

000 001 002 003 004 005 LANGUAGE BOTTLENECK MODELS: A FRAMEWORK 006 FOR QUALITATIVE COGNITIVE DIAGNOSIS 007 008 009

010 **Anonymous authors**
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029

Paper under double-blind review
030
031
032

ABSTRACT

033 Accurately assessing student knowledge is central to education. Cognitive Di-
034 agnosis (CD) models estimate student proficiency at a fixed point in time, while
035 Knowledge Tracing (KT) methods model evolving knowledge states to predict fu-
036 ture performance. However, CD and probabilistic KT models represent knowledge
037 states via quantitative estimates of knowledge concept mastery, limiting expressiv-
038 ity, while deep learning-based KT methods prioritize predictive accuracy at the cost
039 of interpretability. We propose Language Bottleneck Models (LBMs), a general
040 framework for producing textual knowledge state summaries that retain predictive
041 power. LBMs use an encoder LLM to produce minimal textual descriptions of
042 a student’s knowledge state, which a decoder LLM then uses to reconstruct past
043 responses and predict future performance. This natural-language bottleneck yields
044 human-interpretable summaries that go beyond the quantitative outputs of CD
045 models and capture nuances like misconceptions. Experiments show zero-shot
046 LBMs rival state-of-the-art CD and KT accuracy on synthetic arithmetic bench-
047 marks and real-world datasets (Eedi and XES3G5M). We also show the encoder
048 can be finetuned with reinforcement learning, using prediction accuracy as reward,
049 to improve summary quality. Beyond matching predictive performance, LBMs
050 reveal qualitative insights into student understanding that quantitative approaches
051 cannot capture, showing the value of flexible textual representations for educational
052 assessment.
053

1 INTRODUCTION

033 **Knowledge state modeling** A fundamental objective in education is accurately assessing what a
034 student knows, identifying misconceptions, and understanding how their knowledge evolves over
035 time (Posner et al., 1982; Larkin, 2012; Chen et al., 2020). Teachers intuitively achieve this through
036 diagnostic reasoning: by observing students’ answers, they infer not merely correctness, but deeper
037 patterns reflecting conceptual mastery or specific misunderstandings.
038

039 **Limitations of Cognitive Diagnosis and Knowledge Tracing** Cognitive Diagnosis (CD) (Templin
040 et al., 2010; Wang et al., 2024) models produce diagnostic reports but are constrained to proficiency
041 estimates over a fixed set of knowledge concepts. In parallel, Knowledge Tracing (KT) (Shen et al.,
042 2024) models excel at predicting future performance based on observed past responses, yet remain
043 limited to either estimating mastery over knowledge concepts (Corbett & Anderson, 1994; Zhou
044 et al., 2024) or operating as black-box models requiring post-hoc interpretability (e.g. DKT (Piech
045 et al., 2015; Ghosh et al., 2020)).
046

047 **Limitations of existing LLM-based approaches** Recent approaches leveraging large language
048 models (LLMs) have shown that LLMs can produce sensible predictions of students’ future behavior
049 when provided with relevant information (Li et al., 2024a; Kim et al., 2024), help mitigate the KT’s
050 cold-start problem (Lee et al., 2024) and be finetuned to improve accuracy on KT tasks (Wang
051 et al., 2025). Nonetheless, these LLM-based methods still lack rigorous interpretability, as they
052 either treat the model as a black-box or rely on free-form explanations that are susceptible to
053 hallucination (Bender et al., 2021). As a result, such methods fail to provide grounded, reliable
representations of knowledge that can be trusted in educational practice. This gap motivates a
principled approach where interpretability is an intrinsic design constraint.
054

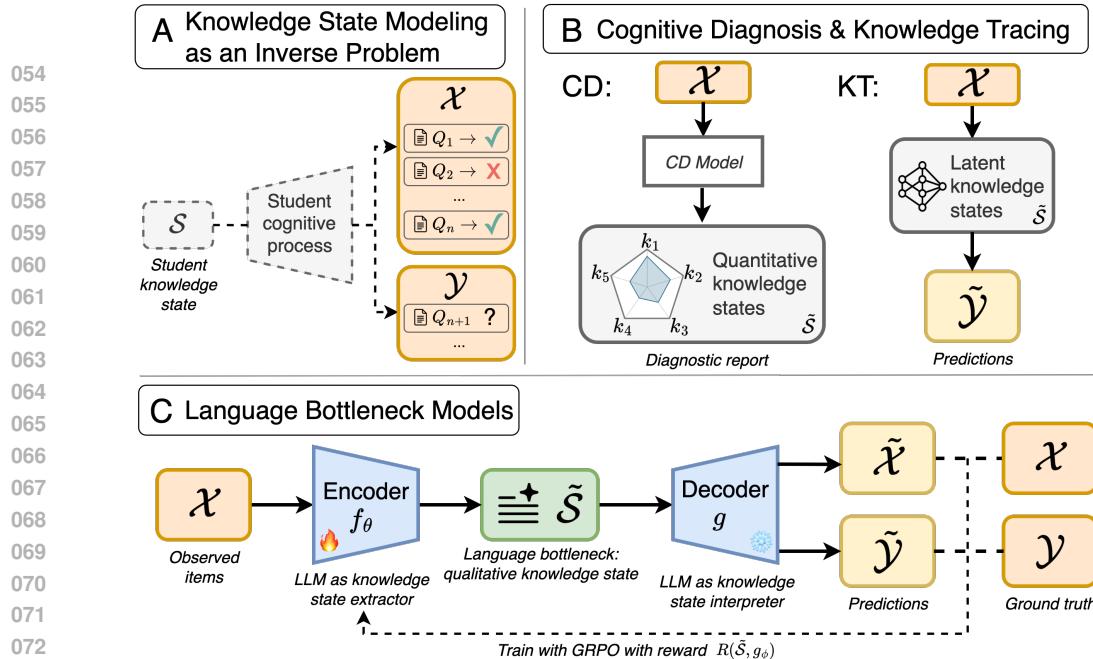


Figure 1: **Language Bottleneck Models for Knowledge Modeling.** (A) Past and future behavior \mathcal{X} and \mathcal{Y} are caused by a certain knowledge state \mathcal{S} held by the student when answering questions. (B) CD and KT models represent the knowledge state via quantitative proficiency vectors or opaque latent embeddings. (C) LBMs approximate the knowledge state using natural language summaries which are used to predict past and future behavior.

Knowledge state modeling as an inverse problem Assessing student knowledge can be framed as an *inverse problem* (Figure 1A): observable answers are generated by an underlying knowledge state \mathcal{S} , and the goal is to infer a faithful approximation $\tilde{\mathcal{S}}$. CD models constrain $\tilde{\mathcal{S}}$ to fixed quantitative proficiency estimates over knowledge concepts (Figure 1B), providing interpretability but lacking flexibility to capture nuanced knowledge states. KT models prioritize predictive performance but have uninterpretable latent representations (Figure 1B). We adopt the inverse-problem framing but allow $\tilde{\mathcal{S}}$ to be expressed as concise, free-form natural language, [so knowledge states remain predictive while avoiding the constraints of a fixed set of knowledge concepts](#).

Language Bottleneck Models We introduce *Language Bottleneck Models (LBMs)* as a general framework for compressing a student’s interaction history into a predictive, text-based knowledge state. As shown in Figure 1C, an encoder LLM maps a student’s interaction history \mathcal{X} into a natural language summary $\tilde{\mathcal{S}}$, while a frozen decoder LLM must reconstruct past responses and predict future ones using only that summary. In an educational setting, they constitute a language-based approach to Cognitive Diagnosis, not bound to fixed skill vocabularies and can flexibly describe nuanced insights such as misconceptions.

Contributions.

- We cast knowledge state modeling as an inverse problem over open-ended textual representations—building on ideas from Cognitive Diagnosis but replacing rigid concept proficiencies with a flexible natural-language knowledge state.
- To instantiate this, we introduce Language Bottleneck Models (LBMs), which encode observed student behavior into predictive text-based summaries of knowledge states.
- We extensively evaluate LBMs on synthetic and real-world datasets against 14 KT and CD baselines across 7 open- and closed-source LLM backbones, present a detailed case study of qualitative differences with CD knowledge states, and demonstrate that LBMs can be effectively trained and steered via their textual summaries.

108

2 KNOWLEDGE STATE MODELING AS AN INVERSE PROBLEM

109

2.1 PRELIMINARIES AND NOTATION

110 We consider data consisting of student interactions with educational questions. At each time step t ,
 111 a student is presented with a question $q_t \in \mathcal{Q}$ and optionally knowledge concept (KC) information
 112 $k_t \in \mathcal{K}$ and provides a response $r_t \in \mathcal{R}$, which is evaluated for correctness $c_t \in \{0, 1\}$. We
 113 represent an interaction as $x_t = (q_t, k_t, r_t, c_t)$, and a student’s interaction history up to time t as
 114 $H_t = (x_1, \dots, x_t)$.

115 The standard predictive task, often referred to as *Knowledge Tracing (KT)* (Corbett & Anderson, 1994;
 116 Shen et al., 2024), is to estimate $p(c_{t+1} \mid q_{t+1}, H_t)$, the probability that the student will answer a
 117 new question q_{t+1} correctly, conditioned on their interaction history. This task definition captures the
 118 forward-prediction aspect of knowledge state modeling, but does not yet address how the underlying
 119 knowledge that drives these responses should be represented.

120

2.2 KNOWLEDGE STATE MODELING AS AN INVERSE PROBLEM

121 An alternative view is to frame knowledge state modeling as an *inverse problem*: observed responses
 122 are generated by a latent knowledge state \mathcal{S} through the student’s cognitive process, and the goal is to
 123 recover an approximation $\tilde{\mathcal{S}}$ of this state from the observed responses.

124 This perspective is central to *Cognitive Diagnosis (CD)* models (Reckase, 2006; De La Torre,
 125 2009; Templin et al., 2010; Wang et al., 2022; 2024), which produce diagnostic reports of concept
 126 mastery from observed responses. However, CD typically restricts $\tilde{\mathcal{S}}$ to quantitative mastery or
 127 proficiency vectors based on predefined or inferred concepts, limiting expressivity. Meanwhile,
 128 deep learning-based KT methods prioritize predictive accuracy without explicitly recovering $\tilde{\mathcal{S}}$,
 129 representing knowledge states as high-dimensional embeddings that lack transparency (Piech et al.,
 130 2015; Zhang et al., 2017; Pandey & Karypis, 2019; Ghosh et al., 2020). **Bayesian KT (Corbett &**
 131 **Anderson, 1994)** as well as recent KT extensions introduce diagnostic reports similar in spirit to CD,
 132 or use interpretable latent states (Yeung, 2019; Minn et al., 2022; Chen et al., 2023; Park et al., 2024;
 133 Zhou et al., 2024). However these approaches remain bound to estimates of KC mastery or rely on
 134 post-hoc interpretability.

135 **Key assumption: constant knowledge state** A practical assumption underlying this formulation is
 136 that a knowledge state can be treated as constant within short diagnostic windows (e.g., unit tests,
 137 placement exams, or tutoring sessions). This aligns with CD models (see §2.1 in Wang et al. (2024)),
 138 and distinguishes them from KT approaches which model the evolution of \mathcal{S}_t across longer periods
 139 where the underlying knowledge state is expected to change.

140

2.3 NATURAL LANGUAGE AS THE INTERFACE

141 Our formulation follows CD approaches in adopting the inverse problem framing of inferring a
 142 diagnostic report from observed responses, but instead of restricting $\tilde{\mathcal{S}}$ to quantitative mastery scores,
 143 we model it through concise textual summaries. Natural language provides an interpretable and
 144 expressive medium—capable of describing arbitrary reasoning patterns or misconceptions, a key
 145 focus in education research (Smith III et al., 1994; Wang et al., 2020; King et al., 2024)—while
 146 remaining human-understandable. In the next section, we introduce *Language Bottleneck Models*
 147 (*LBM*s), which operationalizes this idea by compressing student interaction histories into concise
 148 textual representations that preserve predictive information.

149

3 LANGUAGE BOTTLENECK MODELS

150

3.1 FORMAL DEFINITION

151 We propose *Language Bottleneck Models* (*LBM*s) for Knowledge State Modeling via textual sum-
 152maries: an LLM-based, two-stage architecture designed to infer a predictive text-based knowledge
 153 state from a student’s interaction history.

Let $X^{\text{enc}} \subseteq H_t$ denote a subset of observed interactions used by the *encoder*. An encoder LLM f_θ maps this history to a natural-language summary: $\tilde{\mathcal{S}} = f_\theta(X^{\text{enc}})$. This summary serves as the sole representation of the student’s knowledge state. A *decoder* LLM g_ϕ then conditions only on $\tilde{\mathcal{S}}$ to predict the probability that the student will answer a question $q \in \mathcal{Q}$ correctly: $g_\phi(q, \tilde{\mathcal{S}}) = p(c | q, \tilde{\mathcal{S}})$, that is, given a question q and a summary $\tilde{\mathcal{S}}$, the decoder predicts the probability that the student will answer correctly.

In principle, both encoder f_θ and decoder g_ϕ could be trained. However, we show with the following motivating experiments that decoding is not the hard part: when given high-quality knowledge state summaries, off-the-shelf LLMs can achieve near-perfect prediction accuracy. **For this reason, we limit the scope of this work to learning an encoder that produces faithful, predictive, and interpretable summaries, keeping g_ϕ fixed.**

3.2 MOTIVATING OBSERVATIONS

We motivate the design of LBMs by two observations.

Observation 1: Given a good knowledge state summary, strong LLMs can decode with high fidelity. To test this in an idealized setting, we used a Synthetic dataset where each student’s knowledge state is programmatically generated. Figure 2 evaluates the performance of different decoder models when given direct access to this perfect, “ground-truth” summary of the student’s latent knowledge (example knowledge state and summaries are shown in Figure A1 in the Appendix, and full dataset details are in §5). Stronger models like GPT-4o achieve nearly perfect accuracy (98%), indicating that the bottleneck representation is indeed sufficient to drive effective downstream prediction for closed-form questions —provided it captures the right information.

Observation 2: Summarizing knowledge states from raw interactions is non-trivial. Standard LLM summarization approaches can capture high-level skill mastery but often fail to identify crucial latent patterns like student misconceptions (see an example comparison on our synthetic dataset Figure A1 in the Appendix).

Together, these observations suggest that the key challenge lies in learning an encoder that produces faithful summaries, rather than in the decoding step itself.

3.3 TRAINING OBJECTIVE AND OPTIMIZATION

We propose a reinforcement learning-based approach to train an encoder to produce more faithful and predictive summaries by using downstream decoder accuracy as reward.

Summary generation and prediction Given an interaction history $H_t = (x_1, \dots, x_t)$, the encoder f_θ maps a subset of interactions $\mathcal{X}_{\text{enc}} \subseteq H_t$ to a textual summary $\tilde{\mathcal{S}} = f_\theta(\mathcal{X}_{\text{enc}})$. The frozen decoder g then conditions on $\tilde{\mathcal{S}}$ to predict responses for two sets \mathcal{X} and \mathcal{Y} : interactions used for reconstruction and prediction, respectively. In practice, \mathcal{X} and \mathcal{Y} may be chosen flexibly to include held-out past responses, future responses, or both.

Reward function Given predicted interactions $\tilde{\mathcal{X}} = \{g(q, \tilde{\mathcal{S}}), q \in \mathcal{X}\}$ and $\tilde{\mathcal{Y}} = \{g(q, \tilde{\mathcal{S}}), q \in \mathcal{Y}\}$, the reward for a summary $\tilde{\mathcal{S}}$ is defined as

$$R(\tilde{\mathcal{S}}; g) = \phi\left(\text{Acc}(\tilde{\mathcal{X}}, \mathcal{X}), \text{Acc}(\tilde{\mathcal{Y}}, \mathcal{Y}), |\tilde{\mathcal{S}}|, \Omega(\tilde{\mathcal{S}})\right), \quad (1)$$

where $\text{Acc}(\cdot, \cdot)$ measures accuracy, $|\tilde{\mathcal{S}}|$ penalizes overly long summaries, and $\Omega(\tilde{\mathcal{S}})$ enforces optional structural constraints (e.g., inclusion of a Misconceptions section). The function Φ balances these components using hyperparameters or indicator functions to enforce constraints as needed.

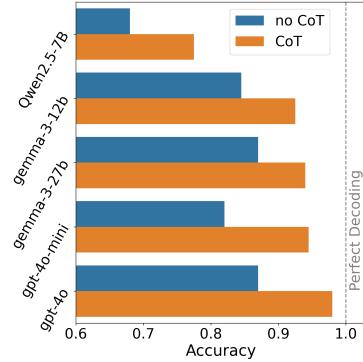


Figure 2: Accuracy on synthetic dataset given ground truth knowledge state summaries.

216 **Optimization via GRPO** We optimize the encoder f_θ with Group Relative Policy Optimization
 217 (GRPO) (Shao et al., 2024). For each input \mathcal{X}_{enc} , the encoder generates G candidate summaries
 218 $\{\tilde{\mathcal{S}}^1, \dots, \tilde{\mathcal{S}}^G\}$, each evaluated by $R(\tilde{\mathcal{S}}^i; g)$. We then compute group-relative advantage and update
 219 parameters:

$$221 \quad A(\tilde{\mathcal{S}}^i) = \frac{R(\tilde{\mathcal{S}}^i; g) - \mu}{\sigma}, \quad \nabla_\theta J(\theta) = \frac{1}{G} \sum_{i=1}^G A(\tilde{\mathcal{S}}^i) \nabla_\theta \log p_\theta(\tilde{\mathcal{S}}^i \mid \mathcal{X}_{\text{enc}}), \quad (2)$$

223 where μ and σ are the mean and standard deviation of rewards within the group.

225 3.4 STEERABILITY OF THE ESTIMATED KNOWLEDGE STATE

227 The natural-language summaries generated by LBMs allow for various human-model interactions
 228 (detailed in Appendix D). **(1) Prompt engineering the encoder.** Since the encoder f_θ is itself an
 229 LLM, its behavior can be shaped through prompt design, such as system instructions or in-context
 230 examples (Brown et al., 2020). **(2) Steering via reward signals.** Rewards to steer the encoder
 231 towards human preferences can be incorporated through the $\Omega(S)$ term in Eq. 1. **(3) Augmenting**
 232 **with student-specific information.** Educators can supplement the model with additional knowledge
 233 not present in observed data—either by augmenting encoder inputs or by editing the generated
 234 summary before decoding. This enables integration of recent classroom observations or specific
 235 misconceptions identified through in-person interactions.

237 4 RELATED WORK

239 We review related works from the Cognitive Diagnosis and Knowledge Tracing literature, as well as
 240 concept bottleneck models. See Appendix F for an extended review of related works, and Table F1
 241 for a high-level comparison of LBMs with CD and KT.

243 **Cognitive Diagnosis** Cognitive Diagnosis Models (CDMs) infer student knowledge states from
 244 observed responses. Classical approaches include Item Response Theory (IRT) and Multidimensional
 245 IRT which measure continuous proficiency scores (Rasch, 1993; Reckase, 2006), and the DINA
 246 model and its variants which estimate binary mastery of knowledge concepts (De La Torre, 2009).
 247 Recent deep learning variants like NeuralCDM (Wang et al., 2022) and RCD (Gao et al., 2021) use
 248 neural networks and graph architectures to model complex relationships between students, questions,
 249 and knowledge concepts. However, these models typically operate within predefined or inferred
 250 knowledge frameworks and provide only quantitative skill mastery estimates.

251 **Knowledge Tracing** Knowledge Tracing methods model student learning to predict future per-
 252 formance. Deep learning approaches like DKT (Piech et al., 2015), DKVMN (Zhang et al., 2017)
 253 and AKT (Ghosh et al., 2020) employ neural architectures. Despite strong predictive performance,
 254 these models represent knowledge as abstract latent vectors, limiting interpretability. **Several recent**
 255 **works have proposed more interpretable KT.** Early Bayesian approaches (Corbett & Anderson, 1994;
 256 Käser et al., 2017) and recent probabilistic models (Minn et al., 2022; Zhou et al., 2024) learn
 257 interpretable latent states representing student proficiency over knowledge concepts, while IRT-based
 258 methods (Yeung, 2019; Chen et al., 2023) combine deep learning with item response theory for
 259 meaningful latent representations. Other works improve interpretability through learned question
 260 relationships (Tong et al., 2022), explainable subsequences (Li et al., 2023), or option tracing (Ghosh
 261 et al., 2021). However, these approaches remain fundamentally constrained to quantitative concept
 262 proficiency estimation or require post-hoc interpretability. Finally, recent LLM-based approaches
 263 have shown promise for knowledge tracing tasks (Li et al., 2024a; Wang et al., 2025), but they
 264 generally remain opaque, either treating LLMs as black boxes with no interpretable intermediate
 265 representation or relying on model-generated explanations susceptible to hallucination.

266 **Concept Bottleneck Models** Concept Bottleneck Models (CBMs) (Koh et al., 2020) improve
 267 interpretability by using human-understandable concept activations as intermediates between inputs
 268 and predictions. However, CBMs typically rely on finite predefined concept sets, limiting applicability
 269 to complex tasks like knowledge tracing. Recently, Explanation Bottleneck Models (XBMs) (Yamaguchi
 & Nishida, 2024) use textual rationales as intermediates for vision classification. While

270 our Language Bottleneck Models (LBMs) adopt this language bottleneck concept, they differ fundamentally: unlike XBMs’ instance-specific rationales, LBM summaries capture implicit knowledge
 271 states that generalize to future, unknown questions, requiring holistic, adaptable summaries rather
 272 than task-specific rationales.
 273

275 5 EXPERIMENTS

277 **Datasets** We evaluate LBMs and baseline
 278 models on a synthetic arithmetic benchmark
 279 and two real-world datasets (Table 1). Our Synthetic
 280 dataset (Appendix B.3.1) simulates learners
 281 answering addition, subtraction, multiplication,
 282 and division questions; each student is as-
 283 signed mastered skills, unmastered skills, and
 284 systematic misconceptions. We filter *Eedi* and
 285 *XES3G5M* for single-session trajectories (max
 286 inter-question gaps of 3 and 10 minutes respec-
 287 tively) with at least 40 and 34 interactions re-
 288 spectively to approximate the static knowledge
 289 state assumption common in CD (Wang et al.,
 290 2024). We evaluate *XES3G5M* in both Chinese
 291 and English using translations from Ozyurt et al. (2024) (see Appendix B.3).

292 **Models** We evaluate LBMs across LLM backbones of different sizes and capabilities, both open-
 293 source (Qwen 2.5 3B and 7B (Team, 2024), Gemma 3 12B and 30B (Team, 2025)), and closed-source
 294 (GPT-4o-mini, GPT-4o and GPT-5 (Achiam et al., 2023)). Unless noted otherwise, we run the *instruct*
 295 variants of each open-source model, use the same backbone LLM for both the encoder and decoder,
 296 and prompt all models to provide their response directly without chain-of-thought. We run GPT-5
 297 with `reasoning_effort=minimal` configuration. Hyper-parameters and prompt templates are
 298 provided in Appendix B.

299 **Baselines** We compare LBMs against 9 Knowledge Tracing methods: DKT (Piech et al., 2015),
 300 DKVMN (Zhang et al., 2017), SAKT (Pandey & Karypis, 2019), AKT (Ghosh et al., 2020), Deep
 301 IRT (Yeung, 2019), SAINT (Choi et al., 2020), SimpleKT (Liu et al., 2023a), QIKT (Chen et al.,
 302 2023), GKT (Nakagawa et al., 2019) implemented with the PYKT library (Liu et al., 2022) and 5
 303 Cognitive Diagnosis methods (IRT (Rasch, 1993), MIRT (Reckase, 2006), DINA (Junker & Sijtsma,
 304 2001), KaNCD and NeuralCDM (Wang et al., 2022)) implemented with the EduCDM library (bigdata
 305 ustc, 2021). We also run each LLM via *direct prompting*, where the LLM predicts answers from the
 306 full interaction history without a bottleneck. Training details for all baselines are given in Appendix B.

307 **Systematic evaluation of knowledge state summaries** We systematically evaluate the quality of
 308 the knowledge state summaries produced by different encoder models on the Synthetic dataset using
 309 LLM-as-a-judge (Li et al., 2025). Specifically we prompt a GPT-5 model to compare each summary
 310 to the corresponding ground-truth knowledge state across the following dimensions: **Global score**
 311 (*overall alignment with the true state*); **Construct accuracy** (*correctly assessed constructs*); **Mis-
 312 conception detection** (*correctly identified student misconceptions*); **Misconception false positives**
 313 (*misconceptions mentioned but absent in the ground truth*); **Specificity** (*precision vs. vagueness of
 314 the description*); and **Confidence calibration** (*degree of over- or under-confidence*). See Appendix C
 315 for full details and prompts.

316 5.1 QUALITATIVE INSIGHTS

317 5.1.1 CASE-STUDY: COMPARING CD AND LBM KNOWLEDGE STATE REPRESENTATIONS

318 A key advantage of LBMs over CD models is the ability to capture nuanced insights about the student
 319 knowledge state, such as misconceptions. We illustrate this with a case-study Figure 3. We train
 320 a state-of-the-art CD model (NeuralCDM) trained on our Synthetic dataset, which achieves strong
 321 predictive performance (AUC: 0.96, Acc: 0.90 on the test set). From this model, we extract proficiency
 322 estimates across knowledge concepts using the learned student embedding vector, producing a typical
 323

CD diagnostic report. We then compare this to the knowledge state summary generated for the same student by an LBM (trained Gemma-12B encoder with frozen Gemma-27B decoder, as in §5.2).

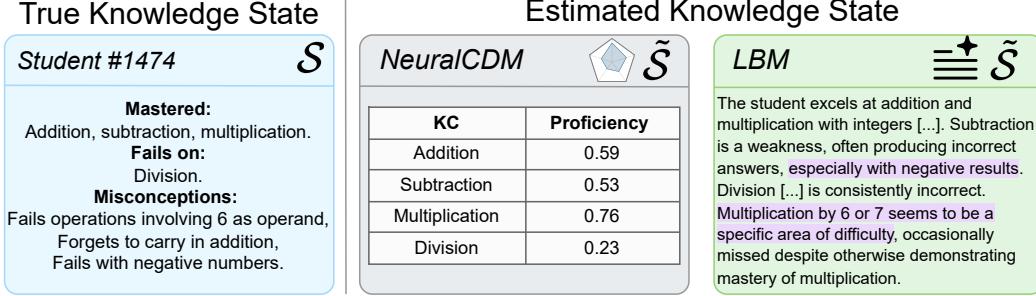


Figure 3: **Case study: comparing CD and LBM knowledge states.** Given a student from the Synthetic dataset, we compare proficiency estimates across knowledge concepts (KCs) obtained from a trained NeuralCDM model to the text-based knowledge state generated by a trained LBM model.

While NeuralCDM reliably captures general KC proficiency, its estimates are influenced by misconceptions without explicitly identifying them. In contrast, LBMs capture overall proficiency and uncover specific misconceptions (e.g., errors with negative numbers or with operand-6). This ability to provide nuanced, qualitative insights into student knowledge states sets LBMs apart from CD methods.

5.1.2 SYSTEMATIC EVALUATION OF KNOWLEDGE SUMMARIES

To further assess the interpretability of the knowledge states produced by LBMs, we systematically compare their summaries to ground truth knowledge states from the Synthetic dataset using GPT-5 as LLM-as-a-judge (Li et al., 2025). The results are shown Figure 4. Figure 4(a) shows the overall alignment with the ground truth summary (score from 1='mostly incorrect' to 5='strongly aligned'), and the construct mastery accuracy. Model capability correlates with performance on both metrics, with GPT-5 achieving the highest scores. Figure 4(b) plots the misconception-detection rate against the misconception false positive rate. Qwen2.5 and GPT-4o/4o-mini exhibit lower hallucination but also lower detection; Gemma-3 detects more misconceptions but at the cost of more false positives; GPT-5 outperforms all models on both dimensions. Figure 4(c) evaluates summary specificity and confidence calibration. Gemma-3 models produce more specific summaries than Qwen2.5, but both families tend to be over-confident relative to GPT models, with GPT-5 displaying the best overall specificity and calibration.

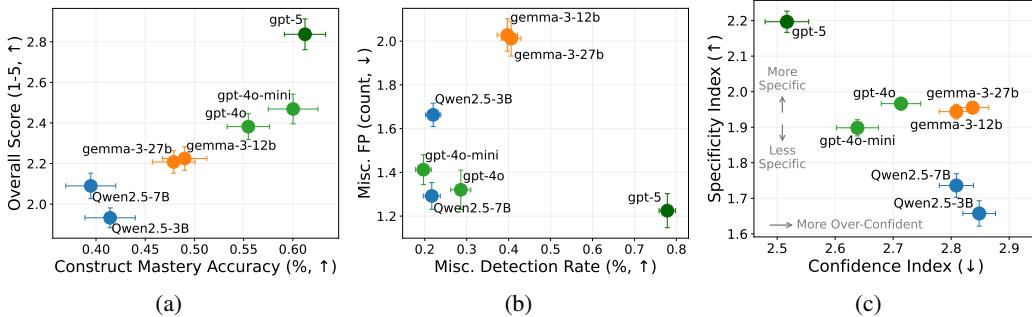


Figure 4: **Systematic evaluation of the summaries produced by different encoder LLMs over 200 test students from the Synthetic dataset:** (a) Average construct mastery accuracy across constructs (%) and average overall score (1-5 scale); (b) Average misconception detection rate (%) and average number of misconception false positives per summary (count); (c) Confidence Index (1-3, 1=under-confident, 2=appropriately calibrated, 3=over-confident) and Specificity Index (1-3, higher is more specific). Error bars represent the standard error (N=200).

378 5.2 TRAINING IBM ENCODERS

As outlined in §3.3, downstream accuracy can serve as a reward signal to train encoders to produce increasingly accurate summaries. We demonstrate this by training a Gemma3-12B encoder with GRPO alongside a frozen Gemma3-27B decoder on 800 students. We set the reward as the decoder accuracy across $|\mathcal{Y}| = 20$ unseen questions $R(\tilde{\mathcal{S}}; g) = \text{Acc}(\tilde{\mathcal{Y}}, \mathcal{Y})$, train with a LoRA adapter (Hu et al., 2022) and evaluate on 200 unseen test students.

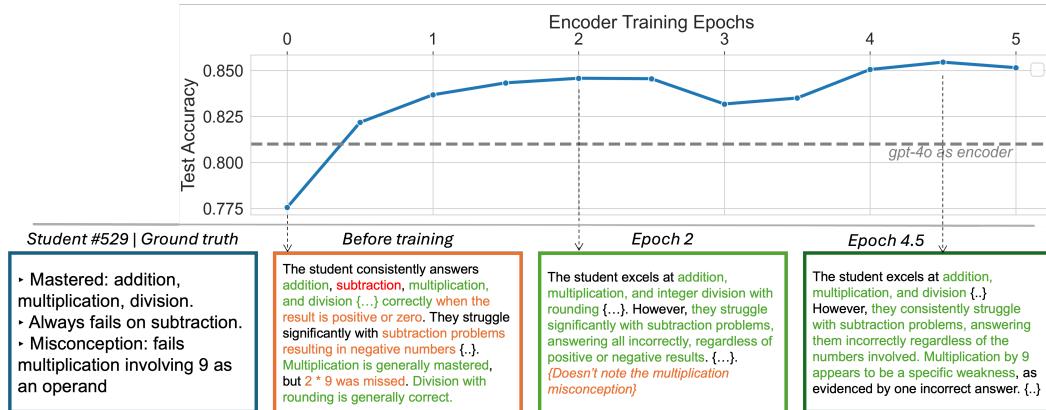


Figure 5: Training the encoder with the decoding accuracy on the synthetic dataset. Evolution of the test accuracy as a Gemma3-12B encoder is trained as described in § 3.3, with a fixed Gemma3-27B decoder. Trained on 800 training students and tested on 200 students, with $|\mathcal{X}| = 50$ questions per input trajectory and $|\mathcal{Y}| = 20$ questions to predict per student. The bottom row shows the evolution of the generated summary over the course of training for an example student. Text is colored green (exact), orange (approximate), or red (false) based on ground-truth.

Effect on accuracy Figure 5 shows the encoder progressively improving summary quality and quickly outperforming GPT-4o. The figure illustrates this through an example student who mastered all constructs except subtraction and fails any multiplication involving 9. The initial summary contains inaccuracies and misses this systematic misconception, while the final summary successfully captures the student’s complete knowledge state. Stratifying by knowledge state complexity, we observe larger gains for more complex cases (Appendix A.3.1).

Qualitative evolution of summaries We use the same LLM-as-a-judge approach as in the previous section to systematically evaluate the generated summaries over training. Figure 6 shows that the summaries gradually better capture concept mastery and align more with the ground truth knowledge state over training. Moreover, as summaries also become more specific and better calibrated. Misconception detection and false positive rates do not significantly change over training (Appendix A.3.2).

5.3 LBM VS KNOWLEDGE TRACING METHODS

We present comprehensive results comparing LBMs to baseline methods across all datasets (detailed results in Table A1 in the Appendix). Here we highlight key findings and insights from these experiments.

Performance Figure 7 compares the performance of LBMs against the top three KT and CD models on the Synthetic, EEDJ, and XES3G5M datasets. The LBMs are evaluated in a zero-shot

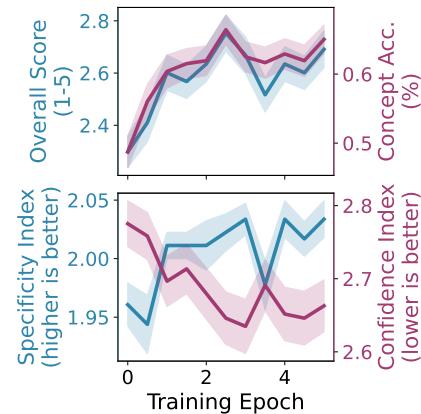


Figure 6: **Systematic evaluation of generated summaries over encoder training.** Evaluation axes are the same as Figure 4(a) and (c). Confidence intervals represent the standard error (N=200).

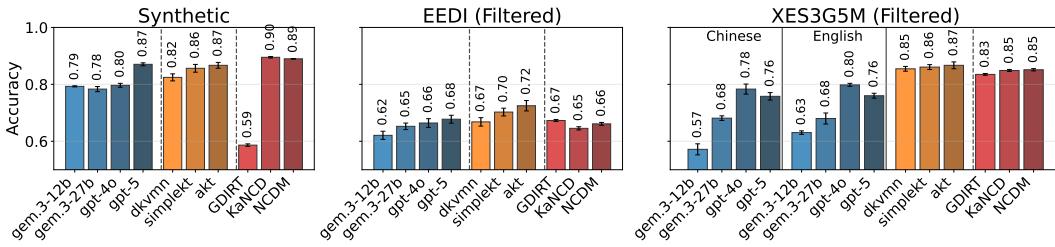


Figure 7: **Accuracy of LBM vs the top-3 KT and CD models across datasets.** Models are grouped by LBMs (blue), KT models (orange) and CD models (red). Top 3 KT and CD models are selected based on average accuracy across all three datasets. Full results for other models are available Table A1 in the Appendix.

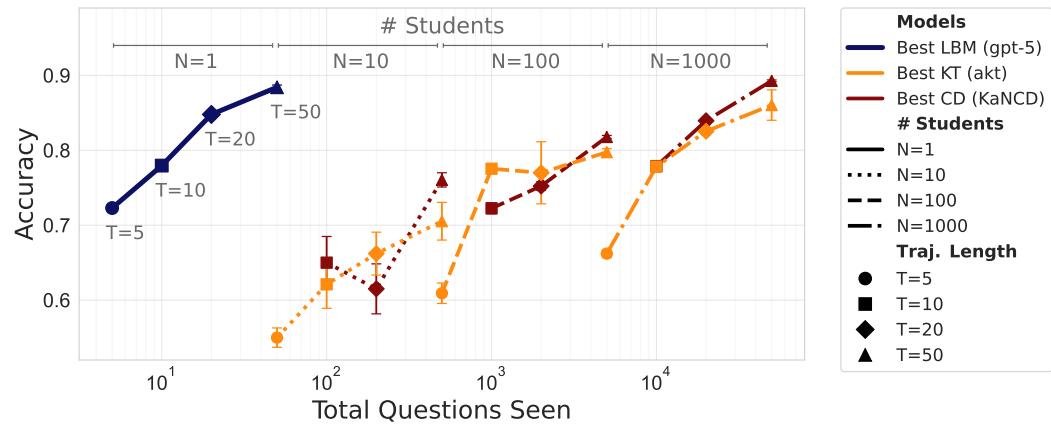


Figure 8: **Accuracy of the best LBM, KT and CD models on the Synthetic dataset.** The x axis shows the total number of question seen by the model, i.e. "Traj. Length" x "# Students". Note that #Students = 1 for LBMs as they are evaluated zero-shot. CD models are evaluated on held-out test interactions from the same students used during training, while KT models and LBMs are evaluated on 200 unseen students. Results are averaged across N=10 runs for KT and CD models and N=3 for LBMs, with error bars showing the standard error. GPT-5 is run with `reasoning_effort=minimal`.

setting, whereas the KT/CD models are trained on data from hundreds of students. As expected, LBM performance is strongly tied to the strength of the underlying LLM: with powerful backbones such as GPT-4o and GPT-5, LBMs approach the accuracy of the best KT and CD models across all three datasets. The largest performance gap arises on XES3G5M. However, this dataset has an average accuracy of 85%, implying that even a constant predictor would achieve 85% accuracy. Unlike KT and CD models, LBMs operate zero-shot and thus cannot exploit such dataset-level statistics, which likely explains their lower accuracy but competitive AUC (see Table A1). Finally, thanks to the multilingual capabilities of modern LLMs, LBMs achieve comparable results on both the English and Chinese versions of XES3G5M.

Sample efficiency Figure 8 compares the performance of LBM models to traditional Knowledge Tracing methods on the Synthetic dataset. LBMs with a GPT-5 backbone achieves comparable accuracy to KT methods with significantly less training data. Since CD/KT methods rely on statistical patterns, they require substantially more observations before reaching strong predictive power, while LBMs demonstrate strong zero-shot performance.

5.4 STEERING LBM BEHAVIOR

We demonstrate multiple steering strategies described in §3.4. Providing explicit misconception information during encoder training produces substantially stronger learning effects than adding the same information at the decoder stage (Appendix A.4.1), suggesting that the encoder uses this additional

486 context to better interpret patterns in student responses. We also show the encoder can be steered
 487 during training to explicitly mention misconceptions through reward signals (Appendix A.4.2), and
 488 illustrate how supplementing the summary with information not present in the input can significantly
 489 improve decoder accuracy (Appendix A.4.3).
 490

491 5.5 ABLATION EXPERIMENTS

493 **How much information is lost by the bottleneck?** Table 2 compares the accuracy of LBM
 494 models to directly predicting new questions from the observed student data. Despite the information
 495 bottleneck, LBM accuracy typically remains within 2% of direct prediction—and often surpasses it.
 496 Figure 9 shows that this gap decreases for longer bottleneck token limits, highlighting a trade-off
 497 between conciseness and predictive accuracy.

498 Table 2: Accuracy results for Direct and LBM methods. **Bold** indicates models for which the LBM accuracy is no more than 2%
 499 below the Direct baseline (Welch’s t-test, details in Appendix B.4.1).
 500

	synthetic			eedi		
	Direct	LBM	Δ	Direct	LBM	Δ
Qwen2.5-3B	.56 ± .00	.61 ± .01	+.06	.35 ± .00	.38 ± .00	+.03
Qwen2.5-7B	.64 ± .00	.65 ± .01	+.01	.65 ± .00	.58 ± .01	-.07
gemma-3-12b	.63 ± .00	.79 ± .00	+.17	.58 ± .00	.62 ± .01	+.04
gemma-3-27b	.62 ± .00	.78 ± .01	+.16	.67 ± .00	.65 ± .01	-.02
gpt-4o-mini	.81 ± .01	.78 ± .01	-.03	.67 ± .01	.61 ± .02	-.05
gpt-4o	.85 ± .01	.80 ± .01	-.05	.66 ± .00	.66 ± .02	+.00
gpt-5	.87 ± .00	.87 ± .01	+.00	.71 ± .02	.68 ± .01	-.03

511 **Which of the encoder or decoder is most critical for LBMs?** We evaluate LBMs with different
 512 encoder–decoder pairings (Appendix A.5.1). Using a strong model (GPT-4o) as the encoder with
 513 weaker models as decoders yields accuracies 5 – 10% higher than when the stronger model is used
 514 as the decoder. This confirms our hypothesis that extracting relevant information for the summary
 515 (encoding) is more challenging than predicting future answers given a summary (decoding).

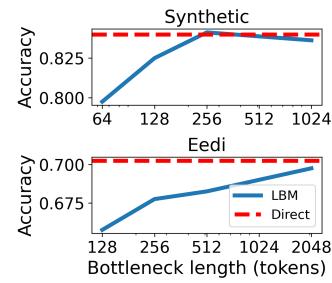
517 **Do LBMs require knowledge concept information?** Table A6 compares LBMs with and without
 518 KC information in the input prompt on the Synthetic and EEDI (Filtered) datasets. Performance does
 519 not significantly change, demonstrating that LBMs do not fundamentally require KC information.

520 6 DISCUSSION

522 **Why can’t we just prompt GPT-4o directly?** The split encoder-decoder architecture of LBMs
 523 offers three key advantages over direct LLM prompting: it creates a global student model with a
 524 single latent summary shared across all predictions rather than isolated per-question reasoning; it
 525 ensures faithful summaries through a closed-loop decoding objective that penalizes non-predictive
 526 summaries; and it provides an explicit interface layer that teachers can read, steer and intervene on.

528 **Wider applicability** LBMs extend beyond education to any task requiring compact, human-
 529 readable summaries with predictive power. The minimal ingredients needed are: (1) a sequence
 530 of observations about an entity, (2) a need to predict future behaviors of that entity, and (3) value
 531 in having interpretable representations. For example, clinical decision support could distill patient
 532 data into textual state descriptions that forecast outcomes while remaining auditible; preventive
 533 maintenance could compress sensor logs into explanations predicting machine failure; customer
 534 success teams could summarize interaction histories to forecast churn.

535 **Limitations and Future Work** LBMs face several constraints including context length limitations,
 536 requirements for textual question content, and substantial computational costs. Future extensions
 537 could address these through iterative encoding for longer inputs, active sensing for optimal question
 538 selection, adaptation for evolving knowledge states, expansion beyond question-answer data, and
 539 integration with pedagogical techniques like LearnLM (Team et al., 2024). These limitations and
 extensions are discussed in detail in Appendix G.



511 Figure 9: Evolution of LBM
 512 accuracy with bottleneck
 513 length (GPT-4o backbone).

540 REPRODUCIBILITY STATEMENT
541542 We are committed to ensuring the reproducibility of our research. All models, datasets, and experi-
543 mental settings are described in detail in the paper and the appendix.544
545

- Datasets: We use one synthetic and two real-world datasets. The generation process for
546 the synthetic data, along with preprocessing steps for the Eedi and XES3G5M datasets, are
547 detailed in Appendix B.3. The source code for the synthetic data will be made available in
548 an online repository upon publication of this work.
- Implementation Details: Our LBM framework is presented Section 3. The specific LLM
549 backbones, baseline models, training hyperparameters, and software libraries used in the
550 experiments are described in Appendix B. All prompt templates used for the LBM encoder
551 and decoder are provided in Appendix B.5.
- Code: The source code for generating the synthetic data, training the models, and running
552 all experiments will be made publicly available in an online repository upon publication of
553 this work.

554555 REFERENCES
556557 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman,
558 Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report.
559 *arXiv preprint arXiv:2303.08774*, 2023.560 Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the
561 dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM*
562 *conference on fairness, accountability, and transparency*, pp. 610–623, 2021.563 bigdata ustc. Educdm. <https://github.com/bigdata-ustc/EduCDM>, 2021.564 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
565 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are
566 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.567 Chen Chen, Gerhard Sonnert, Philip M Sadler, Dimitar Sasselov, and Colin Fredericks. The impact of
568 student misconceptions on student persistence in a mooc. *Journal of Research in Science Teaching*,
569 57(6):879–910, 2020.570 Jiahao Chen, Zitao Liu, Shuyan Huang, Qiongqiong Liu, and Weiqi Luo. Improving interpretability
571 of deep sequential knowledge tracing models with question-centric cognitive representations. In
572 *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 14196–14204, 2023.573 Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha,
574 Dongmin Shin, Chan Bae, and Jaewe Heo. Towards an appropriate query, key, and value computa-
575 tion for knowledge tracing. In *Proceedings of the seventh ACM conference on learning@ scale*, pp.
576 341–344, 2020.577 Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural
578 knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, December 1994. ISSN
579 1573-1391. doi: 10.1007/BF01099821. URL <https://doi.org/10.1007/BF01099821>.580 Jimmy De La Torre. Dina model and parameter estimation: A didactic. *Journal of educational and*
581 *behavioral statistics*, 34(1):115–130, 2009.582 Zhiang Dong, Jingyuan Chen, and Fei Wu. Knowledge is power: Harnessing large language models
583 for enhanced cognitive diagnosis. In *AAAI*, 2025.584 Weibo Gao, Qi Liu, Zhenya Huang, Yu Yin, Haoyang Bi, Mu-Chun Wang, Jianhui Ma, Shijin Wang,
585 and Yu Su. Rcd: Relation map driven cognitive diagnosis for intelligent education systems. In
586 *Proceedings of the 44th international ACM SIGIR conference on research and development in*
587 *information retrieval*, pp. 501–510, 2021.

594 Aritra Ghosh, Neil Heffernan, and Andrew S Lan. Context-aware attentive knowledge tracing. In
 595 *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data*
 596 *mining*, pp. 2330–2339, 2020.

597 Aritra Ghosh, Jay Raspat, and Andrew Lan. Option tracing: Beyond correctness analysis in knowledge
 598 tracing. In *International Conference on Artificial Intelligence in Education*, pp. 137–149. Springer,
 600 2021.

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

603 Brian W Junker and Klaas Sijtsma. Cognitive assessment models with few assumptions, and
 604 connections with nonparametric item response theory. *Applied Psychological Measurement*, 25(3):
 606 258–272, 2001.

607 Tanja Käser, Severin Klingler, Alexander G Schwing, and Markus Gross. Dynamic bayesian networks
 608 for student modeling. *IEEE Transactions on Learning Technologies*, 10(4):450–462, 2017.

609 JongWoo Kim, SeongYeub Chu, Bryan Wong, and Mun Yi. Beyond right and wrong: Mitigating
 610 cold start in knowledge tracing using large language model and option weight. *arXiv preprint*
 611 *arXiv:2410.12872*, 2024.

612 Jules King, L Burleigh, Simon Woodhead, Panagiota Kon, Perpetual Baffour, Scott Crossley, Walter
 613 Reade, and Maggie Demkin. Eedi - mining misconceptions in mathematics. <https://kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics>, 2024.
 614 Kaggle.

615 Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and
 616 Percy Liang. Concept bottleneck models. In *ICML*, 2020.

617 Douglas Larkin. Misconceptions about “misconceptions”: Preservice secondary science teachers’
 618 views on the value and role of student ideas. *Science Education*, 96(5):927–959, 2012.

619 Unggi Lee, Jiyeong Bae, Dohee Kim, Sookbun Lee, Jaekwon Park, Taekyung Ahn, Gunho Lee, Damji
 620 Stratton, and Hyeoncheol Kim. Language model can do knowledge tracing: Simple but effective
 621 method to integrate language model and knowledge tracing task. *arXiv preprint arXiv:2406.02893*,
 622 2024.

623 Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer
 624 Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for
 625 natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*,
 626 2019.

627 Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita
 628 Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. From generation to judgment:
 629 Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on*
 630 *Empirical Methods in Natural Language Processing*, pp. 2757–2791, 2025.

631 Haoxuan Li, Jifan Yu, Yuanxin Ouyang, Zhuang Liu, Wenge Rong, Juanzi Li, and Zhang Xiong.
 632 Explainable few-shot knowledge tracing. *arXiv preprint arXiv:2405.14391*, 2024a.

633 Jiatong Li, Qi Liu, Fei Wang, Jiayu Liu, Zhenya Huang, Fangzhou Yao, Linbo Zhu, and Yu Su.
 634 Towards the identifiability and explainability for personalized learner modeling: an inductive
 635 paradigm. In *Proceedings of the ACM Web Conference 2024*, pp. 3420–3431, 2024b.

636 Qing Li, Xin Yuan, Sannyuya Liu, Lu Gao, Tianyu Wei, Xiaoxuan Shen, and Jianwen Sun. A genetic
 637 causal explainer for deep knowledge tracing. *IEEE Transactions on Evolutionary Computation*, 28
 638 (4):861–875, 2023.

639 Zitao Liu, Qiongqiong Liu, Jiahao Chen, Shuyan Huang, Jiliang Tang, and Weiqi Luo. pykt: A python
 640 library to benchmark deep learning based knowledge tracing models. In *Thirty-sixth Conference*
 641 *on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

648 Zitao Liu, Qiongqiong Liu, Jiahao Chen, Shuyan Huang, and Weiqi Luo. simplekt: a simple but
 649 tough-to-beat baseline for knowledge tracing. *arXiv preprint arXiv:2302.06881*, 2023a.
 650

651 Zitao Liu, Qiongqiong Liu, Teng Guo, Jiahao Chen, Shuyan Huang, Xiangyu Zhao, Jiliang Tang,
 652 Weiqi Luo, and Jian Weng. Xes3g5m: A knowledge tracing benchmark dataset with auxiliary
 653 information. *NeurIPS*, 2023b.

654 Max Ruiz Luyten and Mihaela van der Schaar. A theoretical design of concept sets: improving the
 655 predictability of concept bottleneck models. In *NeurIPS*, 2024.
 656

657 Haiping Ma, Manwei Li, Le Wu, Haifeng Zhang, Yunbo Cao, Xingyi Zhang, and Xuemin Zhao.
 658 Knowledge-sensed cognitive diagnosis for intelligent education platforms. In *Proceedings of the*
 659 *31st ACM international conference on information & knowledge management*, pp. 1451–1460,
 660 2022.

661 Sein Minn, Jill-Jênn Vie, Koh Takeuchi, Hisashi Kashima, and Feida Zhu. Interpretable knowledge
 662 tracing: Simple and efficient student modeling with causal relations. In *Proceedings of the AAAI*
 663 *conference on artificial intelligence*, volume 36, pp. 12810–12818, 2022.

664

665 Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. Graph-based knowledge tracing: modeling
 666 student proficiency using graph neural network. In *IEEE/WIC/aCM international conference on*
 667 *web intelligence*, pp. 156–163, 2019.

668 Tuomas Oikarinen, Subhro Das, Lam Nguyen, and Lily Weng. Label-free concept bottleneck models.
 669 In *ICLR*, 2023.
 670

671 Yilmazcan Ozyurt, Stefan Feuerriegel, and Mrinmaya Sachan. Automated knowledge concept annotation
 672 and question representation learning for knowledge tracing. *arXiv preprint arXiv:2410.01727*,
 673 2024.

674 Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. In *12th International*
 675 *Conference on Educational Data Mining, EDM 2019*, pp. 384–389. International
 676 Educational Data Mining Society, 2019.

677

678 Soonwook Park, Donghoon Lee, and Hogun Park. Enhancing knowledge tracing with concept map
 679 and response disentanglement. *Knowledge-Based Systems*, 302:112346, 2024.

680

681 Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J
 682 Guibas, and Jascha Sohl-Dickstein. Deep Knowledge Tracing. In *NeurIPS*, volume 28, 2015. URL
 683 https://papers.nips.cc/paper_files/paper/2015/hash/bac9162b47c56fc8a4d2a519803d51b3-Abstract.html.
 684

685 George J Posner, Kenneth A Strike, Peter W Hewson, and William A Gertzog. Accommodation of a
 686 scientific conception: Toward a theory of conceptual change. *Science education*, 66(2):211–227,
 687 1982.

688

689 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi
 690 Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text
 691 transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

692

693 Georg Rasch. *Probabilistic models for some intelligence and attainment tests*. ERIC, 1993.

694

695 Mark D Reckase. 18 multidimensional item response theory. *Handbook of statistics*, 26:607–642,
 2006.

696

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

700

701 Shuanghong Shen, Qi Liu, Zhenya Huang, Yonghe Zheng, Minghao Yin, Minjuan Wang, and Enhong
 Chen. A survey of knowledge tracing: Models, variants, and applications. *IEEE Transactions on*
Learning Technologies, 2024.

702 Sungbin Shin, Yohan Jo, Sungsoo Ahn, and Namhoon Lee. A closer look at the intervention procedure
 703 of concept bottleneck models. In *ICML*, 2023.

704

705 John P Smith III, Andrea A DiSessa, and Jeremy Roschelle. Misconceptions reconceived: A
 706 constructivist analysis of knowledge in transition. *The journal of the learning sciences*, 3(2):
 707 115–163, 1994.

708 Gemma Team. Gemma 3. 2025. URL <https://goo.gle/Gemma3Report>.

709

710 LearnLM Team, Abhinit Modi, Aditya Srikanth Veerubhotla, Aliya Rysbek, Andrea Huber, Brett Wilt-
 711 shire, Brian Veprek, Daniel Gillick, Daniel Kasenberg, Derek Ahmed, et al. Learnlm: Improving
 712 gemini for learning. *arXiv preprint arXiv:2412.16429*, 2024.

713 Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL <https://qwenlm.github.io/blog/qwen2.5/>.

714

715 Jonathan Templin, Robert A Henson, et al. *Diagnostic measurement: Theory, methods, and applica-
 716 tions*. Guilford press, 2010.

717

718 Jonathan L Templin and Robert A Henson. Measurement of psychological disorders using cognitive
 719 diagnosis models. *Psychological methods*, 11(3):287, 2006.

720

721 Hanshuang Tong, Zhen Wang, Yun Zhou, Shiwei Tong, Wenyuan Han, and Qi Liu. Introducing
 722 problem schema with hierarchical exercise graph for knowledge tracing. In *Proceedings of the
 723 45th international ACM SIGIR conference on research and development in information retrieval*,
 724 pp. 405–415, 2022.

725 Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yu Yin, Shijin Wang, and Yu Su. Neuralcd: a general
 726 framework for cognitive diagnosis. *IEEE Transactions on Knowledge and Data Engineering*, 35
 727 (8):8312–8327, 2022.

728

729 Fei Wang, Weibo Gao, Qi Liu, Jiatong Li, Guanhao Zhao, Zheng Zhang, Zhenya Huang, Mengxiao
 730 Zhu, Shijin Wang, Wei Tong, et al. A survey of models for cognitive diagnosis: New developments
 731 and future directions. *arXiv preprint arXiv:2407.05458*, 2024.

732

733 Zichao Wang, Angus Lamb, Evgeny Saveliev, Pashmina Cameron, Yordan Zaykov, José Miguel
 734 Hernández-Lobato, Richard E Turner, Richard G Baraniuk, Craig Barton, Simon Peyton Jones,
 735 Simon Woodhead, and Cheng Zhang. Diagnostic questions: The neurips 2020 education challenge.
 736 *arXiv preprint arXiv:2007.12061*, 2020.

737

738 Ziwei Wang, Jie Zhou, Qin Chen, Min Zhang, Bo Jiang, Aimin Zhou, Qinchun Bai, and Liang He.
 739 Llm-kt: Aligning large language models with knowledge tracing using a plug-and-play instruction.
 740 *arXiv preprint arXiv:2502.02945*, 2025.

741

742 Shin’ya Yamaguchi and Kosuke Nishida. Toward explanation bottleneck models. In
 743 *MINT@NeurIPS2024: Foundation Model Interventions*, 2024.

744

745 Chun-Kit Yeung. Deep-irt: Make deep learning based knowledge tracing explainable using item
 746 response theory. *arXiv preprint arXiv:1904.11738*, 2019.

747

748 Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. Dynamic key-value memory networks
 749 for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*,
 750 pp. 765–774, 2017.

751 Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted
 752 gap-sentences for abstractive summarization. In *ICML*, 2020.

753

754 Hanqi Zhou, Robert Bamler, Charley M Wu, and Álvaro Tejero-Cantero. Predictive, scalable and
 755 interpretable knowledge tracing on structured domains. In *ICLR*, 2024.

APPENDIX

A EXTENDED RESULTS

A.1 MOTIVATING OBSERVATION 2

Figure A1 compares summaries produced by different LLMs when prompted to describe a student’s knowledge state from 50 question-answer pairs from a synthetic dataset (see Section 5 for details). While all models capture high-level skill mastery, only one correctly identifies a misconception (errors with negative numbers) out of the four existing ones, illustrating that standard summarization approaches often miss crucial latent patterns.

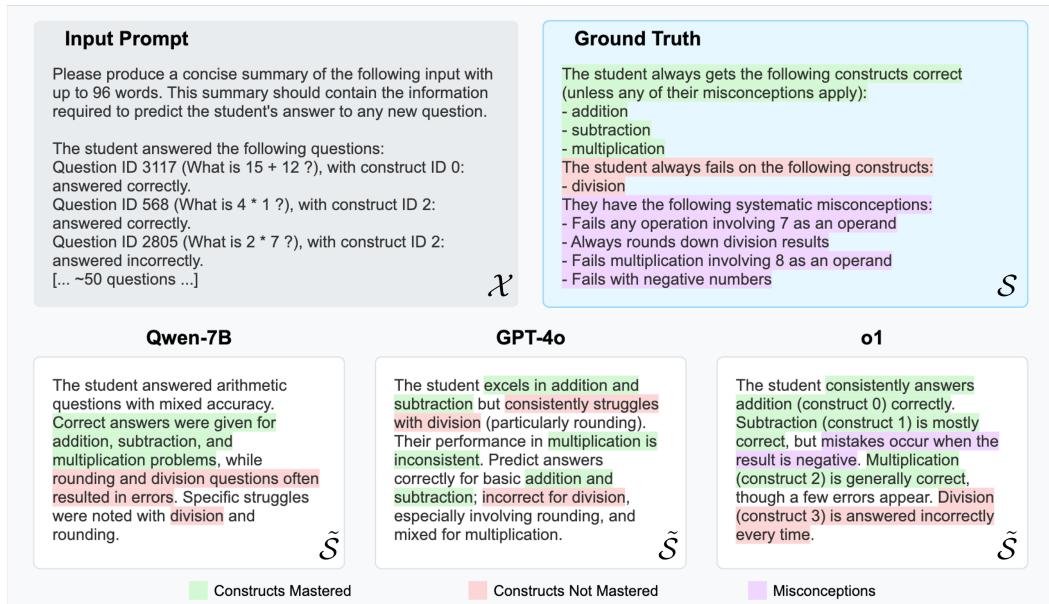


Figure A1: **Zero-shot knowledge state encoding compared to ground truth.** LLM models with different capabilities are prompted to write a summary of the knowledge state (top left panel) of a student, given 50 observed questions and answers provided in text. The ground truth knowledge state (top right panel) describing the student behavior has three main components: constructs mastered, constructs not mastered, and misconceptions. All three models capture the construct mastery correctly, but are not able to capture any misconception, beside o1 which notices the negative numbers misconception (bottom row). Bottom right notations correspond to components in Figure 1.

A.2 FULL RESULTS

Table A1 compares the zero-shot performance of LLMs to all the baselines across all datasets.

Note: The surprisingly low accuracy of some LLMs on EEDI and XES3G5M likely reflects the strong class imbalance in these datasets (68% correct in EEDI; 85% in XES3G5M; cf. Table 1) and the fact that both LLMs and Direct models are evaluated zero-shot, preventing them from adjusting to this imbalance. Consequently, models whose predictions skew toward the minority class (“incorrect”) can score below random accuracy. The high AUCs for some of these models indicate that they still capture some meaningful predictive signal.

Table A1: Results across all datasets and methods. We report the mean and standard deviation across N=10 runs for KT and CD models and N=3 runs for LLM-based models (Direct, LBM). XES3G5M-E and XES3G5M-C denote the English and Chinese versions of the XES3G5M dataset, respectively; note that since the KT and CD models do not use the question content, their results are shared across both versions of the dataset. The KT and CD models are trained on a train set and evaluated on a test set, while LLM Direct and LBM methods are run zero-shot on a test set. AUC unavailable for closed-source LLM models as they require access to the model's output logits. Results for QIKT on Synthetic and MIRT on EEDI are omitted due to implementation issues.

Model Type	Model Name	Synthetic		EEDI (Filt.)		XES3G5M-E (Filt.)		XES3G5M-C (Filt.)	
		ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
KT Models (Trained)	akt	.87±.01	.87±.01	.72±.02	.64±.02	.87±.01	.68±.03	.87±.01	.68±.03
	deepIRT	.82±.02	.82±.01	.67±.01	.59±.01	.85±.01	.64±.03	.85±.01	.64±.03
	dkt	.82±.02	.81±.02	.66±.01	.57±.02	.85±.01	.64±.03	.85±.01	.64±.03
	dkvmn	.82±.01	.82±.01	.67±.01	.59±.02	.85±.01	.67±.03	.85±.01	.67±.03
	gkt	.71±.07	.67±.08	.61±.03	.57±.02	.86±.01	.60±.06	.86±.01	.60±.06
	qikt	—	—	.66±.00	.58±.00	.82±.00	.53±.00	.82±.00	.53±.00
	saint	.66±.02	.65±.02	.67±.03	.59±.07	.87±.01	.66±.03	.87±.01	.66±.03
	sakt	.82±.02	.81±.02	.66±.03	.56±.02	.85±.01	.60±.02	.85±.01	.60±.02
CD Models (Trained)	simplekt	.86±.01	.85±.01	.70±.01	.62±.01	.86±.01	.68±.02	.86±.01	.68±.02
	DINA	.57±.01	.61±.01	.51±.00	.55±.01	.50±.01	.55±.01	.50±.01	.55±.01
	IRT	.59±.00	.62±.01	.67±.00	.62±.01	.83±.00	.68±.01	.83±.00	.68±.01
	KaNCD	.90±.00	.95±.00	.65±.01	.64±.01	.85±.00	.81±.00	.85±.00	.81±.00
	MIRT	.61±.00	.65±.01	—	—	.71±.01	.66±.01	.71±.01	.66±.01
	NCMD	.89±.00	.95±.00	.66±.01	.67±.00	.85±.00	.80±.00	.85±.00	.80±.00
	LLM Direct (Zero-shot)	Qwen2.5-3B	.56±.00	.79±.00	.35±.00	.54±.00	.19±.00	.56±.00	.19±.00
		Qwen2.5-7B	.64±.00	.73±.00	.65±.00	.62±.00	.76±.00	.63±.00	.75±.00
LBM (Zero-shot)	gemma-3-12b	.63±.00	.96±.00	.58±.00	.71±.00	.30±.00	.71±.00	.29±.00	.70±.00
	gemma-3-27b	.62±.00	.94±.00	.67±.00	.76±.00	.33±.00	.78±.00	.32±.00	.79±.00
	gpt-4o	.85±.01	—	.66±.00	—	.82±.00	—	.79±.01	—
	gpt-4o-mini	.81±.01	—	.67±.01	—	.74±.08	—	.75±.07	—
	gpt-5	.87±.00	—	.71±.02	—	.81±.01	—	.79±.01	—
	Qwen2.5-3B	.61±.01	.65±.01	.38±.00	.55±.01	.27±.01	.58±.01	.30±.01	.63±.02
	Qwen2.5-7B	.65±.01	.69±.01	.58±.01	.55±.01	.75±.01	.63±.00	.74±.01	.62±.02
	gemma-3-12b	.79±.00	.85±.00	.62±.01	.64±.02	.63±.01	.67±.01	.57±.02	.65±.01
	gemma-3-27b	.78±.01	.85±.01	.65±.01	.67±.01	.68±.02	.70±.01	.68±.01	.70±.02
	gpt-4o	.80±.01	—	.66±.02	—	.80±.01	—	.78±.02	—
	gpt-4o-mini	.78±.01	—	.61±.02	—	.70±.08	—	.70±.07	—
	gpt-5	.87±.01	—	.68±.01	—	.76±.01	—	.76±.01	—

A.3 TRAINING LBM ENCODERS

A.3.1 DIFFICULTY STRATIFICATION

To investigate whether the accuracy gains seen during LBM training (Figure 5) vary across students in the dataset, we stratify students according to how many misconception they hold: 0, 1, 2 or 3+. Since each misconception represent additional "irregularities" in the student's response pattern beyond simple mastery of constructs, this effectively stratifies different complexity levels across student knowledge states. Table A2 shows the change in accuracy relative to the GPT-4o baseline across difficulty levels.

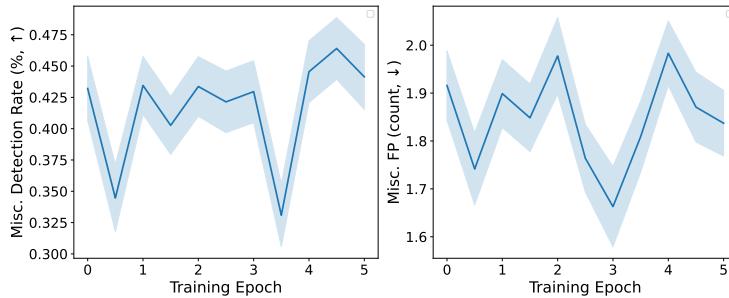
Table A2: Relative difference in accuracy between trained Gemma3-12B and the GPT-4o baseline, with students grouped by number of misconceptions.

# of Misconceptions	Accuracy (mean ± std)			Relative Diff. (%)		
	Baseline	Pre-training	Post-training	Pre	Post	Δ
0 (N = 22)	0.98 ± 0.05	0.92 ± 0.12	0.99 ± 0.05	-5.80	0.90	6.70
1 (N = 17)	0.89 ± 0.09	0.87 ± 0.15	0.91 ± 0.09	-2.00	3.00	5.00
2 (N = 23)	0.81 ± 0.16	0.80 ± 0.12	0.89 ± 0.08	-0.80	10.20	11.00
3 (N = 135)	0.77 ± 0.12	0.74 ± 0.14	0.82 ± 0.11	-4.80	5.70	10.50

864 The model shows 10 – 11% increased accuracy relative to GPT-4o for students with 2 or 3+ misconceptions, compared to just 5 – 7% for students with 0-1 misconceptions. This suggests the encoder
 865 becomes particularly better at handling complex knowledge states.
 866
 867

868 A.3.2 EVOLUTION OF MISCONCEPTION DETECTION AND FALSE POSITIVE 869

870 Figure A2 shows the evolution of the misconception detection rate and misconception false positive
 871 counts over the training of the encoder from §5.2.
 872
 873



883 **Figure A2: Evolution of misconception detection and false positive over encoder training.**
 884
 885

886 A.4 STEERING LBM BEHAVIOR

888 A.4.1 AUGMENTING WITH STUDENT-SPECIFIC INFORMATION TO ASSIST LBM TRAINING

889 Finally, we demonstrate how providing additional information to the LBM can assist with the training
 890 process. On the Synthetic dataset, we train two identical LBMs (Gemma-12B trainable encoder,
 891 Gemma-27B frozen decoder) while providing 2 misconceptions either to the encoder as part of the
 892 input data, or to the decoder as part of the bottleneck. The models are trained for one epoch on
 893 800 students. We then evaluate the resulting models *without any additional information provided*.
 894 Table A3 shows the accuracy before/after training for both models. The model where additional
 895 information was provided to the encoder during training reaches 84% accuracy after training compared
 896 to 80% when it is provided to the decoder. This suggest that the additional information facilitates
 897 training of the encoder.
 898

899 Table A3: Result of training on the Synthetic dataset when providing misconception information
 900 about each student during training, either in the encoder input or in the bottleneck.
 901

	Before training	After 1 epoch training
Information in X	0.802 ± 0.001	0.840 ± 0.004
Information in S	0.794 ± 0.002	0.799 ± 0.006

907 A.4.2 STEERING VIA REWARD SIGNALS

909 To demonstrate the possibility to steer LBMs’ behavior via the reward signal, we consider an example
 910 use-case where a teacher would like the model to pay particular attention to potential misconceptions
 911 held by the student. Following the reward-shaping framework of Section 3.3, we augment the training
 912 objective with an additional term that explicitly encourages the model to surface misconceptions.
 913 Concretely, we set $\Omega(S) = \mathbb{1}_{\mathcal{L}}[\text{“misconception”} \in \tilde{\mathcal{S}}]$, where $\tilde{\mathcal{S}}$ is the textual bottleneck emitted by
 914 the LBM for student state S . This binary reward is added to the accuracy term and optimized with
 915 GRPO. Figure A3 confirms that the policy quickly internalizes this incentive: after only a handful of
 916 training steps, the proportion of summaries that explicitly mention a misconception goes from less
 917 than 80% to >95%, demonstrating that the reward function provides an effective lever for shaping
 higher-level pedagogical behavior.

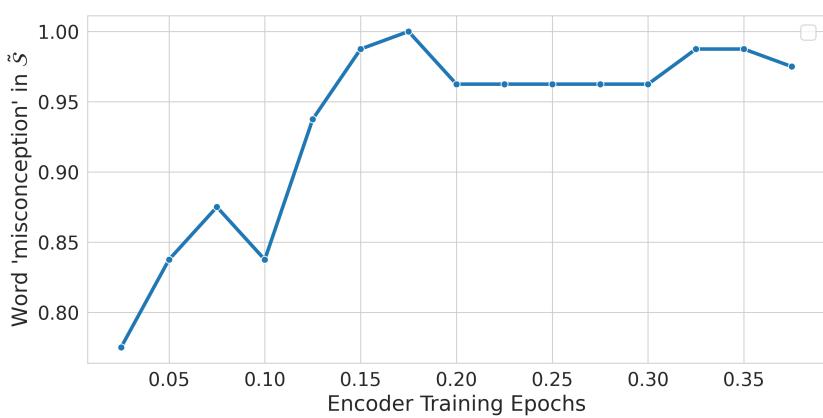


Figure A3: **Presence of the word "misconception" in the LBM's summaries during training of the encoder model steered towards mentioning student misconceptions via the reward signal.**

A.4.3 AUGMENTING LBMs WITH STUDENT-SPECIFIC INFORMATION TO COMPLEMENT INPUT DATA

To demonstrate how an LBM can be actively steered with additional information we run the following ablation on the Synthetic dataset: 1. Each student trajectory originally probes four constructs. For a given trajectory, we remove every question linked to one construct leaving the input data intentionally incomplete. 2. We then run the input data through the encoder and inject a single teacher-supplied sentence describing the student's mastery of the missing construct directly into the model's bottleneck representation (e.g., "The student has mastered addition except in the event of misconceptions").

We repeat this procedure four times—once for each construct—running the LBM both with and without the additional sentence, and report the results Table A4. Naturally, the models complemented with the additional information of the student's mastery of the construct missing in the input data outperform models provided only with the incomplete data. This small experiment illustrates a key advantage of LBMs: because the LBM compresses evidence into a text-based summary, it can be complemented with additional information absent from the input data.

Table A4: Accuracy of a gpt-4o-based LBM on the Synthetic dataset while removing one construct from the input data, with and without providing information about the missing construct in the bottleneck. Each run is repeated across all four constructs; mean and standard deviation are reported.

Without additional bottleneck information	With additional bottleneck information
0.749 ± 0.007	0.791 ± 0.013

A.5 ABLATION EXPERIMENTS

A.5.1 ENCODER/DECODER VARIANTS

Table A5 shows the accuracy of different combinations of encoder-decoder models on the Synthetic dataset. The top row shows the result of using the strongest model evaluated (gpt-4o) as both encoder and decoder. Then, we vary either the encoder or decoder part of the LBM across other LLMs, and report the resulting accuracy. A clear pattern which emerges is that the resulting LBMs are stronger when gpt-4o is used as the *encoder* instead of the *decoder*. This implies that the task of accurately capturing knowledge state information in the bottleneck is harder than predicting answers to future questions when provided with a knowledge state summary.

972 Table A5: Performance of different LBM encoder-decoder combinations on the Synthetic dataset.
973

	Encoder	Decoder	Accuracy
Strongest model	gpt-4o	gpt-4o	0.809
Strongest model as encoder	gpt-4o	gpt-4o-mini	0.821
	gpt-4o	google/gemma-3-12b-it	0.795
	gpt-4o	Qwen/Qwen2.5-7B-Instruct	0.765
	gpt-4o	Qwen/Qwen2.5-3B-Instruct	0.715
Strongest model as decoder	gpt-4o-mini	gpt-4o	0.765
	google/gemma-3-12b-it	gpt-4o	0.775
	Qwen/Qwen2.5-7B-Instruct	gpt-4o	0.666
	Qwen/Qwen2.5-3B-Instruct	gpt-4o	0.649

981 A.5.2 IMPORTANCE OF KNOWLEDGE CONCEPTS IN THE INPUT
982983 Table A6 compares the accuracy of different LBMs with vs without knowledge conception information
984 in the input prompt. The performance generally remains similar to the KC version, and it even
985 increases for most models on the EEDI dataset. This demonstrates that LBMs do not fundamentally
986 require KC information.987 Table A6: Comparison of LBM performance with vs without knowledge concept (KC) information
988 in the input. Bold indicates a difference in accuracy no more than 3% below the full input.

	Synthetic			EEDI (Filtered)		
	w/ KC	w/o KC	Δ	w/ KC	w/o KC	Δ
Qwen2.5-3B	.61 ± .01	.62 ± .05	+.00	.38 ± .00	.38 ± .02	+.01
Qwen2.5-7B	.65 ± .01	.66 ± .02	+.01	.58 ± .01	.58 ± .01	+.00
gemma-3-12b	.79 ± .00	.73 ± .00	-.06	.62 ± .01	.60 ± .02	-.02
gemma-3-27b	.78 ± .01	.75 ± .01	-.03	.65 ± .01	.69 ± .03	.03
gpt-4o-mini	.78 ± .01	.76 ± .00	-.01	.61 ± .02	.65 ± .03	.04
gpt-4o	.80 ± .01	.76 ± .01	-.04	.66 ± .02	.68 ± .01	.02

1006 A.6 EXAMPLE BOTTLENECKS ON THE EEDI DATASET

1007 Figure A4 shows example bottlenecks produced by a gpt-4o-based LBM for a student in the Eedi
1008 dataset. The input data is composed of 30 questions from various constructs, and the decoder predicts
1009 4 test questions. As shown Figure 4 in the main paper, a shorter bottleneck of 128 tokens constrains
1010 the expressive power of the model, and in this example the resulting predictions fail on two of the
1011 test questions. A longer bottleneck of 256 or 512 tokens allows for more nuance and details in the
1012 bottleneck, leading to the decoder correctly predicting all four test questions for this student.

1015 B EXPERIMENTAL DETAILS

1016 B.1 LBMs DETAILS

1017 **LLM backbones** We evaluate the following closed- and open-source models:1018

- GPT-5, GPT-4o and GPT-4o-mini (Achiam et al., 2023);
- Qwen2.5-3B and Qwen2.5-7B (Team, 2024);
- Gemma3-12B and Gemma3-27B (Team, 2025).

1019 For open-source models we extract the activation logits of the “Yes” and “No” tokens and return
1020 the higher value. The logits for the “Yes” and “No” tokens are used to compute the AUC. For

1026				
1027				
1028				
1029				
1030				
1031				
1032				
1033				
1034				
1035				
1036				
1037				
1038				
1039				
1040				
1041				
1042				
1043				
1044				
1045				
1046				
1047				
1048				
1049				
1050				
1051				
1052				
1053	Student #44759 Predicted Knowledge States $\tilde{\mathcal{S}}$			Predictions on 4 test questions
1054	128 tokens			
1055	<p>The student demonstrates accurate skills in writing numbers in standard form, solving timetable problems, identifying equivalent fractions, interpreting bar charts, converting percentages to fractions, algebraic substitution, and various basic arithmetic operations. However, they struggle with more complex fractions, decimal operations, angle calculations, rounding, function machines, and mental addition strategies. Overall, the student has a solid foundation in key areas but needs improvement in dealing with fractions of different denominators, negative integers, and decimal calculations.</p>			   
1056	256 tokens			
1057	<p>The student has demonstrated robust skills in working with basic fractions, decimals, and standard form for numbers between 0 and 1. They correctly answered questions involving simple fraction operations, frequency trees, interpreting timetables, substituting values into expressions, function machine outputs, estimation, rounding numbers greater than 1, and calculating non-unit fractions of an amount. However, they struggled with more complex fraction additions, algebraic fractions, and working with decimals involving operations such as multiplication by an integer or other decimals. They also had difficulty with algebraic manipulations, calculating the square of decimals, and understanding angles related to parallel lines and polygons. Despite these difficulties, they correctly handled finding exterior angles of regular polygons and working with integers and negative numbers in simpler multiplication and subtraction problems. The student's proficiency lies in straightforward calculations and procedural tasks but falters slightly with combined algebraic and decimal operations, more complex fraction manipulations, and multi-step angle calculations.</p>			   
1058	512 tokens			
1059	<p>The student demonstrates proficiency in various mathematical areas, with some notable gaps. They excel in reading timetables, identifying equivalent fractions, writing numbers between 0 and 1 in standard form, performing calculator calculations with negative numbers, rounding numbers greater than 1 to one significant figure, identifying non-unit fractions in shapes, subtracting proper fractions with the same denominator, calculating non-unit fractions of an amount, and working out the size of each exterior angle in regular polygons.</p>	<p>6. Multiplying two decimals together. 7. Subtracting proper fractions which share a common factor. 8. Performing estimation and rounding numbers to one significant figure. 9. Calculating the square of a number. 10. Multiplying a decimal by an integer.</p>		
1060	<p>However, the student struggles with several topics:</p> <ul style="list-style-type: none">1. Adding algebraic fractions where one denominator is a multiple of the other.2. Finding incorrect steps in solving for angles on parallel lines using angle facts.3. Carrying out multiplication and division involving negative integers.4. Dividing integers by decimals and understanding efficient strategies for mental addition.5. Writing numbers greater than 1 in standard form.	<p>The student correctly answers questions involving basic algebra (substituting positive integers into expressions), function machines, basic fraction operations (adding fractions with the same denominator), and basic arithmetic operations (subtracting negative integers). They also correctly convert percentages to fractions and add and subtract proper fractions.</p>		
1061		<p>These results suggest a strong understanding of fundamental mathematical concepts, though their skills in more complex and nuanced problems could use improvement. The student would benefit from focused practice on algebraic manipulations, angle calculations, application of decimal operations, and standard form conversion.</p>		
1062				   
1063				

Figure A4: **Example bottlenecks produced by an LBM with gpt-4o as both encoder and decoder for a student in the Eedi dataset, with varying bottleneck lengths.** The input data provided to the encoder is composed of 30 questions across various constructs, and the decoder predicts 4 test questions using the bottleneck.

closed-source models we prompt for a “Yes” or “No” answer and parse the text output. We run GPT models via the OpenAI API, with the following snapshots: GPT-5: 2025-08-07; GPT-4o: 2024-08-06; GPT-4o-mini: 2024-07-18.

Reward function For training the encoder in Section 5, we set the reward function to the decoder accuracy across $|\mathcal{Y}| = 20$ questions:

$$R(\tilde{\mathcal{S}}; g) = \text{ACC}(\tilde{\mathcal{Y}}, \mathcal{Y})$$

The RL steering experiment section A.4.2 only rewards the presence of the word “misconception” in the bottleneck:

$$R(\tilde{\mathcal{S}}; g) = \Omega(\tilde{\mathcal{S}}) = \mathbb{1}[\text{'misconception'} \in \tilde{\mathcal{S}}]$$

B.2 KT/CD MODELS

Cognitive Diagnosis models We evaluate the following 5 Cognitive Diagnosis models:

- IRT (Rasch, 1993)
- MIRT (Reckase, 2006)
- DINA (Junker & Sijtsma, 2001)
- KaNCD (Wang et al., 2022)
- NeuralCDM (Wang et al., 2022)

1080 We use the implementation from the EduCDM library (bigdata ustc, 2021). To make sure there is
 1081 enough data per student to train on, we filter out students with < 10 interactions in each
 1082 dataset.
 1083

1084 **Knowledge Tracing models** We evaluate the following 9 Knowledge Tracing models:
 1085

- 1086 • DKT Piech et al. (2015)
- 1087 • DKVMN Zhang et al. (2017)
- 1088 • SAKT Pandey & Karypis (2019)
- 1089 • GKT Nakagawa et al. (2019)
- 1090 • Deep IRT Yeung (2019)
- 1091 • AKT Ghosh et al. (2020)
- 1092 • SAINT Choi et al. (2020)
- 1093 • SimpleKT Liu et al. (2023a)
- 1094 • QIKT Chen et al. (2023)

1095 We use the PYKT implementation Liu et al. (2022) for all of these models with default hyperparameters.
 1096 We only modify the pyKT implementation to additionally compute the accuracy and AUC on
 1097 $N = |\mathcal{Y}|$ questions.
 1098

1100 B.3 DATASETS

1101 B.3.1 SYNTHETIC

1102 Our synthetic dataset simulates students answering basic arithmetic problems. Each student is
 1103 characterised by (i) mastered skills, (ii) unmastered skills, and (iii) systematic misconceptions.
 1104 Questions are arithmetic operations—addition, subtraction, multiplication, or division (rounded to
 1105 the nearest integer)—between two operands drawn from $[0, 15]$ for addition/subtraction or $[1, 10]$ for
 1106 multiplication/division.
 1107

1108 **Misconception pool.** For every student we sample misconceptions uniformly at random from:
 1109

- 1110 • forgets to carry in addition;
- 1111 • fails multiplications involving the number x with $x \sim \mathcal{U}(6, 9)$;
- 1112 • fails any operation involving the number x with $x \sim \mathcal{U}(6, 9)$;
- 1113 • fails whenever an operand > 10 ;
- 1114 • always rounds division results down;
- 1115 • fails with negative numbers.

1116 B.3.2 EEDI

1117 Figure B1 shows the histogram of the number of misconceptions per student across the 2000 students
 1118 of the dataset.
 1119

1120 B.3.2 EEDI

1121 The Eedi dataset is analogous to the one publicly shared via the NeurIPS 2020 Education Chal-
 1122 lenge (Wang et al., 2020) organised by Eedi, but additionally includes the text of each question. While
 1123 the exact version of the dataset used in this work is not available publicly, a very similar version
 1124 including question texts is available via the "Eedi - Mining Misconceptions in Mathematics" Kaggle
 1125 Competition (King et al., 2024).

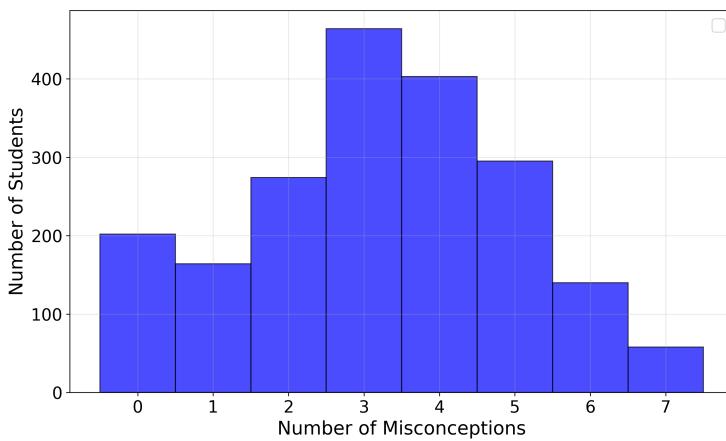


Figure B1: Distribution of the number of misconceptions per student in the Synthetic dataset.

Preprocessing Although many trajectories in the Eedi dataset span several days or months, we retain only single-session sequences to satisfy the constant knowledge state assumption - a common assumption in Cognitive Diagnosis (Wang et al., 2024). We compute sessions of interactions by grouping answers that are within the follow criterias:

- minimum response time: 5 s, to avoid random answers;
- maximum gap between answers: 3 min;
- minimum trajectory length: 40 questions.

This filtering yields a total of 623 individual trajectories with 40+ questions.

B.3.3 XES3G5M

The XES3G5M dataset contains student interaction data from a large-scale online mathematics learning platform (Liu et al., 2023b). The original dataset has question and construct text in Chinese. We use English translation from Ozyurt et al. (2024) to also run on an English version of the dataset.

Preprocessing The XES3G5M preprocessing follows a similar approach to Eedi, with adjusted parameters to accommodate the characteristics of this dataset. We compute sessions of interactions by grouping answers that meet the following criteria:

- minimum response time: 5s, to avoid random answers;
- maximum gap between answers: 10min;
- minimum trajectory length: 34 questions.

The increased maximum gap between answers (10 minutes vs. 3 minutes for Eedi) is to ensure that a sufficient number of sessions are available for training. More stringent filtering make it easier to satisfy the constant knowledge state assumption, but might not produce enough data points for effective training of KT and CD models. This filtering yields a total of 1,245 student-sessions trajectories with 34+ questions, across 996 individual students.

B.4 EXPERIMENT PARAMETERS

B.4.1 TABLE A1, FIGURE 7, TABLE 2, FIGURE 9

Table B1 summarize experimental settings used for Table A1, Figure 7, Table 2 and Figure 9. Runs are aggregated across N=3 for LBMs/Direct LLMs and N=10 for KT/CD models. For Figure 9 the bottleneck size varies along the plot's x axes.

	Synthetic	Eedi (Filtered)	XES3G5M (Filtered)
$ \mathcal{X} $	50	30	30
$ \mathcal{Y} $	4	4	4
Test students	200	100	200
Bottleneck size	128 tokens	512 tokens	512 tokens
CoT prompting	No	No	No

Table B1: Experimental settings for the different datasets.

Statistical Test for Table 2 Table 2 shows in bold the models and datasets for which the LBM variant has an accuracy of no more than 2% below the Direct baseline. We use Welch’s t-test with the null hypothesis: ‘ $\text{accuracy}_{LBM} \geq \text{accuracy}_{\text{direct}} - 0.02$ ’, and report in bold runs where $p < 0.05$.

B.4.2 FIGURE 8

Latent Bottleneck Models (LBMs)

- $|\mathcal{X}| = N, N \in \{5, 10, 20, 50\}, |\mathcal{Y}| = 4$
- Test students: 200
- Bottleneck size: 256 tokens
- Chain-of-thought prompting: **Yes**; reasoning_effort=minimal for GPT-5
- Encoder & decoder: same backbone LLM (varies by row in the table)

CD baselines The CD models were run using the EduCDM implementation (bigdata ustc, 2021). As these models require training on a per-student basis, we evaluated them on a held-out set of test interactions from the same students seen during training. **We used an 80/20 train/test split for the Synthetic dataset and 88/12 for EEDI and XES3G5M of each student’s interaction history.** While this evaluation setup differs from the unseen-student protocol used for KT and LBMs, it allows for a fair comparison of sample efficiency. To ensure stable and reliable accuracy estimates, especially with limited interactions, all results are aggregated and averaged over $N=10$ independent runs.

KT baselines KT models were evaluated using the PYKT implementation Liu et al. (2022), modified to compute accuracy on $|\mathcal{Y}| = 4$ questions, in order to be comparable with our LBM models. For each method we keep the pyKT default hyperparameters. Test accuracy is computed across 200 unseen test students.

B.4.3 FIGURE 5

- Dataset: Synthetic
- $|\mathcal{X}| = 50, |\mathcal{Y}| = 20$
- Train / test students: 800/200
- Bottleneck size: 128 tokens
- Chain-of-thought prompting: No
- Encoder (trained): Gemma3-12B
- Decoder (frozen): Gemma3-27B

Training hyper-parameters.

- Batch size : 5
- $G = 5$
- Learning rate = 1×10^{-4}
- $\beta = 0.04$

1242 • Optimiser: Adam (adamw_torch, default settings)
 1243 • **LoRA configuration**
 1244 – $r = 16, \alpha = 16$
 1245 – target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
 1246 "gate_proj", "up_proj", "down_proj"]
 1247 – Dropout: 0.05
 1248
 1249

1250 B.4.4 ENCODER-DECODER VARIANTS

1251 Identical to the Table A1 set-up, except for the choice of encoder and decoder LLMs.

1253 B.4.5 STEERING EXPERIMENTS

1255 Same dataset and model parameters as for Figure 5.

1257 **Training hyper-parameters.**

1259 • Batch size : 4
 1260 • $G = 4$
 1261 • Learning rate = 1×10^{-4}
 1262 • $\beta = 0.04$
 1263 • Optimiser: Adam (adamw_torch, default settings)
 1264 • **LoRA configuration**
 1265 – $r = 16, \alpha = 16$
 1266 – target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
 1267 "gate_proj", "up_proj", "down_proj"]
 1268 – Dropout: 0.05
 1269
 1270

1271 B.5 PROMPTS

1273 Below are the different prompts used to query LLMs in our experiments.

```

1  template: |
2    Please produce a concise summary of the following input with up to
3    {max_words} words.
4    This summary should contain the information required to predict
5    the student's answer to any new question.
6    {cot_instruction}
7
8    Input:
9    {input_text}
10
11   Once again, please provide a **concise** summary of the student's
12   knowledge state.
13   {cot_instruction}
14   Keep your summary under {max_words} words.
15
16   # with CoT
17   cot_instruction: |
18     Think step by step and lay out your reasoning before you write the
19     final summary. Then, enclose the final summary in <info>...</
20     info>.
21   # without CoT
22   cot_instruction: |
23     Enclose your entire summary in <info>...</info> and do not include
24     anything else.
25
26   input_text: |
```

```

1296 21 The student answered the following questions:
1297 22 {question_1_text}
1298 23 ...
1299 24 {question_n_text}
1300 25
1301 26 # with construct information
1302 27 question_i_text: |
1303 28     Question {question_txt, question_ID}, with construct{construct_txt
1304 29         , construct_ID}: answered {correctness}.
1305 30 # without construct information
1306 31 question_i_text: |
1307 32     Question {question_txt, question_ID}: answered {correctness}.

```

Listing 1: Base Prompt for the Encoder.

1309

```

1310 1 template: |
1311 2     Here is some information about a student:
1312 3     {bottleneck}
1313 4
1314 5     Predict whether the student will answer {new_question} correctly
1315 6         or not.
1316 6     Answer with "Yes" or "No" and nothing else.

```

Listing 2: Base Prompt for the Decoder.

1317

```

1318 1 template: |
1319 2     Please produce a concise summary of the following input with up to
1320 3         {max_words} words.
1321 3     This summary should contain the information required to predict
1322 4         the student's answer to any new question.
1323 4
1324 5     Make sure to mention any misconception held by the student.
1325 6     {cot_instruction}
1326 7
1327 8     Input:
1328 9     {input_text}
1329 10
1330 11     Once again, please provide a **concise** summary of the student's
1331 12         knowledge state.
1332 13     {cot_instruction}
1333 13     Keep your summary under {max_words} words.

```

Listing 3: Base Prompt for the Encoder when steering for mentioning "misconceptions" (Section A.4.2).

1335

1336

B.6 CODE AND HARDWARE

1338

Experiments were ran on four NVIDIA A100 GPUs with 80GB VRAM each, and 880GB of total RAM.

1341

C SYSTEMATIC EVALUATION VIA LLM-AS-A-JUDGE

1343

C.1 METHOD

1345

We use LLM-as-a-judge (Li et al., 2025) with GPT-5 to systematically evaluate generated summaries on the Synthetic dataset. Specifically, for a given student in the Synthetic dataset we provide GPT-5 with the ground truth knowledge state and a summary to be evaluated. The judge LLM is blinded to the model that produced the summary. We then prompt the model to assess the summary across the following dimensions:

- 1350 • **Misconception detection:** for each misconception in the student knowledge state, assess
1351 how whether the summary captures it. Score: 0,1,2, 0 for not mentioned, 1 for partial
1352 mention (implied or vaguely hinted at), 2 for explicitly stated.
- 1353 • **Misconception false positives:** report any instance of misconception stated in the summary
1354 that do not appear in the ground truth knowledge state.
- 1355 • **Construct-level accuracy:** for each construct, determine whether the construct’s true
1356 mastery was accurately captured by the summary. Score: 0,1, 0 for incorrect or missing, 1
1357 for correct.
- 1358 • **Overall score:** generally assess the alignment of the summary with the ground truth. Score:
1359 1 to 5, from major mismatch (1) to very strong alignment (5).
- 1360 • **Confidence calibration:** evaluate whether the summary expresses uncertainty appropriately.
1361 Score: 1 to 3, 1 for under-confident and vague, 2 for well-calibrated, 3 for over-confident.
- 1362 • **Specificity:** evaluate how precise the summary is. Score: 1 to 3, 1 for very generic and
1363 vague, 2 for moderately specific, 3 for highly specific.

1365 C.2 EVALUATION EXAMPLES

1366 Table C1 shows the result of the evaluation for different summaries generated for student #1585.

1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403

Table C1: Comparison of generated student summaries and GPT-5-based LLM-as-a-judge evaluation (Synthetic dataset, student #1585). **Keys:** **Q.:** Overall Score (1-5, higher is better). **Constructs** (0/1): Add (Addition), Sub (Subtraction), Mul (Multiplication), Div (Division). **Misconceptions** (0-2, 0=missed, 1=implied, 2=explicitly noted): Div R (Rounds down division), Add C (Forgets carry), Neg (Fails negatives), x7 (Fails x7). **FP:** False Positives count. **Cal:** Confidence Calibration (1-3, 1=under-confident, 2=appropriately calibrated, 3=over-confident). **Spec:** Specificity (1-3, higher is more specific).

Model	Generated Summary	Q. (1-5)	Constructs (0/1)				Misconceptions (0-2)				FP	Cal	Spec
			Add	Sub	Mul	Div	Div R	Add C	Neg	x7			
Ground Truth	The student has mastered all skills (Add, Sub, Mul, Div) but has systematic misconceptions: Always rounds down division; Forgets to carry in addition; Fails with negative numbers; Fails multiplication involving 7.	-	1	1	1	1	2	2	2	2	-	-	-
gpt-5	Student is strong in basic addition (mostly correct for sums up to ~25, but missed 3+8, 8+12, 6+15, 15+8). Subtraction: competent with small positive results (e.g., 10-4, 14-12, 15-13, 5-1), but frequently incorrect on negatives/zero-minus cases (3-6, 0-12, 3-15, 3-9, 0-15, 5-11, 4-9, 3-7, 1-15). Multiplication: generally solid (facts like 2×2, 4×1, 4×5, 6×5, 6×10, 9×8 correct) but missed 5×7. Division (rounded): good with whole-number quotients ($\div 1$, $10 \div 2$, $4 \div 2$, $6 \div 1$) and $6 \div 5$, but struggles with non-integer cases ($8 \div 5$, $6 \div 4$, $5 \div 3$).	5	1	1	1	1	2	1	2	1	0	2	3
gpt-4o	The student consistently answers addition (construct ID 0) and multiplication (construct ID 2) questions correctly, reflecting strong competence in these areas. They show varying performance in subtraction (construct ID 1), struggling with negative results. In division (construct ID 3), they often answer correctly when the result is a whole number but struggle when rounding is required. Their overall performance indicates proficiency in basic arithmetic, particularly addition and multiplication, with some difficulty in questions involving subtraction with negative results and non-whole arithmetic operations.	4	1	1	1	1	1	0	2	0	1	2	2

1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469

Table C1 – continued from previous page

Model	Generated Summary	Q. (1-5)	Constructs (0/1)				Misconceptions (0-2)				FP	Cal	Spec
			Add	Sub	Mul	Div	Div	R	Add	C	Neg		
gemma-3-12b	The student demonstrates strong addition and multiplication skills (constructs 0 & 2) with occasional errors. Subtraction (construct 1) is a significant weakness, showing frequent mistakes across various numbers. Division with rounding (construct 3) is inconsistent; the student excels with simple divisions but struggles with others. Performance is notably worse with negative results or larger numbers in subtraction and occasionally addition. The student appears to grasp basic arithmetic operations but lacks fluency and accuracy in subtraction.	2	0	0	0	0	1	1	2	0	2	3	2
Qwen2.5-3B	The student has strong accuracy in arithmetic operations like addition and multiplication, particularly when constructs are 0, 2, or 3. However, they struggle with subtraction and division, especially when rounding is involved (constructs 1 and 3).	2	0	0	1	0	1	0	0	0	1	3	2
gpt-4o-mini	The student struggles with basic subtraction problems, having answered incorrectly on many related questions. They demonstrate understanding of addition and multiplication, scoring correctly on multiple questions in those areas. The student has shown some success in division, rounding, but struggles with specific datasets. They can solve problems involving simple addition, like 10 - 4, but make errors with negative results and more complex subtraction like 3 - 15. Overall, their strengths lie in addition and multiplication, while they need improvement in subtraction and certain division problems.	2	0	0	1	0	1	0	2	0	2	3	2
Qwen2.5-7B	The student struggles with subtraction (construct ID 1) and division (construct ID 3), while performing well in addition (construct ID 0) and multiplication (construct ID 2). Correct answers include 10/2 (5), 11+10 (21), 0+6 (6), 6/5 (1), 12+1 (13), 4*5 (20), 9*4 (36), 15-13 (2), 13+12 (25), 6/1 (6), 8*1 (8), 5*7 (35), 2+13 (15), 9*8 (72), 9*8 (72), 4/2 (2).	1	0	0	0	0	0	0	0	0	0	3	2

1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1499
1500
1501
1502

Table C1 – continued from previous page

Model	Generated Summary	Q. (1-5)	Constructs (0/1)				Misconceptions (0-2)				FP	Cal	Spec
			Add	Sub	Mul	Div	Div	R	Add	C	Neg		
gemma-3-27b	The student reliably performs addition and multiplication with smaller positive integers (constructs 0 & 2). They struggle with subtraction and division, especially when results are negative or not whole numbers (constructs 1 & 3). Accuracy decreases with larger numbers in subtraction and addition. Division rounding is often incorrect. Performance on construct 0 is ~85% correct, construct 1 is ~40% correct, construct 2 is ~75% correct, and construct 3 is ~50% correct.	1	0	0	0	0	1	0	2	0	2	3	2

1503 **C.3 PROMPTS**

1504

1505 We use the following system prompt and prompt template to instruct and query the judge model.

1506

1507

1508

1509

1510

1511

1512

1513

1514

1515

1516

1517

1518

1519

1520

1521

1522

1523

1524

1525

1526

1527

1528

1529

1530

1531

1532

1533

1534

1535

1536

1537

1538

1539

1540

1541

1542

1543

1544

1545

1546

1547

1548

1549

1550

1551

1552

1553

1554

1555

1556

1557

1558

1559

1560

1561

1562

1563

1564

1565

1566

1567

1568

1569

1570

1571

1572

1573

1574

1575

1576

1577

1578

1579

1580

1581

1582

1583

1584

1585

1586

1587

1588

1589

1590

1591

1592

1593

1594

1595

1596

1597

1598

1599

1600

1601

1602

1603

1604

1605

1606

1607

1608

1609

1610

1611

1612

1613

1614

1615

1616

1617

1618

1619

1620

1621

1622

1623

1624

1625

1626

1627

1628

1629

1630

1631

1632

1633

1634

1635

1636

1637

1638

1639

1640

1641

1642

1643

1644

1645

1646

1647

1648

1649

1650

1651

1652

1653

1654

1655

1656

1657

1658

1659

1660

1661

1662

1663

1664

1665

1666

1667

1668

1669

1670

1671

1672

1673

1674

1675

1676

1677

1678

1679

1680

1681

1682

1683

1684

1685

1686

1687

1688

1689

1690

1691

1692

1693

1694

1695

1696

1697

1698

1699

1700

1701

1702

1703

1704

1705

1706

1707

1708

1709

1710

1711

1712

1713

1714

1715

1716

1717

1718

1719

1720

1721

1722

1723

1724

1725

1726

1727

1728

1729

1730

1731

1732

1733

1734

1735

1736

1737

1738

1739

1740

1741

1742

1743

1744

1745

1746

1747

1748

1749

1750

1751

1752

1753

1754

1755

1756

1757

1758

1759

1760

1761

1762

1763

1764

1765

1766

1767

1768

1769

1770

1771

1772

1773

1774

1775

1776

1777

1778

1779

1780

1781

1782

1783

1784

1785

1786

1787

1788

1789

1790

1791

1792

1793

1794

1795

1796

1797

1798

1799

1800

1801

1802

1803

1804

1805

1806

1807

1808

1809

1810

1811

1812

1813

1814

1815

1816

1817

1818

1819

1820

1821

1822

1823

1824

1825

1826

1827

1828

1829

```

1557 49 "misc_false_positive": [..., ...]
1558 50
1559 51 3. *Construct-level accuracy (0 1 )*
1560 52 There are four constructs: addition, subtraction, multiplication,
1561 53 division.
1562 54 Score whether the bottleneck correctly reflects the student's
1563 55 mastery status, **taking misconceptions into account**. E.g.,
1564 56 if a construct is mastered except when a misconception applies
1565 57 , statements like good at X except in specific cases
1566 58 count as correct.
1567 59 Score 1 if the construct is accurately described; score 0 if
1568 60 inaccurate or missing.
1569 61 Output:
1570 62 "construct_scores": {"addition": <0/1>, "subtraction": <0/1>, "
1571 63     "multiplication": <0/1>, "division": <0/1>}
1572 64
1573 65 4. *Overall quality (1 5 )*
1574 66 Holistic score of how well the bottleneck aligns with the entire
1575 67 ground truth.
1576 68 1 = Major mismatch
1577 69 2 = Several substantial errors
1578 70 3 = Mixed; partial alignment
1579 71 4 = Mostly correct with minor issues, such as rare
1580 72 misconceptions
1581 73 5 = Very strong alignment
1582 74 Output:
1583 75 "overall_score": <1 5 >
1584 76
1585 77 5. *Confidence calibration (1 3 )*
1586 78 Evaluate whether the bottleneck expresses uncertainty
1587 79 appropriately:
1588 80 - 1 = Underconfident (hedges excessively; too vague or
1589 81     noncommittal)
1590 82 - 2 = Well-calibrated (claims are proportional, avoids strong
1591 83     unwarranted assertions)
1592 84 - 3 = Overconfident (makes strong claims not justified by the data
1593 85     )
1594 86 Output:
1595 87 "confidence_calibration": <1 3 >
1596 88
1597 89 6. *Specificity (1 3 )*
1598 90 Evaluate how specific or generic the bottleneck is:
1599 91 - 1 = Very generic; vague statements without detail
1600 92 - 2 = Moderately specific; some detail, but missing precision
1601 93 - 3 = Highly specific; clearly articulated patterns that closely
1602 94     match the ground truth
1603 95 Output:
1604 96 "specificity": <1 3 >
1605 97
1606 98 # OUTPUT FORMAT
1607 99
1608 100 Return only a dictionary in the following structure:
1609 101 {
1610 102     "misc_scores": {"0": 2, ...},
1611 103     "misc_false_positive": ['struggles with rounding', ...]
1612 104     "construct_scores": {"addition": 1, "subtraction": 0, "
1613 105         "multiplication": 1, "division": 1},
1614 106     "overall_score": 4,
1615 107     "confidence_calibration": 2,
1616 108     "specificity": 3,
1617 109     }
1618 110
1619 111 Do not include anything else in your answer.
1620 112
1621 113 user_prompt_template: |

```

```

1611 99 # Bottleneck
1612 100 {bottleneck}
1613 101
1614 102 # Ground Truth Constructs
1615 103 {constructs}
1616 104
1617 105 # Ground Truth Misconceptions
1618 106 {misconceptions}

```

Listing 4: Prompts used in LLM-as-a-judge evaluation.

D STEERABILITY OF THE ESTIMATED KNOWLEDGE STATE

A key advantage of LBMs is the ability for humans to interact with the model to steer the estimated student knowledge states and complement the model with additional information. Here, we further discuss the three mechanisms for human-model interaction in the LBM framework mentioned in Section 3.4.

Prompt engineering the encoder. The most straightforward approach is directly shaping how the encoder generates summaries through prompt engineering, for example via system instructions or in-context examples (Brown et al., 2020). By modifying the instruction prompt given to the encoder, educators can influence the format, emphasis, and level of detail in the generated knowledge state summaries. For instance, instructing the encoder to highlight specific types of misconceptions or to focus on particular subject areas can yield more targeted summaries. Examples of good and bad knowledge states can also be provided to the encoder to steer its behavior via in-context learning.

Steering via reward signals during training. When training the encoder, human preferences can be incorporated through the reward function by using the $\Omega(S)$ term from Eq. 1. This allows for systematic enforcement of desirable summary properties across the model’s outputs. For example, if educators find that explicit enumeration of misconceptions is particularly valuable for intervention planning, the reward function can be designed to favor summaries that consistently identify and articulate student misconceptions – as demonstrated Figure A.4.2.

Augmenting with student-specific information. Perhaps the most powerful form of interaction involves supplementing the model with additional student-specific information not present in the observed interaction data X . This can occur in two ways:

- **Augmenting encoder input:** Educators can provide supplementary information alongside the observed interactions X , such as notes about recent classroom activities not yet reflected in assessment data, or observations about a student’s learning process not captured in their answers.
- **Modifying the generated summary:** After the encoder produces a knowledge state summary S , educators can directly edit this summary based on their domain expertise and student-specific knowledge before passing it to the decoder. For example, a teacher who noticed a student struggling with negative numbers during an in-class exercise could add this observation to the generated summary, even if the available assessment data contains few questions involving negative numbers.

E COMPUTATION COST ANALYSIS

Table E1 compares the average wall-clock runtime of different model types across our experiments. LBMs take up to 10x more time to run than traditional methods due to LLM inference costs, and roughly 2x longer than the Direct variant (as it requires inference on both the encoder and decoder). Despite higher computational costs, LBMs offer unique qualitative interpretability, nuanced insights and zero-shot capabilities that justify such overhead for educational applications where interpretability is key and training data is limited.

1665 Table E1: Average computation time across (in seconds) of each model across our experiments,
 1666 aggregated by model type (mean \pm std). KT models are run on an NVIDIA RTX 6000 48GB, the CD,
 1667 Direct and LBM models are run on an NVIDIA A100 80GB.

Model Type	Synthetic	EEDI (Filtered)	XES3G5M (Filtered)
CDM (Training + Inference)	24.8 ± 22.5	69.3 ± 70.5	131.0 ± 195.9
KT (Training + Inference)	180.5 ± 333.8	149.0 ± 237.1	242.2 ± 498.9
Direct (Inference)	268.6 ± 262.4	629.0 ± 863.0	479.3 ± 371.8
LBM (Inference)	445.0 ± 629.2	1384.3 ± 2091.1	1352.7 ± 2091.7

F EXTENDED RELATED WORK

This section provides an extended review of the literature related to our proposed approach, spanning traditional knowledge tracing, recent advances in LLM-based student modeling, and concept bottleneck models (CBMs), and text summarization models.

F.1 COGNITIVE DIAGNOSIS

Cognitive Diagnosis Models (CDMs) aim to infer a student’s latent knowledge state from their observed responses to test questions Wang et al. (2024).

F.1.1 PSYCHOMETRICS-BASED CD MODELS

Classical models for Cognitive Diagnosis originate from psychometrics, including Item Response Theory (IRT) and its multidimensional variant (MIRT), which model continuous scores of knowledge proficiency using logistic functions Rasch (1993); Reckase (2006). The DINA model estimates binary mastery variables of knowledge concepts, assuming that students must master all required skills to answer correctly, while accounting for slips and guesses Junker & Sijtsma (2001). Other variants have been proposed with different assumptions, such as DINO (Templin & Henson, 2006) which considers that students will correctly answer the item if at least one required knowledge concept is mastered. A critical input for these methods is the Q-matrix, which describe which knowledge concept is required for each question.

F.1.2 DEEP LEARNING-BASED CD MODELS

More recent deep learning approaches offer greater flexibility in modeling complex relationships. The Neural Cognitive Diagnosis Model (NCDM) uses neural networks to learn interaction functions between student proficiency vectors and item characteristics Wang et al. (2022). Extensions include Kernel-based Neural Cognitive Diagnosis (KaNCD), which models latent associations between knowledge concepts Wang et al. (2022), and Knowledge-Sensitive Cognitive Diagnosis (KSCD), which learns intrinsic relations between knowledge concepts from student responses Ma et al. (2022). Graph neural networks have also been incorporated, with frameworks like RCD capturing relationships between students, questions, and knowledge concepts Gao et al. (2021). Recent encoder-decoder architectures, such as ID-CDF, enable inductive diagnosis by directly encoding student responses into ability vectors Li et al. (2024b). While these deep learning models provide enhanced predictive power and can handle diverse data types, they still typically operate within predefined knowledge concept frameworks and are limited to quantitative estimates of skill mastery.

F.2 KNOWLEDGE TRACING

Compared to Cognitive Diagnosis which assumes a constant knowledge state, Knowledge Tracing methods aim at estimating evolving knowledge states as students answer questions. We similarly review Knowledge Tracing methods Shen et al. (2024), as well as recent efforts to enhance the interpretability of KT models.

1719 F.2.1 DEEP KNOWLEDGE TRACING
1720

1721 Deep learning-based approaches like Deep Knowledge Tracing (DKT) Piech et al. (2015), Dynamic
1722 Key-Value Memory Networks (DKVMN) (Zhang et al., 2017) and Attentive Knowledge Tracing
1723 (AKT) Ghosh et al. (2020) uses neural networks or attention-based architecture to learn contextual
1724 representation of questions and student knowledge states. Despite their strong predictive performance,
1725 these models represent a student’s knowledge state as an abstract, high-dimensional latent vector,
1726 which poses significant challenges in interpretability and actionable feedback for educators.

1727 F.2.2 INTERPRETABLE KNOWLEDGE TRACING
1728

1729 Several recent works have proposed more interpretable Knowledge Tracing methods.

1730
1731 **Bayesian methods for Knowledge Tracing** Early KT approaches used structured models where
1732 learned parameters allow for direct interpretations. For example, in Bayesian KT (Corbett & Anderson,
1733 1994) and dynamic Bayesian KT (Käser et al., 2017), the latent variables learned represent the
1734 student’s evolving proficiency across knowledge concepts (KCs). Interpretable Knowledge Tracing
1735 (IKT) (Minn et al., 2022) introduces a causal probabilistic student model based on skill mastery, ability
1736 profiles, and problem difficulty. While this approach provides clearer connections between model
1737 components and predictions, interpretability remains tied to quantitative skill mastery estimates. PSI-
1738 KT (Zhou et al., 2024) similarly learns a student’s latent knowledge state over KCs via a probabilistic
1739 hierarchical state-space model and psychology-inspired learning dynamics. While these methods
1740 provide interpretable latent states, they remain limited to estimating the student’s proficiency over
1741 KCs.

1742 **IRT-based Knowledge Tracing** Other works combine deep learning architectures with classical
1743 item-response theory to make latent representation interpretable. Deep-IRT (Yeung, 2019) leverages
1744 the powerful abilities of deep learning architectures by training a DKVMN model to estimate student
1745 ability on each KC and item difficulty and predict future answers via an IRT model. QIKT (Chen et al.,
1746 2023) employs IRT functions as the final prediction layer, combining question-centric knowledge
1747 acquisition, knowledge mastery scores, and knowledge application scores through encoder modules.
1748 Despite meaningful latent representations, the diagnostic output of these methods remains constrained
1749 to proficiency estimates over KCs.

1750 **Learned Questions Relationships** HGKT (Tong et al., 2022) addresses limitations of concept-
1751 based proficiency by modeling hierarchical relationships between questions and introducing problem
1752 schemas as additional links between questions. Problem schemas are discovered through hierarchical
1753 clustering of question embeddings via BERT encodings, providing a means of grouping questions
1754 orthogonal to knowledge concepts and therefore enabling more detailed diagnostic reports than at the
1755 concept proficiency level. However, the interpretability of these learned schemas requires post-hoc
1756 interpretation using TextRank to infer schema descriptions from clusters of questions. While this
1757 approach moves beyond simple concept proficiency, it still relies on learned embeddings that cannot
1758 directly articulate student reasoning patterns without post-hoc manipulations.

1759 **Explainable subsequences** Explainable subsequences provide an alternative to interpretability
1760 via concept proficiency by identifying which past questions are most relevant for predicting future
1761 responses. For example, Li et al. (2023) proposes a genetic algorithm to identify explainable
1762 subsequences in student interaction histories better than with standard Deep Learning explanation
1763 methods such as Shapley values or gradient-based saliency maps. While this allows for a different kind
1764 of interpretability from concept mastery, the explanations remain at the level of question relevance
1765 rather than underlying reasoning processes. This approach can indicate which questions matter but
1766 cannot explain why they matter or what misconceptions they reveal.

1767 **Option Tracing** Option Tracing (Ghosh et al., 2021) moves beyond binary correctness by modeling
1768 which specific option a student selects in multiple-choice questions, enabling finer-grained analysis
1769 of misconceptions through patterns in distractor choices. Similarly, Park et al. (2024) leverages MCQ
1770 responses and concept maps to disentangle student understanding at the concept level. While their
1771 approach is motivated by misconception detection, their IRT-based predictions remain grounded in
1772 concept-level proficiency prediction and do not explicitly validate misconception identification.

1773 F.2.3 LLM-BASED KNOWLEDGE TRACING
1774

1775 Recent studies have begun integrating Large Language Models (LLMs) into the KT framework. For
 1776 example, Li et al. (2024a) demonstrated that LLMs are able to make sensible predictions about student
 1777 responses when prompted with adequate information. Other works Lee et al. (2024); Kim et al.
 1778 (2024) have studied how LLMs can help mitigate the cold-start problem compared to traditional KT
 1779 approaches, while Wang et al. (2025) demonstrated state-of-the-art performances in KT by combining
 1780 LLMs with sequence interaction models. However, these methods generally remain opaque: they
 1781 either treat the LLM as a black-box, or rely on model-generated explanations that are susceptible to
 1782 hallucination (Bender et al., 2021).

1783 KCQRL (Ozyurt et al., 2024) leverages language models to improve the embedding of any deep
 1784 learning KT method by encoding semantic information about question content, while KCD (Dong
 1785 et al., 2025) refines a cognitive diagnosis model using the general knowledge an LLM. While these
 1786 methods can improved the predictive accuracy of KT and CD models, their interpretability remains
 1787 limited to knowledge concept proficiency.

1790 F.3 COMPARISON OF LBMs TO KT AND CD
1791

1792 Table F1 gives a high-level comparison of LBMs and KT/CD models. We acknowledge that this
 1793 summary table is a simplification of the KT and CD fields, as each contains many works that have
 1794 been proposed to tackle these individual limitations (for example, using textual question contents
 1795 to improve embeddings (Ozyurt et al., 2024), or using specialized Multiple Choice Questions to
 1796 reveal misconceptions via KT-type methods(King et al., 2024)). Nevertheless, it clarifies our paper’s
 1797 similarities with these two fields, while contrasting key differences that enable LBMs to address
 1798 educational scenarios where neither traditional KT nor CD methods are well-suited —particularly for
 1799 misconception identification or when working with limited data.

1800
1801 F.4 CONCEPT BOTTLENECK MODELS
1802

1803 Concept Bottleneck Models (CBMs) Koh et al. (2020) improve interpretability through human-
 1804 understandable concept activations as intermediates, with extensions exploring unsupervised concept
 1805 learning Oikarinen et al. (2023), test-time interventions Shin et al. (2023), and theoretical analyses
 1806 of concept set design Luyten & van der Schaar (2024). However, CBMs typically rely on finite
 1807 predefined concept sets, limiting their applicability to complex tasks like knowledge tracing. Recently,
 1808 Yamaguchi & Nishida (2024) introduced Explanation Bottleneck Models (XBMs), which use textual
 1809 rationales as intermediates for vision classification, justifying a single known label per input. While
 1810 our Language Bottleneck Models (LBMs) adopt this language bottleneck concept, they differ funda-
 1811 mentally. Unlike XBMs’ instance-specific, task-specific rationales, LBM summaries aim to capture
 1812 an *implicit* knowledge states—as emphasized by our inverse problem formulation—and generalize to
 1813 future, unknown questions. These requirements necessitate holistic, adaptable summaries rather than
 1814 XBMs’ task-specific rationales, leading us to introduce Language Bottleneck Models (LBMs) as a
 1815 distinct framework with broader applicability.

1816
1817 F.5 TEXT SUMMARIZATION MODELS
1818

1819 Recent text summarization models such as BART Lewis et al. (2019), T5 Raffel et al. (2020), and
 1820 PEGASUS Zhang et al. (2020) effectively produce concise summaries based on explicitly available
 1821 textual content. However, these approaches differ fundamentally from knowledge tracing, where
 1822 summaries must infer implicit student knowledge states not directly observable in the input. Standard
 1823 summarization metrics (e.g., ROUGE, BLEU) rely on explicit reference summaries, making them
 1824 unsuitable for evaluating latent knowledge inference tasks. In contrast, our Language Bottleneck
 1825 Models generate textual summaries of the student’s implicit knowledge state, optimized for predictive
 1826 accuracy on downstream questions rather than syntactic overlap with the observed input.

1827 Table F1: Comparison of Cognitive Diagnosis, Knowledge Tracing and our proposed Language
 1828 Bottleneck Models.
 1829

Aspect	Knowledge Tracing (KT)	Cognitive Diagnosis (CD)	Language Bottleneck Models (LBMs)
<i>Interpretability Capabilities</i>			
Interpretable Output	Quantitative proficiency across knowledge concepts	Quantitative proficiency across knowledge concepts	Qualitative text summaries
Misconception Detection	No	No	Yes
Requires Predefined or Inferred Concepts for Interpretability	Yes	Yes	No
<i>Data Requirements</i>			
Primary Input Modality	Question and/or Construct IDs	Question and Construct IDs (Q-matrix)	Any textual information
Training Data Requirements	High	High	Low/Zero-shot
<i>Modeling Characteristics</i>			
Knowledge State Assumption	Dynamic	Static	Static
Human-in-the-Loop	Limited	Limited	High (steerable & editable)
<i>Fundamental Distinctions</i>			
Core Question Addressed	“Can we predict a student’s responses over time?”	“Can we quantitatively estimate a student’s proficiency across concepts?”	“Can we qualitatively estimate a student’s knowledge state, including knowledge concepts and misconceptions?”
Case Study (Figure 3)	<i>Similar to CD</i>	<i>Proficiency vector: KC1 (Add): 0.59, KC2 (Sub): 0.53, KC3 (Mul): 0.76, KC4 (Div): 0.23</i>	<i>“...excels at addition and multiplication... Subtraction is a weakness... Multiplication by 6 or 7 seems to be a specific area of difficulty...”</i>

G EXTENDED DISCUSSION

G.1 EXTENDED DISCUSSION

1871 **Parallel with Kolmogorov Complexity** The inverse problem formulation of Knowledge State
 1872 Modeling draws a natural parallel with algorithmic information theory. The Kolmogorov complexity
 1873 $K(x)$ of a string x is defined as the length of the shortest program that outputs x when run on a
 1874 universal Turing machine. Traditional KT methods focus on learning statistical regularities between
 1875 questions and responses, without looking for parsimonious explanations for observed behavior. In
 1876 contrast, LBMs explicitly search for minimal natural language descriptions that can both reconstruct
 1877 past interactions and predict future responses. The knowledge state summary \mathcal{S} is akin to the minimal
 1878 program, and the decoder LLM functions is akin to the universal interpreter that unpacks \mathcal{S} to generate
 1879 the observed question-answer patterns. This connection suggests that effective knowledge state
 1880 summaries should capture the algorithmic essence of student behavior—the underlying “program”
 of knowledge and misconceptions that generates observable responses—rather than merely fitting

1881 surface-level patterns. While true Kolmogorov complexity is uncomputable, LBMs approximate
 1882 this ideal through the natural language bottleneck constraint, encouraging summaries that balance
 1883 compression with predictive power.
 1884

1885 **Computational Architecture and Design Choices** The encoder-decoder architecture reflects
 1886 a deliberate separation of concerns: extraction versus interpretation of knowledge states. Our
 1887 experiments demonstrate that these two tasks have different intrinsic complexity, with encoding
 1888 (summary generation) being more challenging than decoding (prediction from summaries). This
 1889 finding has practical implications for model selection and computational resource allocation. The use
 1890 of GRPO for encoder training represents a novel application of reinforcement learning to interpretable
 1891 AI, where the reward signal directly measures the downstream utility of explanations rather than their
 1892 surface-level quality.
 1893

1894 G.2 EXTENDED LIMITATIONS

1895 **Context Length Constraints** A practical limitation of LBMs is the context length restrictions of
 1896 current LLMs. Even though modern models can handle thousands of tokens, comprehensive student
 1897 histories in real educational settings can easily exceed these limits. As student trajectories grow more
 1898 complex and include more information, the context required for the encoder might go beyond what
 1899 current LLMs are capable of. A natural solution could be an iterative encoding process, where the
 1900 encoder iterates over the text bottleneck while going over the input data by windows.
 1901

1902 **Computational Cost and Resource Requirements** Since LBMs involve two LLMs as the encoder
 1903 and decoder, they are typically more computationally intensive to run than traditional KT methods.
 1904 Each inference call with LBMs requires two LLM calls (encoding and decoding) with extensive
 1905 context windows. Training LBMs with GRPO is particularly costly, as it requires iteratively generating
 1906 and evaluating multiple candidate summaries per training example. This cost structure may limit the
 1907 practical deployment of LBMs to high-stakes educational contexts where the interpretability benefits
 1908 justify the computational expense.
 1909

1910 **Dependence on Textual Question Content** Many existing CD/KT datasets provide only question
 1911 identifiers rather than full question text. This limits the direct applicability of LBMs, as they require
 1912 the questions content in order to generate meaningful summaries about student knowledge. While
 1913 question content is increasingly available in modern educational platforms, this dependency creates a
 1914 barrier to applying LBMs to historical datasets or systems that rely primarily on item response theory
 1915 frameworks.
 1916

1917 **Constant Knowledge State Assumption** The constant knowledge state assumption underlying the
 1918 inverse problem formulation, while reasonable for short diagnostic sessions, does not cover longer
 1919 time horizons. Real learning involves continuous knowledge acquisition, forgetting, and misconception
 1920 evolution that our current framework cannot capture. This limitation restricts current LBMs to
 1921 diagnostic rather than contexts of assessment throughout the learning process. The assumption also
 1922 fails to account for contextual factors (fatigue, motivation, external stressors) that can significantly
 1923 impact student performance within even short sessions.
 1924

G.3 EXTENDED FUTURE WORK

1925 **Iterative Encoding Strategies** A natural extension to address context length limitations involves
 1926 iteratively applying the encoder while chunking input data. At each step, the encoder would process
 1927 a portion of the interaction history X along with one or more previously generated bottleneck
 1928 summaries, creating an updated summary that incorporates new information from the latest chunk.
 1929 This approach could maintain both detailed recent context and compressed historical patterns, enabling
 1930 LBMs to handle arbitrarily long student trajectories through sequential refinement of knowledge state
 1931 representations.
 1932

1933 **Active Learning and Question Selection** Building on iterative encoding, active learning strategies
 1934 could provide encoders with the most informative input questions. Rather than processing all available
 1935 interactions, the system could strategically select questions that maximize information gain about

1935 uncertain aspects of student knowledge. This could leverage uncertainty quantification methods or
 1936 use the encoder LLM itself to recommend the most diagnostically valuable questions. Such active
 1937 sensing would improve both efficiency and diagnostic power by focusing computational resources on
 1938 the most insightful student responses.

1939

1940 **Dynamic Knowledge State Modeling** Extending LBMs to handle evolving knowledge states
 1941 represents a significant next step necessary to account for progressive learning and forgetting effects.
 1942 A natural relaxation of the constant knowledge state assumption involves "piecewise-constant" knowl-
 1943 edge states that can evolve between question sessions but remain static within sessions. This extension
 1944 poses exciting challenges in developing training objectives that balance within-session consistency
 1945 with between-session learning dynamics, potentially requiring new approaches to temporal modeling
 1946 in natural language representations.

1947

1948 **Multimodal Input Integration** Question-answer pairs represent a relatively narrow information
 1949 source for inferring student knowledge states. Richer data sources such as student-tutor interactions,
 1950 self-reported explanations for answers, timing patterns, or hint usage could provide deeper insights
 1951 into student understanding. While LBMs naturally extend to text-based inputs, future work should
 1952 investigate optimal strategies for combining these varied data sources and evaluate their relative
 1953 contributions to knowledge state inference accuracy and interpretability.

1954

1955 **Pedagogical Alignment** Recent work like LearnLM Team et al. (2024) demonstrates the potential
 1956 for making LLMs more pedagogically aligned. Incorporating similar pedagogical principles into
 1957 LBM components could help encoders better interpret educational interactions and generate sum-
 1958 maries that align with expert teaching practices. This might involve training on educator-annotated
 1959 examples, incorporating educational taxonomies into summary structure, or using reward functions
 1960 that emphasize pedagogically relevant aspects of student knowledge states. Such alignment could
 1961 bridge the gap between computational convenience and educational validity.

1962

STATEMENT: USE OF LARGE LANGUAGE MODELS

1963

1964 Large Language Models (LLMs) were used as an assistive tool in the preparation of this manuscript
 1965 and the development of the accompanying code.

1966

- LLMs were used to improve the clarity, grammar, and flow of the text. This included rephrasing sentences, correcting typographical errors, and ensuring a consistent tone.
- LLMs were used to generate boilerplate code for data processing scripts, assist in debugging, and suggest implementations for standard machine learning components.

1967

The authors take full responsibility for the final content of this paper.

1968

1969

1970

1971

1972

1973

1974

1975

1976

1977

1978

1979

1980

1981

1982

1983

1984

1985

1986

1987

1988