DePHI: Dynamically Evolving System Framework for PHI-Safe and Auditable LLM Generation

Anonymous Author(s)
Anonymous Institution
anonymous@example.com

Abstract

Ensuring regulatory compliance of large language models (LLMs) remains a critical challenge in safety-sensitive domains such as healthcare, where violations involving protected health information (PHI) may lead to severe legal and ethical consequences. Existing approaches, including heuristic filtering, static de-identification pipelines and post-hoc safety classifiers, provide limited support for adaptive enforcement, system-level feedback, and reproducible audit evidence tied to explicit regulatory rules. We propose DePHI (**D**ynamically **E**volving system framework for **PHI**-safe and auditable LLM generation), a multi-layer system that supports HIPAA-oriented PHI compliance through rule modeling, multi-stage detection, and iterative threshold adaptation. At its core, DePHI dynamically adjusts compliance decisions by integrating evidence-driven rule modeling, multi-stage PHI detection, and feedback-based threshold adaptation, enabling the system to respond to changing inputs and observed errors. We evaluate DePHI on clinical text from MIMIC-IV and EHRNoteQA, measuring (i) convergence of regulatory rule sets extracted from HIPAA documents, (ii) PHI detection under controlled and adversarial settings, and (iii) system behavior across utility-safety tradeoffs. We observe: (i) consistent convergence behavior of the Rule Graph (Jaccard similarity from 0.095 to 1.0), (ii) the observed false negative rate stays within 0.04–0.14 across iterative adversarial audit cycles, and (iii) accurate PHI detection (F1 = 0.9904) with near-zero false negative rates under the EHRNoteQA controlled-injection protocol. DePHI demonstrates that regulatory compliance in LLM systems can be operationalized through structured, multi-layer designs that combine rule grounding, detection, and feedback adaptation, providing a practical and auditable foundation for evaluating PHI-safe LLM generation in healthcare settings.

1 Introduction

Large language models (LLMs) have demonstrated strong capabilities across a wide range of natural language processing tasks [8, 47], including clinical text understanding, medical question answering, and decision support [33, 24, 23]. Advances in instruction tuning and alignment have further improved their practical usability in real-world applications [29, 37, 5].

However, deploying LLMs in healthcare introduces critical risks related to the generation or leakage of protected health information (PHI). Prior work shows that LLMs can memorize and reproduce sensitive data [9], while more recent studies highlight privacy risks arising from contextual inference, adversarial prompting, and red-teaming scenarios [26, 31, 48]. Under regulatory frameworks such as HIPAA [40], even rare failures can have significant legal and ethical consequences, making reliable compliance a central requirement for deployment.

Existing mitigation approaches fall into three broad categories. First, de-identification pipelines remove sensitive entities using rule-based or hybrid methods [27, 28]. Second, privacy-preserving training and contextual privacy learning aim to reduce leakage during model training or adaptation [13, 1, 49]. Third, post-hoc filtering systems and guardrails detect violations after generation [38, 31, 26].

Preprint.

While effective in isolation, these approaches are typically static, provide limited traceability to regulatory rules, and do not adapt based on observed failures or evolving policy requirements.

We instead view compliance as a *dynamic, audit-driven control problem*, where regulatory rules, detection signals, and observed failures jointly determine system behavior over time. This perspective requires integrating (i) rule formalization grounded in regulatory text, (ii) robust detection of PHI, and (iii) feedback mechanisms that adjust enforcement based on empirical outcomes. Recent work has explored policy-to-rule translation and regulatory grounding using LLMs [15, 12], but these efforts focus on rule extraction rather than system-level enforcement and adaptation.

To address this gap, we propose DePHI (**D**ynamically **E**volving system framework for **PHI**-safe and auditable LLM generation), a multi-layer framework for HIPAA-oriented PHI-safe generation in LLM systems. DePHI integrates five components into a closed-loop architecture. First, it constructs a Rule Graph from statutory text using a stochastic Bernoulli gating mechanism, which we define as the Bernoulli Regulatory Bottleneck (BRB), to admit rules based on supporting evidence [16, 21, 36]. Second, it detects PHI using keyword, pattern-based, and neural signals [3, 28]. Third, it applies fail-closed enforcement by rejecting outputs identified as high-risk. Fourth, it adapts the detection threshold at each audit cycle using the empirical false negative rate (FNR) via constrained optimization [2]. Finally, it produces versioned compliance records with uncertainty summaries using statistical estimation methods [14, 35].

A key aspect of DePHI is that rule construction and enforcement are not treated as one-shot processes: rules are versioned across regulatory snapshots, while audit feedback adapts the runtime threshold. We evaluate DePHI on clinical text from MIMIC-IV [20] and EHRNoteQA [23], as well as controlled adversarial scenarios designed to stress PHI detection. Empirically, we observe that the Rule Graph stabilizes as regulatory evidence increases, FNR dynamics remain feedback-responsive under iterative auditing, and the system exposes a controllable tradeoff between utility and safety under different operating regimes.

In summary, our contributions are threefold: we formulate PHI compliance in LLM systems as a dynamic, closed-loop control problem integrating rule modeling, detection, and feedback; we design a multi-layer architecture that combines symbolic rules, probabilistic filtering, and neural detection, while empirically identifying the semantic boundary cases that remain challenging under adversarial and evolving conditions; and we provide empirical evaluation on MIMIC-IV and EHRNoteQA, demonstrating stable rule construction, strong detection of explicit and structured PHI, and controllable utility-safety tradeoffs under controlled settings. The code is available at this anonymous repository: https://anonymous.4open.science/DePHI.

2 Related Work

2.1 PHI Detection and Clinical NLP

A large body of work has focused on detecting PHI in clinical text. Early approaches relied on rule-based or hybrid systems combining dictionaries and pattern matching [27]. Neural models such as BioBERT [24] and ClinicalBERT [3] are widely used for clinical entity recognition and text understanding. More recent work has shifted toward LLM-based clinical reasoning and evaluation, including medical question answering and realistic EHR-based benchmarks such as EHRNoteQA [33, 23]. Hybrid systems such as Philter [28] combine symbolic and learning-based components and achieve high recall in de-identification tasks. However, these methods are typically standalone pipelines that do not integrate regulatory rules or support adaptive, system-level compliance enforcement. In contrast, our proposed DePHI incorporates PHI detection as one component of a larger compliance system, combining symbolic and neural methods within a multi-stage pipeline integrated with rule-based constraints.

2.2 Regulatory Text Processing and Rule Extraction

Prior work has explored automated extraction of normative phrases from regulatory text, achieving strong performance in identifying policy-relevant statements [22]. More recent work has extended legal and regulatory text processing through domain-adapted pretrained language models, LLM-based legal reasoning benchmarks, privacy-law grounding frameworks, and policy-to-rule formalization pipelines [10, 18, 15, 12]. Recent work includes GoldCoin [15], which grounds LLMs in privacy laws, and P2T [12], which translates policy documents into executable rules using LLM-based extraction and validation. These works are closely related to DePHI's rule-construction layer, but

primarily address rule grounding or policy translation rather than integrating rule construction with downstream enforcement and feedback-driven adaptation. DePHI differs by treating rule construction as an incremental, evidence-driven process: candidate rules are probabilistically admitted via the BRB, versioned across regulatory snapshots, and linked to downstream audit evidence.

2.3 LLM Privacy, Safety, and Guardrails

Ensuring safe behavior in LLMs has been widely studied through reinforcement learning from human feedback [29], constitutional AI [5], and red-teaming or adversarial evaluation [31, 48]. Recent privacy-focused work shows that LLMs can fail to preserve contextual privacy even when information is not conventionally considered sensitive, motivating benchmarks such as CONFAIDE [26]. Other work explores LLMs as contextual privacy protection learners, showing that privacy-sensitive behavior can be improved through instruction tuning [49]. These approaches focus on model-internal alignment, whereas DePHI operates as an external, auditable control layer.

2.4 Constrained Optimization and Uncertainty Quantification

Constrained optimization has been widely used to enforce safety constraints in machine learning systems. Approaches based on constrained Markov decision processes [4] and constrained policy optimization [2] provide principled frameworks for balancing performance and safety. Our DePHI builds on these ideas by formulating compliance as a constraint on the FNR and using dual ascent to dynamically adjust detection thresholds. Our setting operates as an inference-time control loop over a compliance pipeline, rather than a full environment interaction problem. Providing reliable uncertainty estimates is also essential for safety-critical systems. Bootstrap methods [14], exact binomial confidence intervals [7], and conformal prediction under distribution shift [35] provide statistical tools for quantifying uncertainty. DePHI integrates these techniques to produce auditable compliance certificates, using uncertainty estimates as regulatory evidence rather than merely as prediction confidence.

3 Method

3.1 Problem Formulation

Let $x \in \mathcal{X}$ denote an input, such as a clinical query, retrieved context, or patient-record excerpt, and let $y = \pi(x)$ denote the output generated by a fixed LLM policy π. We assume inputs are drawn from an evaluation or deployment distribution $\mathcal{D}$ over clinical contexts. DePHI does not optimize or update π; instead, it applies and adapts an inference-time filtering mechanism that decides whether a generated output should be allowed or rejected.

Let $U(y, x)$ denote downstream task utility, such as whether the generated answer preserves the correct EHRNoteQA response. Let $Y_\tau(y) \in \{0, 1\}$ denote the DePHI detection decision under threshold τ, where $Y_\tau(y) = 1$ indicates that the output is detected as PHI-positive and should be

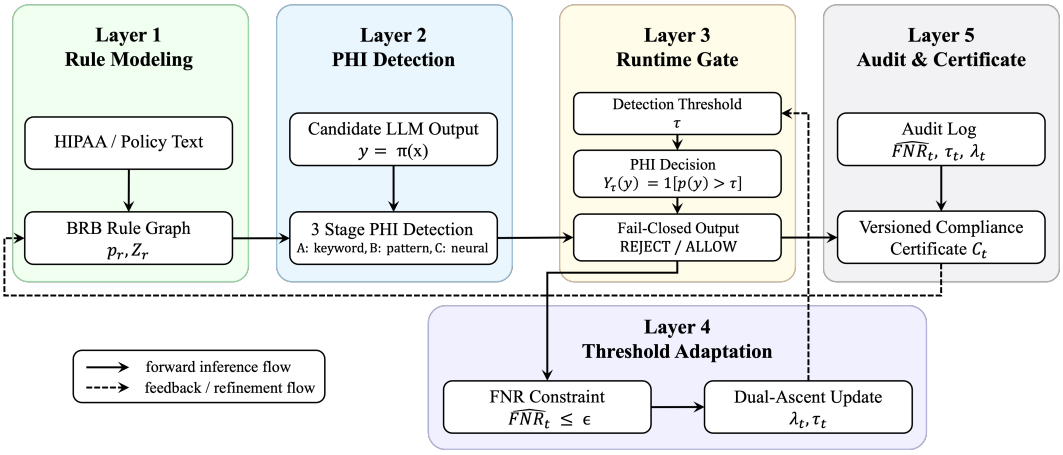

Figure 1: Overview of DePHI. Layer 1 converts regulatory text into a BRB-calibrated Rule Graph; Layer 2 evaluates candidate LLM outputs using a three-stage PHI detector; Layer 3 applies a fail-closed runtime decision; Layer 4 updates the operating threshold from empirical audit feedback; and Layer 5 records versioned audit evidence and compliance certificates. **Key takeaway:** DePHI frames HIPAA compliance as a closed-loop control problem rather than a static post-hoc filter.

rejected, and $Y_\tau(y) = 0$ indicates that no PHI is detected by the compliance gate. We formulate compliance-aware generation as a constrained decision problem over the filtering threshold:

$$\max_{\tau} \ \mathbb{E}_{x\sim\mathcal{D}}[U(\pi(x), x)] \quad \text{s.t.} \quad \mathrm{FNR}(\tau) \le \epsilon, \tag{1}$$

where τ is the detection threshold applied to output y, and ϵ is the target upper bound on the FNR [32]:

$$\mathrm{FNR}(\tau) = \Pr(Y_\tau(y) = 0 \mid y \text{ contains PHI}) = \frac{\mathbb{E}[\mathbb{1}(y \text{ contains PHI} \wedge Y_\tau(y) = 0)]}{\mathbb{E}[\mathbb{1}(y \text{ contains PHI})]}. \tag{2}$$

In practice, DePHI estimates $\mathrm{FNR}(\tau_t)$ using $\widehat{\mathrm{FNR}}_t$, as defined in equation (6), from injected PHI annotations or audited outputs. The LLM weights remain fixed throughout.

3.2 DePHI Architecture

Figure 1 summarizes the five-layer inference-time pipeline of DePHI.

Layer 1: Rule Graph construction.

DePHI first extracts candidate regulatory rules $\mathcal{R} = \{r_1, \ldots, r_N\}$ from statutory and policy text. Each rule r is associated with an anchor set $\mathcal{A}_r$, which contains evidence about whether the extracted statement corresponds to a prohibitive PHI-related constraint. In our implementation, anchors are aligned to Code of Federal Regulations (CFR) identifiers. Let $n_r = |\mathcal{A}_r|$ denote the number of anchors aligned to rule r, and let k_r denote the number of those anchors indicating that r expresses a relevant PHI restriction. Rule validity is estimated using the Beta-Binomial posterior mean:

$$p_r = \frac{\alpha_0 + k_r}{\alpha_0 + \beta_0 + n_r}, \tag{3}$$

where $p_r \in [0, 1]$ is the admission probability. We use a prior distribution $\mathrm{Beta}(\alpha_0, \beta_0)$ with $\alpha_0 = 24$ and $\beta_0 = 1$, corresponding to a prior mean of approximately 0.96. This prior is motivated by prior work reporting that approximately 96% of extracted normative statements from HIPAA regulatory text were judged correct under their annotation protocol [22]. Importantly, this value is used only as an initialization for the BRB posterior; it does not assume that all extracted statements are valid prohibitive PHI constraints. Candidate rules are subsequently filtered using evidence anchors and Bernoulli admission.

We introduce the *Bernoulli Regulatory Bottleneck* (BRB), a DePHI-specific stochastic rule-admission mechanism inspired by Bayesian uncertainty modeling and stochastic bottlenecks [36, 21, 16]. Specifically, for each candidate rule, BRB samples $Z_r \sim \mathrm{Bernoulli}(p_r)$ and admits r when $Z_r = 1$ and $p_r \ge \tau_{\text{rule}}$. We set $\tau_{\text{rule}} = 0.50$ as a minimum evidence threshold, so candidates with posterior support below chance-level admission are rejected; rules with $p_r \ge 0.85$ are admitted deterministically to avoid unnecessary stochastic rejection of high-confidence candidates. The admitted rules form a versioned Rule Graph $\mathcal{G} = (\mathcal{R}', \mathcal{E})$, where $\mathcal{R}' \subseteq \mathcal{R}$ and $\mathcal{E}$ encodes shared PHI categories, source citations, and regulatory dependencies. Operationally, we add an edge $(r_i, r_j) \in \mathcal{E}$ when two admitted rules share a CFR citation, share a PHI category, or have the Jaccard similarity, $J(r_i, r_j) = |T_{r_i} \cap T_{r_j}|/|T_{r_i} \cup T_{r_j}|$, above $\delta = 0.30$ over trigger-condition token sets T_{r_i} and T_{r_j}. This yields a versioned, evidence-weighted rule state. Additional implementation details on anchor construction and Rule Graph assembly are provided in Appendix A.

Layer 2: Multi-stage PHI detection. Given a candidate output y, DePHI estimates PHI risk using three complementary detection signals. Stage A performs keyword-trie matching over 78 PHI-related terms organized by the PHI categories represented in the Rule Graph, producing $s_A(y) \in \{0, 1\}$; Stage B applies 20 structured regex templates for identifiers, dates, phone numbers, and address-like expressions, producing $s_B(y) \in \{0, 1\}$; and Stage C uses a fine-tuned Bio_ClinicalBERT to produce a neural PHI score $p_C(y) \in [0, 1]$. These signals are combined using a conservative max aggregation:

$$p(y) = \max(s_A(y), s_B(y), p_C(y)), \qquad Y_\tau(y) = \mathbb{1}[p(y) > \tau], \tag{4}$$

where $p(y)$ is the aggregated PHI risk score and $Y_\tau(y)$ is the thresholded PHI detection decision. Detailed setup and fine-tuning information are provided in Appendix B.

Layer 3: Fail-closed runtime enforcement. Layer 3 converts the PHI detection decision into an output-level compliance action:

$$\mathrm{Decision}(y) = \begin{cases} \text{REJECT}, & Y_\tau(y) = 1, \\ \text{ALLOW}, & Y_\tau(y) = 0. \end{cases} \tag{5}$$

The gate is fail-closed: invalid detector states or unreliable decisions default to REJECT, reducing missed disclosures at the cost of possible over-rejection. When an output is rejected, DePHI returns a compliance refusal message and does not expose the original generated text to the user.

Layer 4: Constrained threshold adaptation. DePHI adapts the detection threshold τ using audit feedback. We use the standard FNR definition $\mathrm{FNR} = \mathrm{FN}/(\mathrm{FN} + \mathrm{TP})$, which measures the fraction of true PHI-containing outputs that are incorrectly passed by the gate [32]. At audit cycle t, DePHI estimates $\mathrm{FNR}(\tau_t)$ in (2) from injected-PHI or audited samples by

$$\widehat{\mathrm{FNR}}_t = \frac{\sum_i \mathbb{1}[y_i \text{ contains PHI} \wedge Y_{\tau_t}(y_i) = 0]}{\sum_i \mathbb{1}[y_i \text{ contains PHI}]}. \tag{6}$$

Here, $Y_{\tau_t}(y_i) = 1$ denotes that the gate flags y_i as PHI-positive, so the denominator is the true PHI-containing set rather than the predicted-negative set. Thus, $\widehat{\mathrm{FNR}}_t$ is the sample estimate of $\mathrm{FNR}(\tau_t)$ and differs from false non-discovery rate or false-omission-style quantities, which condition on predicted negatives (e.g., $\mathrm{FN}/(\mathrm{FN} + \mathrm{TN})$) [6]. DePHI maintains a nonnegative dual variable $\lambda_t \geq 0$ and updates the operating threshold using

$$\lambda_{t+1} = \mathrm{clip}\Big(\lambda_t + \eta(\widehat{\mathrm{FNR}}_t - \epsilon), 0, \lambda_{\max}\Big), \qquad \tau_{t+1} = \mathrm{clip}\left(\frac{\tau_t}{1 + \lambda_{t+1}}, \tau_{\min}, \tau_{\max}\right). \tag{7}$$

We use $\eta = 0.01$, $\lambda_{\max} = 10$, $\tau_{\min} = 0.1$, and $\tau_{\max} = 0.9$. For adversarial-cycle experiments, we set the target upper bound $\epsilon = 0.05$ as a conservative operational risk target for threshold adaptation. This threshold is used for empirical control in our experiments and should not be interpreted as a legal safe harbor. If $\widehat{\mathrm{FNR}}_t > \epsilon$, the dual variable increases and τ_t decreases, making the gate more conservative; otherwise, the threshold stabilizes. Only τ_t is adapted; LLM weights remain fixed.

Layer 5: Audit records and uncertainty quantification. Layer 5 records audit statistics and attaches an uncertainty interval to the estimated FNR. For regulatory-style reporting, DePHI uses the Clopper–Pearson exact binomial interval [11] as the default confidence interval because it is conservative and appropriate for finite-sample binomial error counts. Let $\mathrm{CI}_t = [L_t, U_t]$ denote the Clopper–Pearson interval computed from the number of false negatives and the number of PHI-positive audited samples at audit cycle t. DePHI stores a versioned audit certificate

$$\mathcal{C}_t = (\widehat{\mathrm{FNR}}_t, [L_t, U_t], \tau_t, \lambda_t, \mathrm{RuleGraphVersion}_t), \tag{8}$$

which records the empirical FNR estimate, conservative uncertainty interval, operating threshold, dual variable, and Rule Graph version used at cycle t. The purpose of this certificate is traceability and reproducibility across system versions, rather than a formal proof of universal compliance. Formal interval definitions and audit-record construction details are provided in Appendix E.

4 Experiments

4.1 Experimental Setup

We evaluate DePHI on clinical text and controlled adversarial scenarios, focusing on system-level behavior rather than isolated module performance. Experiments use clinical notes derived from MIMIC-IV [20], a large-scale de-identified EHR dataset from ICU patient records, and EHRNoteQA [23] for downstream utility evaluation. DePHI uses the full three-stage detection pipeline: keyword matching, structured pattern matching, and neural PHI detection. The neural detector is initialized from Bio_ClinicalBERT [3] and fine-tuned for token-level PHI/non-PHI classification on MIMIC-IV text with synthetic PHI injections; full details are given in Appendix B.

The three detection stages are designed to capture complementary failure modes. Stage A is expected to detect explicit lexical PHI mentions with minimal latency, but can miss obfuscated or format-shifted identifiers. Stage B targets structured PHI such as dates, IDs, phone numbers, and address-like patterns, but is less effective for indirect or semantic references. Stage C provides a neural semantic backstop for weakly lexicalized PHI, while its effectiveness depends on the coverage of fine-tuning data and the availability of representative PHI annotations. We further conduct a within-system component ablation study to isolate the marginal contribution of each detection stage under a fixed gate, metric, and injection protocol.

DePHI runs as an inference-time compliance layer without updating LLM weights. Stage A and B perform dictionary and regex matching over 98 rules/patterns, adding 2.2 ms on CPU; Stage C

invokes a 107.7M-parameter Bio_ClinicalBERT detector when neural scoring is required. On an NVIDIA H200 GPU, Stage C takes 3.3 ms for medium-length outputs, and because Stage A and B intercept 40% of outputs, the effective Stage C overhead is about 2.0 ms, yielding approximately 4.2 ms total overhead per output. The one-time three-model BRB annotation protocol costs 150 API calls, about 24,000 tokens, and about $0.06 for 50 candidate items; additional profiling is provided in Appendix F.

4.2 Experiment 1: Rule Graph Convergence and Annotation Stability

Because deontic classification in legal text is ambiguous, we evaluate both Rule Graph stability under incremental regulatory expansion and annotation reliability for BRB candidates.

We process six sequential HIPAA-related document snapshots, denoted $t_1, \ldots, t_6$, through the same Layer 1 pipeline: candidate rule extraction, anchor construction, BRB posterior scoring, Bernoulli admission, and Rule Graph construction. The snapshots progressively expand the regulatory corpus, starting from the HIPAA statutory baseline and Privacy Rule materials, and then incorporating HITECH and Omnibus updates, OCR access guidance, Security Rule guidance, and the 2024 Reproductive Health Care Privacy final rule ([40, 43, 39, 41, 44, 45, 42], Appendix G.1). Across the full corpus, the pipeline produces 186 candidate rules and 872 anchor samples, from which BRB admits a final library of 89 rules; posterior scores are used only as graded evidence for admission rather than as deterministic acceptance of extracted statements.

To evaluate annotation reliability, we apply a TinyV-inspired three-annotator LLM verification protocol [50] using Claude Sonnet 4.6 Extended, Gemini 3.1 Pro, and GPT-5.5. Each annotator classifies 50 BRB candidates as prohibitive or non-prohibitive under explicit deontic definitions; only full-consensus items are used as deterministic ground truth, while disputed cases remain probabilistic BRB labels. The three-model protocol reaches full consensus on 47/50 items, with 24 unanimously prohibitive items and 23 unanimously non-prohibitive items. Three items remain disputed: `ITEM_007`, `ITEM_046`, and `ITEM_049`, specified in Appendix G.2. Agreement remains high, with Fleiss' $\kappa = 0.92$, corresponding to almost perfect agreement. Pairwise Jaccard similarity over the YES sets ranges from 0.889 to 0.960: Claude vs. Gemini = 0.960, Claude vs. GPT-5.5 = 0.889, and Gemini vs. GPT-5.5 = 0.923. Because all three annotators evaluated the same candidates under identical deontic definitions and questionnaire prompts, this high agreement supports the stability of the BRB annotation protocol, while prior work shows that deontic modality classification in legal text remains challenging even in expert or legal-domain settings [25]. This result supports the use of a probabilistic BRB gate: residual disagreement reflects genuine deontic uncertainty rather than annotation failure.

Let $\mathcal{R}_t$ denote the set of rules admitted into the Rule Graph after processing regulatory snapshot t, and let $\mathcal{R}_6$ denote the final Rule Graph constructed from the full document set. We quantify convergence by measuring the Jaccard similarity between each intermediate rule set and the final rule set:

$$J(\mathcal{R}_t, \mathcal{R}_6) = \frac{|\mathcal{R}_t \cap \mathcal{R}_6|}{|\mathcal{R}_t \cup \mathcal{R}_6|}.$$

This measures how closely the Rule Graph constructed from a partial regulatory corpus matches the Rule Graph obtained after all regulatory documents are incorporated. As shown in Table 1(a), Jaccard similarity increases monotonically from 0.095 to 1.000 across the six regulatory snapshots, and the Rule Graph stabilizes at 21 regulatory sections in later stages. This indicates that BRB-based rule admission becomes increasingly stable as regulatory evidence accumulates, rather than oscillating as new documents are added. Together, these results show stable Rule Graph construction while preserving genuine legal ambiguity when it arises; extended details are in Appendix G.1.

4.3 Experiment 2: FNR Dynamics Under Adversarial Augmentation

Layer 4 adapts the detection threshold under changing PHI risk. We evaluate this behavior using an adversarial PHI stream over 20 audit cycles, with 50 generated samples per cycle, for a total of 1000 samples. Samples are balanced across five PHI categories: `NAME`, `DATE`, `ID`, `CONTACT`, and `LOCATION`. Across cycles, the adversarial generator increases difficulty through abbreviation, partial masking, paraphrasing, format variation, and compositional mixing.

At each cycle t, DePHI applies the full detection pipeline, estimates $\widehat{\mathrm{FNR}}_t$ against injected PHI ground truth, and updates τ_t using the Layer 4 dual-ascent rule with target $\epsilon = 0.05$. Throughout the 20 audit cycles, the underlying LLM generator, the three-stage detection modules, and the Rule

Table 1: Experiments 1–2: Rule stability, annotation agreement, and FNR dynamics.

(a) Rule Graph + annotation stability

Metric	t_1	t_2	t_3	t_4	t_5	t_6
$J(\mathcal{R}_t, \mathcal{R}_6)$	0.095	0.210	0.440	0.670	0.880	1.000

Annot.	Cons.	Y/N	Disputed	Jaccard	κ
3-model	47/50	24/23	3	0.889–0.960	0.92

(b) FNR dynamics under adversarial cycles

Metric	Value
Cycles / samples / PHI types	20 / 1000 / 5
Target ϵ / FNR range	0.05 / [0.04, 0.14]
Mean FNR, early / late	0.071 / 0.086
Peak / final FNR	0.14 / 0.06
Hotspot	NAME (0.20–0.40)

Graph are kept fixed, so no model weights or regulatory rules are updated during this experiment; after each cycle, Layer 4 updates only the runtime gate threshold τ_t using the audited $\widehat{\mathrm{FNR}}_t$.

As shown in Table 1(b), the observed per-cycle $\widehat{\mathrm{FNR}}_t$ stays within [0.04, 0.14] across all 20 cycles. The mean FNR increases from 0.071 in cycles 1–10 to 0.086 in cycles 11–20, reflecting harder adversarial inputs. Although several cycles exceed the target $\epsilon = 0.05$, this is expected because ϵ is used as a feedback target for threshold adaptation rather than a hard per-cycle guarantee; each cycle contains only 50 adversarial samples, so category composition and obfuscation difficulty can cause transient deviations. After these violations, the controller tightens the operating threshold, and the final-cycle FNR decreases to 0.06. This indicates feedback-driven correction after observed violations, not strict per-cycle satisfaction of ϵ or a formal convergence guarantee at this sample scale.

The main hotspot is `NAME` (FNR 0.2–0.4), where adversarial variants appear as initials, abbreviations, partial mentions, or context-dependent references. In contrast, Stages A and B—keyword matching and structured pattern matching—more reliably detect explicit identifiers, dates, contacts, and location-like strings. These failures motivate Stage C, the Bio_ClinicalBERT neural detector, which targets weakly lexicalized or semantic PHI mentions not captured by deterministic rules.

Thus, Experiment 2 shows empirical stabilization under the evaluated adversarial stream, not a formal guarantee that all future FNR values remain below ϵ; full per-cycle and category-level results are provided in Appendix G.3.

4.4 Experiment 3: Utility-Safety Tradeoff

We evaluate whether DePHI exposes the tradeoff between downstream utility and PHI safety under controlled injection. Using 100 EHRNoteQA samples, we inject two PHI entities into each context and prompt Claude Sonnet 4.6 Extended to generate free-text responses. We compare an ungated baseline against DePHI at two thresholds, $\tau = 0.4$ and $\tau = 0.7$. Utility is measured as *post-gate effective* EHRNoteQA multiple-choice accuracy, where rejected outputs are counted as non-answers rather than evaluated for clinical reasoning; safety is measured as string-match FNR against injected PHI ground truth, and block rate is the fraction of outputs rejected by the fail-closed gate.

As shown in Table 2(a), the ungated baseline achieves 0.43 accuracy but has high leakage risk (FNR = 0.60). DePHI reduces FNR to 0.02 at $\tau = 0.4$ and 0.00 at $\tau = 0.7$, with a 97% block rate; the low effective accuracy reflects blocked outputs counted as non-answers, not degraded clinical reasoning. Stage C achieves 0.9904 F1 on the held-out synthetic PHI validation split. These results reflect a conservative safety-first regime: injected PHI is mostly explicit names, dates, and identifiers intercepted by Stages A and B, so both thresholds yield the same 97% block rate and τ mainly affects residual cases. A detailed interpretation is provided in Appendix G.4.

4.5 Experiment 4: Adversarial Robustness

We evaluate whether DePHI detects PHI under adversarial transformations while avoiding false positives on clean clinical text. Unlike Experiment 2, which studies FNR dynamics across audit cycles, this experiment is designed as a taxonomy-based failure-mode audit: each adversarial class corresponds to a distinct PHI-obfuscation mechanism. We evaluate 10 transformation classes, including prompt injection, split-token obfuscation, written dates, spoken identifiers, abbreviation, Unicode substitution, profession-name ambiguity, contextual embedding, indirect references, and spoken phone numbers. For each class, we generate 50 PHI-positive variants, yielding 500 adversarial samples in total, and evaluate false positives on 500 PHI-negative clean clinical samples.

The full DePHI pipeline is used for evaluation, with Stage A keyword matching, Stage B structured pattern matching, Stage C neural PHI detection, and the fail-closed runtime gate enabled. A sample is counted as detected if any stage identifies PHI and the runtime gate rejects the output. This

Table 2: Experiments 3–4: Utility–safety tradeoff and adversarial robustness. In Experiment 3, accuracy denotes post-gate effective task accuracy, where rejected outputs are counted as non-answers. Experiment 4 evaluates both attack-class coverage and sample-level detection under controlled adversarial PHI transformations.

(a) Utility–safety tradeoff

System	Eff. Acc.	FNR	Block
Ungated	0.43	0.60	0%
DePHI ($\tau = 0.4$)	0.02	0.02	97%
DePHI ($\tau = 0.7$)	0.03	0.00	97%

Protocol: 100 EHRNoteQA items, 2 PHI/item; blocked outputs count as non-answers.

(b) Adversarial robustness

Metric	Outcome
Attack classes	10
Adversarial samples	500
Detected classes	8/10
Detected samples	400/500
Clean false positives	0/500
Primary failures	indirect ref.; spoken phone

Protocol: 50 variants/class; 500 clean samples.

Table 3: Stage-level ablation of the DePHI detection cascade on an expanded balanced diagnostic set. The PHI-positive set contains 500 samples stratified as Stage-A pool: 200, Stage-B pool: 200, and Stage-C pool: 100, with 500 PHI-negative clean samples.

Variant	**TP/FN**	**FP/TN**	**Pool detection (A/B/C)**	**FNR ↓**	**F1 ↑**
Stage A only	226/274	0/500	166/40/20	0.548	0.623
Stages A+B	326/174	0/500	166/140/20	0.348	0.789
Full DePHI (A+B+C)	326/174	0/500	166/140/20	0.348	0.789

protocol measures both attack-class coverage and sample-level robustness under controlled adversarial transformations.

As shown in Table 2(b), DePHI detects all samples in 8 out of 10 adversarial transformation classes, corresponding to 80% attack-class coverage and 400/500 detected adversarial samples. It also produces zero false positives on 500 clean samples. The detected classes include prompt injection, split obfuscation, written dates, spoken SSNs, abbreviation, Unicode substitution, profession-name ambiguity, and context-embedded PHI.

The two failure classes are indirect references and spoken-form phone numbers. Both require semantic normalization beyond surface lexical or structural matching: indirect references may omit complete identifiers, while spoken phone numbers require mapping word sequences back to numeric contact patterns. Thus, the failures are concentrated in weakly lexicalized semantic cases rather than distributed across all attack types. This supports the multi-stage design of DePHI: deterministic Stages A and B provide strong coverage for explicit and structured PHI, while the remaining errors identify concrete targets for improving semantic Stage C detection and future deployment-scale red-teaming.

4.6 Ablation Study

We conduct a within-system ablation to quantify how each stage of the runtime detection cascade contributes to PHI-safe output decisions. The ablation is evaluated on an expanded balanced diagnostic set with 500 PHI-positive adversarial samples and 500 PHI-negative clean samples. The PHI-positive subset is deliberately stratified into three pools: 200 keyword-dominant samples targeting Stage A, 200 regex-dominant samples targeting Stage B, and 100 semantic-only samples targeting Stage C. This design evaluates component behavior under controlled failure-mode coverage rather than estimating deployment-wide FNR. Importantly, the available public clinical data are highly de-identified and professionally curated, so they contain few naturally occurring, messy, weakly lexicalized PHI cases that would allow Stage C to show its full semantic advantage; the Stage-C pool is therefore used as a diagnostic stress test rather than a claim of complete semantic coverage.

To isolate the marginal contribution of the detection cascade under a fixed interface, we evaluate three executable configurations: **Stage A only**, which uses keyword matching; **Stages A+B**, which adds structured pattern matching for identifiers, dates, contact expressions, and related formats; and **Full DePHI (Stages A+B+C)**, which additionally includes the neural PHI detector. We do not report unexecuted variants such as "w/o BRB" in this quantitative table: BRB is evaluated separately in Experiment 1 through Rule Graph convergence and annotation agreement, while this ablation focuses on runtime PHI detection.

Table 3 shows that Stage A alone detects 226/500 PHI-positive samples, with strongest performance on the keyword-dominant pool. Adding Stage B increases detection from 226 to 326 true positives and reduces FNR from 0.548 to 0.348, primarily by improving coverage on the regex-dominant pool from 40/200 to 140/200. This confirms that structured pattern matching contributes substantially beyond keyword matching.

Full DePHI matches the aggregate performance of Stages A and B in this diagnostic set, which should be interpreted as a data-boundary result rather than evidence that Stage C is unnecessary. Public clinical benchmarks such as MIMIC-IV are already carefully de-identified, and our injected PHI cases are necessarily controlled; as a result, most detectable positives are either lexical or structurally recoverable by Stages A and B. The remaining false negatives align with the same semantic boundary observed in Experiment 4, especially indirect references and spoken-form contact information. Thus, this ablation shows that Stage C does not introduce false positives under the current threshold, while also identifying the need for stronger semantic normalization and expert-labeled messy PHI data before Stage C can contribute measurable aggregate gains. Ablation scope and limitations are discussed in Appendix H.

4.7 Summary

Across all experiments, DePHI shows stable Rule Graph construction, feedback-responsive FNR dynamics, and an explicit utility–safety tradeoff under fail-closed enforcement. The adversarial and ablation results further localize the main residual failures to semantic transformation classes, especially indirect references and spoken-form contact information, rather than broad failures across PHI categories. These findings support our central claim that compliance behavior in LLM systems is a system-level property arising from rule modeling, detection, feedback control, and auditing, while also identifying Stage C semantic normalization as the main target for future improvement.

5 Discussion and Conclusion

DePHI addresses a gap left by standard PHI de-identification systems, generic LLM guardrails, and one-shot regulatory rule extraction methods. In healthcare deployment, HIPAA-oriented PHI compliance requires more than detecting PHI on a fixed benchmark: the system must connect regulatory rules to runtime decisions, adapt to observed failures, and produce reproducible audit evidence. DePHI operationalizes this view by linking regulatory grounding, multi-stage PHI detection, fail-closed enforcement, threshold adaptation, and versioned audit records into a closed-loop compliance pipeline. This is the central distinction of our framework: compliance is treated as a system-level control problem rather than a static post-hoc filtering task.

This design provides practical advantages over existing approaches. Compared with standalone de-identification pipelines, DePHI treats PHI detection as one component of a broader compliance mechanism. Compared with generic LLM safety filters or alignment methods, it operates as an external inference-time layer, making it compatible with fixed, proprietary, or non-retrainable LLMs. Compared with legal-rule extraction or policy-to-rule translation systems, DePHI uses rule construction as input to runtime enforcement rather than as a terminal output. Empirically, Rule Graph convergence and three-model annotation agreement support the stability of regulatory grounding; adversarial FNR dynamics show that Layer 4 acts as a feedback stabilizer rather than a monotonic-error guarantee; and utility–safety results expose the expected deployment tradeoff under fail-closed enforcement. The expanded adversarial and ablation studies further localize residual failures to semantic boundary cases, such as indirect references and spoken-form contact information, rather than broad failures across PHI categories.

We discuss several scope limitations to help define concrete next steps. Rule labels are based on high-agreement three-model consensus and should be further validated by legal experts for disputed or institution-specific cases; adversarial evaluation is diagnostic and should be expanded through larger red-teaming studies; and public datasets such as MIMIC-IV and EHRNoteQA cannot fully simulate long-horizon institutional feedback dynamics. We discuss deployment extensions and benchmark directions in Appendix I. Overall, DePHI demonstrates that HIPAA compliance for LLM systems is better understood as a systems-level property emerging from rule grounding, detection, enforcement, feedback control, and auditability, rather than as a model-level attribute alone.

References

[1] Martin Abadi et al. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[3] Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, 2019.

[4] Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.

[5] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[6] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1): 289–300, 1995.

[7] Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 2001.

[8] Tom Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[9] Nicholas Carlini et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security)*, 2021.

[10] Ilias Chalkidis et al. Legal-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

[11] Charles J. Clopper and Egon S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

[12] Gautam Varma Datla, Anudeep Vurity, Tejaswani Dash, Tazeem Ahmad, Mohd Adnan, and Saima Rafi. Executable governance for AI: Translating policies into rules using LLMs, 2025. URL `https://arxiv.org/abs/2512.04408`. arXiv preprint.

[13] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3-4):211–487, 2014.

[14] Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7 (1):1–26, 1979.

[15] Wei Fan, Haoran Li, Zheye Deng, Weiqi Wang, and Yangqiu Song. GoldCoin: Grounding large language models in privacy laws via contextual integrity theory. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3321–3343, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.195. URL `https://aclanthology.org/2024.emnlp-main.195/`.

[16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

[17] Deep Ganguli et al. Red teaming language models to reduce harms. *arXiv preprint arXiv:2209.07858*, 2022.

[18] Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, Jessica Wu, Joe Nudell, Joel Niklaus, Jonathan H. Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom Zur, Varun Iyer, and Zehua Li. LegalBench: A

collaboratively built benchmark for measuring legal reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 44123–44279, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/89e44582fd28ddfea1ea4dcb0ebbf4b0-Abstract-Datasets_and_Benchmarks.html.

[19] Dan Hendrycks et al. Aligning AI with shared human values. In *International Conference on Learning Representations (ICLR)*, 2021.

[20] Alistair Johnson et al. MIMIC-IV: A freely accessible electronic health record dataset. *Scientific Data*, 2024.

[21] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *NeurIPS*, 2015.

[22] Nataliya Kiyavitskaya, Nicola Zeni, Luisa Mich, James R. Cordy, and John Mylopoulos. Towards the development of privacy-aware systems. *Information and Software Technology*, 50 (4):244–253, 2008.

[23] Sunjun Kweon et al. EHRNoteQA: An LLM benchmark for real-world clinical practice using discharge summaries. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

[24] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[25] Kristina Minkova, Shashank M. Chakravarthy, and Gijs van Dijck. Low-resource deontic modality classification in eu legislation. In *Proceedings of the Natural Legal Language Processing Workshop 2023*, 2023.

[26] Niloofar Mireshghallah, Hyunwoo Kim, Xuhui Zhou, Yulia Tsvetkov, Maarten Sap, Reza Shokri, and Yejin Choi. Can LLMs keep a secret? testing privacy implications of language models via contextual integrity theory. In *International Conference on Learning Representations (ICLR)*, 2024.

[27] Ishna Neamatullah, Margaret M. Douglass, Li-Wei H. Lehman, Andrew Reisner, Mauricio Villarroel, William J. Long, Peter Szolovits, George B. Moody, Roger G. Mark, and Gari D. Clifford. Automated de-identification of free-text medical records. *BMC Medical Informatics and Decision Making*, 8(32), 2008.

[28] Beau Norgeot, Kiley Muenzen, Tracy A. Peterson, Xiaobing Fan, Benjamin S. Glicksberg, Gregory Schenk, Eric Rutenberg, Boris Oskotsky, Marina Sirota, Jinoos Yazdany, Gabriela Schmajuk, David Ludwig, Tristan Goldstein, and Atul J. Butte. Protected health information filter (Philter): Accurately and securely de-identifying free-text clinical notes. *npj Digital Medicine*, 3(57), 2020.

[29] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[30] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[31] Ethan Perez et al. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[32] David M. W. Powers. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011. ISSN 2229-3981. URL https://researchnow.flinders.edu.au/en/publications/evaluation-from-precision-recall-and-f-measure-to-roc-informednes/.

[33] K. Singhal et al. Towards expert-level medical question answering with large language models. *Nature*, 2023.

[34] Amber Stubbs and Özlem Uzuner. Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/UTHealth corpus. *Journal of Biomedical Informatics*, 58:S20–S29, 2015.

[35] Ryan J. Tibshirani, Rina Foygel Barber, Emmanuel J. Candès, and Aaditya Ramdas. Conformal prediction under covariate shift. In *Advances in Neural Information Processing Systems*, 2019.

[36] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. In *Allerton Conference on Communication, Control, and Computing*, 2000.

[37] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[38] Imdad Ullah, Najm Hassan, Sukhpal Singh Gill, Basem Suleiman, Tariq Ahamed Ahanger, Zawar Shah, Junaid Qadir, and Salil S. Kanhere. Privacy preserving large language models: ChatGPT case study based vision and framework, 2023. URL `https://arxiv.org/abs/2310.12523`. arXiv preprint.

[39] United States Congress. Health Information Technology for Economic and Clinical Health Act, 2009. Title XIII of the American Recovery and Reinvestment Act of 2009.

[40] U.S. Congress. Health Insurance Portability and Accountability Act (HIPAA). Public Law 104-191, 1996.

[41] U.S. Department of Health and Human Services. Modifications to the HIPAA Privacy, Security, Enforcement, and Breach Notification Rules Under the HITECH Act and the Genetic Information Nondiscrimination Act; Other Modifications to the HIPAA Rules, 2013. Final Rule, 78 FR 5566.

[42] U.S. Department of Health and Human Services. Hipaa privacy rule to support reproductive health care privacy. `https://www.federalregister.gov/documents/2024/04/26/2024-08503/hipaa-privacy-rule-to-support-reproductive-health-care-privacy`, 2024. Final rule, 89 Fed. Reg. 32976, Apr. 26, 2024.

[43] U.S. Department of Health and Human Services, Office for Civil Rights. Summary of the hipaa privacy rule. `https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html`, 2024. Accessed 2026-05-04.

[44] U.S. Department of Health and Human Services, Office for Civil Rights. Individuals' right under hipaa to access their health information. `https://www.hhs.gov/hipaa/for-professionals/privacy/guidance/access/index.html`, 2024. Accessed 2026-05-04.

[45] U.S. Department of Health and Human Services, Office for Civil Rights. Summary of the hipaa security rule. `https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html`, 2024. Accessed 2026-05-04.

[46] U.S. Department of Health and Human Services, Office for Civil Rights. Hipaa regulatory initiatives. `https://www.hhs.gov/hipaa/for-professionals/regulatory-initiatives/index.html`, 2025. Accessed 2026-05-04.

[47] Ashish Vaswani, Noam Shazeer, et al. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[48] Jason Wei et al. Jailbroken: How does LLM safety training fail? *arXiv preprint arXiv:2307.02483*, 2023.

[49] Yijia Xiao, Yiqiao Jin, Yushi Bai, Yue Wu, Xianjun Yang, Xiao Luo, Wenchao Yu, Xujiang Zhao, Yanchi Liu, Quanquan Gu, et al. PrivacyMind: Large language models can be contextual privacy protection learners. *arXiv preprint arXiv:2310.02469*, 2023.

[50] Zhangchen Xu, Yuetai Li, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. Tinyv: Reducing false negatives in verification improves rl for llm reasoning. *arXiv preprint arXiv:2505.14625*, 2025.

A Layer 1: Rule Graph Construction Details

Layer 1 constructs a versioned Rule Graph from HIPAA-related regulatory text. The purpose of this appendix is to specify the data sources, candidate rule extraction procedure, anchor construction, and BRB admission rule used to reproduce the Rule Graph experiments in the main paper.

A.1 Data Sources

We use publicly available HIPAA-related regulatory documents from the U.S. Department of Health and Human Services (HHS), including the Privacy Rule summary, full Privacy Rule text, Security Rule guidance, Right of Access documentation, and the 2024 Reproductive Health Privacy update. These documents form the regulatory corpus used for candidate rule extraction and incremental Rule Graph construction. In addition, we use pre-computed HIPAA Annotator outputs from the P2T framework as structured candidate rule representations extracted from statutory text, and we construct anchor evidence from the GoldCoin HIPAA compliance dataset, which provides labeled regulatory applicability examples mapped to CFR sections.

A.2 Candidate Rule Extraction

Candidate rules are obtained from structured regulatory parsing outputs. We adopt P2T HIPAA extraction outputs where available, which provide YAML-formatted rule candidates containing source citations, conditions, requirements, and semantic attributes such as hazard and obligation type. To ensure compatibility with our inference and BRB pipeline, we also implement a constrained extraction module that maps regulatory text spans into a fixed JSON schema. This schema standardizes rule fields across sources and allows candidate rules to be passed directly into anchor alignment and BRB scoring.

A.3 Anchor Set Construction

The anchor set provides evidence for estimating whether a candidate rule corresponds to a prohibitive PHI-related constraint. We construct anchors from the GoldCoin HIPAA dataset by mapping each annotated example to a CFR rule identifier and converting its regulatory outcome into a binary signal: forbidden disclosures are treated as PHI-positive anchors, while permitted disclosures are treated as non-prohibitive anchors. For each rule r, let A_r denote the subset of anchor samples mapped to the same CFR identifier. We compute

$$k_r = \sum_{a \in A_r} \mathbb{1}[a \text{ supports a prohibitive PHI constraint}], \qquad n_r = |A_r|, \tag{9}$$

where k_r is the number of supporting anchors and n_r is the total number of aligned anchors for rule r. These GoldCoin-derived anchors are used for BRB scoring. Separately, the 50-item three-model annotation protocol reported in Experiment 1 is used to validate annotation reliability for disputed BRB candidates, rather than to replace the full anchor set used for Rule Graph construction.

A.4 Bernoulli Regulatory Bottleneck

The Bernoulli Regulatory Bottleneck (BRB) models rule admission as an evidence-weighted stochastic decision. For each candidate rule r, we estimate an admission probability using the Beta-Binomial posterior mean:

$$p_r = \frac{\alpha_0 + k_r}{\alpha_0 + \beta_0 + n_r}. \tag{10}$$

Consistent with the main text, we use $\mathrm{Beta}(\alpha_0, \beta_0) = \mathrm{Beta}(24, 1)$, corresponding to a prior mean of approximately 0.96. This prior is motivated by prior work reporting high precision in extracting normative statements from HIPAA regulatory text [22]. The prior is used only to initialize BRB scoring; it does not imply that every extracted normative statement is a valid prohibitive PHI constraint.

We use a floor threshold $\tau_{\text{floor}} = 0.50$ and a hard-admission ceiling $\tau_{\text{ceil}} = 0.85$. Rules with $p_r < \tau_{\text{floor}}$ are rejected. Rules with $p_r \geq \tau_{\text{ceil}}$ are admitted deterministically. For rules in the intermediate region $\tau_{\text{floor}} \leq p_r < \tau_{\text{ceil}}$, BRB samples

$$Z_r \sim \mathrm{Bernoulli}(p_r),$$

and admits the rule only when $Z_r = 1$. This design prevents candidate extraction from being treated as automatic rule acceptance while preserving uncertainty for weakly supported or legally ambiguous rules.

A.5 Rule Graph Construction Pipeline

The Layer 1 pipeline proceeds as follows. Regulatory documents are first parsed into text spans, from which candidate clauses are extracted and mapped into structured rule representations. Each rule is then aligned to CFR identifiers using citation and regex-based matching. Anchor samples are aggregated by CFR identifier to compute (k_r, n_r), after which BRB posterior scoring and Bernoulli admission determine the active rule set. Admitted rules are assembled into a versioned Rule Graph whose nodes correspond to retained regulatory constraints and whose metadata records source citations, PHI categories, anchor counts, posterior scores, and document version. Edges are added between admitted rules when they share a CFR citation, share a PHI category, or have Jaccard similarity above $\delta = 0.30$ over trigger-condition token sets. In the Jaccard convergence experiment, stability is computed over admitted rule sets rather than graph edges, so the reported Jaccard values do not depend on edge-weight choices.

A.6 Reproducibility Notes

Reproducing Layer 1 requires the HIPAA regulatory documents from HHS, GoldCoin HIPAA examples for anchor construction, and P2T HIPAA Annotator outputs for candidate rules. The released codebase includes scripts for regulatory text parsing, candidate rule normalization, anchor alignment, BRB estimation, and Rule Graph generation. These scripts reproduce the Layer 1 statistics reported in the main paper, including the 186 candidate rules, 872 anchor samples, BRB admission counts, and Jaccard convergence across regulatory versions.

B Layer 2: Multi-Stage PHI Detection Details

Layer 2 implements the PHI detection component of DePHI. Its purpose is to map a candidate output y into a PHI-risk signal that can be consumed by the Layer 3 fail-closed gate and the Layer 4 threshold adaptation mechanism. The detector combines deterministic symbolic matching with neural semantic detection, so that explicit PHI patterns can be captured cheaply while ambiguous or weakly lexicalized PHI mentions can still be handled by a learned model.

B.1 Detection Interface

Given an output text y, Layer 2 computes three signals: a keyword signal $s_A(y) \in \{0, 1\}$, a structured pattern signal $s_B(y) \in \{0, 1\}$, and a neural PHI score $p_C(y) \in [0, 1]$. These signals are combined using the same conservative aggregation rule as in the main text:

$$p(y) = \max(s_A(y), s_B(y), p_C(y)), \qquad Y_\tau(y) = \mathbb{1}[p(y) > \tau]. \tag{11}$$

Here, $Y_\tau(y) = 1$ denotes that the output is flagged as PHI-positive under threshold τ. The max aggregation ensures that strong evidence from any detector cannot be suppressed by weaker signals from another detector. In practice, this creates two operating regimes: a hard-rule dominant regime where Stage A or Stage B triggers deterministically, and a probabilistic regime where the neural score $p_C(y)$ and threshold τ govern the decision.

B.2 Stage A: Keyword Matching

Stage A performs deterministic keyword matching using 78 PHI-related keyword entries organized by the PHI categories represented in the Layer 1 Rule Graph. The matcher does not traverse graph edges at inference time; the Rule Graph provides the regulatory category structure, while Stage A uses a lightweight keyword trie for runtime detection. Each PHI category, such as `NAME`, `DATE`, `ID`, `CONTACT`, and `LOCATION`, is associated with lexical triggers extracted from rule conditions, requirements, and PHI-category metadata. The matcher is implemented as a trie-based lookup so that exact lexical triggers can be detected efficiently at inference time. Stage A returns $s_A(y) = 1$ if any keyword trigger is found and $s_A(y) = 0$ otherwise. This stage is designed for high precision on explicit PHI mentions and low runtime overhead, but it is not expected to detect paraphrased, obfuscated, or context-dependent PHI.

B.3 Stage B: Pattern-Based Detection

Stage B applies regular-expression templates for structured PHI categories. The pattern library is derived from established clinical de-identification practice, including Philter-style rules [28], and extended with templates for identifiers, dates, contact information, and structured location expressions. Stage B returns $s_B(y) = 1$ if any structured pattern is matched and $s_B(y) = 0$ otherwise. This stage complements keyword matching by covering PHI that follows recognizable formats, such as SSNs, medical record numbers, account identifiers, phone numbers, emails, dates, and address-like expressions.

Prior work on Philter reports high empirical recall on clinical de-identification benchmarks, including UCSF clinical notes and i2b2 data [28]. We use these results as motivation for including a symbolic pattern layer, not as a formal recall guarantee for DePHI under all distributions. Stage B remains limited to syntactic regularities and may fail when PHI is expressed through paraphrase, indirect reference, or semantic implication.

B.4 Stage C: Neural PHI Detection

Stage C uses a Bio_ClinicalBERT-based detector [3] to identify PHI-like content that may not be captured by keyword or pattern rules. Given a tokenized output $y = (y_1, \ldots, y_n)$, the model produces token-level PHI probabilities $p_C(y_i) = \Pr(\text{PHI} \mid y_i)$, which are aggregated into a sequence-level score

$$p_C(y) = \max_i p_C(y_i). \tag{12}$$

The maximum-token aggregation is conservative: a single high-risk token is sufficient to raise the output-level neural PHI score.

The neural detector is initialized from the Bio_ClinicalBERT checkpoint of ClinicalBERT and fine-tuned with a token-level PHI/non-PHI objective. Training uses MIMIC-IV clinical text augmented with synthetic PHI injection data. Synthetic examples are constructed by inserting PHI entities, such as names, dates, identifiers, contact information, and locations, into clean clinical contexts. This allows us to control ground-truth spans and expose the model to variants such as formatting changes, partial mentions, and obfuscated identifiers. Labels are represented at the token level, with synthetic labels generated directly from the injection process and clinical labels aligned to PHI categories when available.

The fine-tuning objective is token-level cross-entropy:

$$\mathcal{L} = -\sum_i \sum_{c \in \mathcal{C}} y_{i,c} \log p_{i,c}, \tag{13}$$

where i indexes tokens, $\mathcal{C}$ denotes the PHI/non-PHI or BIO-style label set, $y_{i,c}$ is the target label, and $p_{i,c}$ is the predicted probability. In our main experiments, fine-tuning uses AdamW with learning rate 2×10^{-5}, batch size 16, and 5 epochs, with validation F1 used for model selection. We use a held-out validation split to monitor overfitting and include class balancing and PHI-span perturbations to reduce sensitivity to rare or overly regular synthetic spans.

Stage C is intended to complement rather than replace Stages A and B. It is most useful for cases where PHI is weakly lexicalized, paraphrased, or context-dependent, such as indirect identifiers or nonstandard references. Conversely, when explicit structured PHI is present, Stages A and B often dominate the decision before the neural threshold becomes decisive.

B.5 Cascade Behavior and Dominance Regimes

The final Layer 2 decision can be written equivalently as

$$Y_\tau(y) = \mathbb{1}\left[s_A(y) = 1 \ \vee \ s_B(y) = 1 \ \vee \ p_C(y) > \tau\right]. \tag{14}$$

This form clarifies the role of symbolic dominance. If Stage A or Stage B fires, the output is flagged independently of the neural score. In this hard-rule dominant regime, the threshold τ has limited influence, and the system tends toward conservative rejection. This explains the high block rate observed in the utility–safety experiment. In the probabilistic regime, where Stages A and B do not fire, the neural score $p_C(y)$ determines whether the output crosses the threshold; here, Layer 4 threshold adaptation has direct control over the precision–recall tradeoff.

This separation is important for interpreting DePHI. A high rejection rate in explicit-PHI settings indicates that deterministic rules are dominating, whereas threshold adaptation is most meaningful when detection depends primarily on the neural component.

B.6 Failure Modes and Coverage Gaps

Layer 2 is designed to cover complementary PHI realizations, but it does not eliminate all failure modes. Keyword matching may miss abbreviations, partial names, or lexical substitutions. Pattern matching may miss spoken forms, paraphrases, or identifiers expressed outside expected formats. Neural detection may improve coverage for weakly lexicalized PHI, but it can still fail on cases requiring deeper contextual inference. In particular, inferential disclosure, where identity or sensitive

status can be inferred from multiple non-identifying clues, remains outside the current scope of the detector and is treated as future work.

These failure modes motivate the multi-stage design rather than contradict it: Stages A and B provide low-cost coverage for explicit and structured PHI, while Stage C addresses residual semantic cases. The audit and threshold-adaptation layers then measure and respond to remaining false negatives under the evaluation protocol.

B.7 Reproducibility Notes

Reproducing Layer 2 requires the Layer 1 Rule Graph output, the keyword dictionary derived from admitted rules, the regex pattern library, and the Bio_ClinicalBERT checkpoint used for Stage C initialization. The released codebase includes modules for trie construction, regex matching, transformer-based token classification, score aggregation, and the unified detection interface used by the runtime gate. The same interface produces the output-level signal $Y_\tau(y)$ used in the FNR, robustness, utility–safety, and ablation experiments.

C Layer 3: Runtime Gate

Layer 3 converts the PHI-risk signal from Layer 2 into an operational decision. While Layers 1–2 determine which regulatory constraints and PHI signals are active, Layer 3 defines how the system handles a candidate LLM output at inference time. Given an output y and the Layer 2 decision $Y_\tau(y)$, the runtime gate applies the fail-closed rule

$$\text{Decision}(y) = \begin{cases} \text{REJECT}, & Y_\tau(y) = 1, \\ \text{ALLOW}, & Y_\tau(y) = 0. \end{cases} \tag{15}$$

If PHI is detected, or if the detector returns an invalid or unreliable state, the output is rejected. A rejected output is replaced with a compliance refusal message, and the original generated text is not returned to the user. This conservative policy is intended to reduce the risk of undetected PHI disclosure at the cost of possible over-rejection.

The fail-closed design reflects the asymmetric risk structure of compliance systems. In healthcare deployment, a false negative corresponds to an unsafe release of PHI, whereas a false positive corresponds to suppressing or regenerating an otherwise safe output. Because these errors have different operational consequences, DePHI prioritizes reducing missed PHI disclosures over maximizing permissiveness. This is consistent with conservative enforcement practices in safety-critical systems and red-team-informed LLM safety evaluation [19, 17].

Layer 3 also provides the interface between detection and adaptation. In hard-rule dominant regimes, Stages A and B trigger rejection deterministically through keyword or structured-pattern matches. In probabilistic regimes, where symbolic detectors do not fire, the neural score from Stage C is compared with the threshold τ, and Layer 4 can adjust τ using audit feedback. The resulting decisions are logged by Layer 5 and used to compute the empirical FNR estimate for subsequent threshold updates.

The runtime gate is implemented as a lightweight post-generation decision module. It does not require additional model inference beyond the Layer 2 detectors and can be integrated either as a post-processing filter on generated outputs or as an intermediate control layer in a generation pipeline. In our implementation, the gate itself adds negligible overhead relative to LLM generation and neural PHI detection.

D Layer 4: Constrained Threshold Adaptation

Layer 4 implements the feedback controller that adjusts DePHI's operating point in response to measured PHI-detection failures. Its purpose is not to train the underlying LLM or optimize task accuracy directly, but to adapt the detection threshold used by the compliance pipeline so that the empirical FNR remains near a target risk tolerance.

D.1 Control Objective

Let y denote a generated output and $Y_\tau(y)$ denote the Layer 2 detection decision under threshold τ, where $Y_\tau(y) = 1$ means that y is flagged as PHI-positive. DePHI treats compliance control as an inference-time constrained decision problem:

$$\max_\tau \ \mathbb{E}_{x\sim\mathcal{D}}[U(y)] \quad \text{s.t.} \quad \text{FNR}(\tau) \le \epsilon, \tag{16}$$

where $\mathcal{D}$ is the evaluation or deployment distribution, $U(y)$ denotes downstream utility, and ϵ is the target upper bound on FNR. The underlying LLM policy remains fixed; only the compliance threshold τ is adapted.

At audit cycle t, DePHI empirically estimates FNR from audited or injected-PHI samples using the standard definition $\mathrm{FNR} = \mathrm{FN}/(\mathrm{FN} + \mathrm{TP})$:

$$\widehat{\mathrm{FNR}}_t = \frac{\sum_i \mathbb{1}[y_i \text{ contains PHI} \wedge Y_{\tau_t}(y_i) = 0]}{\sum_i \mathbb{1}[y_i \text{ contains PHI}]}. \tag{17}$$

Here, the denominator is the true PHI-containing set under the audit labels, not the set of predicted negatives. Thus, this quantity is a sample estimate of $\mathrm{FNR}(\tau_t)$ rather than a false non-discovery or false omission-style metric.

D.2 Dual-Ascent Threshold Update

DePHI maintains a nonnegative dual variable λ_t that accumulates evidence of constraint violation. When the empirical FNR estimate exceeds the target ϵ, λ_t increases and the threshold is tightened. The update is:

$$\lambda_{t+1} = \mathrm{clip}\Big(\lambda_t + \eta(\widehat{\mathrm{FNR}}_t - \epsilon), 0, \lambda_{\max}\Big), \qquad \tau_{t+1} = \mathrm{clip}\left(\frac{\tau_t}{1+\lambda_{t+1}}, \tau_{\min}, \tau_{\max}\right). \tag{18}$$

Here, η is the dual learning rate, $\lambda_{\max}$ prevents unbounded multiplier growth, and $(\tau_{\min}, \tau_{\max})$ define the allowable threshold range. We use

$$\tau_0 = 0.45, \qquad \epsilon = 0.05, \qquad \lambda_{\max} = 10, \qquad \tau_{\min} = 0.1, \qquad \tau_{\max} = 0.9.$$

When $\widehat{\mathrm{FNR}}_t > \epsilon$, the update increases λ_t and decreases τ_t, making the detector more conservative. When the empirical FNR estimate remains below the target, the multiplier stops increasing and the threshold stabilizes. This update is inspired by constrained optimization and CPO-style feedback control [2], but it is not a full reinforcement-learning procedure: DePHI does not update LLM weights and does not require environment interaction beyond the audit batches.

We set the dual learning rate to $\eta = 0.01$, with $\lambda_{\max} = 10$, $\tau_{\min} = 0.1$, and $\tau_{\max} = 0.9$, matching the main-text configuration.

D.3 Why Adapt the Threshold

Layer 4 adapts τ rather than editing rules or retraining the detector because τ is the lowest-latency control variable in the system. Updating it does not require rerunning Layer 1 rule construction or fine-tuning the Stage C detector. It also directly controls the precision–recall tradeoff when the neural detector is decisive. This preserves modularity: Layers 1–3 define the rule library, detector, and runtime gate, while Layer 4 adjusts the operating point using audit feedback.

The effect of threshold adaptation depends on which detection stage dominates. In a hard-rule dominant regime, Stage A keyword matching or Stage B pattern matching triggers rejection deterministically, so changing τ has limited influence. In a probabilistic regime, where Stages A and B do not fire and Stage C provides the decisive score, τ becomes the main control variable. This distinction explains why threshold adaptation has modest effect when explicit PHI patterns dominate, but becomes more important when PHI is weakly lexicalized or semantically expressed.

D.4 Stability Intuition

Layer 4 should be interpreted as a stabilizing feedback mechanism rather than as a guarantee of monotonic improvement. The update creates a negative-feedback loop:

$$\widehat{\mathrm{FNR}}_t \uparrow \Rightarrow \ \lambda_{t+1} \uparrow \Rightarrow \ \tau_{t+1} \downarrow \Rightarrow \ \text{stricter detection} \ \Rightarrow \ \widehat{\mathrm{FNR}}_{t+1} \downarrow .$$

In adversarial or small-batch settings, stochastic variation in the audit batch may temporarily dominate the correction signal, so FNR need not decrease at every cycle. The intended behavior is controlled recovery around the target region, not pointwise monotonic reduction. This is consistent with the Experiment 2 dynamics, where FNR is non-monotonic but remains in a controlled range under the target-driven feedback rule, although individual audit cycles may exceed ϵ due to stochastic adversarial variation.

D.5 Audit Cycle Protocol

Layer 4 is updated once per audit cycle. Each cycle consists of collecting an audit batch, computing $\widehat{\mathrm{FNR}}_t$ under the current threshold, updating (λ_t, τ_t), and logging the new operating point for Layer 5. In the adversarial experiments, one audit cycle corresponds to one batch of generated PHI-positive samples. This makes the controller interpretable at the batch level and allows the reported FNR dynamics to be reproduced from the same sequence of audit batches.

D.6 Failure Modes and Practical Limits

Layer 4 has three practical limitations. First, if deterministic stages dominate the decision, τ has limited control authority because the runtime gate is triggered before the neural score becomes decisive. Second, if the audit batch is small or highly variable, $\widehat{\mathrm{FNR}}_t$ can be noisy, causing temporary over-correction or under-correction. Third, the controller only acts on the measured FNR under the audit definition; risks outside the measured label space, such as inferential disclosure from multiple weak signals, may remain unaddressed. These limitations define the scope of the controller rather than contradictions of the method.

D.7 Reproducibility Notes

To reproduce Layer 4 behavior, one must specify the audit batch ordering, batch size, empirical FNR estimator, initial threshold τ_0, initial dual variable λ_0, target ϵ, learning rate η, and clipping bounds $(\tau_{\min}, \tau_{\max}, \lambda_{\max})$. Our implementation uses a lightweight scalar update and records $(\widehat{\mathrm{FNR}}_t, \tau_t, \lambda_t)$ at each audit cycle. The controller adds negligible computational overhead and changes only the compliance threshold, not the underlying LLM or detector weights.

Algorithm 1 Layer 4 threshold adaptation

Require: threshold τ_t, dual variable λ_t, target ϵ, learning rate η, empirical FNR estimate $\widehat{\mathrm{FNR}}_t$

1: $\lambda_{t+1} \leftarrow \mathrm{clip}(\lambda_t + \eta(\widehat{\mathrm{FNR}}_t - \epsilon), 0, \lambda_{\max})$
2: $\tau_{t+1} \leftarrow \mathrm{clip}(\tau_t/(1 + \lambda_{t+1}), \tau_{\min}, \tau_{\max})$
3: **return** $(\tau_{t+1}, \lambda_{t+1})$

E Layer 5: Audit Records and Uncertainty Quantification

Layer 5 records DePHI's behavior across audit cycles and produces versioned audit artifacts. Its role is not to prove universal compliance, but to make measured system behavior traceable, reproducible, and statistically interpretable under the evaluation distribution. While Layer 4 uses the empirical FNR estimate to adapt the threshold, Layer 5 stores the measurement context, uncertainty summaries, and system configuration used at each audit cycle.

E.1 Empirical FNR Estimation

At audit cycle t, let $\mathcal{D}_t = \{y_i\}_{i=1}^{n_t}$ denote the audited outputs, and let $G_i \in \{0, 1\}$ indicate whether output y_i contains PHI according to injected ground truth or audit annotation. Given the Layer 2 decision $Y_{\tau_t}(y_i)$, where $Y_{\tau_t}(y_i) = 1$ means that the output is flagged as PHI-positive, DePHI empirically estimates the FNR as

$$\widehat{\mathrm{FNR}}_t = \frac{\sum_{i=1}^{n_t} \mathbb{1}[G_i = 1 \wedge Y_{\tau_t}(y_i) = 0]}{\sum_{i=1}^{n_t} \mathbb{1}[G_i = 1]}. \tag{19}$$

This is the sample estimate of the standard FNR, $\mathrm{FNR} = \mathrm{FN}/(\mathrm{FN} + \mathrm{TP})$, and the denominator is the true PHI-containing set under the audit labels. This estimate is passed to Layer 4 for threshold adaptation and stored by Layer 5 for reporting.

E.2 Uncertainty Summaries

Layer 5 reports uncertainty summaries for the measured FNR. We use these summaries as complementary diagnostics rather than as formal guarantees of compliance outside the audited distribution.

First, DePHI computes bootstrap intervals to characterize finite-sample variability. Given the audit set $\mathcal{D}_t$, we draw B bootstrap resamples with replacement and recompute $\widehat{\mathrm{FNR}}_t^{(b)}$ on each resample. For confidence level $1 - \alpha$, the percentile bootstrap interval is

$$\mathrm{CI}_{\mathrm{boot}} = \left[Q_{\alpha/2}\left(\{\widehat{\mathrm{FNR}}_t^{(b)}\}_{b=1}^{B} \right), Q_{1-\alpha/2}\left(\{\widehat{\mathrm{FNR}}_t^{(b)}\}_{b=1}^{B} \right) \right], \tag{20}$$

where $Q_q(\cdot)$ denotes the empirical q-quantile. In practice, we use $B = 2000$ and $\alpha = 0.05$ as auxiliary diagnostic settings; the main text reports the Clopper–Pearson interval as the default regulatory-style uncertainty summary.

Second, because false negatives may be rare and audit batches may be small, DePHI also reports Clopper–Pearson exact binomial intervals [11, 7]. Let $m_t = \sum_i \mathbb{1}[G_i = 1]$ be the number of PHI-positive audit samples and let $e_t = \sum_i \mathbb{1}[G_i = 1 \wedge Y_{\tau_t}(y_i) = 0]$ be the number of false negatives. The Clopper–Pearson interval for the underlying binomial error probability is

$$\mathrm{CI}_{\mathrm{CP}} = \left[\mathrm{Beta}^{-1}\Big(\frac{\alpha}{2}; e_t, m_t - e_t + 1\Big), \mathrm{Beta}^{-1}\Big(1 - \frac{\alpha}{2}; e_t + 1, m_t - e_t\Big)\right], \tag{21}$$

with the usual boundary conventions when $e_t = 0$ or $e_t = m_t$. We report this interval because it is conservative and interpretable for small samples.

Finally, DePHI can compute conformal recalibration diagnostics when a calibration split is available [35]. For PHI-positive calibration examples, we define nonconformity scores using the detector confidence, for example

$$s_i = 1 - p(y_i), \tag{22}$$

where $p(y_i)$ is the aggregated PHI risk score from Layer 2. A conformal quantile of these scores can be used to adjust the operating threshold under exchangeability assumptions. We treat this as a distribution-aware recalibration diagnostic, not as a claim of unconditional robustness under arbitrary adversarial shift.

E.3 Audit Certificate

Each audit cycle produces a versioned certificate that records the system state used to make compliance decisions:

$$C_t = \Big(\widehat{\mathrm{FNR}}_t, CI_t, \tau_t, \lambda_t, \mathrm{RuleGraphVersion}_t\Big).$$

Here, CI_t is the Clopper–Pearson interval used for regulatory-style reporting. Detector version, audit dataset identifier, bootstrap summaries, and conformal recalibration diagnostics are stored as auxiliary metadata, but are not part of the minimal certificate tuple reported in the main text.

E.4 Interaction with Layer 4

Layer 5 closes the feedback loop by providing the measurement signal used by Layer 4. At each audit cycle, Layer 5 computes $\widehat{\mathrm{FNR}}_t$ from audited outputs, records uncertainty summaries, and passes the empirical FNR estimate to the dual-ascent threshold update. Layer 4 then updates (λ_t, τ_t), and the resulting operating point is stored in the next certificate. This separation keeps measurement and control modular: Layer 5 estimates and records risk, while Layer 4 adapts the threshold.

E.5 Limitations and Reproducibility

Layer 5 inherits the limitations of statistical auditing. If the audit set is not representative of deployment inputs, the FNR estimate may be biased. If PHI-positive samples are rare, intervals may be wide. If the distribution shifts strongly or adversarially, conformal recalibration assumptions may not hold. These limitations are reported as part of the audit artifact rather than hidden from the system record.

To reproduce Layer 5 results, one must specify the audit dataset, PHI ground-truth construction method, threshold τ_t, bootstrap seed and number of resamples, confidence level α, binomial interval method, and calibration split if conformal recalibration is used. Given these inputs and the recorded detector and Rule Graph versions, the FNR estimate, uncertainty intervals, and certificate fields are deterministic up to bootstrap resampling randomness.

F Runtime Profiling and Annotation Cost

DePHI is implemented as an inference-time compliance layer and does not update the underlying LLM weights. Its runtime overhead is therefore post-generation and component-dependent. Stage A and B perform dictionary and regular-expression matching over 98 rules/patterns, consisting of 78 keyword entries and 20 regular-expression templates. Together with the fail-closed runtime gate, these symbolic stages add 2.2 ms of latency on CPU, measured as the median over 50 trials.

Stage C invokes a fine-tuned Bio_ClinicalBERT detector with 107.7M parameters only when neural scoring is required. On an NVIDIA H200 GPU, Stage C median latency is 3.0 ms for short outputs

($\sim$42 characters), 3.3 ms for medium-length outputs ($\sim$154 characters), and 7.7 ms for long outputs ($\sim$704 characters), measured with batch size 1. In our profiling, Stage A and B intercept 40% of outputs before Stage C is invoked. Under the medium-length setting, this yields an effective Stage C overhead of approximately 2.0 ms per output. Combined with Stage A and B and gate overhead, the total DePHI overhead is approximately 4.2 ms per output.

We do not report a single percentage overhead relative to the base LLM because that value depends on the underlying model, decoding configuration, hardware, and output length. Instead, we report component-level latency to make the overhead attributable and reproducible. The one-time three-model BRB annotation protocol costs 150 API calls, approximately 24,000 tokens, and about $0.06 for 50 candidate items. This cost is amortized across Rule Graph versions and is not incurred during ordinary inference.

G Experiments

G.1 Experiment 1: Rule Graph Jaccard Convergence and Annotation Stability

A central concern in iterative rule construction is whether the admitted rule set remains stable as additional regulatory evidence is incorporated, or whether it drifts due to noisy extraction or self-reinforcement. This concern is especially important for legal and regulatory text, where extracted clauses may express obligations, permissions, exceptions, or prohibitions depending on context. Prior work in regulatory rule extraction, legal modality classification, and knowledge-base construction has shown that rule extraction from heterogeneous sources can be unstable and that deontic categories may remain ambiguous even under expert annotation [22, 30, 25]. We therefore evaluate both Rule Graph stability under incremental regulatory expansion and annotation reliability for BRB rule-admission labels.

We process six cumulative versions of HIPAA-related regulatory evidence, denoted $t_1, \ldots, t_6$, through the same Layer 1 pipeline: candidate rule extraction, anchor construction, BRB posterior scoring, Bernoulli admission, and Rule Graph construction. The six versions are constructed as cumulative snapshots of HIPAA-related regulatory materials: they begin with the HIPAA statutory baseline and Privacy Rule materials, then incorporate HITECH and Omnibus updates, OCR Right of Access guidance, Security Rule guidance, additional HHS/OCR interpretive documents, and the 2024 HIPAA Privacy Rule amendment on reproductive health care privacy [40, 43, 39, 41, 44–46, 42]. Across the full corpus, the Layer 1 pipeline produces 186 candidate rules and 872 anchor samples. The initial BRB pass admits 52 rules, and iterative refinement expands the admitted rule library to 89 rules. Across all processed document versions, the final Rule Graph contains 89 admitted rules; the earlier 52-rule library corresponds to the initial regulatory subset before later document versions and anchors are incorporated. BRB scores are used internally for evidence-weighted rule admission, and the released artifacts report the admitted Rule Graph and versioned rule metadata used in the experiments.

G.2 BRB Annotation Protocol and Disputed Items

We evaluate the reliability of BRB ground-truth construction using a TinyV-style multi-verifier protocol [50]. Each of the 50 BRB candidate items is independently labeled by Claude Sonnet 4.6 Extended, Gemini 3.1 Pro, and GPT-5.5 under the same deontic logic definitions used in the main paper. A candidate is included in the deterministic annotation set only when all three annotators agree on whether it is a prohibitive PHI-related constraint. Items without full agreement are not forced into binary ground truth; instead, they are retained as probabilistic BRB cases.

Because frontier LLM APIs are versioned external systems, we report the exact model names, prompts, consensus rule, and disputed items; the protocol is intended to be reproducible at the decision-procedure level even if future API snapshots differ.

The three-model protocol reaches full consensus on 47/50 items. Among consensus items, 24 are labeled prohibitive and 23 are labeled non-prohibitive. The remaining three items exhibit 2–1 disagreement. Fleiss' κ is 0.92, indicating almost perfect agreement, and pairwise Jaccard similarity over the YES-label sets ranges from 0.889 to 0.960. These results indicate that the revised prompt and multi-verifier protocol produce stable annotations while preserving a small number of genuinely ambiguous deontic cases.

The disputed items are informative because they involve clauses that can be read either as operational documentation requirements or as gating conditions on permissible disclosure. For this reason, DePHI

Table 4: Three disputed BRB annotation items under the three-model protocol. These items are excluded from deterministic ground truth and retained as probabilistic BRB labels.

Item	CFR	Claude	Gemini	GPT-5.5 / Condition Summary
ITEM_007	R-164.504	NO	NO	YES; plan documents restrict uses and disclosures of information by the plan sponsor.
ITEM_046	R-164.504	NO	NO	YES; plan documents restrict uses and disclosures of health information by the plan sponsor.
ITEM_049	R-164.508	YES	NO	NO; organization must obtain and record a description of the personal representative's authority to act for the individual.

does not force them into deterministic labels. Instead, they remain BRB-calibrated probabilistic cases, which is consistent with the purpose of the Bernoulli Regulatory Bottleneck: rule admission should represent uncertainty when deontic interpretation is not fully determinate.

Because BRB depends on whether candidate rules correspond to prohibitive PHI-related constraints, we additionally evaluate annotation reliability. We re-annotate 50 BRB candidate items using a TinyV-style verifier protocol [50], extended to three independent LLM annotators: Claude Sonnet 4.6 Extended, Gemini 3.1 Pro, and GPT-5.5. Each annotator classifies a candidate rule as prohibitive or non-prohibitive under explicit deontic logic definitions. Consensus labels are used as deterministic ground truth only when annotators agree; disputed cases are retained as probabilistic BRB labels rather than forced into binary labels.

The three-model protocol reaches full agreement on 47/50 items, with 24 unanimous prohibitive labels and 23 unanimous non-prohibitive labels. Three items remain disputed: ITEM_007, ITEM_046, and ITEM_049. Pairwise Jaccard similarity over the YES/prohibitive sets ranges from 0.889 to 0.960: Claude vs. Gemini = 0.960, Claude vs. GPT-5.5 = 0.889, and Gemini vs. GPT-5.5 = 0.923. Overall three-way agreement yields Fleiss' $\kappa = 0.92$, corresponding to almost perfect agreement. This is substantially above the agreement difficulty reported in prior deontic modality classification studies, where legal modality categories remain challenging even for expert or legal-domain annotators [25]. The remaining disputed items therefore support the BRB design: residual legal ambiguity is represented probabilistically rather than collapsed into a brittle deterministic label.

Let $\mathcal{R}_t$ denote the set of rules admitted into the Rule Graph after processing document version t. We quantify stability using the Jaccard similarity coefficient:

$$J(\mathcal{R}_i, \mathcal{R}_j) = \frac{|\mathcal{R}_i \cap \mathcal{R}_j|}{|\mathcal{R}_i \cup \mathcal{R}_j|}. \tag{23}$$

Jaccard similarity measures set overlap and is commonly used to assess stability in information retrieval and knowledge-base construction settings [30]. In our setting, increasing Jaccard similarity indicates that newly incorporated regulatory evidence refines the admitted rule set rather than causing large oscillations.

We observe a monotonic increase in Jaccard similarity across regulatory document expansions:

$$0.095 \rightarrow 0.210 \rightarrow 0.440 \rightarrow 0.670 \rightarrow 0.880 \rightarrow 1.000.$$

The Rule Graph also stabilizes at a consistent set of 21 regulatory sections in later stages, suggesting that newly added documents do not substantially alter the core admitted rule structure after sufficient regulatory evidence is incorporated. Together with the three-model annotation study, this provides two complementary forms of evidence: BRB-based rule admission becomes stable under incremental regulatory expansion, and the labels used for BRB analysis are highly consistent while still preserving genuinely ambiguous cases.

These results should be interpreted as empirical stability evidence rather than a formal convergence theorem. Jaccard convergence does not by itself prove semantic correctness, and three-model LLM consensus is not a substitute for legal-expert annotation. Instead, the experiment shows that the Rule Graph is stable under the evaluated regulatory expansions and that disputed rule candidates can be handled explicitly through probabilistic BRB labels. Future work should incorporate expert legal review for disputed items and evaluate Rule Graph stability under additional regulatory domains.

G.3 Experiment 2: FNR Dynamics Under Adversarial Augmentation

Layer 4 is designed to adapt the detection threshold under changing PHI risk. In deployment, PHI leakage risk may shift over time as users vary formats, split identifiers, paraphrase sensitive content, or combine multiple obfuscation strategies. We therefore evaluate DePHI under an adversarial PHI stream, focusing on whether the empirical FNR remains controlled around the target region $\epsilon = 0.05$, rather than whether FNR decreases monotonically at every cycle.

We construct an LLM-based adversarial generator that produces PHI-containing samples across five categories: `NAME`, `DATE`, `ID`, `CONTACT`, and `LOCATION`. Each audit cycle contains 50 samples, and the evaluation runs for 20 cycles, yielding 1,000 adversarial samples in total. Across cycles, the generator increases difficulty through abbreviation, partial masking, paraphrasing, format variation, and compositional mixing. At each cycle t, DePHI applies the full detection pipeline, computes $\widehat{\mathrm{FNR}}_t$ against injected PHI ground truth, and updates the threshold τ_t using the Layer 4 dual-ascent rule. The underlying LLM generator, the three-stage detection modules, and the Rule Graph remain fixed throughout the 20 audit cycles; only the runtime gate threshold is updated from the audited FNR estimate.

Table 5 summarizes the FNR dynamics. The empirical FNR estimate ranges from 0.04 to 0.14 across all cycles. The mean FNR increases from 0.071 in cycles 1–10 to 0.086 in cycles 11–20, reflecting the harder adversarial inputs introduced later in the schedule. Although FNR exceeds the target $\epsilon = 0.05$ in some cycles, it does not diverge: the peak FNR is 0.14, and the final-cycle FNR decreases to 0.06 after threshold updates. This behavior is consistent with dual-ascent correction under stochastic adversarial variation, although we do not claim a formal convergence guarantee at this sample scale.

Table 5: FNR dynamics under adversarial PHI augmentation across 20 audit cycles. The target upper bound is $\epsilon = 0.05$.

Metric	**Value**
Audit cycles	20
Samples per cycle	50
Total adversarial samples	1,000
PHI categories	5
Target ϵ	0.05
empirical FNR estimate range	0.04–0.14
Mean FNR, cycles 1–10	0.071
Mean FNR, cycles 11–20	0.086
Peak FNR	0.14
Final-cycle FNR	0.06
Primary hotspot	`NAME`
`NAME`-category FNR	0.20–0.40

The trajectory is non-monotonic, which is expected because each 50-sample audit batch introduces new adversarial variants and finite-sample variation. The relevant observation is therefore stabilization under feedback rather than monotonic reduction. When $\widehat{\mathrm{FNR}}_t$ exceeds ϵ, the dual variable increases and the threshold is tightened; subsequent cycles show recovery rather than uncontrolled growth. This supports the negative-feedback role of Layer 4 under the evaluated protocol, while remaining an empirical observation rather than a formal guarantee.

The main error hotspot is the `NAME` category, where FNR is approximately 0.20–0.40. Adversarial names can appear as initials, abbreviations, partial mentions, or context-dependent references, making them more difficult for Stage A keyword matching and Stage B structured pattern matching than more regular categories such as dates, identifiers, and contact information. This motivates Stage C neural detection as a semantic backstop for weakly lexicalized PHI mentions that are not reliably captured by hard rules.

Experiment 2 shows that DePHI does not eliminate all false negatives and does not guarantee pointwise monotonic improvement. Instead, it empirically estimates FNR at each audit cycle and adjusts the operating threshold to control false-negative behavior under progressively harder adversarial inputs. The remaining `NAME` errors identify a concrete direction for targeted augmentation and improved semantic modeling.

G.4 Experiment 3: Utility–Safety Tradeoff

A deployable compliance layer must expose the tradeoff between downstream task utility and PHI safety. In regulated clinical settings, reducing false negatives is often prioritized over unconstrained generation quality, but overly conservative filtering may suppress useful outputs. We therefore evaluate whether DePHI reduces PHI leakage in generated responses and how much utility is lost under strict fail-closed enforcement.

Setup. We evaluate DePHI on 100 EHRNoteQA samples [23]. Each item contains a clinical question and patient-record context requiring reasoning over clinical notes. For each item, we inject two PHI entities into the context and prompt the underlying LLM to generate a free-text response. We use Claude Sonnet 4.6 Extended as the LLM under evaluation and compare an ungated baseline against DePHI under two threshold settings, $\tau = 0.4$ and $\tau = 0.7$.

Utility is measured using multiple-choice accuracy on EHRNoteQA. Safety is measured using the FNR, computed by string matching generated outputs against the injected PHI ground truth. The block rate records the fraction of generated outputs rejected by the fail-closed gate. This protocol evaluates output-level PHI leakage after generation, rather than only static PHI detection on pre-existing clinical text.

Results. As shown in Table 6, the ungated baseline achieves 0.43 task accuracy but has high PHI leakage, with FNR = 0.60. DePHI reduces FNR to 0.02 at $\tau = 0.4$ and 0.00 at $\tau = 0.7$, but both settings produce a 97% block rate and substantially lower effective utility.

Table 6: Utility–safety tradeoff under controlled PHI injection. Both DePHI configurations yield the same block rate because the injected PHI entities are primarily explicit names, identifiers, and dates intercepted by Stage A and B deterministic matching before Stage C thresholding becomes decisive.

System	**Accuracy** $\uparrow$	**FNR** $\downarrow$	**Block Rate**
Ungated LLM	0.43	0.60	0%
DePHI ($\tau = 0.4$)	0.02	0.02	97%
DePHI ($\tau = 0.7$)	0.03	0.00	97%

The results show that DePHI can sharply reduce PHI leakage under controlled injection, but at a substantial utility cost in this setting. This behavior reflects a hard-rule dominant regime: most injected PHI consists of explicit names, dates, or identifiers, which are captured by Stage A keyword matching or Stage B structured pattern matching and rejected before the probabilistic threshold τ has much influence. Consequently, the utility–safety tradeoff is governed primarily by the detection cascade and fail-closed policy, not by threshold tuning alone.

Experiment 3 should not be interpreted as showing that DePHI always sacrifices utility to this degree. Rather, it identifies a conservative operating point appropriate for high-risk compliance scenarios where explicit PHI leakage must be blocked. In softer probabilistic regimes, where Stage C contributes more decisions and Stage A and B do not dominate, τ provides a more direct mechanism for navigating the utility–safety frontier.

G.5 Experiment 4: Adversarial Robustness

PHI leakage may occur through adversarially transformed or weakly signaled references rather than explicit identifiers. We therefore evaluate whether DePHI detects PHI under prompt injection, lexical obfuscation, format variation, Unicode substitution, and contextual embedding, while avoiding false positives on clean clinical samples. This experiment is designed as a taxonomy-based failure-mode audit: each attack class represents a qualitatively distinct PHI-obfuscation mechanism, and repeated samples within each class test whether the observed behavior is stable across surface variations.

We construct an expanded adversarial diagnostic set with 10 PHI-positive attack classes and 50 samples per class, yielding 500 adversarial PHI-positive samples. The attack classes are: `prompt_injection`, `split_obfuscation`, `written_date`, `spoken_ssn`, `abbreviated`, `unicode_sub`, `profession_name`, `context_embedded`, `indirect_ref`, and `spoken_phone`. We additionally include 500 PHI-negative clean clinical samples to evaluate false positives. The evaluation uses the full DePHI pipeline with Stage A keyword matching, Stage B structured pattern

matching, Stage C neural detection, and the fail-closed runtime gate enabled. A sample is counted as detected if any stage identifies PHI and the gate rejects the output.

Table 7 reports the per-class detection results. DePHI detects all samples in 8 of the 10 adversarial attack classes, corresponding to 400/500 detected PHI-positive samples overall. The two consistently missed classes are `indirect_ref` and `spoken_phone`, each with 0/50 detections. On clean clinical samples, DePHI produces 0/500 false positives.

Table 7: Expanded adversarial robustness evaluation. Each adversarial class contains 50 PHI-positive samples; the clean set contains 500 PHI-negative samples.

Class	**Detected / Total**	**Detection Rate**
`written_date`	50 / 50	100%
`spoken_ssn`	50 / 50	100%
`abbreviated`	50 / 50	100%
`context_embedded`	50 / 50	100%
`unicode_sub`	50 / 50	100%
`prompt_injection`	50 / 50	100%
`split_obfuscation`	50 / 50	100%
`profession_name`	50 / 50	100%
`indirect_ref`	0 / 50	0%
`spoken_phone`	0 / 50	0%
Overall adversarial detection	400 / 500	80%
Clean false positives	0 / 500	0%

The expanded evaluation shows that the failures are not random isolated misses but concentrated in two semantically difficult transformation classes. The `indirect_ref` cases refer to sensitive information indirectly, often without a complete surface identifier. The `spoken_phone` cases require normalizing spoken word sequences into numeric contact patterns. Both failure modes require semantic normalization beyond keyword matching and regular expressions, and remain challenging for the current Stage C detector under the tested threshold and training data. These cases are consistent with known limitations of PHI detection on weakly lexicalized and context-dependent mentions [34, 28].

The results support the role of the detection cascade while clarifying its boundary conditions. Stages A and B provide strong coverage for explicit, structured, and format-regular PHI, including written dates, spoken SSNs, Unicode substitutions, split-token obfuscation, and prompt-injection variants. Stage C does not introduce false positives in this evaluation, but the current model does not yet resolve the two semantic boundary classes. This suggests that future improvements should focus less on expanding lexical rules and more on semantic normalization, spoken-form parsing, and expert-labeled training data for indirect PHI references.

Experiment 4 provides expanded diagnostic evidence that DePHI is robust to most tested adversarial PHI transformations, detecting 8/10 attack classes and 400/500 adversarial samples while maintaining a 0/500 false positive rate on clean clinical text. The remaining failures are concentrated in indirect references and spoken-form phone numbers, identifying concrete targets for future semantic augmentation.

H Ablation Study

H.1 Motivation and Scope

We conduct the ablation study as a within-system component analysis of the DePHI runtime detection cascade. The goal is to isolate how executable detection stages contribute to PHI-safe output decisions while holding the interface, threshold setting, injected ground truth, and output-level evaluation protocol fixed. This differs from a system-level comparison against external tools. External de-identification systems such as Philter, i2b2-style pipelines, or standalone ClinicalBERT NER models use different interfaces, label taxonomies, and output formats, and many are designed for offline batch de-identification rather than real-time output gating. Substituting them into a DePHI stage would require redesigning the runtime gate and would conflate architectural differences with component-level effects.

Table 8: Ablation study of the DePHI detection cascade on an expanded balanced diagnostic set. The PHI-positive set contains 500 samples stratified into Stage-A, Stage-B, and Stage-C pools; the clean set contains 500 PHI-negative samples.

Variant	**TP**	**FN**	**FP**	**TN**	**FNR** $\downarrow$	**F1** $\uparrow$
Stage A only	226	274	0	500	0.548	0.623
Stages A+B	326	174	0	500	0.348	0.789
Full DePHI (A+B+C)	326	174	0	500	0.348	0.789

Table 9: Pool-level detection counts for the ablation study. The Stage-A pool contains 200 keyword-dominant PHI samples, the Stage-B pool contains 200 regex-dominant PHI samples, and the Stage-C pool contains 100 semantic-only PHI samples.

Variant	**Stage-A pool**	**Stage-B pool**	**Stage-C pool**
Stage A only	166/200	40/200	20/100
Stages A+B	166/200	140/200	20/100
Full DePHI (A+B+C)	166/200	140/200	20/100

We therefore report only executable DePHI configurations that were directly measured under the same logging pipeline. We do not report unsupported numerical variants such as *w/o BRB*. BRB is evaluated separately in Experiment 1 through Rule Graph convergence and annotation agreement, while this ablation focuses on the runtime PHI detection cascade.

H.2 Experimental Protocol

The ablation is evaluated on an expanded balanced diagnostic set containing 500 PHI-positive adversarial samples and 500 PHI-negative clean samples. The PHI-positive subset is stratified into three pools: 200 keyword-dominant samples designed to test Stage A, 200 regex-dominant samples designed to test Stage B, and 100 semantic-only samples designed to test Stage C. The PHI-negative subset contains clean clinical samples used to measure false positives. The PHI-positive ratio is therefore A:B:C = 2:2:1.

We evaluate three executable configurations. **Stage A only** uses keyword matching and tests lexical coverage. **Stages A+B** adds structured pattern matching for identifiers, dates, contact expressions, and related formats. **Full DePHI (A+B+C)** additionally includes the fine-tuned Bio_ClinicalBERT detector. For each variant, we report true positives, false negatives, false positives, true negatives, FNR, F1, false positive rate, and pool-level detection counts.

H.3 Results and Analysis

Tables 8 and 9 show a clear contribution from Stage B. Stage A alone detects 226/500 PHI-positive samples, with strongest performance on keyword-dominant cases. Adding Stage B increases the number of true positives from 226 to 326, reduces FNR from 0.548 to 0.348, and improves F1 from 0.623 to 0.789. The improvement comes primarily from the regex-dominant pool, where detection increases from 40/200 to 140/200. This confirms that structured pattern matching provides substantial coverage beyond keyword matching for dates, identifiers, contact expressions, and related formats.

Full DePHI produces the same aggregate result as Stages A and B on this diagnostic set. The Stage-C pool remains at 20/100 detected under both configurations, indicating that the current fine-tuned Stage C detector does not yet resolve the semantic-only cases under the tested threshold and available training data. Importantly, Stage C does not introduce false positives: all configurations produce FP=0 on 500 clean samples. Thus, the current Stage C component is non-degrading but not yet performance-improving on the semantic-only pool.

H.4 Interpretation

The ablation supports two conclusions. First, deterministic symbolic detection is important but not reducible to keyword matching: Stage B provides a measurable gain over Stage A by handling structured PHI that does not appear as simple keyword matches. Second, the remaining errors are concentrated in semantically difficult cases that are not resolved by the current Stage C training setup.

This does not contradict the cascade design; rather, it identifies where additional expert-labeled data, semantic normalization, or stronger neural training objectives are needed.

This result also clarifies the current role of Stage C. In the present implementation, Stage C acts as a semantic backstop that preserves performance and does not increase false positives, but it does not yet improve aggregate detection on the semantic-only diagnostic pool. Future versions of DePHI should target this pool directly through additional training data for indirect references, spoken-form contact information, and weakly lexicalized PHI mentions.

H.5 Limitations

This ablation is diagnostic rather than exhaustive. The expanded balanced set is larger and more controlled than the initial diagnostic set, but it is still constructed by failure-mode pools rather than sampled from a deployment distribution. The results therefore characterize component behavior under controlled attack-class coverage, not a population-level estimate of real-world PHI leakage. Future ablations should separately vary Stage C thresholds, neural training data, semantic PHI categories, and BRB-derived keyword sets to determine when neural detection improves over deterministic symbolic stages.

I Future Extensions and Deployment Considerations

DePHI is designed as an inference-time compliance control layer for HIPAA-oriented LLM deployment. The present evaluation focuses on reproducible offline experiments, including Rule Graph construction, PHI detection, fail-closed gating, threshold adaptation, and audit-record generation. We outline several extensions that are important for deployment-scale validation.

First, future deployments should incorporate legal-expert review for regulatory rule semantics. In this work, BRB rule labels are constructed using a high-agreement three-model LLM consensus protocol, which provides a scalable and reproducible annotation procedure. Expert review remains important for disputed, institution-specific, or semantically ambiguous clauses, especially when adapting the Rule Graph to local compliance policies.

Second, adversarial evaluation should be expanded beyond diagnostic stress testing. Our adversarial experiments are designed to identify failure modes across representative PHI-obfuscation classes. Deployment-level robustness evaluation should include larger attack suites, repeated samples per category, randomized red-teaming, and confidence intervals for attack-class detection rates, so that isolated failures can be distinguished from persistent vulnerability classes.

Third, semantic PHI detection remains an important extension. The current cascade is strongest on explicit and structurally recoverable PHI, while indirect references and spoken-form contact information require stronger semantic normalization. Future versions may incorporate expert-labeled weakly lexicalized PHI examples, spoken-to-structured normalization, retrieval-augmented context checking, and calibration under adversarial paraphrase distributions.

Fourth, long-horizon feedback dynamics should be evaluated under realistic institutional conditions. Public datasets such as MIMIC-IV and EHRNoteQA support reproducible offline evaluation, but they cannot fully simulate policy drift, changing documentation practices, institutional rule interpretations, or production-scale audit feedback. Controlled deployment simulations could evaluate how Rule Graph versions, thresholds, and audit records evolve over time.

Finally, future benchmarks should move beyond token-level PHI detection toward system-level compliance evaluation. Useful benchmarks should measure rule traceability, empirical risk control, audit reproducibility, utility–safety calibration, adversarial robustness, and compatibility with fixed or proprietary LLMs. These properties better reflect the operational requirements of compliance-aware LLM deployment in regulated settings.