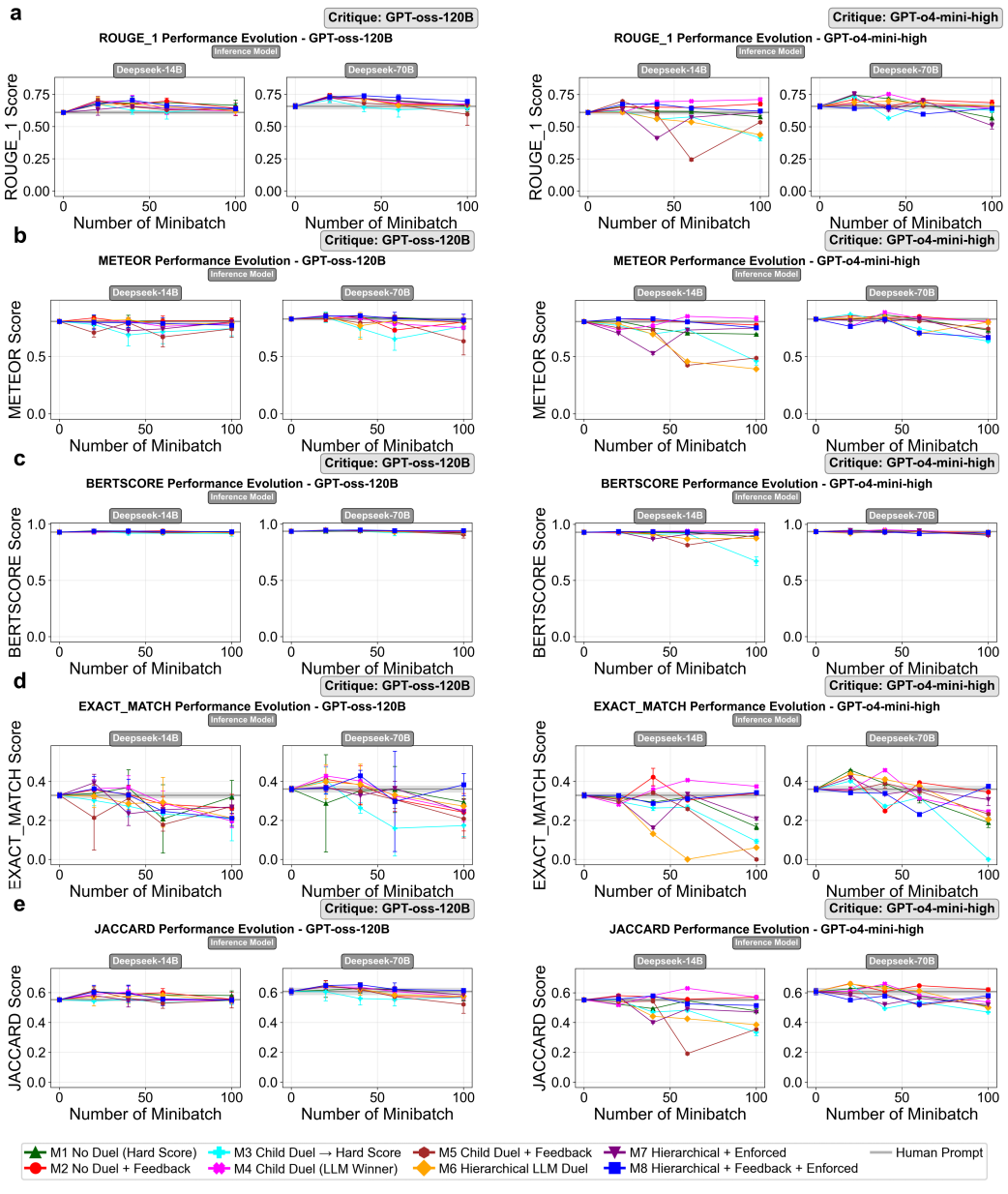

Figure S9: Fold-1 evolution comparisons for GPT-oss-120B vs. GPT-o4-mini-high on METEOR, BERTScore, Exact Match, and Jaccard.

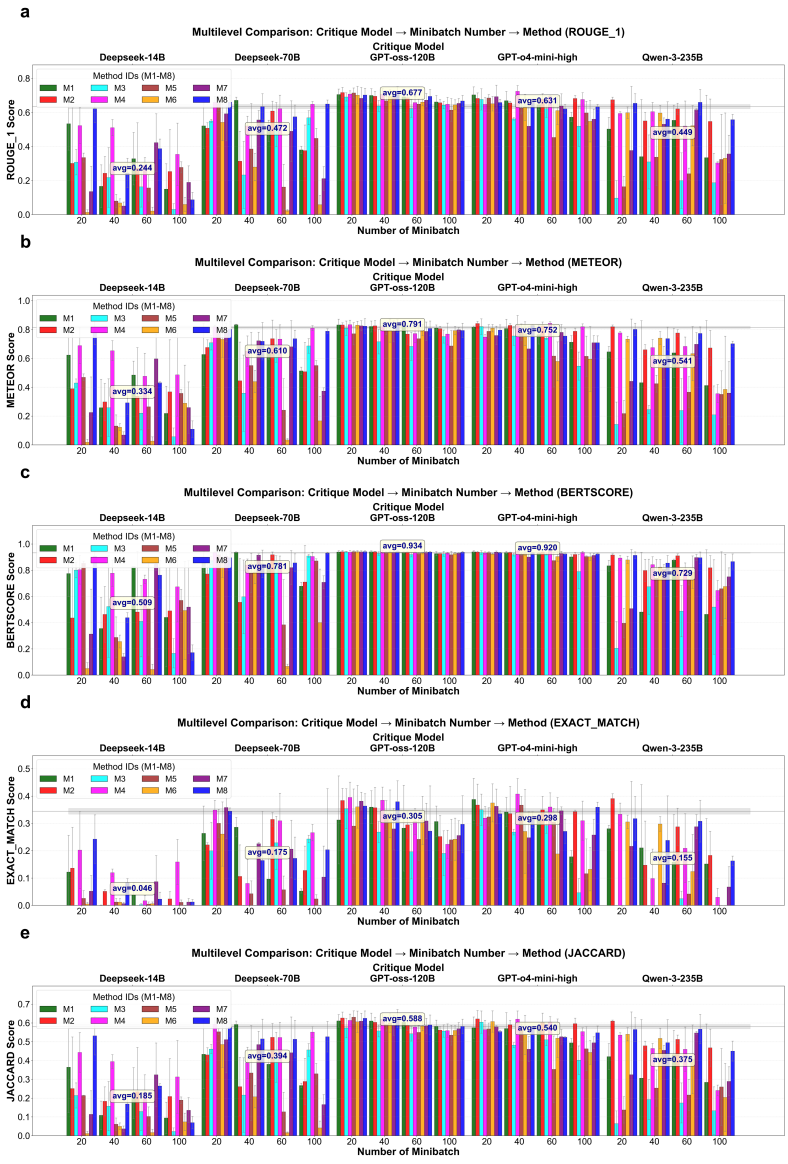

Figure S10: Fold-1 multilevel barplots (with GPT-o4-mini-high included) for ROUGE-1, METEOR, BERTScore, Exact Match, and Jaccard.

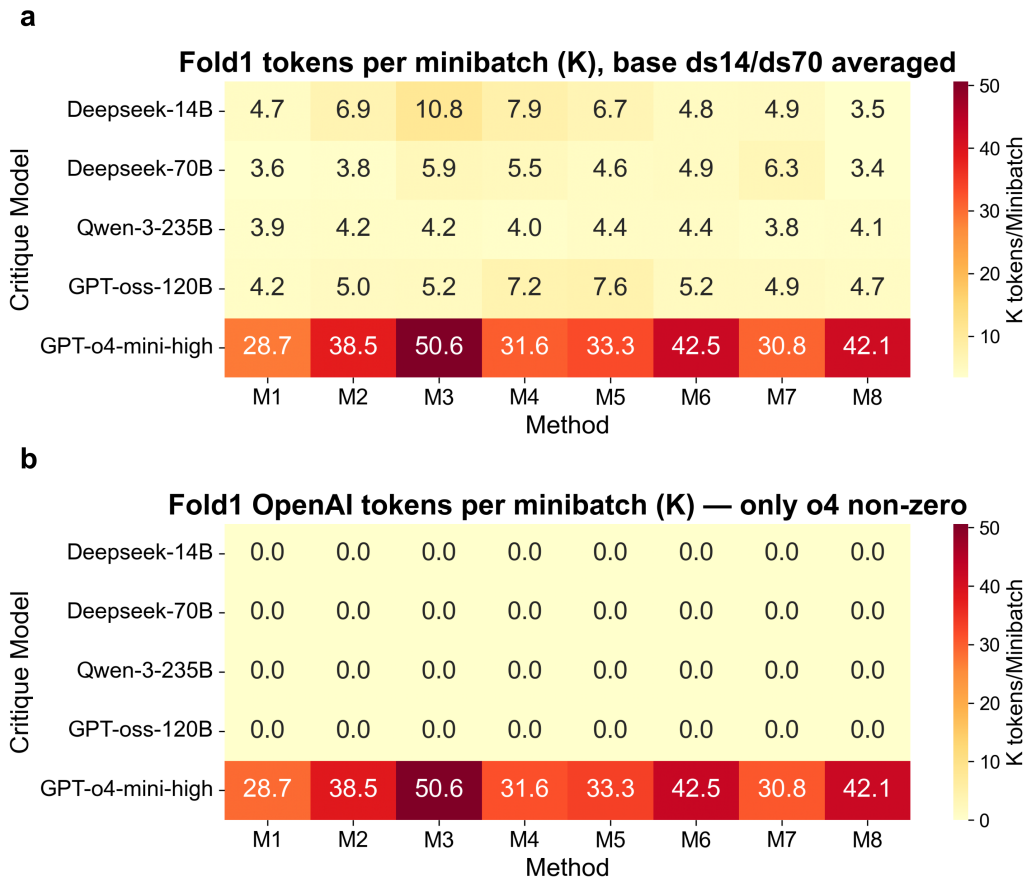

Figure S11: Fold-1 token cost heatmaps: total tokens per minibatch (left) and OpenAI-only tokens (right).

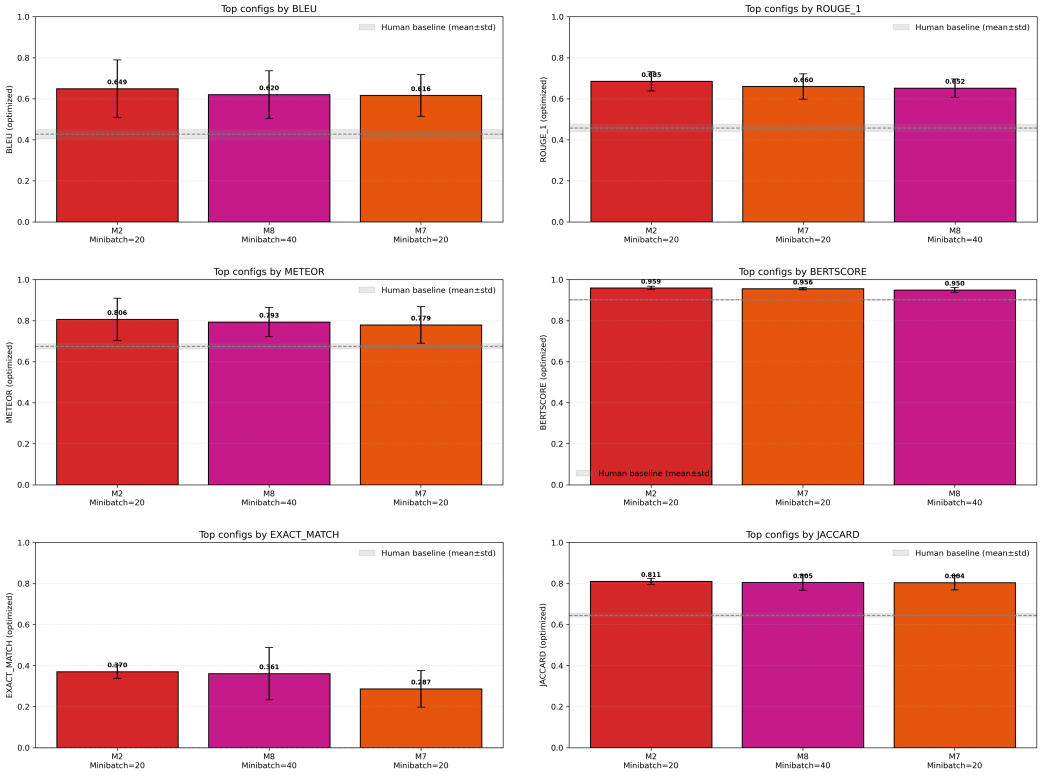

Figure S12: Extended field — Electrode Architecture: top configurations (M8-40, M7-20, M2-20) vs. human baseline across six metrics.

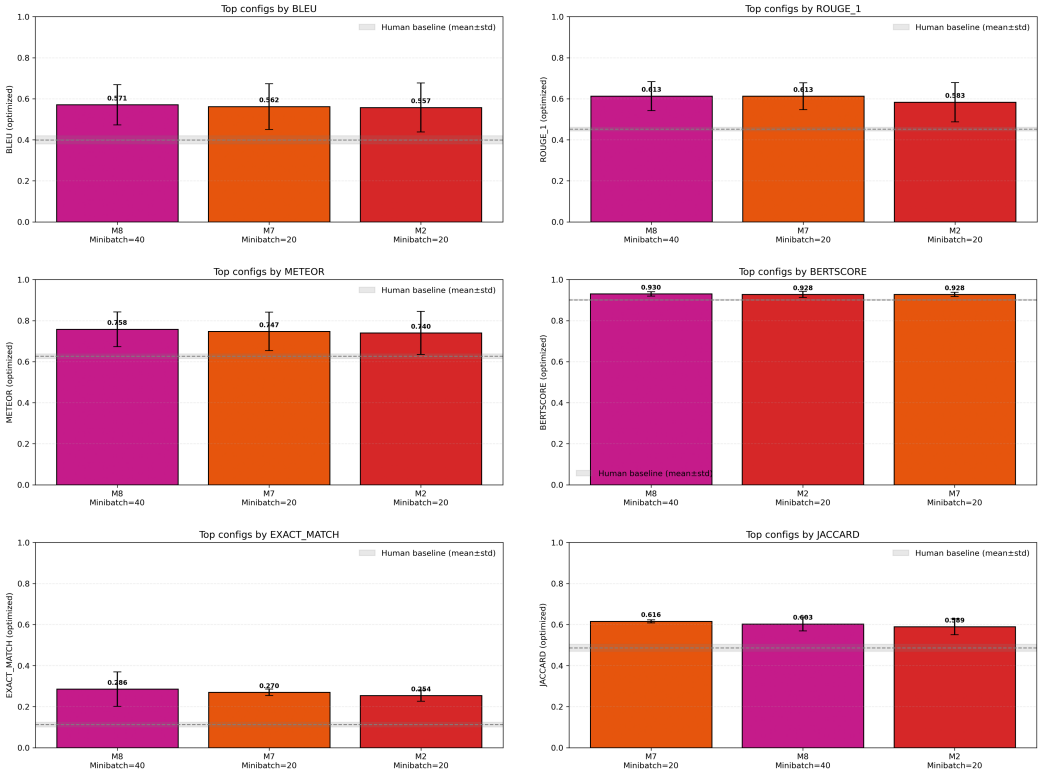

Figure S13: Extended field — Material Layer Composition: top configurations (M8-40, M7-20, M2-20) vs. human baseline across six metrics.

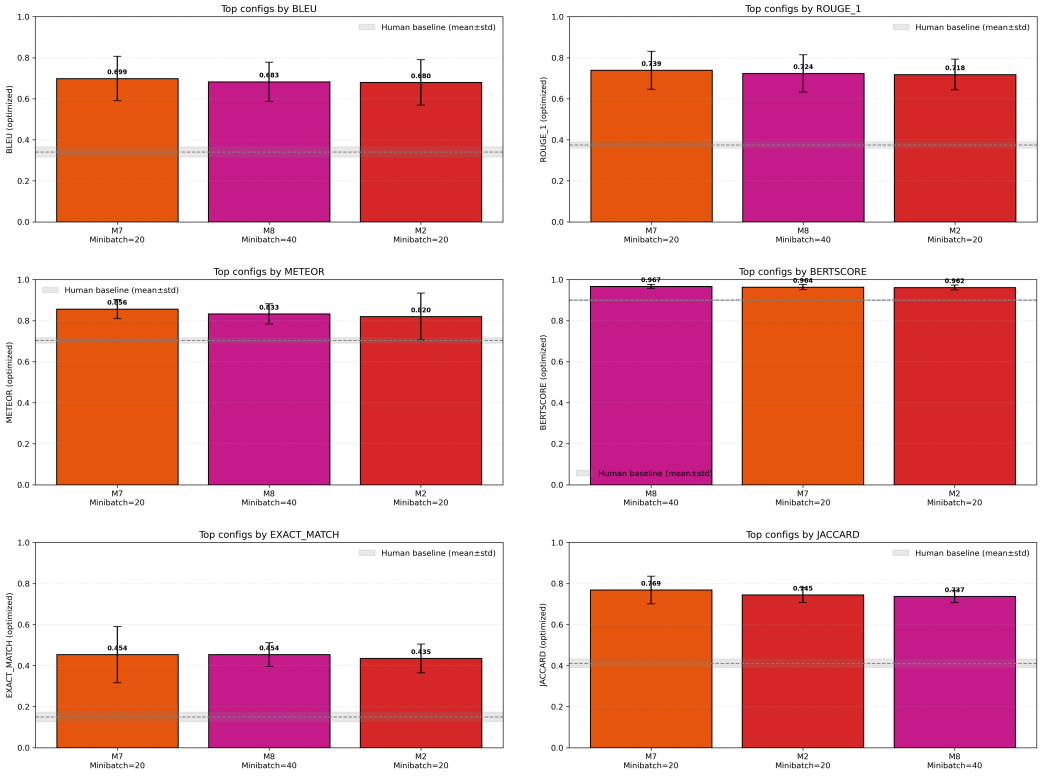

Figure S14: Extended field — Sensitivity and Response: top configurations (M8-40, M7-20, M2-20) vs. human baseline across six metrics.

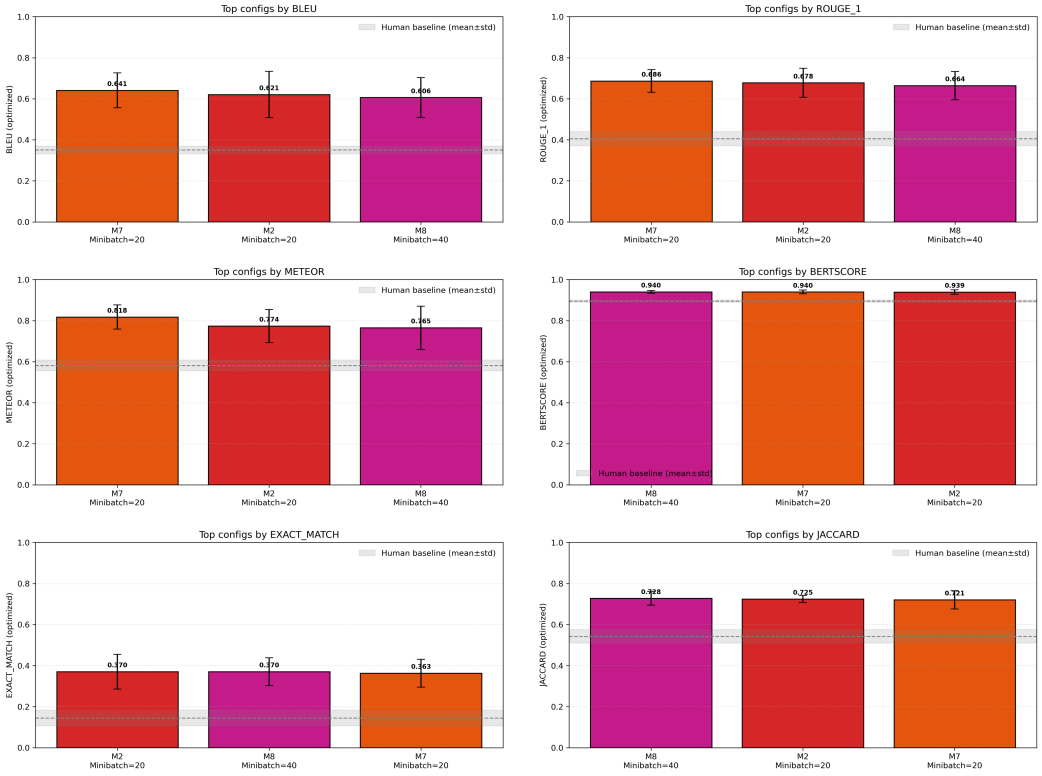

Figure S15: Extended field — Synthesis and Thermal Processing: top configurations (M8-40, M7-20, M2-20) vs. human baseline across six metrics.

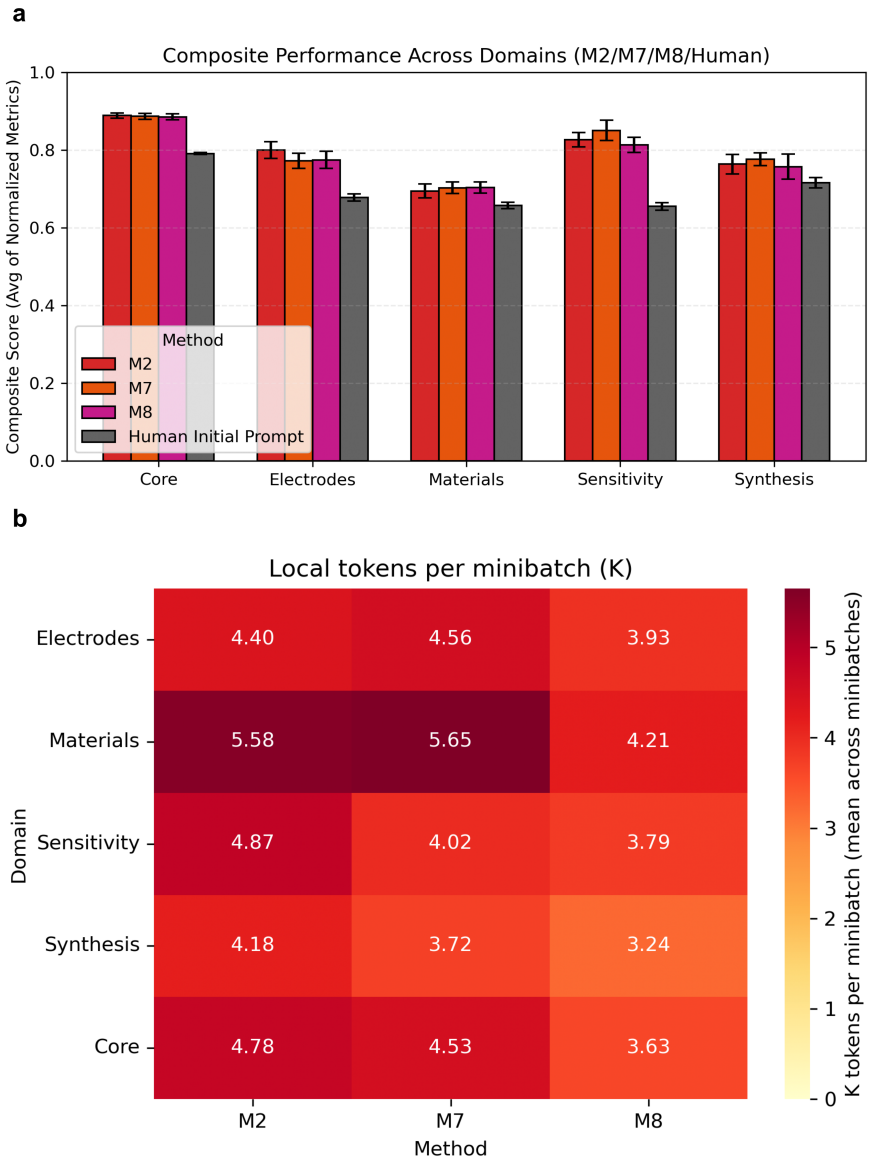

Figure S16: Extended fields summary (panels a–b). (a) Composite scores across five fields for the top-3 strategies vs. Human baseline. (b) Token usage per minibatch (K tokens) aggregated over the four extended fields for the top-3 strategies.

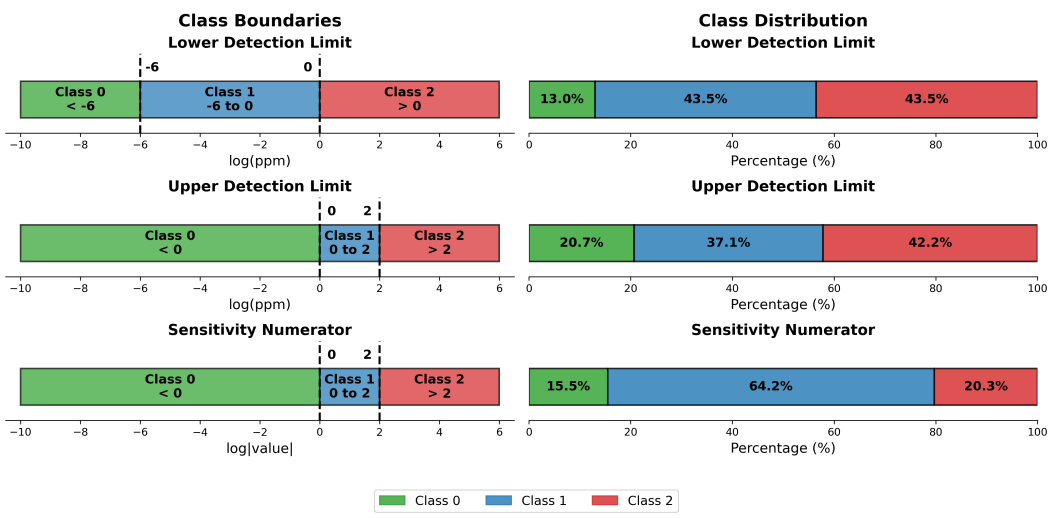

Figure S17: Classification task definitions for Twin-phase device performance prediction. Left: class boundaries on log scale for three prediction targets—Lower Detection Limit (LDL), Upper Detection Limit (UDL), and Sensitivity. Each target is discretized into three classes spanning multiple orders of magnitude. Right: class distribution percentages across the augmented dataset.

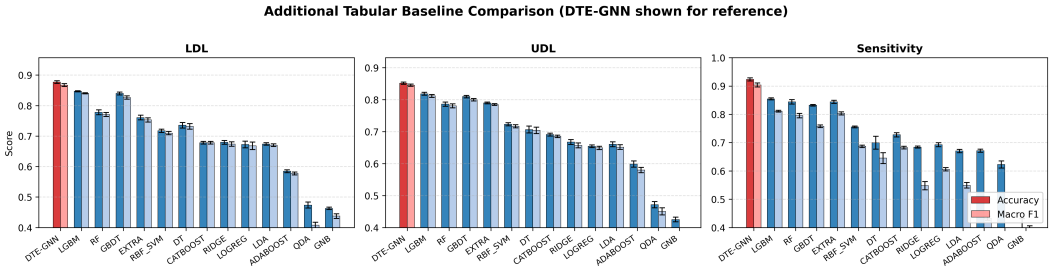

Figure S18: Additional tabular baseline comparison on held-out original data. Performance of 13 tabular machine learning models not shown in the main figure, across LDL, UDL, and Sensitivity prediction tasks. Models are ordered left-to-right by cross-task UCB ranking. While LightGBM (LGBM), Random Forest (RF), and Gradient Boosting (GBDT) achieve competitive performance, all tabular baselines underperform DTE-GNN (Figure 4a), confirming the advantage of heterogeneous graph representation.

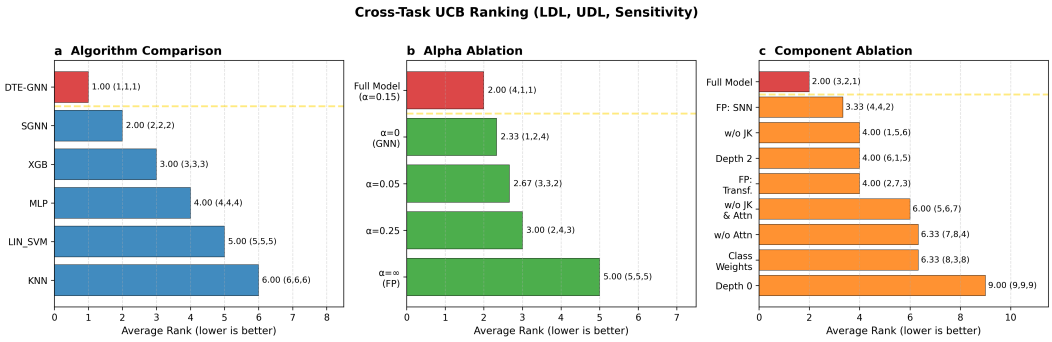

Figure S19: Cross-task UCB ranking visualization (see Section E for methodology). Horizontal bars show the average rank across LDL, UDL, and Sensitivity tasks for each configuration, with per-task ranks shown in parentheses. Lower average rank indicates more consistent top performance. (a) Algorithm comparison: DTE-GNN achieves perfect rank 1 across all tasks. (b) Alpha ablation: Full Model (α=0.15) achieves the best average rank of 2.00. (c) Component ablation: Full Model achieves the best average rank of 2.00, with a 1.33 gap to the second-best configuration.

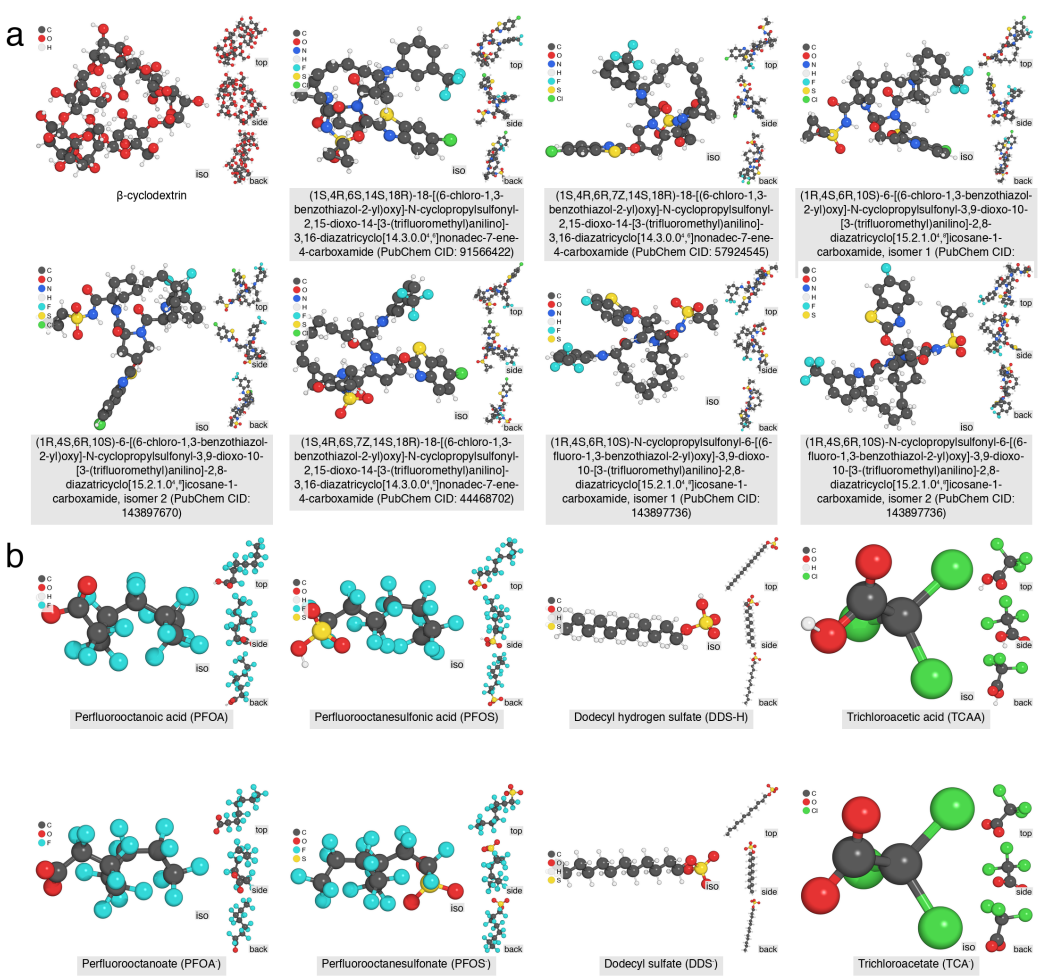

Figure S20: Structure views of the (a) probe and (b) target molecules/anions.

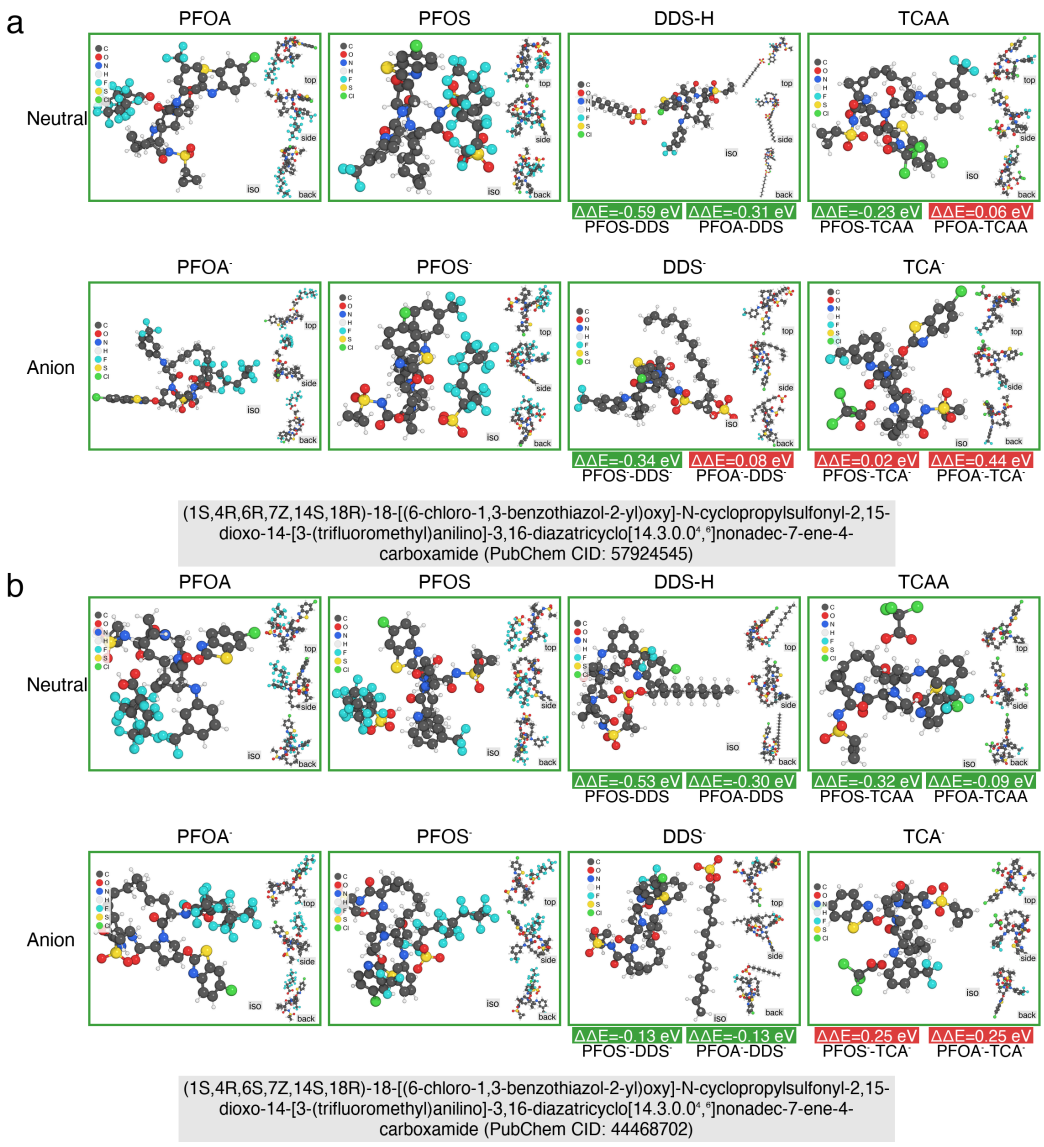

Figure S21: Structure views and DFT-calculated binding energy difference (selectivity) of probe (a) CID-57924545 and (b) CID-44468702 over target molecules/anions.

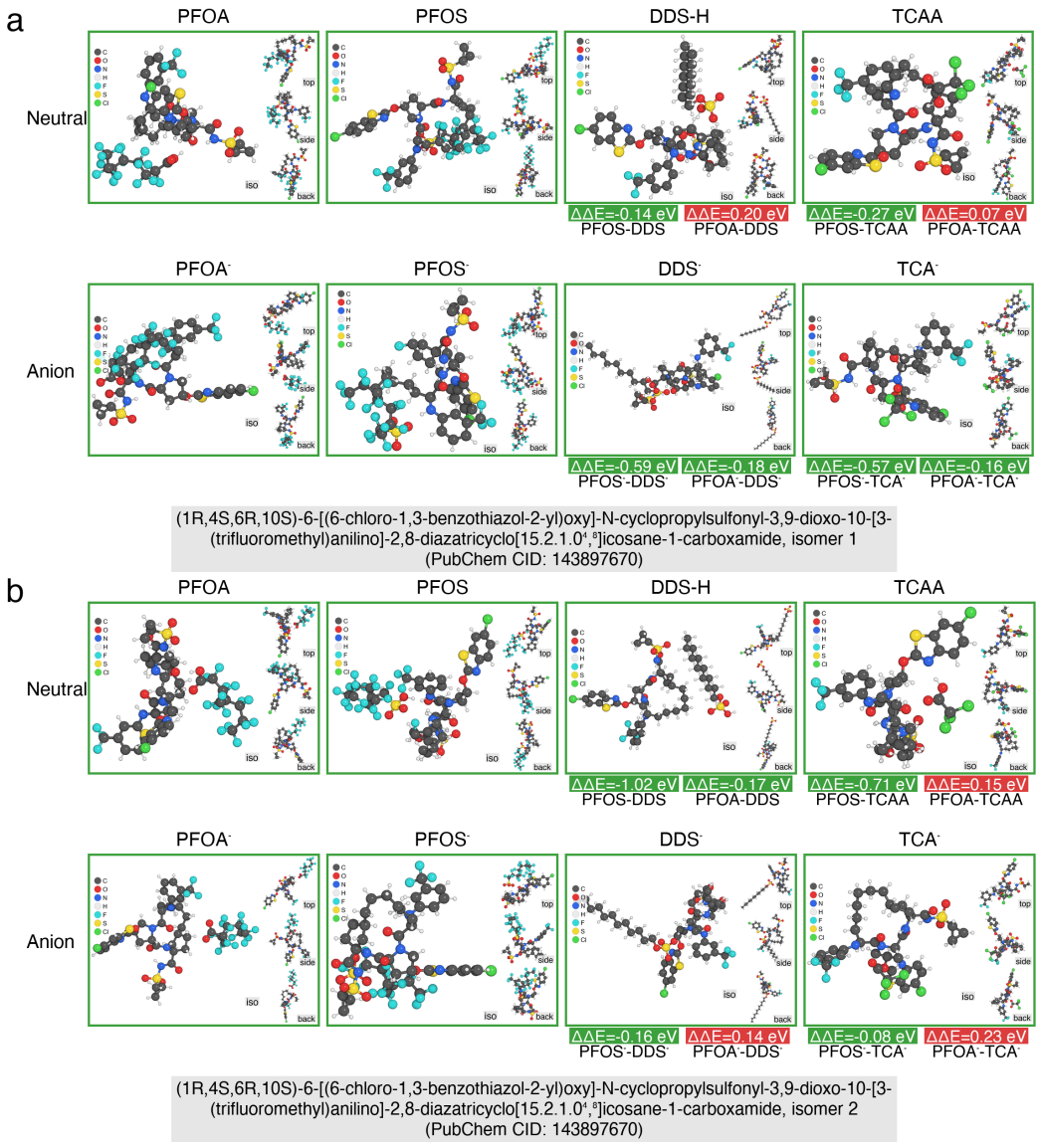

Figure S22: Structure views and DFT-calculated binding energy difference (selectivity) of probe (a) CID-143897670 isomer #1 and (b) CID-143897670 isomer #2 over target molecules/anions.

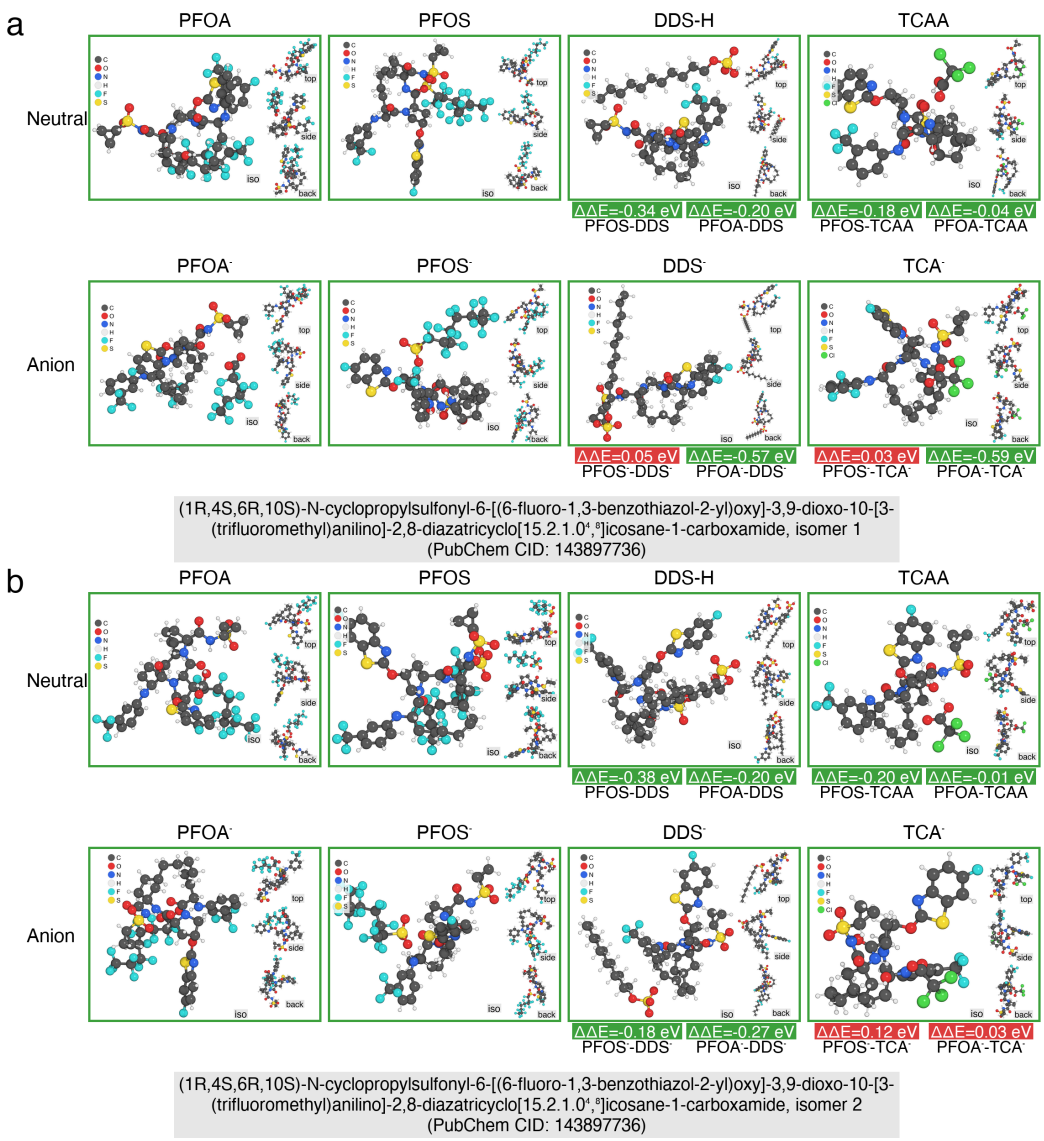

Figure S23: Structure views and DFT-calculated binding energy difference (selectivity) of probe (a) CID-143897736 isomer #1 and (b) CID-143897736 isomer #2 over target molecules/anions.

Table S1: DTE-GNN predicted device-performance scores and DFT binding-energy selectivity ($\Delta\Delta E$, eV) for the top five candidates and β-Cyclodextrin baseline. S_{target} (Eq. 8) is computed for PFOS. Negative $\Delta\Delta E$ indicates preferential PFAS binding (favorable selectivity). Three candidates (CIDs 44468702, 57924545, 91566422) are stereoisomers with identical 2D fingerprints and indistinguishable DTE-GNN scores. Two candidates (CIDs 143897670 and 143897736) have two isomers. Bold marks the only candidate with consistently negative $\Delta\Delta E$ across all conditions.

	DTE-GNN (PFOS)				$\Delta\Delta E$ **Neutral (eV)**				$\Delta\Delta E$ **Anion (eV)**			
Probe	P_0^{LDL}	P_2^{UDL}	P_2^{Sens}	S_{target}	PFOS–DDS	PFOA–DDS	PFOS–TCAA	PFOA–TCAA	PFOS^-–DDS^-	PFOA^-–DDS^-	PFOS^-–TCA^-	PFOA^-–TCA^-
β-Cyclodextrin	0.998	0.000	0.003	∼0	−1.90	−0.43	−1.57	−0.10	+0.54	−0.21	+0.68	−0.07
CID-143897670-isomer1	0.912	0.653	0.200	0.130	−0.14	+0.20	−0.27	+0.07	−0.59	−0.18	−0.57	−0.16
CID-143897670-isomer2	0.912	0.653	0.200	0.130	−1.02	−0.17	−0.71	+0.15	−0.16	+0.14	−0.08	+0.23
CID-143897736-isomer1	0.918	0.585	0.193	0.113	−0.34	−0.20	−0.18	−0.04	+0.05	−0.57	+0.03	−0.59
CID-143897736-isomer2	0.918	0.585	0.193	0.113	−0.38	−0.20	−0.20	−0.01	−0.18	−0.27	+0.12	+0.03
CID-91566422	**0.895**	**0.783**	**0.138**	**0.105**	**−1.53**	**−1.04**	**−1.12**	**−0.62**	**−0.31**	**−0.53**	**−0.23**	**−0.46**
CID-44468702	0.895	0.783	0.138	0.105	−0.53	−0.30	−0.32	−0.09	−0.13	−0.13	+0.25	+0.25
CID-57924545	0.895	0.783	0.138	0.105	−0.59	−0.31	−0.23	+0.06	−0.34	+0.08	+0.02	+0.44

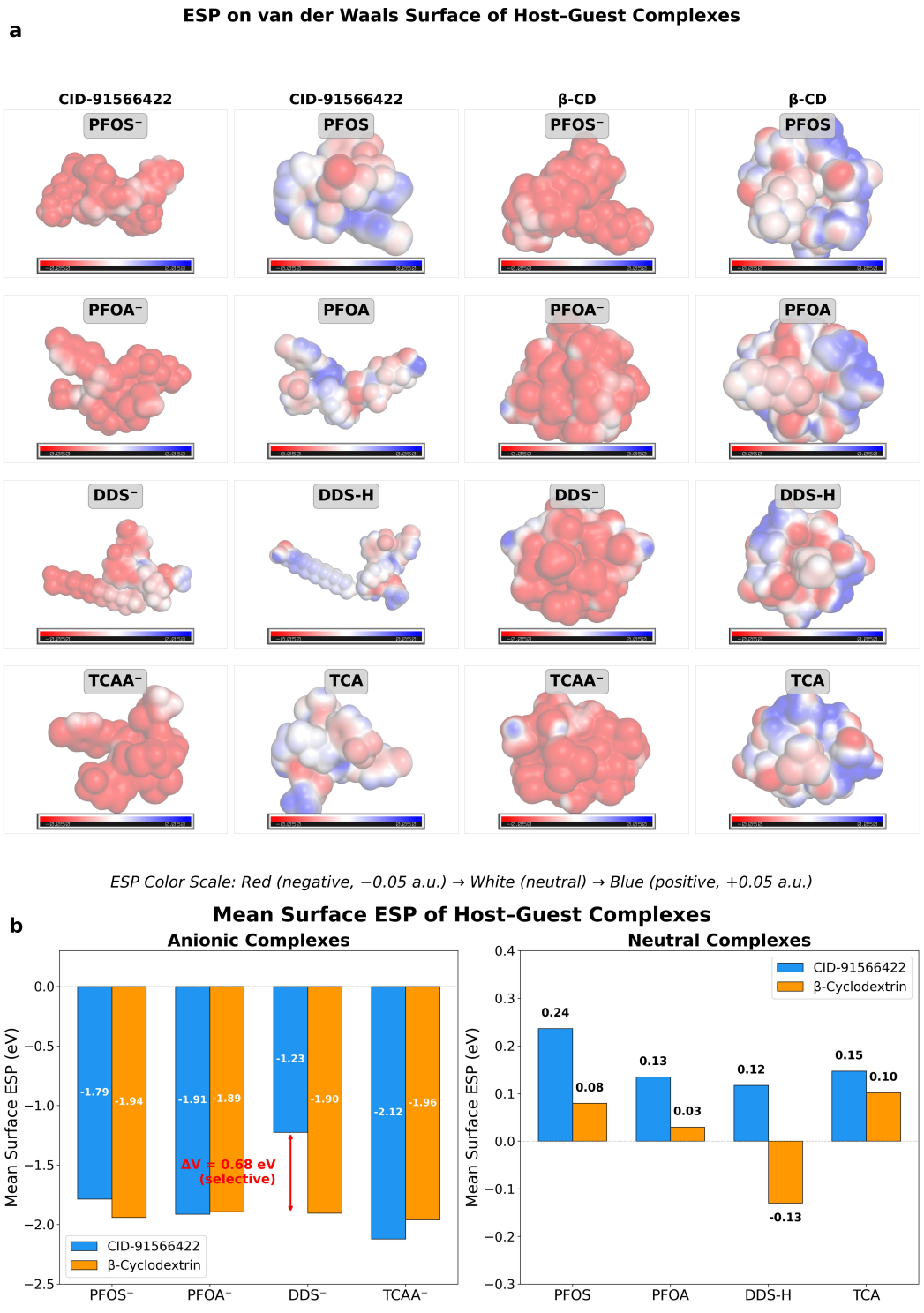

Figure S24: **Electrostatic potential (ESP) analysis of host–guest complexes.** (a) ESP mapped onto the van der Waals surface ($\rho = 0.001$ a.u.) for CID-91566422 and β-Cyclodextrin complexes with four guests under anionic and neutral conditions. Color scale: red (-0.05 a.u.) $\rightarrow$ white (0) $\rightarrow$ blue ($+0.05$ a.u.). (b) Mean surface ESP ($\bar{V}$, eV) comparison. CID-91566422 shows a distinctly less negative $\bar{V}$ for DDS$^-$ (-1.23 eV) compared to PFAS guests (-1.79 to -2.12 eV), while β-Cyclodextrin shows uniform $\bar{V} \approx -1.9$ eV for all guests.

B Additional Related Work

LLM-based Scientific Information Extraction

Scientific information extraction (IE) has long sought to convert the narrative style of papers into machine-actionable records. Earlier materials/chemistry pipelines relied on rule-based parsing and domain heuristics, exemplified by ChemDataExtractor Swain & Cole (2016). With the rise of scientific-language pretraining, transformer encoders such as SciBERT Beltagy et al. (2019) enabled stronger scientific NER and sentence-level semantics, while domain-focused IE systems further emphasized entity normalization and scalable extraction in materials literature Weston et al. (2019). Beyond NER, end-to-end scientific IE has also been framed as joint extraction of entities, relations, and coreference to support structured scholarly representations Luan et al. (2018). Despite these advances, most approaches either require substantial labeled data or must be carefully adapted to each narrow schema and subdomain, a challenge that becomes acute for device papers where fabrication, geometry, interfaces, and performance metrics are tightly entangled.

More recently, generative LLMs have been used for schema-driven structured extraction from scientific text, including large-scale demonstrations in materials and chemistry Dagdelen et al. (2024). However, these systems often depend on extensive fine-tuning, brittle prompt crafting, or post-hoc repair to control output validity and cross-field consistency. Domain-specific pretraining can help but does not eliminate the supervision bottleneck for specialized schemas Trewartha et al. (2022). In contrast, our T^3 framework is designed as an end-to-end pipeline that couples autonomous, prompt-optimized extraction with downstream device-level prediction, directly targeting the extraction–modeling gap for heterogeneous FET sensor literature.

Prompt Optimization and Textual Gradients

Prompting has become a central mechanism for steering frozen LLMs, including reasoning-oriented prompting strategies such as chain-of-thought Wei et al. (2022). To reduce reliance on manual prompt engineering, automated prompt search and refinement has been explored via LLM-driven proposal-and-selection loops. Automatic Prompt Engineer (APE) Zhou et al. (2023) and Optimization by PROmpting (OPRO) Yang et al. (2024) treat prompts as optimizable parameters, iteratively generating candidate instructions and selecting those that improve task performance. In parallel, system-building frameworks such as DSPy Khattab et al. (2023) compile multi-step LLM pipelines and tune prompts programmatically, highlighting a shift from single-prompt tuning to compositional optimization.

A complementary line of work introduces optimization signals that resemble gradients, where critiques or losses are converted into text edits. Automatic Prompt Optimization (APO) operationalizes this idea through minibatch "natural language gradients" plus guided search Pryzant et al. (2023), while TextGrad generalizes the concept by backpropagating LLM-generated feedback through computation graphs to improve upstream components Yuksekgonul et al. (2025b). Most evaluations, however, emphasize generic NLP/reasoning benchmarks, leaving open how to robustly optimize prompts under scientific constraints (units, normalization, nested schemas, and strict output validity). Our contribution extends textual-gradient optimization into a multi-mode TextGrad regime tailored to scientific IE, enabling reliable schema-conformant extraction with minimal human annotation and directly powering the Text→Twin translation in T^3.

Graph Neural Networks for Materials and Molecules

Graph neural networks (GNNs) are now the dominant paradigm for molecular and materials property prediction because they encode relational inductive biases over atoms and bonds. Message Passing Neural Networks (MPNNs) provide a unifying formulation Gilmer et al. (2017), while geometric architectures such as SchNet Schütt et al. (2017) and directional message passing (DimeNet) Klicpera et al. (2020) capture distance- and angle-dependent interactions critical for quantum properties. In materials science, crystal graph models extend message passing to periodic solids Xie & Grossman (2018), and more recent variants incorporate richer geometric primitives (e.g., line graphs) to improve accuracy on benchmark property prediction tasks Choudhary & DeCost (2021). Alongside GNNs, learned and hand-crafted fingerprints remain widely used across modalities, including

extended-connectivity fingerprints for molecules Rogers & Hahn (2010) and composition-based featurizations for inorganic materials Ward et al. (2016).

Despite strong performance at the molecule/crystal level, most GNN work assumes a single connected structure as the prediction object, whereas real sensing devices are multi-component systems whose performance emerges from topology, interfaces, and coupled functional modules. Generic relational modeling (e.g., multi-relation GNNs) can represent typed edges and interactions Schlichtkrull et al. (2018b), but has rarely been specialized to device-scale, physics-informed topologies. Our Twin component closes this gap by constructing a physics-informed heterogeneous graph for FET sensors (device topology + cross-domain fingerprints), enabling device-level predictions rather than single-material property inference.

FET Sensor Modeling and Performance Prediction

FET sensors and related BioFET/ISFET devices have traditionally been analyzed using physics-based models that connect surface charge, electrostatics, and carrier transport to measurable signals. Foundational ISFET work established the core sensing principle in transistor form Bergveld (1970), and subsequent analyses studied fundamental limits arising from screening and electrolyte effects Stern et al. (2007) as well as performance ceilings for nanoscale BioFET sensors Nair & Alam (2006). These approaches are physically interpretable, but high-fidelity simulation and parameter calibration can be costly and difficult to scale across diverse materials, geometries, and functionalization strategies; moreover, device physics references emphasize that performance depends on coupled choices across stacks and interfaces rather than isolated parameters Sze & Ng (2006).

In response, data-driven methods have been increasingly used for sensor modeling and inference, including ML for biosensor signal processing and performance characterization Cui et al. (2020). Yet most learning-based pipelines flatten devices into tabular features, discarding explicit topology and making it hard to generalize across architectures (e.g., gate stacks, channel classes, probe layers) or to propagate fabrication/process effects through coupled interfaces. Building on prior graph-based progress for FET chemical sensor prediction Ferreira et al. (2025), T^3 introduces a device-topology-aware, physics-informed heterogeneous representation that can be populated automatically from literature and trained end-to-end for device-level performance prediction.

Scientific Knowledge Graphs and Literature Mining

Scientific knowledge graphs (KGs) and scholarly corpora aim to represent literature at scale to support retrieval, reasoning, and discovery. Large infrastructure efforts have produced heterogeneous literature graphs Ammar et al. (2018) and open corpora with structured full text and metadata Lo et al. (2020), providing foundations for downstream IE and linking. In materials science, major open databases such as the Materials Project Jain et al. (2013) and AFLOW Curtarolo et al. (2012) have accelerated property-driven research, but they primarily capture computed or curated structured data rather than the full experimental detail embedded in narrative papers.

To complement curated resources, literature mining has generated domain-specific datasets by extracting experimental procedures and properties directly from text, including synthesis recipe databases Kononova et al. (2019) and auto-generated magnetic transition temperature resources Court & Cole (2018). Beyond symbolic KGs, representation learning over text (e.g., word embeddings) has been shown to recover latent materials knowledge and even anticipate discoveries Tshitoyan et al. (2019), and surveys have emphasized the broader opportunity and challenges of NLP/IE for materials databases Olivetti et al. (2020). A persistent limitation, however, is that extracted entities and relations are often not wired into downstream predictive models—KGs remain largely descriptive. T^3 directly addresses this disconnect by designing extraction schemas that instantiate a predictive device-twin graph, linking literature mining to topology-aware performance models in a single end-to-end pipeline.

C METHOD AND TECHNICAL DETAILS

C.1 METHOD AND TECHNICAL DETAILS FOR "TEXT"

C.1.1 DATASET AND TASK DEFINITION

FET Sensor Literature Corpus Building upon prior work on neuromorphic spiking graph neural networks for FET sensor design Ferreira et al. (2025), we leverage a curated collection of 844 peer-reviewed scientific publications on field-effect transistor (FET) sensors. These articles were sourced from major scientific databases including IEEE Xplore, ACS Publications, Royal Society of Chemistry, Elsevier ScienceDirect, and Springer Nature. Each publication contains the full manuscript text, including experimental sections, results, and supplementary materials where available. This corpus represents a comprehensive cross-section of the FET sensor literature published over the past two decades, capturing both materials innovation and device engineering advances.

Importantly, the corpus is filtered to include only experimental studies (excluding computational simulations such as DFT or AIMD), and focuses specifically on concentration-detection applications spanning gas-phase, chemical, and biological FET sensors. Publications concerning non-concentration measurements (e.g., pressure sensing) are excluded from this dataset.

In that prior study Ferreira et al. (2025), domain experts manually curated structured metadata from these publications, establishing a high-quality ground-truth dataset. Each paper was annotated to extract eight core performance and operational parameters that characterize FET sensor functionality. These expert annotations serve as the gold standard for evaluating automated extraction methods in the present work.

Structured Information Extraction Task The primary objective of this work is to develop an automated pipeline capable of extracting structured sensor metadata from unstructured scientific text with fidelity approaching human expert performance. Specifically, given the full text of a scientific publication, the system must generate a structured JSON output containing the following fields:

- sensor_type: Classification of the sensor modality (gas, bio, or liquid)
- detect_target: Chemical or biological species being detected (e.g., ammonia, glucose, DNA)
- lower_detection_limit: Minimum detectable concentration with units (e.g., 5 ppb, 1 nM)
- upper_detection_limit: Maximum measurable concentration before saturation
- probe_material: Active sensing material or functionalization layer (e.g., metal oxide, antibody, polymer)
- test_operating_temperature: Experimental temperature in degrees Celsius
- pH_value: Solution pH for liquid/bio sensors, or -1 for gas-phase measurements
- test_medium: Environmental context of sensing experiments (e.g., air, phosphate buffer, serum)

This extraction task is challenging due to several factors: (1) high variability in reporting conventions across journals and research groups, (2) implicit information requiring domain knowledge inference (e.g., room temperature defaults, pH assumptions), (3) multi-scale unit conversions (ppb, ppm, molarity), and (4) disambiguation when multiple sensor variants are presented in a single publication. Moreover, detection limits are often reported indirectly through figures or cited without explicit numerical statements, requiring contextual reasoning beyond simple pattern matching.

Data Partitioning To ensure robust evaluation and generalization assessment, we employ a three-fold cross-validation strategy. The 844 publications are partitioned into three non-overlapping folds, each maintaining representative coverage of sensor types and target analytes. For each fold, the data is divided into:

- Training set: 583 publications (approximately 69 percent) for prompt optimization and model development

- Test set: 261 publications (approximately 31 percent) held out for final performance evaluation

The fold assignments are deterministic and fixed across all experiments to enable fair comparison between different optimization strategies. Critically, the test set remains completely isolated from the training process; no information from test set papers influences prompt refinement or model selection. The development set is available but not actively utilized in the current study, as our TextGrad-based optimization operates exclusively on training set feedback.

All data splits preserve the original expert annotations, ensuring that each publication in the corpus has a corresponding ground-truth JSON record. This pairing enables automated metric computation by comparing system-generated outputs against expert-curated references across multiple dimensions: lexical overlap (BLEU, ROUGE), semantic similarity (BERTScore), and structured field accuracy (Exact Match, Jaccard Index).

Extended Field Extraction Beyond the eight core fields annotated in the prior work Ferreira et al. (2025), we extend the extraction scope to four additional field categories encompassing 20 supplementary parameters:

Electrode Architecture (4 fields) Device electrode configuration including gate, source, and drain electrode materials, along with structure design type (e.g., planar, vertical, coplanar, back-gated, top-gated, interdigitated).

Material Layer Composition (7 fields) Layer-by-layer device structure comprising substrate material and thickness, channel (active layer) material, dielectric layer material and thickness, surface functionalization molecules, and structure dimensionality (0D/1D/2D/3D nanostructure classification).

Sensitivity and Response Characteristics (4 fields) Dynamic performance metrics including response time, recovery time, and sensitivity definition (numerator representing signal change such as ΔR, ΔI, or ΔG, and denominator representing reference value such as baseline resistance or analyte concentration).

Synthesis and Thermal Processing (5 fields) Fabrication conditions encompassing annealing temperature, duration, and atmosphere (air, N_2, Ar, O_2, vacuum, H_2), as well as hydrothermal synthesis temperature and duration.

For these extended fields, expert annotations are performed on a smaller corpus of 171 publications (approximately 20 percent of the core dataset size), reflecting a deliberate strategy to minimize human labeling effort while maintaining sufficient data for prompt optimization.

Our optimization workflow proceeds in two stages. First, the eight optimization modes (M1-M8) are systematically evaluated on the core dataset (844 papers) to identify the top-performing strategies. The three best-performing strategies are then applied to optimize prompts for all five field categories (core plus four extended), yielding 15 optimized prompts in total. In the final deployment phase, these prompts are applied to the full corpus of 1,686 publications. For each field, human experts review the outputs from the three strategy variants and perform lightweight curation to produce the final structured database. This ensemble approach, combining multiple optimization strategies with expert oversight, ensures robust extraction quality while leveraging the complementary strengths of different optimization modes.

Problem Formulation The central research question addressed in this phase is: Can an autonomous prompt optimization framework iteratively refine a natural language prompt to achieve expert-level extraction performance on complex scientific information extraction tasks, without manual trial-and-error? We begin with a minimal human-authored prompt (18 lines, providing only a JSON schema template) and seek to evolve it through data-driven optimization into a comprehensive extraction guideline that encodes domain conventions, edge case handling, and implicit reasoning strategies. Success is measured by comparing the optimized prompt against the baseline across six complementary metrics computed on the held-out test set, with particular emphasis on

the Committee Score (average of BERTScore, ROUGE-1, and METEOR) as the primary quality indicator.

C.1.2 Automatic Prompt Optimization via TextGrad

Overview Traditional prompt engineering for large language models relies on manual trial-and-error refinement, which is labor-intensive, subjective, and difficult to scale across diverse tasks. To address this limitation and enable autonomous prompt refinement, we adopt TextGrad Yuksekgonul et al. (2025b), a gradient-based optimization framework that treats natural language prompts as differentiable parameters. Analogous to how neural network training computes numerical gradients to update model weights, TextGrad computes *textual gradients*—natural language critiques generated by an LLM that describe how the prompt should be modified to improve task performance. This formulation conceptualizes the LLM as a differentiable computational graph where the prompt serves as a learnable input, the extraction task defines the loss function, and the critique model provides the optimization signal, enabling principled iterative refinement without manual intervention.

In this work, we instantiate TextGrad through a two-model architecture specifically designed for scientific information extraction tasks. Our implementation extends the original TextGrad framework by introducing minibatch training, hierarchical evaluation strategies, and multi-round iterative refinement to handle the complexity and variability inherent in real-world scientific literature.

Two-Model Architecture Our optimization pipeline employs a division of labor between two distinct language model components, each serving a specialized function in the prompt refinement process.

Inference Model The Inference Model serves as the production system responsible for performing the actual information extraction task. Given a scientific publication's full text and a candidate prompt, the Inference Model generates structured JSON output containing the eight target fields described in Section 2.1. We evaluate multiple open-source models in this role, primarily focusing on DeepSeek-R1 variants (14B and 70B parameter versions) due to their strong reasoning capabilities and local deployment feasibility.

The Inference Model operates under inference mode only and does not undergo any fine-tuning or parameter updates. Its behavior is entirely controlled by the prompt text, making prompt quality the sole determinant of extraction performance. This constraint aligns with practical deployment scenarios where model retraining is infeasible due to computational costs or proprietary model restrictions.

Critique Model The Critique Model functions as a meta-optimizer that analyzes the performance gap between Inference Model outputs and expert annotations. Operating at a higher level of abstraction, the Critique Model receives as input: (1) the current prompt text, (2) a minibatch of scientific papers, (3) the corresponding Inference Model outputs for those papers, and (4) the ground-truth expert annotations. Its task is to identify systematic deficiencies in the current prompt and generate textual feedback (the "gradient") that guides prompt revision.

We explore several model choices for this critique role, including DeepSeek-R1 (14B/70B), Qwen3 (235B parameters), and proprietary models such as GPT-o4-mini-high. The choice of Critique Model represents a key design decision, as stronger reasoning capabilities in this component can lead to more effective prompt optimization, albeit at higher computational cost.

Critically, the Critique Model does not directly rewrite the prompt. Instead, it produces an intermediate representation: a natural language instruction enumerating specific issues to address (e.g., "The prompt fails to handle pH conversion from logarithmic scale" or "Detection limit extraction ignores P/Pv ratio formats"). This separation ensures that prompt modifications remain interpretable and traceable.

Updater Component Following critique generation, a third component (the Updater, also implemented as an LLM call) consumes the critique's textual gradient alongside the current prompt and produces a revised prompt (Eq. 2). The textual gradient identifies a small number of targeted deficiencies relative to expert annotations (typically 1–3 per round). To encourage incremental re-

finement rather than drastic rewrites, the Updater instruction limits the scope of each revision to a modest fraction of the prompt tokens (e.g., up to roughly 25%), preventing wholesale replacement that might discard functional aspects of the existing prompt. This is a soft constraint conveyed through the Updater's instruction rather than a guaranteed token-level projection, analogous in spirit to gradient descent's incremental update principle adapted to the discrete space of natural language.

Single-Round Optimization Workflow Algorithm 1 formalizes the optimization procedure for a single training round. The process begins by sampling a minibatch of papers from the training set. For each paper, the Inference Model generates an extraction attempt using the current prompt (Phase 1), producing a fixed set of baseline outputs that will be reused throughout the optimization iterations.

The Critique Model then analyzes this minibatch-level performance data, identifying systematic prompt deficiencies that manifest across multiple papers. This aggregation over a minibatch (rather than optimizing on individual papers) is crucial for learning generalizable prompt improvements rather than overfitting to idiosyncrasies of single documents.

The textual gradient produced by the Critique Model serves as input to the Updater, which performs a constrained revision of the prompt (Phase 2). Critically, both optimization iterations analyze the same fixed baseline outputs O_0 rather than regenerating outputs after each prompt revision. Optionally, quantitative metrics computed from O_0 can be included in the critique prompt to guide the analysis, though by default the Critique Model operates on qualitative comparison alone. This design reduces computational cost by avoiding redundant Inference Model calls during the refinement loop, at the trade-off of operating on potentially outdated performance signals.

After completing both iterations, the optimized prompts P_1 and P_2 are applied to the minibatch to generate fresh outputs O_1 and O_2 (Phase 3). Finally, a winner selection mechanism determines which prompt variant will propagate to the next training round based on these newly generated outputs (Phase 4). This selection can be based purely on automated metric scores (hard evaluation) or incorporate LLM-based pairwise comparison (soft evaluation), as discussed in Section 2.3. When enforced child replacement is enabled, an optimized prompt is selected even if the initial prompt achieved higher scores.

Algorithm 1 TextGrad Single-Round Optimization

Require: Training set $\mathcal{D}_{\text{train}}$, current prompt P_0, minibatch size B, iterations K
Ensure: Optimized prompt P^* for next round

Sample minibatch $\mathcal{M} \leftarrow \text{RandomSample}(\mathcal{D}_{\text{train}}, B)$
▷ Phase 1: Generate baseline outputs with initial prompt
$\mathcal{O}_0 \leftarrow \{\text{InferenceModel}(P_0, \text{paper}) \mid \text{paper} \in \mathcal{M}\}$

▷ Phase 2: Iterative prompt refinement via textual gradients
Initialize $P \leftarrow P_0$
for $k = 1$ to K **do**
    $I_k \leftarrow \text{CritiqueModel}(P, \mathcal{M}, \mathcal{O}_0, \mathcal{M}_{\text{expert}})$ ▷ Analyze $\mathcal{O}_0$ vs expert
    $P_k \leftarrow \text{Updater}(P, I_k)$ ▷ Apply textual gradient
    Save P_k to disk
    $P \leftarrow P_k$ ▷ Chain for next iteration
end for

▷ Phase 3: Re-evaluate with optimized prompts
for $k = 1$ to K **do**
    $\mathcal{O}_k \leftarrow \{\text{InferenceModel}(P_k, \text{paper}) \mid \text{paper} \in \mathcal{M}\}$
end for

▷ Phase 4: Compute metrics and select winner
$\mathcal{S} \leftarrow \text{EvaluateMetrics}(\{\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_K\}, \mathcal{M}_{\text{expert}})$
$P^* \leftarrow \text{SelectWinner}(\{P_0, P_1, \ldots, P_K\}, \mathcal{S}, \text{duel\_mode})$
return P^*

Multi-Round Training Strategy The single-round optimization described above is embedded within an outer loop that spans multiple training rounds (typically 20-100 rounds depending on the optimization mode). Each round operates on a randomly sampled minibatch from the training set, ensuring that the prompt is exposed to diverse examples over time rather than overfitting to a fixed subset. Note that the sampling is performed without tracking or deduplication across rounds, meaning that individual papers may appear in multiple minibatches over the course of training. This design prioritizes simplicity and reproducibility (via fixed random seeds) over strict non-replacement sampling.

The winner prompt from round n serves as the initialization P_0 for round $n+1$, creating a trajectory of progressively refined prompts. This multi-round strategy allows the optimization to accumulate improvements incrementally, with early rounds addressing coarse-grained issues (e.g., output format compliance) and later rounds refining subtle edge cases (e.g., unit conversion conventions, disambiguation strategies).

Importantly, the training set is never exhausted; with 583 training papers and a minibatch size of 3, a 20-round optimization exposes the system to only 60 papers (approximately 10 percent of available data). This deliberate undersampling reduces computational cost while maintaining sufficient diversity for generalization. The held-out test set (218 papers) provides an unbiased measure of the final prompt's performance on unseen data.

C.1.3 Optimization Modes and Winner Selection Strategies

A key contribution of this work is the systematic exploration of different optimization strategies through a modular framework. Rather than committing to a single winner selection mechanism, we define eight distinct optimization modes (M1-M8) that combine complementary evaluation approaches. This design enables empirical comparison of pure metric-driven optimization versus LLM-assisted qualitative judgment, and allows us to assess the value of hierarchical competition structures.

These modes represent different combinations of configurable components: whether to include hard score feedback in critique prompts, whether to employ LLM-based judging for winner selection, and whether to enforce child replacement regardless of performance. Table S2 summarizes these configurations as a reference taxonomy for the experimental evaluation.

Framework Components Our optimization modes are constructed from four orthogonal design dimensions:

Winner Selection Basis The mechanism for determining which prompt variant advances to the next training round. Two options are available:

- Hard Score: Winner determined solely by automated metric performance via the Committee Score (Eq. 3), which averages over available metrics (default: BERTScore, ROUGE-1, METEOR; alternatively "all": BLEU, ROUGE-1, METEOR, BERTScore, ExactMatch, Jaccard). This approach is deterministic, computationally efficient, and directly optimizes the target evaluation function.
- Duel Result: Winner determined by LLM-based pairwise comparison, where a judge model evaluates which prompt produces outputs closer to expert quality. This approach captures nuanced quality dimensions not fully reflected in lexical metrics.

Duel Mode The competition structure for comparing prompt variants within a training round:

- No Duel: Only automated metrics are computed; no LLM judging is performed. Fastest option, suitable for large-scale experiments.
- Child-Only Duel: The two optimized prompts (iteration-1 vs iteration-2) compete in pairwise LLM judging to determine a child champion. When combined with hard score winner basis, this child champion then competes against the initial prompt using automated metrics. When combined with duel result winner basis, the LLM-selected child champion becomes the final winner directly.

- Hierarchical Duel: A two-stage tournament where (1) iteration-1 competes against iteration-2 via LLM judging to determine a child champion, then (2) the child champion competes against the initial prompt (parent) in a second LLM-judged comparison. This structure tests whether optimization genuinely improves over the baseline through qualitative assessment at both stages.

Feedback Inclusion Whether to provide quantitative performance metrics to the Critique Model during gradient generation:

- Excluded (default): The Critique Model receives only the textual outputs and expert annotations, forcing it to identify deficiencies through qualitative comparison alone. This approach mirrors the original TextGrad formulation and avoids potential metric hacking.
- Included: The Critique Model additionally receives aggregate hard scores for the current prompt (e.g., average BERTScore, ROUGE-1, METEOR, and Committee Score across the minibatch). This summary-level feedback can help prioritize improvement areas but may bias the optimization toward metric-driven refinements rather than semantic quality improvements.

Child Replacement Policy Whether to enforce selection of an optimized prompt regardless of performance:

- Flexible: The initial prompt can win if it outperforms both optimized variants. This conservative approach prevents performance regression.
- Enforced: One of the optimized prompts (iteration-1 or iteration-2) must be selected even if the initial prompt scored higher. This aggressive strategy forces exploration and prevents optimization stagnation.

Mode Definitions Table S2 enumerates the eight optimization modes investigated in this study. The modes are ordered by increasing complexity and computational cost.

Table S2: Optimization mode configurations. Hard score refers to automated metrics (BERTScore, ROUGE, etc.), while duel result indicates LLM-based pairwise judgment.

Mode	Winner Basis	Duel Mode	Feedback	Child Enforced
M1	Hard Score	No Duel	Excluded	No
M2	Hard Score	No Duel	Included	No
M3	Hard Score	Child-Only	Excluded	No
M4	Duel Result	Child-Only	Excluded	No
M5	Hard Score	Child-Only	Included	No
M6	Duel Result	Hierarchical	Excluded	No
M7	Duel Result	Hierarchical	Excluded	Yes
M8	Duel Result	Hierarchical	Included	Yes

Mode Rationale and Expected Behaviors

M1: Hard Score Baseline The simplest configuration, serving as a baseline for pure metric-driven optimization. The Critique Model operates in a ”black box” setting with no quantitative feedback, and winner selection relies entirely on automated metrics. This mode is fastest and most reproducible, but may struggle with quality dimensions poorly captured by lexical similarity measures.

M2: Metric-Aware Critique Extends M1 by providing hard scores to the Critique Model. This allows the critique to diagnose specific metric weaknesses (e.g., ”ROUGE is low due to missing synonyms”) but risks overfitting to metric idiosyncrasies rather than true semantic quality.

M3-M5: Child Competition with Hard Score Selection These modes introduce LLM-based judging for comparing the two optimized prompts (iteration-1 vs iteration-2), with final winner selection based on hard scores. In M3, the LLM first determines which child prompt is superior,

then this child champion competes against the parent prompt using automated metrics. M5 extends M3 by providing aggregate performance metrics to the Critique Model during gradient generation, enabling metric-aware prompt refinement. Both modes leverage LLM judgment for relative child comparison while relying on objective metrics for the critical parent-vs-child decision.

M4: Child Competition with LLM Selection M4 uses LLM-based judging throughout the selection process. The two optimized prompts compete via pairwise LLM comparison, and the winning child is directly selected as the round winner without further comparison against the parent prompt. This aggressive strategy fully trusts LLM judgment to identify improvements, bypassing metric-based validation against the baseline.

M6: Hierarchical Tournament Implements a two-stage competition where (1) the optimized prompts first compete via LLM judging to select a child champion, then (2) the child champion faces the parent prompt in a second LLM-judged comparison. This structure mirrors evolutionary selection pressure, ensuring that optimized variants must demonstrably outperform the baseline through qualitative assessment to propagate.

M7: Forced Exploration Enforces child selection even when the parent outperforms, preventing premature convergence to local optima. This mode is appropriate when early-round regressions are tolerable in pursuit of eventual breakthroughs, or when hard metrics are known to be unreliable guides.

M8: Full System Combines all enhancement components: hierarchical dueling, LLM-based winner selection, metric-aware critique, and enforced exploration. This mode represents maximum computational investment and is expected to achieve the most aggressive optimization, though at risk of instability or overfitting to the critique model's biases.

Computational Complexity The eight modes exhibit substantial variation in computational cost. Modes M1-M2 require only Inference Model inference and metric calculation, while M8 additionally invokes the Critique Model, Updater, and multiple LLM judge calls per training round. For a 20-round optimization on fold 1 (60 papers total), M1 requires approximately 120 LLM calls (60 initial + 60 re-evaluation after two iterations), whereas M8 may require over 500 calls when accounting for critique, update, and judging operations.

This cost-performance trade-off is a central empirical question: Does the additional reasoning provided by LLM-based judging and metric-aware critique justify the 4-5x increase in computational expense? Our experimental results (Section 3) provide evidence to inform this design choice for practical deployments.

C.1.4 Evaluation Metrics

Assessing the quality of automatically extracted information requires comparing system-generated outputs against expert-curated ground truth across multiple dimensions. No single metric can fully capture extraction fidelity, as structured scientific information encompasses lexical accuracy, semantic faithfulness, and field-level completeness. We therefore employ a complementary suite of six automated metrics that collectively evaluate different aspects of extraction performance.

Semantic Similarity: BERTScore Zhang et al. (2020) BERTScore measures semantic similarity by computing contextual embeddings for tokens in both the generated and reference texts, then finding optimal alignment between token pairs using cosine similarity. Unlike surface-form metrics, BERTScore captures paraphrases and synonymous expressions through its pre-trained language model backbone (DistilBERT-base-uncased in our implementation).

For each token in the candidate output, the metric identifies the most similar token in the reference based on embedding distance, and vice versa. Precision, recall, and F1 scores are computed from these alignments, with F1 serving as the primary BERTScore value. This metric is particularly valuable for scientific text where authors may describe the same concept using varied terminology (e.g., "detection limit" versus "sensitivity threshold").

The key advantage of BERTScore is its robustness to surface-form variation while maintaining interpretability through token-level alignments. However, it may occasionally conflate semantically distinct technical terms with similar contextual usage patterns, and its reliance on a fixed embedding model means it cannot adapt to domain-specific terminology beyond its training corpus.

Lexical Overlap: ROUGE-1 Lin (2004) ROUGE-1 (Recall-Oriented Understudy for Gisting Evaluation) measures unigram overlap between generated and reference texts. Originally designed for summarization evaluation, ROUGE-1 quantifies how much of the reference content is captured in the system output, emphasizing recall over precision.

The metric operates by stemming all words to their root forms (using the Porter stemmer), then computing the ratio of overlapping unigrams to total unigrams in the reference. ROUGE-1 is particularly sensitive to completeness: a system that omits critical fields or values will incur steep penalties. However, it is agnostic to word order and semantic nuance, treating ”5 ppb” and ”5 parts per billion” as entirely distinct despite equivalent meaning.

We report the ROUGE-1 F1 score, which balances recall against precision, preventing trivial optimization strategies that maximize overlap by copying entire passages verbatim.

Alignment-Based Evaluation: METEOR Banerjee & Lavie (2005) METEOR (Metric for Evaluation of Translation with Explicit ORdering) extends simple n-gram matching by incorporating stemming, synonymy, and paraphrase recognition. The metric aligns words between candidate and reference texts using multiple matching stages: exact match, stem match, and synonym match (based on WordNet).

After establishing alignments, METEOR computes precision and recall, then combines them into a harmonic mean weighted toward recall. Additionally, METEOR applies a fragmentation penalty that rewards contiguous matches over scattered alignments, implicitly capturing some notion of fluency and coherence.

For structured extraction tasks, METEOR’s synonym awareness is particularly valuable when dealing with chemical nomenclature or equivalent technical expressions. For instance, ”phosphate buffered saline” and ”PBS” would receive partial credit despite sharing no surface tokens. This makes METEOR more lenient than BLEU while remaining more conservative than pure semantic embeddings.

N-Gram Precision: BLEU Papineni et al. (2002) BLEU (Bilingual Evaluation Understudy) measures precision of n-gram matches between generated and reference texts, with a brevity penalty to discourage pathologically short outputs. Originally designed for machine translation, BLEU emphasizes exactness over recall: a system that produces highly accurate but incomplete extractions can still achieve respectable BLEU scores.

We employ the smoothing variant (method 1) to handle cases where higher-order n-grams may have zero matches, preventing undefined scores on short outputs. BLEU’s primary limitation for extraction tasks is its insensitivity to semantic equivalence; functionally correct variations in phrasing or unit representation receive no credit unless they exhibit surface-level overlap.

Despite these limitations, BLEU remains a useful signal for detecting prompt refinements that improve format consistency and terminology standardization.

Structured Field Accuracy: Exact Match Exact Match evaluates field-level correspondence in the structured JSON outputs. For each of the eight target fields (sensor type, detect target, detection limits, etc.), we compare the extracted value against the expert annotation and compute the percentage of fields with character-exact agreement.

This metric is unforgiving: ”5.0 ppb” and ”5 ppb” are considered distinct, as are ”ammonia” and ”NH3”. Exact Match therefore captures the most stringent notion of extraction correctness, rewarding systems that perfectly replicate expert formatting conventions and terminology choices.

The metric’s harshness makes it a strong indicator of prompt refinement quality. Improvements in Exact Match signal that the system has learned to standardize units, resolve abbreviations consistently, and adopt expert-preferred field values rather than extracting raw text snippets.

Set Similarity: Jaccard Index The Jaccard Index measures token-level set similarity as the ratio of intersecting tokens to the union of tokens across generated and reference outputs. After tokenization (splitting on whitespace and punctuation), we compute:

$$\text{Jaccard} = \frac{|\text{tokens}_{\text{gen}} \cap \text{tokens}_{\text{ref}}|}{|\text{tokens}_{\text{gen}} \cup \text{tokens}_{\text{ref}}|}$$

This metric is order-agnostic and emphasizes coverage: outputs that mention the same entities and values as the reference, even in different arrangements, receive high scores. Jaccard is less sensitive to verbosity than ROUGE (since it uses union normalization) and less sensitive to phrasing than BLEU (since it ignores n-gram structure).

For multi-field structured outputs, Jaccard provides a complementary signal to Exact Match. While Exact Match requires perfect field-level agreement, Jaccard rewards partial matches and gives credit for extracting most of the relevant information even when formatting details differ.

Metric Complementarity and Composite Scoring The six base metrics above capture orthogonal quality dimensions: BERTScore for semantic fidelity, ROUGE for completeness, METEOR for flexible lexical alignment, BLEU for precision, Exact Match for structural correctness, and Jaccard for entity coverage. No single metric is universally superior; each exhibits distinct failure modes and biases.

To support winner selection decisions, we additionally compute a Committee Score by averaging BERTScore, ROUGE-1, and METEOR. This composite metric (yielding seven total scores per prompt variant) provides a balanced summary emphasizing the most reliable evaluation dimensions. During optimization, these quantitative scores may optionally be provided to the Critique Model to inform its analysis, though by default the critique operates on qualitative output comparison alone.

All final experimental results report the complete metric profile (six base metrics plus committee score), enabling comprehensive assessment of prompt quality improvements across multiple evaluation axes. This multi-metric approach guards against overfitting to any single measure and provides a more robust characterization of extraction fidelity.

C.1.5 Experimental Protocol

Training Phase The prompt optimization process operates over multiple training rounds; we sweep $\{20, 40, 60, 100\}$ rounds for every configuration rather than tying round count to a specific mode. Each round executes the single-round workflow described in Algorithm 1: a minibatch of 3 papers is randomly sampled from the training set, the Inference Model generates outputs using the current prompt, the Critique Model analyzes performance gaps, and the Updater produces a revised prompt. Winner selection (via hard scores, LLM judging, or hierarchical dueling) determines which prompt variant propagates to the subsequent round.

This multi-round strategy ensures exposure to diverse examples over time while avoiding exhaustive iteration over the entire training set. With 583 training papers available and a minibatch size of 3, the round-count sweep spans 60–300 unique paper encounters per trajectory (with possible repeats due to random sampling). This deliberate undersampling reduces computational cost while maintaining sufficient diversity for generalization.

Random sampling is controlled by a fixed seed per experimental run to ensure reproducibility. Each training round operates on a freshly sampled minibatch, preventing overfitting to a static subset of examples and allowing the optimization to encounter different edge cases and reporting conventions throughout the training trajectory.

Test Phase Evaluation Upon completion of multi-round training, the final optimized prompt is evaluated on the held-out test set (261 papers per fold). The Inference Model processes each test paper using both the original human-written prompt and the optimized prompt, generating paired outputs for direct comparison.

For each test paper, we compute the six-metric evaluation suite (BERTScore, ROUGE-1, METEOR, BLEU, Exact Match, Jaccard) by comparing system outputs against expert annotations. Aggregate

statistics (mean, standard deviation, minimum, maximum) are reported across all test papers for each metric, along with per-metric improvement ratios between optimized and baseline prompts.

Critically, no information from test set papers influences the optimization process. Test set DOIs, full texts, and expert annotations remain isolated from training, ensuring that reported performance reflects genuine generalization to unseen scientific literature rather than memorization of training examples.

Experimental Design and Replication To assess optimization robustness and account for stochastic variation, we employ a rigorous replication strategy combining multiple independent runs with cross-validation:

Independent Replications For each configuration (optimization mode, model choice, hyperparameters), we conduct 5 independent rollouts starting from the same human-authored initial prompt. Seeds are fixed for reproducibility, but minibatch sampling order differs across rollouts, yielding distinct optimization trajectories even under identical settings. Final test set performance is aggregated across these 5 runs, with mean and standard deviation reported to characterize typical performance and run-to-run variability.

Cross-Validation The 3-fold data partitioning enables assessment of methodology-level variation beyond single-fold idiosyncrasies. By repeating experiments across all three folds, we obtain performance estimates that account for dataset composition effects. Results tables report both per-fold performance (to demonstrate consistency) and cross-fold aggregates (to summarize overall effectiveness).

This factorial design (5 replications × 3 folds = 15 total runs per configuration) provides statistical rigor for comparing optimization modes and model choices.

C.2 Method and Technical Details for "Twin"

C.2.1 Material Descriptor Representation

Field-effect transistor (FET) sensors incorporate diverse material classes across functional components, including channel semiconductors, dielectric layers, electrodes, substrates, and surface functionalizations. To enable machine learning on these heterogeneous materials, we develop a unified descriptor framework that represents each material as a fixed-dimensional numerical vector comprising 25 macroscopic properties and a 320-dimensional fingerprint embedding.

Building upon prior work on spiking graph neural networks for FET sensor modeling Ferreira et al. (2025), which demonstrates the effectiveness of material descriptors for cheminformatics and materials informatics, we extend the framework to accommodate the broader scope of the present dataset. While the original implementation covered inorganic compounds, organic molecules, and polymers, the expanded corpus from the Text phase now includes biosensors with protein-based recognition elements and nucleic acid aptamers. Accordingly, we introduce two additional material categories—biomolecules (proteins) and nucleic acids (DNA/RNA)—leveraging state-of-the-art biological language models for their representation.

This *Cross-Domain Material Fingerprinting* approach transforms free-text material names extracted during the Text phase into dense numerical vectors, substantially enriching the structured information available for the Twin phase. Each publication may yield one or more experimental records, and our encoding scheme ensures that the full chemical and biological diversity across all material-bearing fields is captured in a unified, model-ready format.

Material Classification and Descriptor Sources Materials extracted from FET sensor literature were classified into five categories based on chemical structure: inorganic compounds, organic molecules, polymers, biomolecules (proteins), and nucleic acids (DNA/RNA). Each category employs domain-specific descriptors optimized for capturing physicochemical properties relevant to sensing performance (Table S3).

For inorganic materials (Table S4), we retrieved formation energies, band gaps, elastic moduli, dielectric constants, and effective masses from the Materials Project database via the OPTIMADE

Table S3: Material descriptor sources and feature dimensions by material category.

Category	Macroproperties (25D)	Fingerprint (320D)	Database
Inorganic	14 physical properties + 11 one-hot encoded (crystal system, electronic class)	MAGPIE composition embedding	Materials Project Jain et al. (2013)
Molecules	Electronic, polarity, molecular size, flexibility, and complexity descriptors	Morgan circular fingerprint	PubChem National Center for Biotechnology Information (2024), ChEMBL European Bioinformatics Institute (2024)
Polymers	7 bulk properties (thermal, mechanical, electrical) + 18 monomer descriptors	Morgan fingerprint of repeat unit	PubChem National Center for Biotechnology Information (2024)
Biomolecules	Sequence length, physicochemical properties, amino acid composition, secondary structure	ESM-2 protein embedding	UniProt UniProt Consortium (2015), ESM-2 Lin et al. (2023)
DNA/RNA	Sequence composition, GC content, melting temperature, thermodynamic stability	DNABERT-S embedding	DNABERT-S Zhou et al. (2025)

API. Crystal system (7 classes) and electronic classification (4 classes: metal, semiconductor, insulator, other) were one-hot encoded to complete the 25-dimensional macroproperty vector. The 320-dimensional fingerprint was derived from MAGPIE (Materials-Agnostic Platform for Informatics and Exploration) composition-based features.

Organic molecules (Table S5) and polymer repeat units (Table S6) are characterized using molecular descriptors from PubChem and ChEMBL, including topological polar surface area (TPSA), partition coefficients (XLogP), hydrogen bond donor/acceptor counts, and 3D pharmacophore features. Morgan circular fingerprints (radius 2, 256 bits) are computed using RDKit and zero-padded to 320 dimensions.

Protein descriptors (Table S7) are computed from amino acid sequences retrieved from UniProt, including isoelectric point, instability index, GRAVY hydropathicity, and amino acid composition fractions. The 320-dimensional embedding is generated using the ESM-2 protein language model (650M parameters), with the mean-pooled representation extracted from the final hidden layer.

Nucleic acid sequences (Table S8) are characterized by length, GC content, purine/pyrimidine ratios, and predicted melting temperatures. Embeddings are computed using DNABERT-S, a species-aware DNA language model, yielding 256-dimensional vectors that are zero-padded to 320 dimensions. Table S9 summarizes the fingerprint and embedding sources for all material categories.

Feature Aggregation Each FET sensor record contains up to 11 material-bearing fields: channel, dielectric layer, gate electrode, source electrode, drain electrode, substrate, probe material, surface functionalization, detection target, test medium, and annealing atmosphere. For each field, the corresponding material is mapped to its descriptor vector (25 + 320 = 345 dimensions). Composite materials (e.g., "zinc oxide/graphene") are decomposed into individual components, and their descriptor vectors are averaged element-wise.

The complete feature representation for each sensor record is constructed by concatenating descriptors from all 11 material fields, yielding a $(11 \times 345) = 3795$-dimensional material descriptor vector. Combined with 7 scalar device parameters (detection limits, response times, etc.) and categorical encodings, the final feature dimension is 3869.

C.2.2 Heterogeneous Graph Representation

To capture the complex multi-component architecture of FET sensors, we represent each device as a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where nodes $\mathcal{V}$ correspond to functional components and edges $\mathcal{E}$ encode physical interactions between them. This representation directly mirrors the physical structure of FET sensors: discrete material components (channel, electrodes, dielectrics) interact

Table S4: Macroscopic properties for inorganic materials (25 dimensions).

#	Property	Unit	Description
Thermodynamic & Electronic (14 dimensions)			
1	formation_energy_per_atom	eV/atom	DFT formation energy from elemental states
2	energy_above_hull	eV/atom	Energy distance to convex hull (0 = stable)
3	band_gap	eV	Electronic band gap
4	density	g/cm^3	Mass density
5	epsilon_x	–	Dielectric constant (x-axis)
6	epsilon_y	–	Dielectric constant (y-axis)
7	epsilon_z	–	Dielectric constant (z-axis)
8	dielectric_total	–	Total static dielectric constant
9	k_vrh	GPa	Bulk modulus (Voigt-Reuss-Hill)
10	g_vrh	GPa	Shear modulus (Voigt-Reuss-Hill)
11	poisson_ratio	–	Poisson's ratio
12	me_avg	m_0	Average electron effective mass
13	mh_avg	m_0	Average hole effective mass
14	magnetization	μ_B/f.u.	Total magnetization
Crystal System One-Hot (7 dimensions)			
15	cs_cubic	0/1	Cubic crystal system
16	cs_hexagonal	0/1	Hexagonal crystal system
17	cs_orthorhombic	0/1	Orthorhombic crystal system
18	cs_tetragonal	0/1	Tetragonal crystal system
19	cs_trigonal	0/1	Trigonal crystal system
20	cs_monoclinic	0/1	Monoclinic crystal system
21	cs_triclinic	0/1	Triclinic crystal system
Electronic Classification One-Hot (4 dimensions)			
22	eg_metal	0/1	Metallic (band gap = 0)
23	eg_semiconductor	0/1	Semiconductor (0 ¡ band gap ¡ 3 eV)
24	eg_insulator	0/1	Insulator (band gap $\geq$ 3 eV)
25	eg_other	0/1	Other/unknown classification

Table S5: Macroscopic properties for organic molecules (25 dimensions).

#	Property	Unit	Description
Electronic Properties (5 dimensions)			
1	Charge	e	Formal molecular charge
2	aromatic_rings	count	Number of aromatic ring systems
3	FeatureRingCount3D	count	3D aromatic ring pharmacophore features
4	FeatureCationCount3D	count	Cationic center count
5	FeatureAnionCount3D	count	Anionic center count
Polarity & Surface Interactions (6 dimensions)			
6	TPSA	$Å^2$	Topological polar surface area
7	XLogP	log units	Octanol-water partition coefficient
8	HBondDonorCount	count	Hydrogen bond donor count
9	HBondAcceptorCount	count	Hydrogen bond acceptor count
10	FeatureDonorCount3D	count	3D H-bond donor features
11	FeatureAcceptorCount3D	count	3D H-bond acceptor features
Molecular Size & Shape (6 dimensions)			
12	MolecularWeight	Da (g/mol)	Molecular weight
13	HeavyAtomCount	count	Non-hydrogen atom count
14	Volume3D	$Å^3$	Van der Waals volume
15	XStericQuadrupole3D	–	X-direction steric quadrupole moment
16	YStericQuadrupole3D	–	Y-direction steric quadrupole moment
17	ZStericQuadrupole3D	–	Z-direction steric quadrupole moment
Flexibility & Dynamics (3 dimensions)			
18	RotatableBondCount	count	Rotatable single bond count
19	EffectiveRotorCount3D	count	Effective rotor count in 3D
20	ConformerModelRMSD3D	Å	Conformer ensemble RMSD
Complexity & Drug-likeness (5 dimensions)			
21	Complexity	–	Bertz complexity score
22	FeatureHydrophobeCount3D	count	Hydrophobic pharmacophore features
23	FeatureCount3D	count	Total pharmacophore features
24	qed_weighted	0–1	Quantitative drug-likeness score
25	np_likeness_score	-5 to $+5$	Natural product likeness

Table S6: Macroscopic properties for polymers (25 dimensions).

#	Property	Unit	Description
Bulk Polymer Properties (7 dimensions)			
1	molar_mass_repeat_g_mol	g/mol	Repeat unit molecular weight
2	density_g_cm3	g/cm^3	Bulk density
3	Tg_C	°C	Glass transition temperature
4	Td_C	°C	Decomposition temperature
5	Tm_C	°C	Melting temperature
6	dielectric_constant	–	Relative permittivity
7	youngs_modulus_GPa	GPa	Elastic modulus
Monomer Descriptors (18 dimensions)			
8	Charge	e	Monomer formal charge
9	TPSA	Å^2	Polar surface area
10	HBondDonorCount	count	H-bond donors
11	HBondAcceptorCount	count	H-bond acceptors
12	MolecularWeight	Da	Monomer molecular weight
13	HeavyAtomCount	count	Non-H atom count
14	RotatableBondCount	count	Rotatable bonds
15	XLogP	log units	Lipophilicity
16	Complexity	–	Molecular complexity
17	FeatureRingCount3D	count	3D ring count
18	FeatureCationCount3D	count	Cationic features
19	FeatureAnionCount3D	count	Anionic features
20	FeatureDonorCount3D	count	3D H-bond donors
21	FeatureAcceptorCount3D	count	3D H-bond acceptors
22	Volume3D	Å^3	Monomer volume
23	EffectiveRotorCount3D	count	Effective rotors
24	FeatureHydrophobeCount3D	count	Hydrophobic features
25	aromatic_rings	count	Aromatic ring count

through well-defined mechanisms (carrier transport, capacitive coupling, surface chemistry) that determine sensing performance.

Node Types and Features We define 16 node types organized into three functional categories:

- **Device components** (9 nodes): channel, gate (top/bottom), dielectric layer (top/bottom), floating gate, source, drain, and substrate. These represent the transistor's electrical architecture.
- **Sensing components** (5 nodes): surface functionalization, probe material, detection target, test medium, and electrolyte. These capture the biochemical sensing interface.
- **Process/condition** (2 nodes): annealing process parameters and operating conditions (temperature, pH, etc.).

Each node's feature vector is constructed from the material descriptors described in Section C.2.1. For material-bearing nodes, we concatenate the 25-dimensional macroproperty vector with a 25-dimensional binary mask indicating feature availability, plus a scalar for the number of constituent materials (for composites), yielding a 51-dimensional node feature. The 320-dimensional fingerprints are processed separately through a dedicated neural branch (Section C.2.4).

For nodes with missing materials, we apply zero-masking: the feature vector is set to zeros while the mask indicates complete absence, allowing the model to distinguish between missing data and zero-valued properties.

Edge Types and Physical Semantics Rather than using a fully connected graph, we define six edge types that encode distinct physical interactions governing FET sensor operation. This physics-informed topology ensures that message passing in the graph neural network follows actual charge transport, electrostatic coupling, and chemical interaction pathways:

Table S7: Macroscopic properties for biomolecules/proteins (25 dimensions).

#	Property	Unit	Description
Sequence Properties (3 dimensions)			
1	sequence_length	aa	Amino acid count
2	molecular_weight_kDa	kDa	Protein molecular weight
3	isoelectric_point	pH	Isoelectric point (pI)
Physicochemical Properties (5 dimensions)			
4	aromaticity	0–1	Aromatic residue fraction (Phe, Trp, Tyr)
5	instability_index	–	Protein stability index (>40 = unstable)
6	gravy	–	Grand average hydropathicity
7	charge_at_pH7	e	Net charge at pH 7.0
8	charge_at_pH5	e	Net charge at pH 5.0
Amino Acid Composition (9 dimensions)			
9	aromatic_fraction	%	Phe, Tyr, Trp fraction
10	aliphatic_fraction	%	Ala, Val, Leu, Ile fraction
11	polar_fraction	%	Ser, Thr, Asn, Gln fraction
12	charged_fraction	%	Asp, Glu, Lys, Arg fraction
13	positive_fraction	%	Lys, Arg, His fraction
14	negative_fraction	%	Asp, Glu fraction
15	cysteine_count	count	Disulfide-forming residues
16	proline_count	count	Structure-breaking residues
17	glycine_fraction	%	Flexible residue fraction
Secondary Structure (3 dimensions)			
18	helix_fraction	0–1	α-helix propensity
19	turn_fraction	0–1	β-turn propensity
20	sheet_fraction	0–1	β-sheet propensity
Spectroscopic & Physical (5 dimensions)			
21	extinction_coefficient_reduced	$M^{-1}cm^{-1}$	Extinction coeff. (reduced Cys)
22	extinction_coefficient_oxidized	$M^{-1}cm^{-1}$	Extinction coeff. (oxidized Cys)
23	average_flexibility	–	B-factor derived flexibility
24	tiny_fraction	%	Gly, Ala, Ser, Cys, Thr fraction
25	large_fraction	%	Phe, Ile, Lys, Leu, Met, Arg, Trp, Tyr fraction

Table S8: Macroscopic properties for DNA/RNA sequences (25 dimensions).

#	Property	Unit	Description
Sequence Composition (9 dimensions)			
1	length	nt	Nucleotide count
2	gc_content	%	Guanine + Cytosine fraction
3	at_content	%	Adenine + Thymine/Uracil fraction
4	a_count	count	Adenine count
5	c_count	count	Cytosine count
6	g_count	count	Guanine count
7	t_count	count	Thymine count
8	u_count	count	Uracil count (RNA only)
9	purine_pyrimidine_ratio	–	(A+G)/(C+T/U) ratio
Thermodynamic Properties (3 dimensions)			
10	tm_celsius	°C	Melting temperature
11	delta_g_kcal_mol	kcal/mol	Folding free energy
12	complexity	–	Sequence complexity index
Sequence Features (4 dimensions)			
13	longest_homopolymer	nt	Longest homopolymer run
14	cpg_count	count	CpG dinucleotide count
15	gc_skew	–	(G−C)/(G+C) strand asymmetry
16	at_skew	–	(A−T)/(A+T) strand asymmetry
Dinucleotide Frequencies (8 dimensions)			
17	dinuc_CG	–	CG dinucleotide frequency
18	dinuc_GC	–	GC dinucleotide frequency
19	dinuc_AT	–	AT dinucleotide frequency
20	dinuc_TA	–	TA dinucleotide frequency
21	dinuc_GG	–	GG dinucleotide frequency
22	dinuc_CC	–	CC dinucleotide frequency
23	dinuc_AA	–	AA dinucleotide frequency
24	dinuc_TT	–	TT dinucleotide frequency
Type Indicator (1 dimension)			
25	is_rna	0/1	RNA indicator (1) vs DNA (0)

Table S9: 320-dimensional fingerprint/embedding sources by material category.

Category	Method	Native Dim	Final Dim	Reference
Inorganic	MAGPIE composition	132	320 (zero-padded)	Ward et al., 2016
Molecules	Morgan fingerprint (r=2)	256	320 (zero-padded)	RDKit
Polymers	Morgan FP of repeat unit	256	320 (zero-padded)	RDKit
Biomolecules	ESM-2 mean pooling	320	320	Lin et al., 2023
DNA/RNA	DNABERT-S embedding	256	320 (zero-padded)	Zhou et al., 2024

1. **Electrical edges**: Connect source, drain, and substrate to the channel, representing carrier transport pathways.
2. **Capacitive edges**: Model gate-channel coupling through the dielectric stack. The topology varies by device design:
   - Standard: gate ↔ dielectric ↔ channel
   - Remote: gate ↔ electrolyte ↔ channel
   - Floating gate: gate ↔ dielectric ↔ floating gate ↔ dielectric ↔ channel
   - Dual gate: parallel top and bottom gate pathways
3. **Chemical edges**: Encode the sensing chain from channel through surface functionalization and probe to the detection target, plus target-medium and medium-channel interactions. We additionally include expert-recommended edges for probe-medium interactions (affecting probe stability and conformation), direct target-channel sensing (relevant for small molecules bypassing Debye screening), and probe-channel charge transfer.
4. **Process edges**: Directed edges from annealing parameters to the channel, encoding thermal treatment effects on material properties.
5. **Condition edges**: Directed edges from operating conditions to channel, gate, and test medium nodes.
6. **Environment edges**: Connect electrolyte to test medium in remote-gate designs, modeling the shared solution environment between the gating electrolyte and the sensing medium.

All chemical, electrical, capacitive, and environment edges are bidirectional to allow message passing in both directions, while process and condition edges are unidirectional (conditions affect components, not vice versa).

C.2.3 Physics-aware Data Augmentation

Following the data protocol established in prior work Ferreira et al. (2025), we address the inherent size limitation of FET sensor datasets extracted from literature. To improve model generalization without introducing physically implausible samples, we develop a physics-aware data augmentation strategy that perturbs device parameters within experimentally reasonable bounds while enforcing domain constraints.

Label-safe Augmentation via Wide Classification Bins A key insight enabling aggressive augmentation is that our classification bins span multiple orders of magnitude. For example, the lower detection limit (LDL) Class 1 spans from 10^{-6} to 10^{0} ppm—a $10^{6}\times$ range. This wide binning provides a substantial safety margin: perturbations causing less than $10\times$ change in the target metric will not alter the class label. We exploit this property by designing perturbations that induce at most $\sim$20–25% changes in device performance metrics, well within the label-safe regime.

The physical basis for each perturbation's impact can be estimated from device physics. For instance, gate dielectric thickness d affects transconductance as $g_m \propto C_{ox} \propto 1/d$, so a 25% thickness perturbation yields approximately 20% change in sensitivity—safely within the bin boundaries. Similarly, pH perturbations of ± 0.6–0.8 units induce surface potential changes of $\Delta\psi \approx 59\ \text{mV} \times \Delta\text{pH} \approx 35$–$47$ mV, which have negligible impact on detection limits for non-pH sensors.

Continuous Parameter Perturbation For each original sample, we generate augmented variants by applying bounded perturbations to continuous parameters. Perturbation factors are sampled from truncated Gaussian distributions centered at unity (for multiplicative noise) or zero (for additive noise), with bounds reflecting typical experimental uncertainties (Table S10).

Physical Constraints To ensure augmented samples remain physically realizable, we enforce constraints that reflect fundamental physical laws:

- **pH range**: Values are explicitly clipped to $[0, 14]$, corresponding to the thermodynamic limits of proton activity in aqueous solutions ($[\text{H}^+] = 1$ M to 10^{-14} M).

Table S10: Perturbation bounds for physics-aware data augmentation.

Parameter	Perturbation Type	Bounds	Physical Basis
Temperature (gas sensors)	Multiplicative	$\times 0.92$–1.08	$\pm 8\%$ calibration uncertainty
Temperature (biosensors)	Additive	± 4°C	Physiological range variation
pH (biosensors)	Additive	± 0.6	Buffer preparation tolerance
pH (liquid sensors)	Additive	± 0.8	Environmental variation
Dielectric thickness	Multiplicative	$\times 0.85$–1.15	$\pm 15\%$ deposition variation
Substrate thickness	Multiplicative	$\times 0.85$–1.15	Wafer tolerance
Annealing temperature	Multiplicative	$\times 0.92$–1.08	Furnace calibration
Annealing time	Multiplicative	$\times 0.85$–1.15	Process window

- **Temperature and thickness**: These parameters use multiplicative perturbation factors ($\times 0.85$–1.15), which inherently preserve the sign of the original positive values, ensuring temperatures remain above absolute zero and layer dimensions remain positive.

These constraints are critical: without them, augmentation can generate samples that violate fundamental physical laws (e.g., negative pH, sub-zero temperatures), leading models to learn spurious correlations from impossible device configurations.

Discrete Augmentation Beyond continuous perturbations, we apply discrete transformations that exploit known physical symmetries and equivalences in FET sensor design:

- **Source/drain interchange**: In the absence of asymmetric doping or geometry, source and drain electrodes are physically interchangeable—the distinction is purely a measurement convention defining current direction. Our dataset confirms this symmetry: 98.5% of samples have identical source and drain materials.
- **Gate stack flip**: In dual-gate and floating-gate architectures, the capacitive coupling between gate stacks and the channel follows symmetric physics. Top and bottom gate/dielectric layer orderings can be reversed without altering the fundamental electrostatic control mechanism.
- **Inert atmosphere substitution**: For carrier gases (in gas sensors) and annealing atmospheres, chemically inert species (N_2, Ar, He, Ne) serve identical purposes—preventing oxidation and providing a controlled environment. We replace nitrogen-based atmospheres with other inert gases using their actual material descriptors from PubChem.

These discrete augmentations generated 3,176 additional samples, each grounded in established semiconductor device physics rather than arbitrary permutations.

Material Descriptor Perturbation Material properties retrieved from databases carry inherent uncertainties from experimental measurements and computational predictions. To improve model robustness against these uncertainties, we perturb material representations:

- **Macroproperty noise**: Gaussian noise ($\sigma = 0.08$, i.e., $\pm 8\%$) is added to the 25-dimensional macroproperty vectors, reflecting typical measurement and prediction uncertainties in database-reported material properties.
- **Fingerprint perturbation**: For binary Morgan fingerprints, we apply stochastic bit flips with probability 0.08. For continuous embeddings (ESM-2, DNABERT-S), additive Gaussian noise is applied.

Baseline Augmentation Strategies To validate the importance of physics-aware constraints, we compare against two random augmentation baselines (Table S11):

- **Same-magnitude random**: Uses identical perturbation bounds as physics-aware augmentation but removes physical constraints (allowing pH < 0, negative temperatures) and discrete augmentations. This ablation isolates the contribution of domain constraints from perturbation magnitude.

Table S11: Comparison of augmentation strategies. Physics-aware augmentation enforces physical constraints and includes discrete transformations, while random baselines ablate these components.

Property	Physics	Same-mag	Destruct.
Perturbation mag.	$\pm$8–15%	$\pm$8–15%	$\pm$80–200%
Physical constraints	✓	×	×
Discrete augment.	✓ (3176)	×	×
Material noise	$\pm$8%	$\pm$8%	$\pm$50%
Variants/sample	2	4	6

- **Destructive random**: Applies large perturbations ($\pm$80–200%) without constraints, representing naive augmentation that ignores physical plausibility.

Empirically (5-fold CV on LDL/UDL/Sensitivity), all three augmentations achieve high accuracy on the augmented test splits ($\geq$ 0.95), but only the physics-aware strategy transfers to held-out original data. Physics-aware augmentation yields Original accuracy of roughly $0.88/0.85/0.92$ on the three tasks, whereas random and same-magnitude baselines collapse to $\sim 0.72/0.67/0.79$ and $\sim 0.73/0.67/0.79$, respectively. This gap demonstrates that domain constraints—not perturbation magnitude—are critical for preserving the real-device distribution.

Rationale. Models trained on physics-aware augmentation generalize to the held-out original distribution, while those trained on magnitude-matched random noise do not. This indicates the GNN is learning physically meaningful device–sensing relationships instead of overfitting to synthetic artifacts. Domain knowledge in the augmentation (valid ranges, symmetric transformations, inert substitutions) constrains message passing to realistic charge/chemical pathways and keeps fingerprint fields semantically aligned with real sensors. Unconstrained noise allows shortcut features that break on real devices. The strong Original-set gains therefore substantiate both the augmentation procedure and the downstream model's use of domain priors.

Evaluation Protocol To validate that augmented data can serve as a legitimate training source, we adopt an evaluation protocol following prior work on spiking GNNs for FET sensors Ferreira et al. (2025). The augmented dataset is split 80:20 for training and testing, while the *entire original dataset* is reserved as a held-out test set.

The primary evaluation metric is **held-out accuracy**: classification accuracy on the complete original dataset. This metric measures whether a model trained on augmented data has learned patterns that transfer to real experimental samples. If the augmentation strategy preserves the underlying physical relationships, the model should generalize to original data; if augmentation introduces artifacts, the model will fail on real experiments.

Comparative experiments between physics-aware augmentation and random baselines (presented in Section D) demonstrate that physics-aware augmentation substantially outperforms random perturbation strategies on held-out accuracy. This validates that physics-aware augmented data captures genuine structure in the FET sensor design space and can be reliably used for training predictive models. Consequently, the held-out accuracy reported for downstream model benchmarking provides a meaningful measure of generalization to real-world sensing experiments.

C.2.4 Graph Neural Network Training

Graphs built in Section C.2.2 (physics-informed augmented set as default) are used to train multi-class classifiers for three tasks: lower detection limit (LDL), upper detection limit (UDL), and sensitivity. Each graph carries both the "augmented" label (training distribution) and the original unperturbed label (held-out robustness evaluation).

Classification Task Definition Following prior work Ferreira et al. (2025), we discretize continuous performance metrics into three ordinal classes to enable robust classification despite measurement uncertainties inherent in literature-reported values. Class boundaries are defined on logarithmic scales to span multiple orders of magnitude (see Figure S17):

- **LDL (Lower Detection Limit)**: Class 0 ($< 10^{-6}$ ppm, best), Class 1 (10^{-6}–10^{0} ppm), Class 2 ($> 10^{0}$ ppm, worst). Lower values indicate better sensitivity to trace analytes.
- **UDL (Upper Detection Limit)**: Class 0 ($< 10^{0}$ ppm, worst), Class 1 (10^{0}–10^{4} ppm), Class 2 ($> 10^{4}$ ppm, best). Higher values indicate broader dynamic range.
- **Sensitivity**: Class 0 (low), Class 1 (medium), Class 2 (high, best). Boundaries depend on the sensitivity definition (e.g., $\Delta R/R_0$, $\Delta I/I_0$) and are normalized per metric type.

This wide binning provides label stability under data augmentation: perturbations causing $< 10\times$ metric changes remain within the same class, enabling physics-aware augmentation without label corruption.

Model Architecture Our primary model is a residual heterogeneous GNN with two branches:

- **GNN branch**: For each edge type (s, r, d) in the heterogeneous graph, we use relation-specific convolution operators aggregated across relations (Eq. 4). Self-loops are optionally added at the graph level; we disable internal self-loop insertion for GATv2Conv. We apply one hetero-convolution layerSchlichtkrull et al. (2018a) with jump-knowledge residuals Xu et al. (2018) and GCNII-styleChen et al. (2020) re-parameterization (Eq. 5), followed by LayerNorm, ReLU, and an attention-based readout. Node inputs are 51-dimensional vectors composed of 25 macroproperties + 25 missing-data masks + 1 material-count scalar (Section C.2.1); the 320-dimensional fingerprints are *not* fed here.
- **Fingerprint branch**: A small MLP that processes the 320-dimensional fingerprints for five fields most relevant to the chemical sensing interface: channel (transducer), detect_target (analyte), probe_material (recognition element), test_medium (sample matrix), and surface_functionalization (interface modifier). Its output is fused with the GNN logits through a learnable gate, initialized to a fixed value and updated in the second training stage.

This architecture preserves physical message-passing paths while allowing complementary fingerprint evidence to modulate the prediction.

Training Protocol We adopt a two-stage schedule: (1) pre-train the GNN branch for E_{gnn} epochs; (2) freeze the GNN and train the fingerprint gate/MLP for E_{fp} epochs. For each fold we report four metrics (accuracy, macro-F1, precision, recall) on (a) augmented train, (b) augmented test, and (c) original held-out graphs; final results are mean $\pm$ std across 5-fold stratified cross-validation.

Key hyperparameters (hidden dimension, dropout, learning rate, weight decay, gate initialization, E_{gnn}, E_{fp}) are selected via Bayesian optimization (BO) with adaptive level-set estimation Zhang et al. (2023), which filters for a high-confidence region of interest (ROI) as a superlevel-set of a Gaussian process surrogate, around a manually tuned seed configuration. The anchor used for all ablations is: hidden_dim=128, dropout=0.2, lr=1×10^{-4}, weight decay=5×10^{-5}, GCNII $\alpha = 0.15$, pretrain epochs $E_{\text{gnn}} = 40$, FP epochs $E_{\text{fp}} = 20$, gate init=0.25, self-loops on. All ablation variants reuse this configuration for fairness. Batches are class-stratified. Optimization uses AdamW Loshchilov & Hutter (2019) with weighted cross-entropy loss (Eq. 7), implemented via `nn.CrossEntropyLoss` which includes the softmax internally. Unless otherwise stated, batch size is 32.

Controls and Ablations To attribute gains, we conduct 14 controlled ablation variants organized into six categories:

- **Architecture components**: toggling jump-knowledge (JK) connections and attention-based readout—full model (JK+Attention), w/o JK, w/o Attention, w/o both.
- **Branch isolation**: GNN-only (no fingerprint branch) and fingerprint-only (FP-only, no graph convolution) to quantify each branch's contribution.
- **Alternative FP encoders**: replacing the default MLP fingerprint branch with Transformer or Spiking Neural Network (SNN) encoders, sharing anchor hyperparameters.
- **GNN depth**: varying the number of hetero-convolution layers (0, 1 [default], 2) to assess depth effects.

- **GCNII residual strength**: varying the initial residual coefficient $\alpha \in \{0.05, 0.15, 0.25\}$; $\alpha = 0.15$ is the anchor default.
- **Training variants**: toggling self-loops in message passing, and enabling inverse-frequency class weights for imbalanced labels.

Additionally, we compare against 17 tabular baselines trained on flattened descriptors, each tuned via the same ROI-based BO protocol: Random Forest (RF), Extremely Randomized Trees (ExtraTrees), Decision Tree (DT), XGBoost, LightGBM, Gradient Boosting Decision Tree (GBDT), CatBoost, AdaBoost, Logistic Regression (LogReg), Linear Support Vector Machine (LIN_SVM), Radial Basis Function SVM (RBF_SVM), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Ridge Classifier, and Gaussian Naive Bayes (GNB). We also include a standalone Spiking GNN (SGNN) baseline from prior work Ferreira et al. (2025).

Unless otherwise stated, "anchor" refers to the best BO-selected residual GNN ($\alpha = 0.15$, JK+Attention, 1 GNN layer, self-loops enabled) with key fingerprints and physics-informed augmentation; all ablation variants reuse its hyperparameters for fairness.

C.3 Method and Technical Details for "Translation"

C.3.1 Virtual Screening Methodology

Candidate Molecule Library The screening library comprised 123,239,643 molecules obtained from the PubChem Compound database (CID-SMILES file, accessed December 2025). Each molecule was represented by:

- A 25-dimensional molecular descriptor vector containing physicochemical properties including molecular weight, topological polar surface area (TPSA), hydrogen bond donor/acceptor counts, rotatable bond count, XLogP, and 3D steric parameters from PubChem.
- A 256-bit Morgan fingerprint (radius 2) generated using RDKit for structural encoding.

The library was partitioned into 62 chunks of 2,000,000 molecules each, further split into parts of 500,000 molecules for parallel processing. A total of 235 parallel jobs were executed on GPU nodes (NVIDIA A40), achieving a throughput of approximately 40 molecules per second per job.

Graph Construction for Inference For each candidate molecule, we constructed heterogeneous graphs following the DTE-GNN architecture. The template graph was derived from the training data corresponding to the PFOS detection experiments in Wang et al. (2025), containing the following node types:

- **Probe material**: Replaced with each candidate molecule's features.
- **Detect target**: Set to PFOS, PFOA, dodecylsulfonic acid (DDS), or trichloroacetic acid (TCAA) depending on the screening task.
- **Substrate**: Fixed to the experimental configuration (kept from template).
- **Condition**: Fixed measurement conditions including concentration ranges and environmental parameters.

The edge structure connecting these nodes was preserved from the training template, representing the physical relationships in the FET sensor system.

Target Molecules for Selectivity Analysis Four detect target molecules are used for comprehensive selectivity screening (Table S12):

Model Inference Three pre-trained DTE-GNN models are applied to each candidate-target pair:

1. **LDL Model**: Predicts lower detection limit class (0: low/good, 1: medium, 2: high/poor).
2. **UDL Model**: Predicts upper detection limit class (0: low/poor, 1: medium, 2: high/good).

Target	CID	Role	Description
PFOS	74483	Primary	Perfluorooctanesulfonic acid
PFOA	9554	Target	Perfluorooctanoic acid
DDS	3423265	Interferent	Dodecylsulfonic acid
TCAA	6421	Interferent	Trichloroacetic acid

Table S12: Target molecules used in selectivity screening. Neutral forms are used for GNN screening; anionic forms for DFT validation (Appendix C.3.2).

3. **Sensitivity Model**: Predicts sensitivity class (0: low/poor, 1: medium, 2: high/good), evaluated for both percentage-based and mV-based sensitivity metrics.

Each model outputs a probability distribution $[p_0, p_1, p_2]$ over three classes. Inference is performed in batches of 2,048 molecules using PyTorch and PyTorch Geometric on CUDA-enabled GPUs.

Scoring Functions

SINGLE-TARGET SCORE For a given probe-target pair, the composite score reflects the joint probability of achieving desirable performance across all three metrics:

$$S_{\text{target}} = P(\text{LDL} = 0) \times P(\text{UDL} = 2) \times P(\text{Sensitivity} = 2) \tag{8}$$

where class 0 for LDL indicates low detection limit (desirable), and class 2 for UDL and sensitivity indicates high dynamic range and high sensitivity, respectively.

SELECTIVITY SCORE To identify probes with high specificity for PFAS compounds over common interferents, we computed the selectivity score as:

$$S_{\text{selectivity}} = \frac{S_{\text{PFOS}} \times S_{\text{PFOA}}}{S_{\text{DDS}} \times S_{\text{TCAA}}} \tag{9}$$

This formulation rewards candidates that:

- Achieve high scores for both PFOS and PFOA (numerator, representing target PFAS response).
- Achieve low scores for both DDS and TCAA (denominator, representing interferent response).

Candidates were filtered to require a minimum target score of 0.01 and selectivity ratio greater than 1.0 before ranking.

Computational Resources The complete screening of 123M molecules across four targets required:

- 940 GPU-hours (235 jobs $\times$ 4 targets $\times$ approximately 1 hour each).
- Storage: 120 GB for result files (30 GB per target).
- Peak memory: 4-8 GB per job during chunk processing.

Result Aggregation Results were aggregated using a memory-efficient streaming algorithm:

1. Process one chunk at a time, loading corresponding result files from all four targets.
2. Compute selectivity scores for molecules present in all four result sets.
3. Maintain a min-heap of size 1,000 to track top candidates without storing all results in memory.
4. Output final rankings with complete prediction details for downstream analysis.

C.3.2 DFT Computational Methodology for Host-Guest Binding Energy Calculations

Scope and Rationale The DFT calculations in this work serve as qualitative validation of the DTE-GNN screening predictions, specifically to compare binding selectivity between the model-predicted probe candidates and the experimentally validated β-Cyclodextrin baseline Wang et al. (2025). Given the broad scope of the T^3 framework spanning text mining, GNN modeling, and virtual screening across millions of candidates, we adopt a computationally efficient grid-based conformational sampling approach with implicit solvation rather than expensive classical molecular dynamics (MD) simulations with explicit solvent. This choice is justified for our comparative screening purpose: systematic grid sampling adequately captures the dominant binding modes for rigid macrocyclic hosts, while the SMD implicit solvation model Marenich et al. (2009) provides reliable solvation free energies at a fraction of the computational cost. More rigorous MD-based free energy calculations could be pursued in future studies for the most promising candidates identified here.

Initial Structure Generation Host-guest inclusion complex configurations are systematically generated through geometric sampling to ensure comprehensive coverage of the binding conformational space. For each host-guest pair, three types of configurations are constructed:

1. **Vertical insertion configurations**: The guest molecule is aligned along the host cavity axis and positioned at nine different insertion depths (-8, -6, -4, -2, 0, +2, +4, +6, +8 Å relative to the cavity center). At each depth, two orientations are sampled by flipping the guest molecule 180 degrees, resulting in 18 configurations per pair.
2. **Surface-lying configurations**: The guest molecule is placed horizontally on both the upper and lower surfaces of the host cavity. At each surface, four rotational orientations (0, 90, 180, 270 degrees around the cavity axis) are sampled, yielding 8 configurations per pair.
3. **Side-binding configurations**: The guest molecule is positioned approaching the host from the side at six azimuthal angles (0, 60, 120, 180, 240, 300 degrees), generating 6 configurations per pair.

The host cavity axis is determined using principal component analysis (PCA) on the ring atoms defining the macrocyclic cavity. The direction corresponding to the smallest eigenvalue is identified as the cavity normal vector. Similarly, the guest molecule principal axis is determined as the direction of largest structural variance via PCA.

All initial structures are placed in a cubic simulation box of 50 Å side length to ensure adequate separation from periodic images.

Geometry Optimization A multi-level optimization protocol is employed to efficiently explore the potential energy surface:

1. **Semi-empirical pre-optimization**: Initial structures are first optimized using the PM6 semi-empirical method to remove atomic clashes and obtain reasonable starting geometries.
2. **DFT geometry optimization**: The PM6-optimized structures are further refined using density functional theory at the B3LYP/6-31G(d) level with the SMD implicit solvation model (solvent = water). This step provides accurate equilibrium geometries while maintaining computational tractability for the large host-guest systems.

Single-Point Energy Calculations Final electronic energies are computed at a higher level of theory using single-point calculations on the B3LYP-optimized geometries. The M06-2X functional with the 6-31+G(d) basis set is employed, combined with the SMD solvation model Marenich et al. (2009) to account for aqueous solvation effects. This computational protocol follows established approaches for PFOS thermodynamic calculations Montero-Campillo et al. (2010); Giroday et al. (2014). The M06-2X functional is selected for its reliable description of non-covalent interactions, including dispersion and hydrogen bonding, which are crucial for accurate host-guest binding energetics.

Configuration Selection and Binding Energy Calculation For each host-guest pair, the configuration with the lowest total electronic energy from the single-point calculations is selected as the representative binding geometry. The binding energy is calculated as:

$$\Delta E_{\text{bind}} = E_{\text{complex}} - E_{\text{host}} - E_{\text{guest}} \tag{10}$$

where E_{complex}, E_{host}, and E_{guest} are the single-point energies of the host-guest complex, isolated host molecule, and isolated guest molecule, respectively. All energies are converted from Hartree to kcal/mol using the conversion factor 627.509474 kcal/mol per Hartree.

Treatment of Protonation States To investigate the influence of guest molecule protonation state on binding affinity, calculations are performed for both neutral guest molecules (protonated forms: PFOA, PFOS, DDS-H, TCAA) and their corresponding deprotonated anionic forms (PFOA^-, PFOS^-, DDS^-, TCA^-). For anionic systems, geometry optimization and single-point calculations are performed with a total charge of -1.

Software and Computational Resources All quantum chemical calculations are performed using Gaussian 16. Geometry optimizations employ default convergence criteria. Single-point calculations use tight SCF convergence. Structure generation and analysis scripts are implemented in Python using NumPy and SciPy libraries.

Electrostatic Potential Surface Analysis To visualize and quantify the electrostatic environment of host–guest complexes, electrostatic potential (ESP) surfaces are computed using the Gaussian 16 `cubegen` utility. For each system, a dedicated single-point calculation is performed at the ωB97XD/6-311+G(2d,2p) level with SMD implicit solvation (solvent = water) on the lowest-energy binding configuration, with `density=current` and `pop=full` keywords to ensure the ESP is evaluated from the converged SCF density. The SCF electron density and electrostatic potential are then evaluated on a uniform $100\times100\times100$ grid via `cubegen`. The van der Waals surface is defined as the $\rho = 0.001$ a.u. electron density isosurface, following the convention of Murray & Politzer (2011). ESP values on this isosurface are extracted and converted from atomic units (Hartree/e) to electronvolts (1 Hartree = 27.2114 eV). Quantitative descriptors include the mean surface ESP ($\bar{V}$), standard deviation (σ), fraction of positive/negative surface area, and the Politzer electrostatic balance index Murray & Politzer (2011). ESP surfaces are rendered using PyMOL with a color scale of -0.05 to $+0.05$ a.u. (red–white–blue).

D Augmentation Strategy Ablation

Following the data protocol from Ferreira et al. (2025), we validate the effectiveness of physics-aware data augmentation by comparing against two random baselines: same-magnitude random (identical perturbation bounds but without physical constraints or discrete augmentations) and destructive random (large perturbations of $\pm$80–200% without constraints). All models are trained on augmented data and evaluated on the held-out original dataset to assess generalization to real experimental samples.

Figure S25 summarizes the results across LDL, UDL, and Sensitivity prediction tasks. Physics-aware augmentation consistently outperforms both random baselines by substantial margins. On basic metrics (Accuracy, F1, Precision, Recall), physics-aware augmentation achieves 13–18% absolute improvements over the best random baseline across all three tasks, with UDL showing the largest gap (+17.7% accuracy, +17.9% F1). Notably, same-magnitude random performs comparably to destructive random despite using much smaller perturbations, demonstrating that the performance gap is not simply due to noise magnitude. Rather, physics-aware augmentation generates an entire dataset of physically plausible FET sensor configurations—each augmented sample represents a device that could realistically exist—whereas random perturbations produce nonsensical combinations (e.g., negative pH, impossible material properties) that corrupt the learned representations. This validates that our augmentation strategy creates meaningful synthetic data grounded in semiconductor device physics, enabling models to learn transferable patterns from an expanded but physically coherent training distribution.

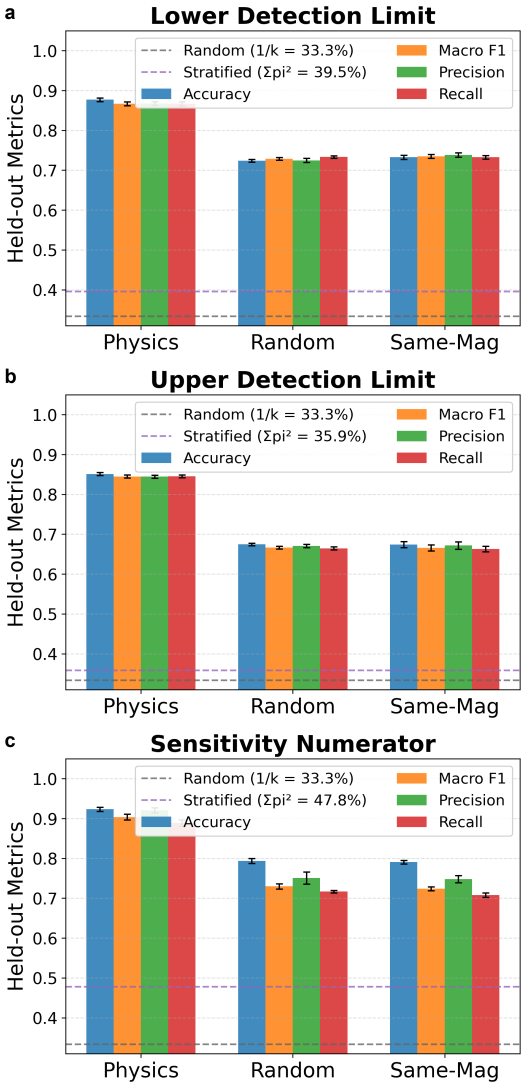

Figure S25: Augmentation strategy comparison on held-out original data. Physics-aware augmentation vs. same-magnitude random and destructive random baselines across LDL, UDL, and Sensitivity prediction tasks. Metrics shown: Accuracy, macro-F1, Precision, and Recall.

E DTE-GNN ARCHITECTURE ABLATION

To understand the contribution of each architectural component in DTE-GNN, we conduct systematic ablation studies across 14 configurations varying readout mechanisms, branch composition, fingerprint encoders, network depth, GCNII residual strength (α), and training strategies (Figure 4b–c).

Multi-Task Ranking Methodology. Since different configurations may excel on different prediction tasks, we adopt a cross-task ranking approach following established practices for comparing classifiers over multiple datasets Demsar (2006). For each configuration c and task t, we compute an optimistic score $s_{c,t} = \mu_{c,t} + \sigma_{c,t}$, where $\mu_{c,t}$ is the mean accuracy and $\sigma_{c,t}$ is the standard deviation across cross-validation folds. This UCB-style scoring, inspired by Gaussian process optimization Derrac et al. (2011), rewards configurations that achieve high mean performance while accounting for estimation uncertainty—an "optimism in the face of uncertainty" principle that balances exploration and exploitation. We rank configurations within each task by their optimistic scores and compute the average rank across all three tasks (LDL, UDL, Sensitivity). The configuration with the lowest average rank is considered the most consistently effective. Under this methodology, the full DTE-GNN model (with JK connections, attention pooling, and $\alpha = 0.15$) achieves the best cross-task ranking in both alpha ablation and component ablation studies, validating it as the most robust architecture choice (Figure S19).

The most striking finding is that the GNN branch is essential: removing it entirely (fingerprint (FP) Only) causes performance to collapse to near-random levels (31–42% accuracy vs. 87–92% for the full model), confirming that the heterogeneous graph structure captures the core device-performance relationships. In contrast, the fingerprint branch provides modest but consistent improvements; GNN Only configurations perform within 1% of the full model, indicating that macroscopic physical properties encoded in graph nodes carry the dominant predictive signal.

Regarding readout mechanisms, the full model with both jumping-knowledge (JK) connections and attention-based pooling achieves the most consistent performance across all tasks. While individual ablations may show marginal improvements on specific tasks (e.g., removing JK yields 88.2% vs. 87.7% accuracy on LDL alone), cross-task evaluation reveals that such gains do not generalize—the full model maintains the best average ranking when evaluated across LDL, UDL, and Sensitivity simultaneously. Network depth matters: zero GNN layers (Depth: 0) substantially degrades performance (5–7% accuracy drop), while two layers perform comparably to one layer, indicating that a single message-passing step suffices to propagate information across the physics-informed graph topology. The GCNII residual strength ($\alpha \in \{0.05, 0.15, 0.25\}$) and fingerprint encoder choice (MLP vs. Transformer vs. SNN) show minimal impact (<0.5% variation), suggesting the architecture is robust to these hyperparameters. Finally, class weighting and self-loop removal provide no consistent benefit, validating our default training configuration.

F PROMPT EVOLUTION

This appendix presents the human-authored initial prompts alongside their autonomously optimized counterparts for all five field categories: the core fields (8 fields) and four extended field groups. For the four extended field groups, we embed the representative top-performing prompts (M8, inference=Deepseek-70B, critique=GPT-oss-120B). Across these prompts, the optimizer generally learned to: (i) tightly scope the extraction window to the main article while ignoring cited abstracts/supplementary fluff; (ii) normalize units and apply explicit defaults (thickness/time/concentration) instead of heuristic guessing; (iii) disambiguate primary device elements from auxiliary layers (e.g., gate dielectric vs. encapsulation vs. reference electrode); (iv) collapse multiple material or device variants into a single representative record when the paper only tweaks minor formulations; (v) extract numeric values directly with required conversions (e.g., pH→[H^+], ppm from P/P_v, seconds from minutes); and (vi) forbid inference—fields remain "" unless explicitly stated. These behaviors collectively reduce hallucination, enforce consistency, and preserve the most representative device configuration per paper.

Concrete improvements observed across the five prompt families include: (1) *Scope control*—original prompts often swept in cited abstracts or supplementary blurbs; optimized prompts now

restrict extraction to the main article body (and, when needed, a specified table or subsection). (2) *Unit rigor*—where initial prompts left units free-form, optimized prompts demand explicit conversions (e.g., minutes→seconds for response/recovery times; P/P_v→ppm; pH→[H^+] for bounds) or defaults when absent. (3) *Device disentangling*—early instructions blurred substrate/dielectric/encapsulation; optimized prompts explicitly separate gate dielectric from protective coatings and require empty strings when not stated. (4) *Variant consolidation*—initial versions invited multiple records for minor material tweaks; optimized prompts collapse variants into a single representative record unless the paper truly reports distinct devices. (5) *Field-specific cues*—material-layer prompts add dopant handling and thickness defaults; sensitivity/response prompts require table parsing and unit rounding; synthesis/thermal prompts stress atmosphere/time/temperature capture while ignoring non-functional processing layers. Collectively, these edits yield cleaner, reproducible JSON with fewer missing-unit errors and no fabricated values.

F.1 Case Study: Prompt Evolution Trajectory

To illustrate how TextGrad-based optimization progressively refines extraction prompts, we trace a single representative run (M8, DeepSeek-70B inference, GPT-oss-120B critique) across 20 training rounds on the core fields. Under our tournament-based selection mechanism (Section C.1), the optimized candidate replaces the current prompt *only* when it achieves a higher committee score on the same minibatch; otherwise the existing prompt is retained unchanged. In this run, 8 of 20 proposals were accepted (40%), reflecting the conservative selection pressure that filters out regressions while accumulating genuine improvements.

Stage 1: General extraction hygiene (Rounds 2–6). The earliest accepted changes address generic extraction errors that manifest across all paper types.

- **Round 2 — Verbatim probe-material names.** The human prompt specifies `"(probe chemical name)"`, which led the LLM to paraphrase or abbreviate materials (e.g., producing "zinc oxide" instead of the paper's "ZnO nanowires"). The optimizer replaced this with `"(exact probe material name as it appears in the paper, copy verbatim)"`, eliminating name mismatches. This instruction persisted unmodified through all 18 subsequent rounds.
- **Round 4 — Unit-bias removal for detection limits.** The original placeholder `"xx ppm (or corresponding units)"` acted as an implicit prior, biasing the LLM toward converting all concentrations to ppm. The optimizer replaced it with `"value with its original unit (or leave empty)"`, preserving the diversity of units (nM, ppb, vol%, etc.) reported across the sensor literature.
- **Round 6 — Scope control against reference hallucination.** A single parenthetical clause was added to the opening instruction: `"(excluding bibliography and reference list)"`. This prevented the LLM from extracting sensor specifications mentioned in cited references rather than the paper's own experiments—a source of phantom records. This round produced the largest single-round committee-score improvement (+0.117).

Stage 2: Domain-specific disambiguation (Rounds 7–12). Once general extraction hygiene is established, subsequent improvements target sensor-science-specific ambiguities that arise only for particular paper types.

- **Round 7 — Humidity sensor classification.** The sensor_type field was augmented with the rule: `"water vapour in air counts as gas"`. This resolved systematic misclassification of humidity sensors, which were previously labeled as "liquid" due to the presence of water.
- **Round 12 — Detection-limit fallback from tested concentrations.** Many papers omit an explicit limit of detection but report a tested concentration range. The optimizer added: `"if not directly stated, use the smallest concentration tested"`. This reduced missing-value rates for detection limits without introducing fabricated numbers.

Between these breakthroughs, two stabilization plateaus occurred (Rounds 8–11 and 13–16), during which proposals were consistently rejected. Notably, several rejected changes directly contradicted previously learned rules—for instance, Round 13 proposed expanding chemical abbreviations to full IUPAC names, conflicting with the "copy verbatim" rule from Round 2. The tournament mechanism correctly preserved the earlier beneficial modification.

Stage 3: Numerical conversion rules (Rounds 17–20). The most domain-specific improvements emerged in the final rounds, when rarer paper types (e.g., pH sensors) exposed new error classes.

- **Round 17 — pH-to-concentration conversion formula.** pH sensors report detection ranges in pH units (e.g., "pH 2–12"), whereas the ground-truth database records molar concentrations. The optimizer discovered and embedded the conversion rule: "`calculate lower_detection_limit = 10^-upper_pH M`", including the non-trivial inversion—upper pH maps to *lower* concentration—without any explicit chemical instruction. This illustrates TextGrad's capacity to learn domain-specific numerical transformations purely from error feedback between LLM outputs and expert annotations.
- **Round 20 — Temperature missing-data handling.** A final instruction was added to output an empty field rather than a default numeric value when no operating temperature is stated, correcting the LLM's tendency to hallucinate "25 °C" as a placeholder.

Table S13 summarizes the trajectory. The evolution proceeds from generic extraction hygiene (verbatim copying, unit preservation, scope control) through domain-specific disambiguation (sensor classification, fallback logic) to numerical conversion rules (pH→concentration). Each accepted modification is strictly additive—no accepted change was ever reverted in a later round. The conservative tournament selection (60% rejection rate) plays a critical role: it prevents regression while allowing incremental accumulation of domain knowledge that would be difficult to engineer manually. Table S14 summarizes the 28 extraction fields organized into five prompt categories.

Table S13: Prompt evolution trajectory for M8 on core fields (20 rounds). Accepted changes are grouped by the type of improvement.

Round	Improvement	Category	ΔScore
2	Verbatim material names	General	+0.000
4	Unit-bias removal	General	+0.012
5	Test medium disambiguation	General	+0.008
6	Scope: exclude bibliography	General	+0.117
7	Humidity sensor rule	Domain	+0.015
12	Detection-limit fallback	Domain	+0.053
17	pH→concentration formula	Numerical	+0.010
20	Temperature missing-data	Numerical	+0.023
Rejected proposals			12/20

F.2 Initial and Optimized Prompts

Core Fields (8 fields)

Input: Full text of a Paper

Expected Output Example (by human expert):

```
{
  "records": [
    {
      "sensor_type": "gas",
      "detect_target": "ammonia",
      "lower_detection_limit": "4 ppm",
      "upper_detection_limit": "60 ppm",
      "probe_material": "6PTTP6(5,5'-Bis(4-hexylphenyl)-2,2'-bithiophene)
          /8-3-NTCDI(N,N'-Bis(3-(perfluorooctyl)propyl)-1,4,5,8-
          naphthalenetetracarboxylic diimide)",
```

Table S14: Summary of LLM extraction fields by prompt category. A total of 28 fields are extracted across five prompt families.

Category	Fields	Field Names
Core	8	sensor_type, detect_target, lower_detection_limit, upper_detection_limit, probe_material, test_operating_temperature, pH_value, test_medium
Electrode Architecture	4	gate, source, drain, structure_design_type
Material Layer	7	substrate, substrate_thickness, channel, dielectric_layer, dielectric_layer_thickness, surface_functionalization, structure_dimensionality
Sensitivity/Response	4	response_time, recovery_time, sensitivity_numerator, sensitivity_denominator
Synthesis/Thermal	5	annealing_temperature, annealing_time, annealing_atmosphere, hydrothermal_temperature, hydrothermal_time
Total	**28**	

```
      "test_operating_temperature (celcius)": "25 \textdegree C",
      "pH_value": "-1",
      "test_medium": "air"
    }
  ]
}
```

Human initial prompt (4 fields)

```
After reading the scientific publication full text provided above, please
     try to generate formatted JSON file for extraction of important
    information. Leave "" for not available fields.

For instance:
{
"records": [
    {
      "sensor_type": "(gas/bio/liquid; please choose one)",
      "detect_target": "(target chemical name)",
      "lower_detection_limit": "xx ppm (or corresponding units)",
      "upper_detection_limit": "xx ppm (or corresponding units)",
      "probe_material": "(probe chemical name)",
      "test_operating_temperature (celcius)": "xx \textdegree C",
      "pH_value": "(use -1 for gas, otherwise a number, if pH sensor, use
           range like xx-yy)",
      "test_medium": "(the name of the medium, e.g. air, water, or other
          medium)"
    }
   (continue if the publication has recorded multiple different target
       been detected)
  ]
}
```

Optimized prompt (M8, GPT-oss-120B critique, DS70B inference)

```
After reading the scientific publication full text provided above, please
     generate a formatted JSON file for extraction of important
    information. Only use the main article text above; ignore any cited-
    paper abstracts, supplementary sections, or other unrelated excerpts.
     If the article describes multiple sensors, identify the sensor with
    the smallest lower_detection_limit and create a single JSON object
    for that sensor only. Do not create separate records for each
    material variant; combine them into one record representing the
    overall sensor system (use a generic class name for the probe
```

```
    material). When several material variants are tested for the same
    sensor, combine them into one record and describe the probe material
    using its generic class name (e.g., "diketopyrrolopyrrole fluorene
    copolymer"). Leave "" for not available fields.

For instance:
{
"records": [
    {
      "sensor_type": "(gas/bio/liquid; please choose one)",
      "detect_target": "(target chemical name)",
      "lower_detection_limit": "Extract the reported concentration value
          for the target chemical as written; provide only the numeric
          value with its unit, stripping any qualitative qualifiers such
          as 'below', 'above', or '~'.
      Prioritize any explicit detection-limit statement (e.g., 'the
          lowest detection limit is X ppm').
      If no explicit detection limit is given, use the smallest
          concentration that was experimentally tested (or leave '').
      If the paper provides a concentration range (e.g., 'from X ppm to Y
           ppm'), use the lower bound (X) as the lower detection limit.
      If the paper reports a 'limit of detection (LOD)', treat that value
           as the lower detection limit.
      If the value is given as a P/Pv ratio, convert it to ppm by
          multiplying the ratio by the analyte's equilibrium vapor
          pressure (Pv, kPa) and then by 1000 ppm / 101.3 kPa (~9.87).
          Output the numeric ppm value with the unit ppm.
      Otherwise preserve the exact numeric value and unit; do not convert
          , round, or infer other units.
      If the article gives a pH response range, convert the highest pH in
           that range to molar [H+] using [H+]=10^{-pH} M and record that
           value as the lower detection limit.",
      "upper_detection_limit": "The highest concentration of the target
          analyte that is explicitly reported or tested in the paper (i.e
          ., the maximum concentration examined in the experiments).
      Provide only the numeric value with its unit, stripping any
          qualitative qualifiers such as 'below', 'above', or '~'.
      Use the largest reported concentration (or leave '').
      If the paper provides a concentration range (e.g., 'from X ppm to Y
           ppm'), use the upper bound (Y) as the upper detection limit.
      If only a single value is given, leave 'upper_detection_limit'
          empty.
      If the value is given as a P/Pv ratio, convert it to ppm by
          multiplying the ratio by the analyte's equilibrium vapor
          pressure (Pv, kPa) and then by 1000 ppm / 101.3 kPa (~9.87).
          Output the numeric ppm value with the unit ppm.
      Otherwise preserve the exact numeric value and unit; do not convert
          , round, or infer other units.
      If the article gives a pH response range, convert the lowest pH in
          that range to molar [H+] using [H+]=10^{-pH} M and record that
          value as the upper detection limit.",
      "probe_material": "(material that directly interacts with the
          target gas, e.g., catalytic or gate metal layer; if no separate
           layer is described, use the gate-dielectric material; if
          several variants are reported, give the generic class name;
          otherwise leave '' if not available)",
      "test_operating_temperature (celcius)": "xx \textdegree C",
      // If the paper mentions an operating temperature (e.g.,
          'measurements were performed at 25 °C'), extract the numeric
          value (with unit) and place it in 'test_operating_temperature
          (celcius)'; if none, leave the field blank.
      "pH_value": "(if not directly given but can be reasonably inferred
          from the test medium, provide that inferred value; otherwise
          use -1; for gas sensors use -1; otherwise a numeric value; if
          pH sensor, use range like xx-yy)",
```

```
      "test_medium": "(the name of the medium, e.g., air, water, or other
           medium; if multiple media are mentioned, choose the one used
          for the primary sensor performance experiments, ignoring
          ancillary tests unless they are the only measurements reported)
          "
      // If multiple ranges are given, use the primary detection window:
          from the lowest measurable concentration to the highest before
          saturation.
    }
   (continue if the publication has recorded multiple different target
      been detected)
  ]
}
```

EXTENDED FIELDS

Electrode Architecture (4 fields) Human initial prompt

```
After reading the scientific publication full text provided above, please
     try to generate formatted JSON file for extraction of important
    information. Leave "" for not available fields.

For instance:
{
"records": [
    {
      "gate": "(gate electrode material name)",
      "source": "(source electrode material name)",
      "drain": "(drain electrode material name)",
      "structure_design_type": "(e.g. planar, vertical, coplanar, back-
          gated, top-gated, interdigitated)"
    }
   (continue if the publication has recorded multiple different device
      architectures)
  ]
}
```

Optimized prompt (M8, GPT-oss-120B critique, DS70B inference)

```
After reading the scientific publication full text provided above, **
    extract** the gate, source, drain materials **including any dopant
    element specified** and the structure-design type that are **
    explicitly stated** in the text, and generate a JSON file **(with
    fields `gate`, `source`, `drain`, and `structure_design_type`)**.

**The `gate` field must contain the material of the gate electrode (or
    reference electrode for electrolyte-gated devices), not any
    functionalizing molecules or catalysts.**

*gate*: the material of the **reference electrode** (or gate electrode
    for non-electrolyte devices) as described in the paper.

**For extended-gate (remote) configurations, the gate material is the
    base metal of the extended gate electrode itself (e.g., the copper
    pad on a PCB), not the sensing membrane or functional layer attached
    to it.**

If the paper only mentions a gate-related material such as a catalyst or
    substrate, use that material for the gate field.

**Normalize material names to canonical forms** (e.g., "Si substrate with
     HfO2" -> "phosphorus/silicon"; "Ti/Au" -> "titanium/gold").

If any required field (gate, source, drain, structure_design_type) is not
     explicitly stated, fill them with defaults: gate = "phosphorus/
```

```
    silicon", source = "gold", drain = "gold", structure_design_type = "
    Standard".

**Note:** For electrolyte-gated devices, the gate material is the
    reference electrode (e.g., Ag/AgCl; if given by a common name such as
     calomel, express it as the chemical pair 'mercury/mercury chloride')
    ; for BioFETs the gate material is the functional membrane
    composition (e.g., polypyrrole/urease). The source and drain are the
    channel contact metals (e.g., Ni/Au).

**Clarification:** The "gate" field refers to the material of the
    reference electrode (or gate electrode for solid-state gates). For
    back-gated devices, the gate electrode is the substrate; extract its
    material from the substrate description (e.g., "n-doped Si" -> "boron
    /silicon").

**Include any dopant name preceding the substrate (e.g., "phosphorus/
    silicon"); if none is mentioned, use just the substrate material.**

**If the text explicitly states that the substrate serves as the gate (e.
    g., "heavily phosphorus-doped silicon wafer (the substrate) serves as
     the bottom-gate electrode"), use the substrate material as the gate
    .**

**For source and drain, extract the semiconductor channel material of the
     FET as explicitly described (e.g., silicon, poly-Si, metal-oxide,
    organic semiconductor), not the contact metal unless the
    semiconductor itself is not specified.**

If the paper mentions more than one distinct FET configuration (different
     gate materials, source/drain metals, or structural designs), create
    a separate record for each configuration.

**Structure Design Type** must be selected from the valid values list,
    for example:
- *Remote (e.g., the gate electrode is physically separated from the
    transistor channel, as in an Extended Gate FET - EGFET)*
- *Electrolyte-Gated (e.g., a gate electrode in direct contact with an
    electrolyte)*
- *Top-Gated*
- *Back-Gated*
- *Planar*
- *Vertical*
- *Coplanar*
- *Interdigitated*

**Decision rule:** If the gate electrode is a distinct, physically
    separate electrode immersed in the same electrolyte as the channel (i
    .e., the gate does **not** sit directly on the channel surface),
    label the `structure_design_type` as **'Remote'**. If the electrolyte
     itself serves as the gate dielectric directly contacting the channel
     surface (e.g., ion-gel or liquid electrolyte on top of the channel),
     label it as **'Electrolyte-Gated'**. Use the remaining terms as
    defined in the original list.

For instance:
{
  "records": [
    {
      "gate": "silver/silver chloride",
      "source": "silver",
      "drain": "silver",
      "structure_design_type": "Floating"
    }
```

```
    // continue if the publication has recorded multiple different device
         architectures
  ]
}
```

Material Layer Composition (7 fields) Human initial prompt

```
After reading the scientific publication full text provided above, please
     try to generate formatted JSON file for extraction of important
    information. Leave "" for not available fields.

For instance:
{
"records": [
    {
      "substrate": "(substrate material name)",
      "substrate_thickness": "xx um",
      "channel": "(channel/active layer material name)",
      "dielectric_layer": "(dielectric layer material name)",
      "dielectric_layer_thickness": "xx nm",
      "surface_functionalization": "(surface modification material or
          molecule name)",
      "structure_dimensionality": "(0D/1D/2D/3D, describing nanostructure
           dimensionality)"
    }
   (continue if the publication has recorded multiple different material
       configurations)
  ]
}
```

Optimized prompt (M8, GPT-oss-120B critique, DS70B inference)

```
After reading the scientific publication full text provided above, please
     try to generate a formatted JSON file for extraction of important
    information.

Leave "" for not available fields, and for `substrate_thickness` and `
    dielectric_layer_thickness` extract the exact numeric value and unit
    explicitly given for the substrate or dielectric layer respectively (
    ignore other layer thicknesses).

**If the substrate thickness is mentioned, always include it in the `
    substrate_thickness` field, capturing the exact numeric value and
    unit (e.g., "525 um"). If not stated, apply the following defaults:
    silicon wafer - 525 um; glass - 1 mm; quartz - 500 um; flexible
    polymer (e.g., PET) - 125 um; other substrates - leave the field
    empty.**

**Note: The "channel" field should denote the sensing transistor used in
    the ISFET system (e.g., p-type Si-FET, organic TFT), not the
    reference-electrode metal stack.**

When filling the "substrate" field, **select the material that is
    described as the supporting bulk or wafer on which the device is
    built; do not treat active sensing materials or coatings as the
    substrate.**

If the substrate is listed with multiple components separated by "/",
    retain the full slash-separated string as the substrate name; if the
    substrate is described as 'degenerately doped silicon' (or any
    similar phrasing), treat the substrate as plain silicon and do not
    infer or output the dopant element; otherwise, if the substrate is a
    doped semiconductor, include the dopant element (e.g., phosphorus-
    doped silicon) as part of the substrate description.
```

```
**If the paper mentions multiple substrates, report the substrate that
    hosts the transistor channel (the material on which the FET itself is
     built) and ignore substrates used only for auxiliary components such
     as reference electrodes or packaging.**

**If the paper mentions a generic substrate name such as "glass",
    interpret it as a silicon-based substrate and output the material as
    "silicon dioxide/sodium oxide/calcium oxide"; assign a default
    thickness of "1000 um" unless a specific thickness is explicitly
    provided.**

For the `dielectric_layer` field, **only include the material that
    functions as the gate insulator in the final device; ignore any
    polymer or resist layers that are used solely for processing (e.g.,
    PMMA, photo-resist).**

**Only list a dielectric layer if the paper explicitly mentions a
    separate gate-dielectric material (e.g., Al2O3, HfO2, Si3N4) distinct
     from any native oxide on the substrate.**

If the device contains more than one layer that could be considered a
    dielectric, **do not treat the native oxide layer that directly
    contacts the channel surface as the dielectric unless it is
    explicitly identified as a gate dielectric**; encapsulating polymers
    or protective coatings should be placed under *
    surface_functionalization* (or omitted if not a functionalization).

**Only fill *surface_functionalization* when a distinct material is
    intentionally added to the gate electrode surface that is different
    from the electrode/contact material; if the functional element is the
     same as the electrode metal or no separate functionalization is
    applied, leave the field empty.**

If the substrate description includes an oxide layer (e.g., Si/SiO2) and
    its thickness is provided, treat that oxide as the dielectric layer
    and populate `dielectric_layer` and `dielectric_layer_thickness`
    accordingly.

For instance:
{
  "records": [
    {
      "substrate": "(substrate material name)",
      "substrate_thickness": "(xx um if mentioned)",
      "channel": "(channel/active layer material name)",
      "dielectric_layer": "(dielectric layer material name)",
      "dielectric_layer_thickness": "xx nm",
      "surface_functionalization": "(surface modification material or
          molecule name)",
      "structure_dimensionality": "(0D/1D/2D/3D, describing nanostructure
           dimensionality)"
      // Create one dictionary for each sensing microneedle on the MMNs (
          Na+, K+, Ca2+, and pH). The order of the dictionaries does not
          matter.
    }
    (continue if the publication has recorded multiple different material
         configurations)
  ]
}

/* When a material is given in the paper as an abbreviation (e.g.,
    ZrO$_2$, PMMA, PMF), output its **full chemical name** in the JSON (e
    .g., "zirconium oxide", "poly(methyl methacrylate)", "poly melamine
    co-formaldehyde"). Preserve the order of components as they appear (e
```

```
    .g., "zirconium oxide/poly(methyl methacrylate)/poly melamine co-
    formaldehyde"). */
```

Sensitivity and Response Characteristics (4 fields) Human initial prompt

```
After reading the scientific publication full text provided above, please
     try to generate formatted JSON file for extraction of important
    information. Leave "" for not available fields.

For instance:
{
"records": [
    {
      "response_time": "xx s",
      "recovery_time": "xx s",
      "sensitivity_numerator": "(the change in signal, e.g. DeltaR,
          DeltaI, DeltaG, or absolute values)",
      "sensitivity_denominator": "(the reference value, e.g. R0, baseline
           current, or gas concentration)"
    }
   (continue if the publication has recorded multiple different
       performance measurements)
  ]
}
```

Optimized prompt (M8, GPT-oss-120B critique, DS70B inference)

```
After reading the scientific publication full text provided above, please
     generate a formatted JSON file for extraction of important
    information.

**If the required values appear inside a table (including ASCII-style
    tables), parse the table rows to locate the appropriate numbers for
    response time, recovery time, and sensitivity.**

**Extract the response time and recovery time as numeric values in
    seconds, rounding to the nearest whole second, and express them as a
    number followed by the letter "s" (e.g., "25 s"). If the source
    reports these times in minutes (or any other unit), first convert
    them to seconds (e.g., 5 min -> 300 s, multiply by 60).**

**When scanning the paper, look for numeric values immediately followed
    by terms such as "response time", "recovery time", "rise time", "
    settling time", "response", or "recovery" (case-insensitive).** Leave
     "" for not-available fields.

**If any required field cannot be found in the provided text, set that
    field to an empty string ("") and do not guess or fabricate numbers.
    All numeric values must be extracted directly from the supplied
    article; do not infer or calculate values that are not explicitly
    reported.**

**Only include the fields listed below; do not add title, authors,
    journal, or year.** The JSON must contain exactly these fields for
    each record: **response_time, recovery_time, sensitivity_numerator,
    sensitivity_denominator**.

**If the paper reports an absolute Dirac-voltage shift (e.g., 14 V) and
    the corresponding analyte concentration (e.g., 300 pM), compute `
    sensitivity_numerator` as the voltage shift converted to millivolts
    (1 V = 1000 mV) and set `sensitivity_denominator` to the reported
    concentration using the same unit.**

**Do not enclose the JSON in any markdown or code-fence tags. Output the
    raw JSON only.** **Return exactly one JSON object (i.e., a single-
```

```
    element array) summarizing the sensor performance; choose the most
    representative values if several are given.**

**The required sensitivity values are located in *Supplementary Table 1*
    (the row titled "Our work"). Use the percentage value (e.g., 190 %)
    as `sensitivity_numerator` and the corresponding detection-limit
    value (e.g., 20 ppm) as `sensitivity_denominator`; ignore any
    asterisk footnotes.**

**If multiple sensitivity values appear elsewhere in the text, choose the
     one that pertains to the sensor described in the "Current Work"
    section (the dEGFET sensor with electropolished Cu electrodes).** **
    If several sensitivity percentages are reported for that sensor,
    select the highest percentage and use the corresponding concentration
     reported with that maximum value.**

**If the paper gives a sensitivity like "<value> mV/pH" (or "<value> mV
    per pH unit"), put the number with its unit in `sensitivity_numerator
    ` and "1 pH" (or "1 dec" for decade) in `sensitivity_denominator`.
    Use the primary sensor's value if several are listed.**

**Specifically, when a linear relationship is expressed as a slope (e.g.,
     "DeltaV_Dirac = 9.877 mV per decade"), treat the slope value with
    its unit as `sensitivity_numerator` and set `sensitivity_denominator`
     to "1 dec".**

**For the sensitivity fields in FET gas-sensor papers, locate the
    percentage change in drain current reported for the lowest NO2
    concentration (e.g., "6.68% for 1 ppm"). Use that percentage as `
    sensitivity_numerator` and the corresponding concentration (including
     its unit) as `sensitivity_denominator`.**

If the paper reports more than one sensor (different MOF films, gases, or
     operating conditions), **output a JSON record for the sensor with
    the highest NO2 detection sensitivity, filling in the fields for that
     sensor.**

**Pattern rule:** For each required field, find the **first** numeric
    value that is **immediately followed** by the appropriate unit--'s'
    for `response_time` and `recovery_time`, '%' for `
    sensitivity_numerator`, and a concentration unit such as 'ppm', 'ppb
    ', 'ppbv', 'uM', 'mM', or 'M' for `sensitivity_denominator`. Apply
    any unit conversions described above. If no such pattern exists, set
    the field to an empty string.

**If the denominator is given in log[DA], convert it to ppm** using the
    conversion factor provided in the paper; if the paper does not supply
     a factor, assume 1 log[DA]$\approx$0.153 ppm.

For instance:
{
  "records": [
    {
      "response_time": "xx s",
      "recovery_time": "xx s",
      "sensitivity_numerator": "(absolute change in drain current DeltaI
          per pH unit)",
      "sensitivity_denominator": "(the reference value, e.g. R0, baseline
           current, or gas concentration). **If expressed as a
          concentration (M, mM, uM), convert to ppm before reporting.**"
    }
    // continue if the publication has recorded multiple different
        performance measurements
  ]
}
```

Synthesis and Thermal Processing (5 fields) Human initial prompt

```
After reading the scientific publication full text provided above, please
     try to generate formatted JSON file for extraction of important
    information. Leave "" for not available fields.

For instance:
{
"records": [
    {
      "annealing_temperature": "xx degC",
      "annealing_time": "xx h",
      "annealing_atmosphere": "(e.g. air, N2, Ar, O2, vacuum, H2)",
      "hydrothermal_temperature": "xx degC",
      "hydrothermal_time": "xx h"
    }
   (continue if the publication has recorded multiple different thermal
       processing conditions)
  ]
}
```

Optimized prompt (M8, GPT-oss-120B critique, DS70B inference)

```
Here is the JSON file capturing the hydrothermal synthesis details for
    the ZnO nanosheets described in the passage. After reading the
    scientific publication full text provided above, extract any
    explicitly labeled annealing (if present) and hydrothermal heating
    details (temperature, duration, atmosphere) and generate a formatted
    JSON file. If a liquid-phase treatment (e.g., mixed acid, aqueous
    solution) is given with temperature and time, record them as
    hydrothermal_temperature and hydrothermal_time; be sure to extract
    the hydrothermal reaction duration (e.g., "90 degC for 1.5 h") and
    place it in hydrothermal_time. Extract the relevant records and
    format them as JSON, filling each field only when the information is
    present.

For instance:
{
  "records": [
    {
      "annealing_temperature": "xx degC",
      "annealing_time": "xx h",
      "annealing_atmosphere": "(e.g. air, N2, Ar, O2, vacuum, H2)",
      "hydrothermal_temperature": "xx degC",
      "hydrothermal_time": "xx h"
    }
    (continue if the publication has recorded multiple different thermal
       processing conditions)
  ]
}
If the manuscript does not mention an annealing step, **do not infer or
    estimate**; set annealing_temperature, annealing_time, and
    annealing_atmosphere to empty strings (""). Likewise, if no
    hydrothermal step is described, **do not infer or estimate**; leave
    hydrothermal_temperature and hydrothermal_time empty. If no explicit
    values are found, **do not infer or estimate**; leave the
    corresponding fields empty (""), and if no annealing or hydrothermal
    step is described, output a list containing a single record where all
     fields are empty strings. **Return only a valid JSON object (no
    markdown code fences).**
```