Can Post-Training Turn LLMs into Good Medical Coders? An Empirical Study of Generative ICD Coding

Anonymous ACL submission

Abstract

Automated International Classification of Diseases (ICD) coding is a core medical-coding task for billing, epidemiology, and clinical decision support. Generative large language models (LLMs) are often reported as weak medical coders, but this finding mainly comes from inference-time settings such as prompting, retrieval, reranking, or tool use, leaving the role of task-specific post-training underexplored. We present a controlled empirical study of post-training for generative ICD coding, comparing discriminative baselines with LLM coders across prompting, supervised fine-tuning, and reinforcement learning under a common protocol and metric set. To our knowledge, this is the first study to evaluate RL-based post-training for generative LLM coders in ICD coding. We further introduce `PHI`, a diagnostic curriculum that extends GRPO to refine missed-code cases. Our results show that prompting-only evaluation substantially underestimates the potential of LLMs for ICD coding. SFT provides the main capability jump, GRPO further improves code-set prediction beyond SFT, and `PHI` provides targeted gains on macro-level performance. These findings suggest that the main bottleneck is not the generative formulation alone, but how the model is adapted and optimized for full-taxonomy recall. We release our code, data splits, and checkpoints at `https://anonymous.4open.science/r/LLM4ICD`.

1 Introduction

Automated International Classification of Diseases (ICD) coding, a central form of medical coding, aims to map each clinical note to a set of standardized diagnosis and procedure codes (Mullenbach et al., 2018; Teng et al., 2022; Ji et al., 2024). These codes are widely used for medical billing, epidemiology, and clinical decision support (Teng et al., 2022; Ji et al., 2024). The task is difficult because a single discharge summary can span thousands of tokens and must be mapped to a subset of an enormous taxonomy, with over 70,000 codes in ICD-10-CM (Mullenbach et al., 2018; Vu et al., 2020). As a result, ICD coding is an extreme multi-label problem with a highly imbalanced label distribution, making accurate code prediction a major challenge (Edin et al., 2023).

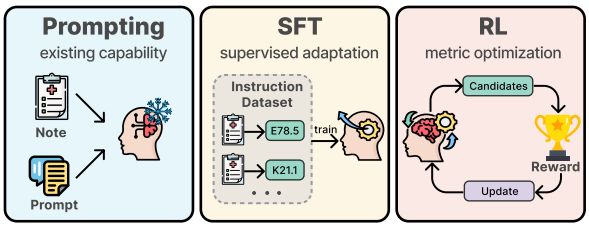

Figure 1: **From prompting to post-training for ICD coding.** Prompting relies on an LLM's existing inference-time capability. SFT adapts the model from labeled note-code demonstrations, teaching a parseable output schema and an empirical code prior. RL further optimizes generated candidate code sets with a sample-level F1 reward computed from the parsed codes.

Prior work has mainly addressed this challenge with discriminative ICD coders. Built on pretrained language model (PLM) encoders and label-wise prediction heads, these systems score codes in a fixed ICD label space and remain strong baselines for long-document coding (Mullenbach et al., 2018; Vu et al., 2020; Huang et al., 2022; Edin et al., 2023). In contrast, generative LLMs have often been reported as weak medical coders (Soroush et al., 2024). Most existing generative studies evaluate LLMs through inference-time use, including zero-shot or few-shot prompting, chain-of-thought reasoning, retrieval, reranking, or tool use, rather than adapting the model itself for ICD coding (Soroush et al., 2024; Boyle et al., 2023; Kwan, 2024; Baksi et al., 2025). These settings often lead to hallucinated or invalid codes and poor exact-code performance, but they leave open whether the same generative models can become accurate coders after task-specific post-training. This distinction matters because the generative formula-

tion remains attractive: LLMs provide a natural-language interface, can follow task instructions, and can emit ICD code sets directly as text.

Task-specific post-training offers a natural path forward, since it adapts the model itself rather than relying only on inference-time prompting or tools. In recent LLM development, supervised fine-tuning (SFT) has become a standard first stage for this adaptation: the model is trained on instruction-response pairs so that it learns the task format, domain style, and output conventions (Ouyang et al., 2022; Wei et al., 2021). For ICD coding, SFT can teach the model to produce valid code lists in the required format while learning the empirical distribution of medical codes. However, SFT is still trained by maximum likelihood. It optimizes next-token prediction rather than the non-differentiable set-level metrics, such as precision, recall, and F1, that determine coding performance (Ranzato et al., 2015). Reinforcement learning (RL) provides a complementary post-training stage: a complete model output is scored by a reward function, and the policy is updated to increase outputs with higher reward. This reward-based view has proven useful in modern LLM post-training, from human-preference alignment to mathematical reasoning and other verifiable tasks (Schulman et al., 2017; Ouyang et al., 2022). ICD coding is a natural fit for this framework because each generated code set can be scored directly by an F1-based reward. Yet RL-based post-training for generative LLM coders in ICD coding remains unexplored.

We address this gap through a controlled empirical progression across two MIMIC datasets, two ICD code systems, Top-50 and Full label settings, and multiple LLM backbones. As illustrated in Figure 1, we organize the study as a staged post-training ladder: ❶ **SFT** establishes the output schema and empirical code prior; ❷ **GRPO** uses a sample-level F1 reward to optimize generated code sets; and ❸ `PHI` (Progressive Hint Injection) extends GRPO with a diagnostic missed-code curriculum, using codes missed by earlier checkpoints as stochastic training-time hints while keeping inference hint-free. To our knowledge, this is the first study to apply RL-based post-training to generative LLM coders for ICD coding. Across this progression, we find that prompting-only evaluation substantially underestimates the potential of LLMs for ICD coding. SFT provides the main capability jump, GRPO further improves code-set prediction beyond SFT, especially in Full label settings, and `PHI` provides targeted gains on remaining missed-code cases and macro-level performance. Our contributions are:

- **Empirical reframing of generative ICD coding.** We show that prompting-only evaluation substantially underestimates the potential of generative LLMs for ICD coding. Under task-specific post-training, the conclusion changes from near-unusable prompting performance to competitive code-set prediction under controlled evaluation. We release code, data splits, and checkpoints to support reproducible comparison.
- **First RL-based post-training study for generative ICD coding.** To our knowledge, this is the first study to apply GRPO-style reinforcement learning to post-train generative LLMs for ICD code-set prediction. All generative methods use the same datasets, splits, parser, and metric set.
- **Diagnostic curriculum for missed-code recall.** We introduce `PHI`, a training-time missed-code curriculum that extends GRPO with codes missed by earlier checkpoints while keeping inference hint-free, providing targeted refinement for remaining missed-code cases.

2 Related Work

Discriminative ICD coding. Automated ICD coding has traditionally been formulated as extreme multi-label classification over long clinical documents. CAML introduced code-specific attention to connect each ICD prediction with supporting spans in the note (Mullenbach et al., 2018). Later discriminative models improved either the document encoder or the label representation: MultiResCNN uses multi-filter residual convolutions for long notes (Li and Yu, 2020), LAAT applies label attention with hierarchical learning for infrequent codes (Vu et al., 2020), convolutional attention models target long-tailed clinical document classification (Liu et al., 2021), and label-correlation rerankers model dependencies among ICD codes (Tsai et al., 2021). Encoder-based pretrained language models further strengthen this paradigm, with BERT-XML and PLM-ICD adapting contextual encoders to large ICD label spaces and long inputs (Zhang et al., 2020; Huang et al., 2022). Edin et al. (2023) show that these systems remain strong baselines on clean MIMIC-III and MIMIC-IV splits when preprocessing and thresholding are controlled. Our work uses these discriminative coders as strong reference points for

evaluating whether task-specific post-training can make generative LLM coders competitive under the same protocol.

Generative ICD coding. Generative ICD coding treats code assignment as a text-generation problem: an LLM reads a clinical note and emits the applicable ICD codes as text. Existing work has mainly explored this formulation through inference-time use of the model, including zero-shot or few-shot prompting, chain-of-thought reasoning, retrieval, reranking, and tool use (Boyle et al., 2023; Soroush et al., 2024; Kwan, 2024; Baksi et al., 2025). These studies show that generative coders can be flexible, but they often struggle with exact medical-code selection and can produce invalid or hallucinated codes (Soroush et al., 2024). Other work has explored domain-specific fine-tuning or rationale-based supervision for improving generative medical coding (Hou et al., 2025; Li et al., 2026). RL has also been applied to ICD coding, but prior work uses it for multi-agent path search over the ICD hierarchy with a discriminative policy network rather than to post-train a generative LLM (Lu et al., 2025). Our work focuses on this missing post-training axis by evaluating how prompting, SFT, GRPO, and `PHI` change generative ICD coding under a controlled protocol.

Post-training methods. Post-training has become a standard stage for adapting pretrained language models to downstream tasks. Supervised fine-tuning (SFT) trains models on instruction-response pairs and is commonly used to teach task format, domain style, and output conventions (Ouyang et al., 2022; Wei et al., 2021). However, maximum-likelihood training optimizes next-token prediction rather than non-differentiable task metrics, creating a mismatch between token-level learning and sequence- or set-level evaluation (Ranzato et al., 2015). Reinforcement-learning-based post-training addresses this mismatch by scoring complete model outputs with explicit rewards and updating the policy toward higher-reward generations. PPO is widely used in RLHF because its clipped policy objective stabilizes updates while a KL penalty keeps the policy close to a reference model (Schulman et al., 2017; Ouyang et al., 2022). More recent methods such as GRPO remove the learned value model and estimate advantages from groups of sampled responses, making them attractive for metric-driven post-training (Shao et al., 2024). Medical NLP has also studied domain and task adaptation for biomedical language models (Alsentzer et al., 2019; Gu et al., 2021). For generative ICD coding, however, post-training remains underexplored: existing work has mainly evaluated inference-time prompting, tool-augmented coding pipelines, or domain-specific fine-tuning, rather than a staged comparison of SFT and RL-based post-training under the same setup. Our work brings this post-training perspective to ICD coding by comparing SFT and GRPO with the same backbone, data, and evaluation protocol, then extending GRPO with `PHI`.

3 Preliminaries

Task formulation. Given a clinical note $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and a predefined ICD code taxonomy $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$, automated ICD coding aims to predict the subset of applicable codes $Y \subseteq \mathcal{C}$. Equivalently, Y can be represented by a binary vector $\mathbf{y} \in \{0,1\}^L$, where $y_j = 1$ indicates that code c_j applies to the note. We write the coding function abstractly as

$$f : \mathbf{x} \mapsto \hat{Y}, \qquad \hat{Y} \subseteq \mathcal{C},$$

where $\hat{Y}$ is the predicted code set. Since L can contain thousands of diagnosis and procedure codes, ICD coding is commonly treated as an extreme multi-label prediction problem.

Discriminative coders instantiate f by assigning a score to each code in the fixed taxonomy and selecting positive labels with a validation-tuned threshold (Mullenbach et al., 2018; Huang et al., 2022). Generative coders instead instantiate f through conditional text generation: given $\mathbf{x}$, an LLM autoregressively produces a structured response containing ICD codes, such as a `<code>...</code>` span, which is parsed into $\hat{Y}$. We use the same deterministic parser for all generative methods. Duplicated codes are deduplicated, and malformed or invalid codes are excluded from $\hat{Y}$. All post-training methods in this paper operate on this generative formulation, differing only in how the LLM is adapted before producing the parsed code set.

4 Method

We post-train a generative LLM for ICD coding through a staged pipeline (Figure 2). First, supervised fine-tuning (SFT) teaches the model to emit a parseable code span and provides an empirical code prior (Sec. 4.1). Second, GRPO optimizes the SFT

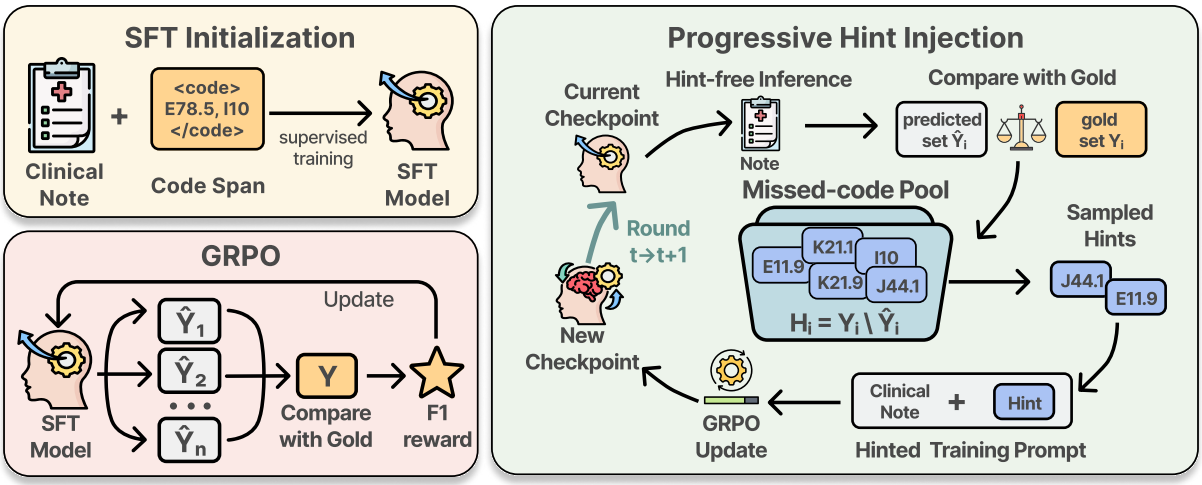

Figure 2: **Overview of our post-training pipeline.** A generative LLM is first supervised fine-tuned to emit a parseable code span, then optimized with GRPO using a sample-level F1 reward computed from the parsed code set. Progressive Hint Injection (`PHI`) iterates over rounds: it runs the current checkpoint without hints, collects missed codes $Y_i \setminus \hat{Y}_i$ into a per-sample hint pool, and samples training-time hints from this pool for the next GRPO round. Hints are used only during training. At inference, the model receives only the clinical note.

policy with a sample-level F1 reward computed from the parsed code set (Sec. 4.2). Third, Progressive Hint Injection (`PHI`) extends GRPO with an iterative curriculum that focuses training on codes missed by earlier checkpoints (Sec. 4.3). All stages output a single <code>$c_1, c_2, \ldots, c_n$</code> span, which we parse into the predicted set $\hat{Y}$. Hints are used only during training. At inference, the model receives only the clinical note, matching the hint-free setting used by all generative baselines.

4.1 SFT Initialization

Prompting relies on the base model's existing capability, but ICD coding requires a constrained output schema and an empirical code prior. We therefore use SFT as the first post-training stage, turning the instruction-tuned base model into a parseable generative ICD coder before RL.

Each SFT example consists of a user prompt $\mathbf{x}_i$, containing a fixed coding instruction, the discharge summary, and the required output format, paired with an assistant response $\mathbf{z}_i$ that serializes the gold code set Y_i inside a single <code>...</code> span. Let $\mathcal{D}_{\text{SFT}} = \{(\mathbf{x}_i, \mathbf{z}_i)\}_{i=1}^{N}$ denote these note-to-code demonstrations. We optimize the masked next-token objective

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{i=1}^{N} \sum_{t=1}^{|\mathbf{z}_i|} \log \pi_\theta(z_{i,t} \mid \mathbf{x}_i, \mathbf{z}_{i,<t}).$$

The loss is computed only over assistant response tokens, while user-prompt tokens are masked and do not contribute to the gradient. We implement this adaptation with LoRA and use the resulting policy $\pi_{\theta_{\text{SFT}}}$ to initialize both GRPO and `PHI`.

4.2 GRPO with a Sample-Level F1 Reward

SFT provides a valid generative coder, but its maximum-likelihood objective still trains the model at the token level. ICD coding, however, is evaluated after the generated text is parsed into a code set. We therefore use RL as a second post-training stage: each complete response is parsed into $\hat{Y}$, scored against the gold set Y, and used to update the generative policy. This lets the model optimize the quality of the predicted code set rather than only imitate the gold code string.

Reward. For each sampled response, we parse the predicted code set $\hat{Y}$ and compare it with the gold set Y. We use the sample-level F1 as the reward:

$$P = \frac{|\hat{Y} \cap Y|}{|\hat{Y}|}, \quad R = \frac{|\hat{Y} \cap Y|}{|Y|}, \quad r_{\text{F1}} = \frac{2PR}{P+R}.$$

For malformed responses or predictions with no valid code span, we set $r_{\text{F1}} = 0$. Our main evaluation reports corpus-level micro-F1, but this metric is aggregated over many examples and is not suitable as a rollout reward. GRPO needs response-level rewards to compare sampled outputs within each group. We therefore use sample-level F1 as a per-response proxy that rewards the same precision–recall tradeoff over parsed code sets, while avoiding token-level likelihood alone.

Optimization. Starting from the SFT policy $\pi_{\theta_{\text{SFT}}}$, we optimize the model with GRPO (Shao et al., 2024). GRPO is well suited to our setting because the reward is programmatically computed from the parsed code set, so no separate reward model is needed. It also avoids training a value critic by estimating relative advantages within a group of sampled responses, making it practical for RL post-training on large ICD datasets.

For each prompt $\mathbf{x}$, we sample a group of G responses $\{o_1, \ldots, o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$. Each response is parsed into a code set and scored with the sample-level F1 reward. GRPO then normalizes rewards within the group to obtain

$$A_j = \frac{r_j - \text{mean}(\{r_1, \ldots, r_G\})}{\text{std}(\{r_1, \ldots, r_G\})}. \tag{1}$$

The policy is updated with a clipped objective and a KL penalty toward a reference policy:

$$\mathcal{J}(\theta) = \mathbb{E}_j\Big[\min\left(\rho_j A_j,\ \text{clip}(\rho_j, 1-\epsilon, 1+\epsilon)A_j\right)\Big] - \beta_{\text{KL}}\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}), \tag{2}$$

where

$$\rho_j = \frac{\pi_\theta(o_j \mid \mathbf{x})}{\pi_{\theta_{\text{old}}}(o_j \mid \mathbf{x})}.$$

Here ϵ is the clipping range, and the KL penalty keeps the updated policy close to π_{ref}. Running this stage without hints gives our GRPO baseline.

4.3 Progressive Hint Injection

GRPO improves code-set prediction by optimizing sampled responses with an F1 reward, but some gold codes can remain repeatedly missed by later checkpoints. `PHI` turns these false negatives into a training-time curriculum. The model is first evaluated without hints to reveal the codes it currently misses, and later GRPO rounds expose sampled missed codes as in-context hints. The reward target remains the full gold set, not the hint subset.

Missed-code curriculum. We maintain a per-sample hint pool $H_i^{(t)}$ for each training example. The round-0 pool is initialized from the SFT model's hint-free predictions:

$$H_i^{(0)} = Y_i \setminus \hat{Y}_i^{\text{SFT}}. \tag{3}$$

At the end of round t, we run the current checkpoint on the training set without hints, parse the predicted code set $\hat{Y}_i^{(t)}$, and refresh the pool for the next round:

$$H_i^{(t+1)} = Y_i \setminus \hat{Y}_i^{(t)}. \tag{4}$$

Samples with empty pools are skipped in the next `PHI` round, so training concentrates on examples with remaining false negatives.

Stochastic hint injection. During `PHI` training, each retained prompt either remains hint-free or receives a sampled hint subset $h_i \subseteq H_i^{(t)}$. We sample hints without replacement from the pool. The sampling distribution gives higher priority to codes that are infrequent in the corpus or have low recall under the latest checkpoint, with clipping and temperature smoothing to prevent a few rare codes from dominating training. The hinted prompt states that the listed codes were missed previously but are confirmed applicable, includes their descriptions, and instructs the model to include the hinted codes while adding any other applicable codes supported by the note. Varying the number of injected hints and keeping a fraction of prompts hint-free reduces reliance on the hint list alone and keeps the model exposed to note-only training cases.

Test-time invariance. Hints are training-time supervision only. At inference, the model receives only the clinical note and must generate the full code set without hints, matching the hint-free setting used by all generative baselines.

5 Experiments

We evaluate the staged post-training progression for generative ICD coding on MIMIC-III (Johnson et al., 2016) and MIMIC-IV (Johnson et al., 2023). Our experiments ask three questions. First, can task-specific post-training move generative LLM coders beyond weak prompting-only performance and toward strong discriminative baselines? Second, how much does each stage, SFT, GRPO, and `PHI`, contribute to this progression? Third, how does `PHI` affect the remaining missed-code cases, including rare codes that prior benchmarks identify as especially challenging?

5.1 Experimental Setup

Datasets. We evaluate on MIMIC-III with ICD-9-CM codes and MIMIC-IV with ICD-10-CM codes (Johnson et al., 2016, 2023). Both datasets contain discharge summaries from the Beth Israel Deaconess Medical Center. We follow the stratified *clean* splits introduced by Edin et al. (2023) and report results on both the Top-50 and Full label settings. Our preprocessing preserves punctuation and document structure, which provide useful

Table 1: **MIMIC-III ICD-9-CM Results.** Methods are grouped by paradigm: **PLM Baselines**, **LLM (Prompting)**, and **LLM (Post-training)**. Within each split we report Micro / Macro Recall (R), Precision (P), and F1. All values are percentages. **Bold** and underline mark the best and second-best result in each column, respectively.

Method	Backbone	Top-50						Full					
		Micro			Macro			Micro			Macro		
		Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
❶ PLM Baselines													
CNN	CNN	59.7	65.6	62.5	52.4	57.4	52.6	40.3	48.9	44.2	8.5	11.3	8.8
GRU	Bi-GRU	36.3	56.1	44.1	30.0	44.2	33.5	42.1	49.0	45.3	9.1	12.1	9.4
CAML	CNN	45.1	69.0	54.5	38.9	56.8	44.4	48.8	53.1	50.9	17.9	20.6	18.0
MultiResCNN	ResNet	62.3	65.6	63.9	58.1	57.7	56.2	51.7	55.2	53.4	16.3	17.1	15.3
LAAT	Bi-LSTM	59.7	71.4	65.0	53.0	64.3	56.0	52.2	57.3	54.6	20.5	24.7	20.7
PLM-ICD	RoBERTa	68.1	68.0	68.1	63.3	63.7	63.5	**58.4**	61.0	**59.7**	**29.0**	**31.4**	**28.3**
❷ LLM (Prompting)													
Zero-shot	Qwen2.5-1.5B	0.8	29.0	1.6	0.4	4.7	0.7	0.1	15.1	0.2	0.0	0.1	0.0
	Qwen3-4B	5.5	52.7	10.0	2.7	11.7	4.0	1.6	24.2	3.0	0.2	0.4	0.2
Few-shot	Qwen2.5-1.5B	2.2	16.1	3.9	1.4	5.6	1.9	0.0	3.9	0.0	0.0	0.0	0.0
	Qwen3-4B	4.6	33.4	8.0	2.8	12.1	3.9	1.1	11.8	2.0	0.1	0.4	0.1
Zero-shot+CoT	Qwen2.5-1.5B	0.8	28.0	1.5	0.3	6.9	0.6	0.0	15.1	0.1	0.0	0.1	0.0
	Qwen3-4B	5.8	61.5	10.6	2.8	18.0	4.2	1.9	30.8	3.5	0.2	0.6	0.2
Few-shot + CoT	Qwen2.5-1.5B	5.1	17.8	8.0	3.3	4.6	2.8	0.6	6.9	1.1	0.0	0.1	0.0
	Qwen3-4B	11.0	52.8	18.2	5.9	18.2	7.8	3.2	22.9	5.7	0.3	0.8	0.3
❸ LLM (Post-training)													
SFT	Qwen2.5-1.5B	73.8	65.4	69.4	68.8	64.1	65.2	30.1	45.7	36.3	9.1	14.1	10.1
	Qwen3-4B	**75.2**	70.1	72.6	**70.3**	67.9	68.3	39.3	55.0	45.8	15.9	22.7	17.4
GRPO	Qwen2.5-1.5B	69.6	71.9	70.7	64.1	67.8	64.6	46.4	52.1	49.1	13.8	15.0	13.0
	Qwen3-4B	71.7	**75.6**	**73.6**	66.5	**72.2**	68.1	51.4	62.8	56.5	19.4	22.7	19.4
PHI (Ours)	Qwen2.5-1.5B	72.3	68.9	70.6	67.0	66.3	65.3	45.0	55.7	49.8	15.9	15.2	14.1
	Qwen3-4B	73.7	73.3	73.5	68.7	70.4	**68.6**	51.4	**62.9**	56.6	20.5	23.0	20.0

cues for section boundaries, abbreviations, negation, and clinical lists in autoregressive generation. To ensure a fair comparison, we train or evaluate all discriminative and generative methods on the same preprocessed corpus. Dataset statistics, preprocessing details, and split sizes are provided in Appendix A.

Baselines. We compare two families of ICD coders. **Discriminative methods** include CNN, GRU, and CAML (Mullenbach et al., 2018), MultiResCNN (Li and Yu, 2020), LAAT (Vu et al., 2020), and PLM-ICD (Huang et al., 2022). **Generative methods** use `Qwen2.5-1.5B` and `Qwen3-4B` as shared backbones and cover prompting variants, supervised fine-tuning, GRPO (Shao et al., 2024), and `PHI`. Prompting variants include zero-shot, few-shot, and chain-of-thought prompting (Wei et al., 2022). All generative variants are evaluated with the same parser, output format, and metric set to ensure a controlled comparison. Baseline configurations and prompt templates are in Appendix C.

Evaluation Metrics. We report precision, recall, and F1 at both the micro and macro levels, with all values shown as percentages. Micro scores are computed by pooling true positives, false positives, and false negatives over the test set, while macro scores are computed as the arithmetic mean of per-class scores following Edin et al. (2023). For discriminative models, we tune a single decision threshold on the validation set to maximize micro-F1. Generative models do not use a decision threshold. Full metric definitions are provided in Appendix G.

Implementation Details. For discriminative baselines, we adopt the tuned hyperparameters of Edin et al. (2023) and retrain each model on our preprocessed corpus. For generative methods, we use LoRA for SFT and GRPO-based RL training with vLLM-backed rollouts. All post-training variants share the same backbone, parser, and reward implementation. Hardware, hyperparameters, and training-time details are provided in Appendix F.

5.2 Main Results

Tables 1 and 2 show a clear shift in how generative LLM coders should be interpreted. Under prompting alone, both backbones perform far below discriminative coders, especially in the Full label setting, which is consistent with prior reports that LLMs are weak medical coders. However, this comparison mainly reflects the limitation of prompting-only evaluation rather than the capabil-

Table 2: **MIMIC-IV ICD-10-CM Results.** Methods are grouped by paradigm: **PLM Baselines**, **LLM (Prompting)**, and **LLM (Post-training)**. Within each split we report Micro / Macro Recall (R), Precision (P), and F1. All values are percentages. **Bold** and underline mark the best and second-best result in each column, respectively.

Method	Backbone	Top-50						Full					
		Micro			Macro			Micro			Macro		
		Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
❶ PLM Baselines													
CNN	CNN	71.6	69.8	70.7	66.0	65.9	64.4	41.8	53.2	46.8	5.6	9.4	6.3
GRU	Bi-GRU	71.9	70.3	71.1	66.9	65.5	65.4	43.4	54.6	48.3	8.6	13.2	9.4
CAML	CNN	70.3	68.6	69.4	64.3	64.6	63.4	51.5	58.1	54.6	14.9	18.0	15.1
MultiResCNN	ResNet	71.7	70.5	71.1	67.2	65.1	64.8	52.0	58.7	55.2	16.9	17.9	15.7
LAAT	Bi-LSTM	71.9	71.6	71.8	66.9	67.2	65.9	53.8	59.3	56.4	18.8	21.7	18.3
PLM-ICD	RoBERTa	73.3	73.6	73.5	68.9	69.8	68.3	**57.0**	62.4	**59.6**	**24.1**	**26.8**	**23.7**
❷ LLM (Prompting)													
Zero-shot	Qwen2.5-1.5B	0.1	15.0	0.3	0.1	17.0	0.2	0.1	14.4	0.1	0.0	0.6	0.0
	Qwen3-4B	0.4	34.3	0.9	0.4	24.4	0.8	0.3	18.9	0.6	0.1	0.7	0.1
Few-shot	Qwen2.5-1.5B	1.1	13.5	2.0	0.8	6.0	1.1	0.4	5.7	0.7	0.0	0.1	0.0
	Qwen3-4B	5.2	17.0	7.9	5.9	27.3	6.2	0.1	4.9	0.2	0.0	0.3	0.0
Zero-shot + CoT	Qwen2.5-1.5B	0.2	13.3	0.4	0.2	18.0	0.3	0.3	17.1	0.6	0.0	0.5	0.0
	Qwen3-4B	1.8	45.3	3.5	1.9	29.8	3.2	1.6	24.2	3.0	0.2	1.2	0.2
Few-shot + CoT	Qwen2.5-1.5B	0.5	15.2	1.0	0.3	4.5	0.6	0.2	6.3	0.3	0.0	0.1	0.0
	Qwen3-4B	7.3	28.6	11.7	6.9	27.4	8.5	1.5	11.1	2.6	0.2	0.9	0.2
❸ LLM (Post-training)													
SFT	Qwen2.5-1.5B	**74.3**	69.6	71.9	71.7	67.6	69.2	50.3	48.8	49.5	12.8	13.1	11.8
	Qwen3-4B	**74.3**	71.7	73.0	**71.8**	69.6	**70.3**	46.8	58.5	52.0	19.3	24.2	20.0
GRPO	Qwen2.5-1.5B	71.1	74.1	72.6	68.5	71.0	68.8	49.1	58.5	53.3	10.7	13.8	10.8
	Qwen3-4B	71.5	**75.9**	**73.6**	68.8	**73.1**	69.8	54.2	**63.7**	58.6	20.2	24.9	20.6
PHI **(Ours)**	Qwen2.5-1.5B	73.8	70.8	72.3	71.1	68.6	69.3	51.0	57.0	53.8	13.2	13.9	12.3
	Qwen3-4B	73.7	73.2	73.4	70.8	71.1	70.1	55.0	63.0	58.7	21.0	25.0	21.2

ity of task-adapted generative coding. After task-specific post-training, the conclusion changes substantially. SFT provides the main capability jump, GRPO improves code-set optimization in the larger label space, and PHI acts as a focused refinement stage for remaining missed codes. Overall, the results suggest that generative LLMs are not inherently poor ICD coders. Rather, prompting-only evaluation conflates the limits of prompting with the potential of task-adapted generative coding.

SFT provides the main capability jump. The largest improvement comes from supervised fine-tuning. Prompted LLMs often fail not only because they lack ICD-specific knowledge, but also because they do not reliably follow the required output schema, generate invalid codes, or place codes outside the parseable <code>...</code> span. SFT addresses this first bottleneck by training the model on note-code demonstrations, teaching both a parseable output format and the empirical ICD code distribution. For example, on MIMIC-III Top-50, Qwen3-4B improves from the best prompting micro-F1 of 18.2 to 72.6 after SFT, already surpassing PLM-ICD. Similar jumps appear on MIMIC-IV and in the Full setting, where SFT brings the model from near-unusable prompting performance into a strong working range. This shows that the main barrier for prompted LLMs is not the generative formulation itself, but the lack of task-specific adaptation.

GRPO improves code-set prediction beyond SFT. After SFT establishes a usable generative coder, GRPO provides the main metric-oriented refinement. Its effect is most visible in the Full setting, where the model must balance precision and recall over a much larger ICD taxonomy. Across both MIMIC-III and MIMIC-IV, GRPO consistently improves Full-setting micro-F1 over SFT, with especially large gains on MIMIC-III. In contrast, Top-50 results are already strong after SFT, so GRPO mainly adjusts the precision–recall trade-off rather than producing another large jump. This pattern suggests that reward-based post-training is most useful once the model has learned the coding format and code prior, but still needs to optimize complete code-set quality in a large label space.

PHI refines the remaining missed-code cases. Unlike SFT and GRPO, this stage is designed for focused refinement rather than broad capability acquisition. It starts from the SFT-initialized policy and repeatedly trains on cases where the current

checkpoint still misses gold codes. Samples whose hint pools become empty are skipped in later PHI rounds, which makes training faster and concentrates updates on unresolved cases. This design can reduce repeated exposure to easy or already-solved cases that contribute heavily to micro-F1. As a result, PHI is often close to GRPO on headline micro-F1 rather than uniformly higher. Its benefit is more visible in the Full setting and in macro-level performance, where remaining missed and lower-frequency codes matter more. This makes PHI a targeted complement to GRPO: GRPO improves overall code-set quality, while PHI redirects later training toward the codes that the current policy still fails to recover.

Post-trained LLMs narrow the gap to PLM-ICD. PLM-ICD remains a strong reference point, especially in the Full label setting. Still, post-training substantially changes the comparison. In Top-50 settings, post-trained LLMs reach or exceed PLM-ICD on MIMIC-III and nearly match it on MIMIC-IV. In Full settings, they do not uniformly surpass PLM-ICD, but GRPO and PHI close much of the gap while retaining the generative formulation. Model size helps, especially after SFT and GRPO in the Full setting, but it is not the main driver: prompted Qwen3-4B remains far below post-trained Qwen2.5-1.5B. Thus, the main conclusion is not that generative LLMs replace discriminative coders outright, but that task-specific post-training makes them competitive in settings where prompting alone fails.

6 Future Directions

Our results suggest that future generative ICD coders should focus on the Full label setting, where the key bottleneck is rare-code recall. Discriminative PLM coders score every label with an explicit classifier head, giving them a natural mechanism for label coverage and threshold-based recall control. Generative LLM coders are different: they must actively recall and emit the correct codes from a large taxonomy, so rare codes can be omitted even when the note contains supporting evidence. PHI shows that training-time exposure to previously missed codes is a promising way to improve this behavior, but stronger mechanisms are needed to fully close the Full-setting gap.

Agentic RL with retrieval. A natural direction is to let the model learn when to consult external coding resources before producing the final code set. During RL, the policy could decide whether to retrieve ICD descriptions, hierarchy neighbors, or similar coded cases, then use this evidence to improve recall. This is especially relevant for rare codes, where the model may not have enough parametric knowledge to generate the correct code without assistance.

Hybrid PLM–LLM coding. Another direction is to combine the label coverage of discriminative coders with the flexibility of generative LLMs. A PLM classifier can provide a high-recall candidate set over the full taxonomy, while the LLM verifies, completes, or explains the final prediction. Such systems may be especially useful in the Full setting, where exhaustive label scoring remains a major advantage of discriminative models.

Recall-oriented rewards and decoding. Our GRPO reward uses sample-level F1, which balances precision and recall. Future work could explore rewards and decoding strategies that explicitly emphasize rare-code recall, such as F_β rewards, class-balanced rewards, bonuses for repeatedly missed codes, or recall-calibrated stopping rules. The key challenge is to improve low-frequency code recovery without encouraging broad overprediction and precision collapse.

7 Conclusion

We presented a controlled empirical study of task-specific post-training for generative ICD coding. By comparing prompting, SFT, and RL-based post-training under the same protocol, this study provides the first evaluation of whether RL-based post-training can improve generative LLM coders for ICD code-set prediction. Our results show that prompting-only evaluation substantially underestimates the potential of LLM coders: SFT provides the main capability jump, GRPO improves code-set prediction in large label spaces, and PHI offers a targeted curriculum for remaining missed-code cases. While strong discriminative PLM coders remain highly competitive, task-specific post-training narrows the gap and makes generative LLMs viable ICD coders under controlled evaluation. More broadly, these findings shift the central question from whether LLMs can code from prompting alone to how post-training, retrieval, and reward mechanisms should be designed to improve reliable recall over the full ICD taxonomy.

8 Limitations

Our evaluation is bounded in several ways. First, we use MIMIC-III and MIMIC-IV, which are standard ICD coding benchmarks but come from a limited clinical data ecosystem. The results may not fully reflect documentation styles, coding practices, or label distributions across institutions. Second, due to computational constraints, our generative experiments cover two relatively small open-source Qwen backbones. Larger LLMs, other model families, and closed-source models may show different scaling behavior under the same post-training pipeline. Finally, the Full label setting remains challenging, especially for rare-code recall, and post-training does not fully close the gap to PLM-ICD.

References

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical bert embeddings. In *Proceedings of the 2nd clinical natural language processing workshop*, pages 72–78.

Krishanu Das Baksi, Elijah Soba, John J Higgins, Ravi Saini, Jaden Wood, Jane Cook, Jack I Scott, Nirmala Pudota, Tim Weninger, Edward Bowen, et al. 2025. Medcoder: A generative ai assistant for medical coding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pages 449–459.

Joseph S Boyle, Antanas Kascenas, Pat Lok, Maria Liakata, and Alison Q O'Neil. 2023. Automated clinical coding using off-the-shelf large language models. *arXiv preprint arXiv:2310.06552*.

Joakim Edin, Alexander Junge, Jakob D Havtorn, Lasse Borgholt, Maria Maistro, Tuukka Ruotsalo, and Lars Maaløe. 2023. Automated medical coding on mimic-iii and mimic-iv: A critical review and replicability study. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pages 2572–2582.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23.

Zhen Hou, Hao Liu, Jiang Bian, Xing He, and Yan Zhuang. 2025. Enhancing medical coding efficiency through domain-specific fine-tuned large language models. *npj Health Systems*, 2(1):14.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Chao-Wei Huang, Shang-Chi Tsai, and Yun-Nung Chen. 2022. PLM-ICD: Automatic ICD coding with pretrained language models. In *Proceedings of the 4th Clinical Natural Language Processing Workshop*, pages 10–20, Seattle, WA. Association for Computational Linguistics.

Shaoxiong Ji, Xiaobo Li, Wei Sun, Hang Dong, Ara Taalas, Yijia Zhang, Honghan Wu, Esa Pitkänen, and Pekka Marttinen. 2024. A unified review of deep learning for automated medical coding. *ACM Computing Surveys*, 56(12):1–41.

Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. 2023. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Keith Kwan. 2024. Large language models are good medical coders, if provided with tools. *arXiv preprint arXiv:2407.12849*.

Fei Li and Hong Yu. 2020. Icd coding from clinical text using multi-filter residual convolutional neural network. In *proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8180–8187.

Mingyang Li, Viktor Schlegel, Tingting Mu, Wuraola Oyewusi, Kai Kang, and Goran Nenadic. 2026. Evaluation and llm-guided learning of icd coding rationales. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4969–5003.

Yang Liu, Hua Cheng, Russell Klopfer, Matthew R Gormley, and Thomas Schaaf. 2021. Effective convolutional attention network for multi-label clinical document classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5941–5953.

Pengli Lu, Xue Yang, Jingjin Xue, and Fentang Gao. 2025. Enhancing icd code assignment with hierarchical multi-agent collaboration and reinforcement learning. *Biomedical Signal Processing and Control*, 110:109094.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

1 (Long Papers), pages 1101–1111, New Orleans, Louisiana. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Ali Soroush, Benjamin S Glicksberg, Eyal Zimlichman, Yiftach Barash, Robert Freeman, Alexander W Charney, Girish N Nadkarni, and Eyal Klang. 2024. Large language models are poor medical coders—benchmarking of medical code querying. *Nejm Ai*, 1(5):AIdbp2300040.

Fei Teng, Yiming Liu, Tianrui Li, Yi Zhang, Shuangqing Li, and Yue Zhao. 2022. A review on deep neural networks for icd coding. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4357–4375.

Shang-Chi Tsai, Chao-Wei Huang, and Yun-Nung Chen. 2021. Modeling diagnostic label correlation for automatic icd coding. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 4043–4052.

Thanh Vu, Dat Quoc Nguyen, and Anthony Nguyen. 2020. A label attention model for icd coding from clinical text. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 3335–3341.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

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

Zachariah Zhang, Jingshu Liu, and Narges Razavian. 2020. Bert-xml: Large scale automated icd coding using bert pretraining. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 24–34.

A Dataset Statistics and Preprocessing

Sources. All MIMIC-based experiments use credentialed-access clinical data released through PhysioNet. For MIMIC-III (Johnson et al., 2016), we use discharge summaries paired with ICD-9-CM diagnosis and procedure codes. For MIMIC-IV (Johnson et al., 2023), we use MIMIC-IV and MIMIC-IV-Note discharge summaries paired with ICD-10-CM diagnosis and ICD-10-PCS procedure codes. Both datasets require completion of the appropriate PhysioNet credentialing and data-use agreements.

Preprocessing and splits. We follow the preprocessing pipeline of Edin et al. (2023) to convert the raw note and code-assignment tables into sectioned-note datasets. Each processed example contains a discharge summary, its deduplicated target code set, a split identifier, and a note identifier. We use the corresponding clean splits from Edin et al. (2023), which remove rare codes with fewer than ten training occurrences and use multi-label stratified sampling so that evaluated codes appear in both training and test sets. Unlike the original preprocessing, which lowercases text and strips non-alphanumeric characters, we preserve common punctuation while still lowercasing the note text. Punctuation, section boundaries, abbreviations, negation cues, and list structure provide useful signals for autoregressive generation. To keep comparisons fair, all discriminative and generative methods are trained or evaluated on the same punctuation-preserved inputs.

Top-50 vs. Full. For both MIMIC-III and MIMIC-IV, we construct two label-set variants. The **Full** variant retains all diagnosis and procedure codes that remain after preprocessing. The **Top-50** variant restricts the label space to the 50 most frequent codes in the corresponding training set, following the standard top-code evaluation protocol used in prior medical-coding work. Examples with no labels after Top-50 filtering are excluded from the corresponding Top-50 split.

Instruction-format conversion. After preprocessing, each dataset is converted into a ShareGPT-style instruction-tuning format. The user turn contains the discharge summary and an instruction to output all applicable diagnosis and procedure codes. The assistant turn contains the gold code set enclosed in a structured `<code>...</code>` block. The same conversion is applied to MIMIC-III and MIMIC-IV, yielding Full and Top-50 variants for supervised fine-tuning, prompting evaluation, and reinforcement-learning experiments. Because the processed examples contain restricted clinical text, we do not redistribute the note-level data. Reproducing the datasets requires authorized PhysioNet access and rerunning the preprocessing pipeline.

B Models

We use two instruction-tuned Qwen backbones for all generative experiments: `Qwen2.5-1.5B` and `Qwen3-4B`. `Qwen2.5-1.5B` provides a lightweight setting for studying whether post-training can improve a small generative ICD coder, while `Qwen3-4B` provides a stronger backbone that remains practical for SFT and GRPO-based training. Both models are evaluated under the same prompting, SFT, GRPO, and `PHI` settings.

C Baseline Configurations

Discriminative baselines. CNN, GRU, and CAML (Mullenbach et al., 2018) share a Word2Vec embedding layer trained on MIMIC notes. CNN and GRU pool the encoder output before classification. CAML applies the label-aware attention of its original work. MultiResCNN (Li and Yu, 2020) replaces the single-filter convolution with a multi-scale residual variant. LAAT (Vu et al., 2020) uses a Bi-LSTM encoder with a label attention layer. PLM-ICD (Huang et al., 2022) uses a RoBERTa encoder pre-trained on PubMed and MIMIC, with the same label attention as LAAT. All hyperparameters follow Edin et al. (2023) Table 3.

Generative baselines. We use `Qwen2.5-1.5B` and `Qwen3-4B` as shared backbones for all generative methods. The prompting baselines include zero-shot, few-shot, zero-shot chain-of-thought, and few-shot chain-of-thought prompting (Wei et al., 2022). The training-based generative baselines follow the staged post-training pipeline studied in the main paper: *SFT* fine-tunes the backbone on note-to-code demonstrations, *GRPO* further optimizes the SFT policy with a sample-level F1 reward, and `PHI` extends GRPO with progressive training-time hints from missed codes. All generative methods use the same output format, parser, and evaluation metrics.

D Prompt Templates

We summarize the prompt templates used for the prompting baselines below. The placeholders

{CORPUS}, {CODE_SYSTEM}, and {LABEL_SCOPE} are filled according to the dataset and label-set setting: MIMIC-III or MIMIC-IV, ICD-9-CM or ICD-10-CM/PCS, and Top-50 or Full. Importantly, the prompt does not include an explicit candidate code list. The Top-50 and Full settings determine only the dataset construction and evaluation label space. During evaluation, we parse and score only the final codes enclosed in <code>...</code>.

Zero-shot. The zero-shot setting contains no clinical demonstrations. The short code block shown in the prompt is a formatting example only and is not an allowed-code list.

Zero-shot Prompt

You are a clinical coding specialist assigning *{CODE_SYSTEM}* codes from a discharge summary.
Use *{LABEL_SCOPE}*.
Output only the final *{CODE_SYSTEM}* codes in <code>...</code>. Do not include any other text.
Formatting example only, not a clinical example and not codes to copy:
<code>*{FORMAT_EXAMPLE_CODES}*</code>
Now code the following discharge summary.
Discharge Summary:
{DISCHARGE_SUMMARY}
Answer:

Few-shot. The few-shot setting prepends three retrieved training examples, each consisting of a discharge summary and its gold code block. Retrieval is based on note similarity.

Few-shot Prompt

You are a clinical coding specialist assigning *{CODE_SYSTEM}* codes from a discharge summary.
Use *{LABEL_SCOPE}*.
Output only the final *{CODE_SYSTEM}* codes in <code>...</code>. Do not include any other text.
Examples:
Example 1
Discharge Summary:
{SHOT_NOTE_1}
Answer:
<code>*{SHOT_CODES_1}*</code>
Example 2
Discharge Summary:
{SHOT_NOTE_2}
Answer:
<code>*{SHOT_CODES_2}*</code>
Example 3
Discharge Summary:
{SHOT_NOTE_3}
Answer:
<code>*{SHOT_CODES_3}*</code>
Now code the following discharge summary.
Discharge Summary:
{DISCHARGE_SUMMARY}
Answer:

Zero-shot + CoT. The zero-shot CoT setting asks the model to write a brief structured rationale inside <think>...</think> before giving the final code block. Only the codes inside <code>...</code> are scored.

Zero-shot + CoT Prompt

You are a clinical coding specialist assigning *{CODE_SYSTEM}* codes from a discharge summary.
Use *{LABEL_SCOPE}*.
First write a brief structured rationale in <think>...</think>, then output the final *{CODE_SYSTEM}* codes in <code>...</code>.
Do not put codes outside the <code> block.
Formatting example only, not a clinical example and not codes to copy:
<think>Briefly identify the documented diagnoses and procedures, then map them to *{CODE_SYSTEM}* codes.</think>
<code>*{FORMAT_EXAMPLE_CODES}*</code>
Now code the following discharge summary.
Discharge Summary:
{DISCHARGE_SUMMARY}
Answer:

Few-shot + CoT. The few-shot CoT setting combines the same three retrieved demonstrations with the structured rationale format. As above, evaluation ignores the rationale and scores only the final code block.

Few-shot + CoT Prompt

You are a clinical coding specialist assigning *{CODE_SYSTEM}* codes from a discharge summary.
Use *{LABEL_SCOPE}*.
First write a brief structured rationale in <think>...</think>, then output the final *{CODE_SYSTEM}* codes in <code>...</code>.
Do not put codes outside the <code> block.
Examples:
Example 1
Discharge Summary:
{SHOT_NOTE_1}
Answer:
<think>Review the discharge summary for documented diagnoses and procedures, then map the supported findings to *{CODE_SYSTEM}* codes.</think>
<code>*{SHOT_CODES_1}*</code>
Example 2
Discharge Summary:
{SHOT_NOTE_2}
Answer:
<think>Review the discharge summary for documented diagnoses and procedures, then map the supported findings to *{CODE_SYSTEM}* codes.</think>
<code>*{SHOT_CODES_2}*</code>
Example 3
Discharge Summary:
{SHOT_NOTE_3}
Answer:
<think>Review the discharge summary for documented diagnoses and procedures, then map the supported findings to *{CODE_SYSTEM}* codes.</think>
<code>*{SHOT_CODES_3}*</code>
Now code the following discharge summary.
Discharge Summary:
{DISCHARGE_SUMMARY}
Answer:

E Training Details

SFT. We fine-tune with LoRA (Hu et al., 2022) of rank 8, learning rate 1×10^{-5} with cosine schedule, and 3 epochs. The optimizer is AdamW. Each pair is formatted as a single ShareGPT-style user turn followed by the gold code span as the assistant turn. The user segment is masked out of the loss, so the cross-entropy objective is computed only over the assistant response tokens.

GRPO. We sample $G = 8$ trajectories per prompt and form within-group advantages from standardized sample-level F1 rewards. The actor is updated with the clipped GRPO objective and a KL penalty to the reference policy. The KL coefficient is 1×10^{-3} throughout. The maximum response length is 196 tokens, sufficient for any plausible code list. Rollouts use vLLM with top-p sampling.

Reward implementation. For each generated response, we extract the predicted code set $\hat{Y}$ from the final <code>...</code> span, normalize and deduplicate the codes, and compare it with the gold set Y. The scalar reward passed to GRPO is the sample-level F1 between $\hat{Y}$ and Y. Corpus-level micro-F1 and macro-F1 are computed only for logging and evaluation, not used as rollout rewards.

PHI hint training. We run PHI for 3 to 5 rounds, with one GRPO epoch per round. The round-0 hint pool is initialized from the SFT model's hint-free predictions:

$$H_i^{(0)} = Y_i \setminus \hat{Y}_i^{\mathrm{SFT}}.$$

At each subsequent round, we run greedy inference with the previous-round checkpoint on the training set without hints, parse $\hat{Y}_i^{(t-1)}$, and refresh

$$H_i^{(t)} = Y_i \setminus \hat{Y}_i^{(t-1)}.$$

Samples whose hint pool becomes empty are filtered out of the next round. At each training step, a hint is injected with probability 0.5. When injecting, we sample between 1 and 5 missed codes without replacement from $H_i^{(t)}$ using the clipped and temperature-smoothed code weights described below.

Hint sampling. For each missed code c, we assign a rare-and-hard priority based on its corpus frequency and latest-round recall:

$$a_c = \frac{1}{\sqrt{\mathrm{freq}_c}} \cdot \frac{1}{\max(\mathrm{recall}_c, 0.05)}.$$

We clip this priority at three times the median priority and sample hints without replacement from the temperature-smoothed distribution

$$p(c \mid H_i^{(t)}) = \frac{\tilde{w}_c^{1/\tau}}{\sum_{c' \in H_i^{(t)}} \tilde{w}_{c'}^{1/\tau}},$$

where $\tilde{w}_c$ is the clipped priority and $\tau = 3.0$. This preserves a bias toward infrequent and low-recall codes while preventing a few rare codes from dominating training.

Hint prompt. When hints are injected, we insert a training hint before the output-format instruction. The hint lists the previously missed codes, includes their ICD descriptions, and gives a mini example showing that the final answer should include both the hinted codes and any additional codes supported by the note. The template is:

```
Training hint:
The following ICD-{VERSION} codes were
missed previously, but they are
confirmed applicable to this case.
Include these hinted codes in the
final <code> answer, then add any
other applicable ICD-{VERSION} codes
supported by the note.
Hinted codes:
- {CODE_1} ({DESCRIPTION_1})
- {CODE_2} ({DESCRIPTION_2})
...
Mini example:
If the hint says:
- 276.2 (Acidosis)
and the note also supports 401.9 and
584.9, then the final answer should
include both the hinted code and the
other supported codes:
<code>276.2, 401.9, 584.9</code>
```

Descriptions are taken from the ICD description files and truncated to a maximum length. The reward target remains the full gold set Y_i, not the hinted subset.

F Implementation Details

Hardware and runtime. Discriminative baselines are trained on a single A100 80GB GPU. Generative RL experiments are run on $4\times$ H100 80GB GPUs. Stage-1 SFT takes about 24 hours. A Top-50 GRPO run takes about 120 hours, while a Full-code GRPO run takes about 160 hours. Each PHI round adds one additional GRPO epoch initialized from the previous-round checkpoint.

Frameworks. Stage-1 SFT is implemented with LlamaFactory. Stages 2 and 3 use EasyR1 with vLLM-backed rollouts. The reward function, parser, and preprocessing utilities are released with the codebase to support reproducibility.

Long-note truncation. Discharge summaries can exceed the model context window. We therefore apply a tiered middle-out truncation scheme that prioritizes preserving section headers and the discharge-diagnoses block. This keeps high-value clinical structure and diagnosis anchors in the input while fitting long notes into the model context.

G Evaluation Metrics

Definitions. For each predicted set $\hat{Y}$ and gold set Y over the label vocabulary $\mathcal{C}$, we compute precision, recall, and F1 at both the micro and macro levels. Micro scores pool true positives, false positives, and false negatives across the full test set before computing each metric. Macro scores first compute per-class precision, recall, and F1, then report the arithmetic mean over classes following Edin et al. (2023). Macro F1 is therefore the arithmetic mean of per-class F1, not the harmonic mean of macro precision and macro recall. This choice avoids the artificial deflation discussed in prior work.

Decision threshold tuning. For discriminative models, we tune a single decision threshold on the validation set to maximize micro-F1, then apply the same threshold at test time. We do not tune separate thresholds per class. Generative models do not use a decision threshold; their predictions are obtained by parsing the generated `<code>...</code>` span into a code set.

H LLM Usage

We used Large Language Models (ChatGPT/-Claude/Gemini) exclusively for grammatical correction in this manuscript. The LLMs played no role in research ideation, methodology, or scientific content generation. All technical contributions and scientific insights are original work by the authors.