INFERRING THE INVISIBLE: NEURO-SYMBOLIC RULE DISCOVERY FOR MISSING VALUE IMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

One of the central challenges in artificial intelligence is reasoning under partial observability, where key values are missing but essential for understanding and modeling the system. This paper presents a neuro-symbolic framework for latent rule discovery and missing value imputation. In contrast to traditional latent variable models, our approach treats missing grounded values as latent predicates to be inferred through logical reasoning. By interleaving neural representation learning with symbolic rule induction, the model iteratively discovers—both conjunctive and disjunctive rules—that explain observed patterns and recover missing entries. Our framework seamlessly handles heterogeneous data, reasoning over both discrete and continuous features by learning soft predicates from continuous values. Crucially, the inferred values not only fill in gaps in the data but also serve as supporting evidence for further rule induction and inference—creating a feedback loop in which imputation and rule mining reinforce one another. Using coordinate gradient descent, the system learns these rules end-to-end, enabling interpretable reasoning over incomplete data. Experiments on both synthetic and real-world datasets demonstrate that our method effectively imputes missing values while uncovering meaningful, human-interpretable rules that govern system dynamics.

1 INTRODUCTION

Neural-symbolic reasoning combines the pattern recognition power of neural networks with the precision and interpretability of symbolic reasoning (Hitzler & Sarker, 2022; Yang et al., 2024). This hybrid paradigm enables AI systems to detect complex patterns in unstructured data while reasoning about them in a structured and explainable manner.

Traditional rule induction methods extract explicit patterns from observed data but often fail when *some observations are missing or incomplete* (Campero et al., 2018; Claire Glanois, 2022). These approaches can effectively learn surface-level rules, yet their ability to fully explain the underlying system is limited when essential data points are absent. For example, in healthcare diagnostics, critical measurements may be missing or noisy, making accurate imputation necessary for reliable reasoning.

Probabilistic models such as Markov Logic Networks (MLNs) (Richardson & Domingos, 2006) handle missing data by treating unobserved facts as latent predicates. However, they typically rely on a *fixed* rule base and *expensive joint inference*, limiting scalability and adaptability in large or heterogeneous datasets (Oltramari et al., 2020). In contrast, we propose a neuro-symbolic system that *co-learns rules and imputations* in a single differentiable loop, enabling fast forward-chaining inference and end-to-end learning.

Our core idea is *a closed loop between imputation and rule discovery*. Given partially observed tables with discrete and continuous attributes, we treat each missing, entity-specific entry as an unknown fact and apply learned rules in a *forward-chaining* pass to predict it. These predictions are compared to the observed entries via a supervised loss, and backpropagation updates the rule parameters and soft predicates. Crucially, improved imputations provide additional evidence for discovering and refining rules in subsequent passes. This self-reinforcing loop leads to better imputations, improved rule induction, and stronger downstream inference.

To enable multi-hop reasoning at scale, many targets require compositional explanations in the form of chains and disjunctions. We optimize rule embeddings using *asynchronous coordinate gradient descent*, updating one rule or clause at a time while holding others fixed. This mirrors step-wise reasoning and ensures monotone loss progress on a smooth surrogate. For disjunctive heads, we adopt a sequential covering strategy to harvest diverse clauses, followed by joint fine-tuning using a soft-OR aggregator (LogSumExp) to reconcile interactions. This staged procedure reliably recovers long chains and disjunctive theories under high missingness while keeping computation tractable.

Our framework handles heterogeneous data by learning *soft predicates* for continuous features (using sigmoid thresholds and slopes) and combining them with discrete predicates through differentiable logical operators. Specifically, we use soft-min to approximate logical AND and soft-max to approximate logical OR. This approach enables uniform forward chaining over mixed data types without requiring pre-discretization.

Contributions. We summarize our contributions as follows: (*i*) We introduce a closed-loop neuro-symbolic framework in which imputation and rule discovery mutually reinforce each other, rather than treating imputation as a preprocessing step. (*ii*) We develop a scalable coordinate gradient descent scheme, combined with sequential covering and joint fine-tuning, that enables multi-hop and disjunctive rule learning even under high missingness. (*iii*) We design a unified differentiable forward-chaining engine that handles both discrete and continuous attributes through soft predicates and smooth logical operators. (*iv*) We empirically validate our approach on synthetic chain and disjunction tasks, as well as real-world datasets (Birds, Heart, SPECT), demonstrating that it recovers human-interpretable rules while achieving strong imputation accuracy and downstream prediction performance.

2 Related Work

Our work is at the intersection of neuro-symbolic Inductive Logic Programming (ILP) and missing value imputation.

Neural Embedding-based ILP. Embedding-based models are widely used for Knowledge Base (KB) completion like TransE (Bordes et al., 2013), TransH (Wang et al., 2014), and TransR (Lin et al., 2015). Complex (Trouillon et al., 2016) introduces complex-valued embeddings for asymmetric relations, while multi-hop reasoning methods like Guu et al. (2015) leverage path-based embeddings for traversing knowledge graphs. However, these approaches often face limitations in reasoning power.

Recent advances in ILP integrate symbolic logic with neural networks. Rocktäschel & Riedel (2017) propose *Neural Theorem Proving (NTP)*, which uses a differentiable backward-chaining method. Then, Campero et al. (2018) introduces a neural forward-chaining differentiable rule induction network. However, both rely on hand-designed templates. Claire Glanois (2022) advances these models by incorporating a hierarchical structure, enabling more flexible rule induction. Nevertheless, these methods are primarily designed for fully-observed data and struggle to handle missing values.

Interpretable Rule Learning. Learning interpretable logical rules for classification has been a long-standing goal. Dash et al. (2018) propose **BRCG**, an integer programming approach that uses column generation to efficiently search the exponential space of candidate clauses, explicitly balancing classification accuracy with rule simplicity. Wang et al. (2021) introduce **RRL**, which utilizes a Gradient Grafting mechanism to learn non-fuzzy rule lists within a deep learning framework, ensuring scalability. Qiao et al. (2021) propose **DR-NET** to learn independent decision rules in Disjunctive Normal Form (DNF) by jointly optimizing rule generation and weight learning. More recently, Barbiero et al. (2022) present **LEN**, an end-to-end differentiable neuro-symbolic method that leverages an entropy-based criterion to extract concise First-Order Logic explanations from neural networks. Unlike these methods, which focus primarily on classification tasks with complete data with binary features, our framework integrates rule learning directly with the handling of missing values.

Rule-Based Missing Value Imputation. Traditional missing data imputation methods, ranging from statistical techniques like **MICE** (Multivariate Imputation by Chained-Equations) (van Buuren & Groothuis-Oudshoorn, 2011), **MissForest** (Random Forest based) (Stekhoven & Bühlmann, 2012), and SOFT-IMPUTE (Mazumder et al., 2010) to deep learning models like **GAIN** (GAN-based) (Yoon et al., 2018), **MissDiff** (Diffusion-based)(Ouyang et al., 2023), **mDAE** (DAE-based)(Dupuy et al., 2024), **VAE-based**(Veldkamp et al., 2025) and MMDL (Li et al., 2020), typically rely on statistical patterns and do not leverage explicit logical rules to govern inter-variable relationships (see Appendix A for a detailed overview).

Recent works have started to bridge rule-based reasoning and missing value imputation. For instance, Chen et al. (2023) employ various interpretable machine learning techniques to address the missing value problem, but their methods are not explicitly rule-based. Closer to our approach, MINTY (Stempfle & Johansson, 2024) utilizes a rule-based model to handle missing data; however, it does not leverage neuro-symbolic reasoning to learn the intricate relationships between observed and missing values as we do. Other non-neural approaches, such as the work by Wang et al. (2017) on synthesizing data completion, also tackle the problem but lack of the representation learning capabilities of neural networks. Our work is distinct in its tight integration of neural learning for representation and symbolic reasoning for both rule discovery and imputation, forming a feedback loop where each component enhances the other.

3 Background

Predicate. In the context of logic-based AI systems, a predicate is a fundamental Boolean logic variable used to describe properties of or relationships between entities. Predicate variables are grounded by data, being True or False, and serve as the basic building blocks for logical expressions. For instance, a predicate like $Has\_Fever(Patient)$ denotes whether a patient has a fever, while $Use\_Drug(Patient)$ specifies whether a drug treats a particular patient. These predicates capture essential aspects of the system's state and relationships.

Logic Rules and Forward Chaining. We represent knowledge with **Horn clauses**

$$f: \quad Q \;\leftarrow\; P_1 \wedge P_2 \wedge \cdots \wedge P_h, \tag{1}$$

where $P_1, \ldots, P_h$ (the *body*) are conditions and Q (the *head*) is the conclusion. Given observed facts (the evidence set $\mathcal{E}$), we perform *forward chaining*: whenever all body predicates of a rule are (approximately) satisfied by facts in $\mathcal{E}$, the rule *fires* and adds Q to $\mathcal{E}$. Importantly, newly inferred facts are *immediately recycled as evidence*, enabling *multi-hop reasoning*—cascades of rule applications that derive conclusions not reachable in a single step.

Latent Predicates and Rule Learning. We use the term *latent predicate* to denote an unobserved fact tied to concrete entities (and, when relevant, timestamps) within the same relational schema as observed predicates. Latent predicates may be Boolean or soft-valued (degrees of truth); they represent missing-but-specific facts we wish to infer. Our goal is to learn *Horn rules* of the form Eq. (1) that capture regularities among observed predicates and support inference about latent ones—i.e., rules whose heads or intermediate conclusions may involve latent predicates, enabling principled completion of missing facts.

Expressive Rule Forms. We consider rules that capture rich logical structure, including conjunctions (AND), disjunctions (OR via multiple clauses), and *chained dependencies*. For example, a latent predicate Q_k may be characterized by

$$Q_k \;=\; (P_1 \wedge P_2) \;\vee\; (P_3 \wedge P_4),$$

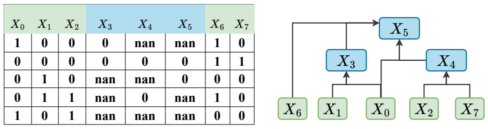
or by multi-hop compositions such as

$$Q_1 = P_1 \wedge P_2, \quad Q_2 = P_3 \wedge P_4, \quad Q_3 = (Q_1 \wedge P_5) \;\vee\; (Q_2 \wedge P_6).$$

This view accommodates both single-step and multi-step (multi-hop) reasoning patterns within a unified Horn-rule framework. We also allow *predicate invention*: introducing unlabeled latent predicates that are not predefined in the schema but are useful intermediates for explaining the data. These invented predicates participate in rules just like observed ones. After rules are discovered, their roles can be *post-hoc interpreted* by inspecting the clauses in which they appear and their relationships to observed predicates.

4 Model: Neuro-Symbolic Forward Chaining Network

Consider problems where some information or features are incomplete. Our goal is to learn a set of logical rules that explain how each predicate with information can be imputed based on evidence from feature space $\mathbf{X}$.



X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
1	0	0	0	nan	nan	1	0
0	0	0	0	0	0	1	1
0	1	0	nan	nan	0	0	0
0	1	1	nan	0	nan	1	0
1	0	1	nan	nan	nan	0	0

Figure 1: Example of missing variables imputation with rule discovery. X_i with *nan* is the predicates with missing information, which can be inferred by the logic rules from $\mathbf{X}$.

These missing variables are inferred through a rule-learning process, allowing the model to uncover hidden relationships in the data. For clarity, we identify the predicates with missing information as $\mathbf{U}$, also named as "latent predicate" in our setting. Though in our experiments, we do not strictly distinguish between feature predicates, as any of them can be incomplete and serve as latent predicates. In more general settings with a predictive label Y, we can view Y as one of the latent predicates, making the rule learning and prediction for Y equivalent to inferring latent predicates $\mathbf{U}$ with rules.

To summarize, our model learns logical rules to infer latent predicates $\mathbf{U}$ by discovering hidden structures within data, as an example illustrated in Figure 1. This rule induction process identifies logical relationships among observable predicates $\mathbf{X}$ and other inferred latent predicates. By explicitly learning these structures, our approach enhances both inference capability and interpretability, offering clear insights into complex, otherwise hidden dependencies. The key idea is summarized in Figure 2, with details presented in the following sections.

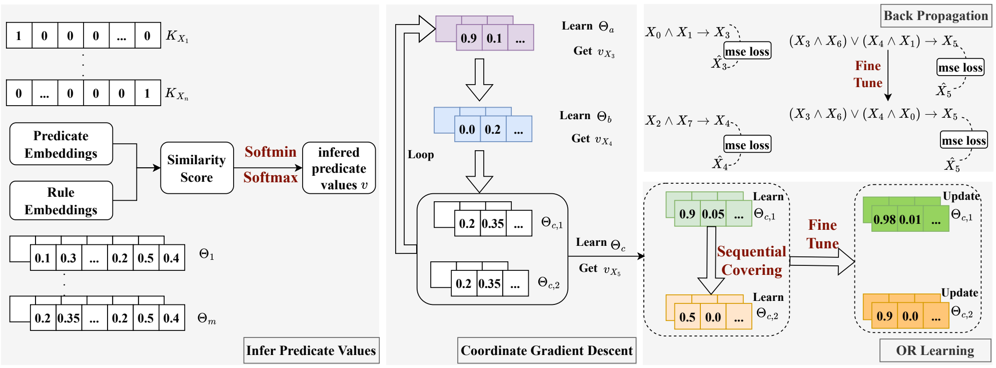


Figure 2: Model framework. Rule embeddings Θ are optimized using coordinate gradient descent. In each learning step, predicate values are inferred via the Softmin-Softmax operation (Eqs. (3, 4,5)). For disjunctive (OR) rule learning, sequential hard covering is applied, followed by fine-tuning of the learned rule embeddings (Section 5.2). Errors are back-propagated using MSE loss between inferred predicate values and the small portion of observed latent predicate samples, constituting a weak-supervision setting.

4.1 Model Preparation: Pretrained Predicate Embeddings

We begin by defining two sets of predicates: $\mathbf{X} = \{X_1, \ldots, X_n\}$ represents the set of *observable* predicate variables, and $\mathbf{U} = \{U_1, \ldots, U_m\}$ denotes the set of predicate variables with *missing information* that the model aims to discover and define. Our framework is designed to handle **both binary (categorical) and continuous features within a unified logical structure**. Binary features are treated as standard logical predicates. For continuous features, we introduce a mechanism to derive a "soft" truth value, effectively creating learnable predicates from them. This allows the model to reason over heterogeneous data types, as detailed in Section 4.2.

As mentioned before, we do not distinguish $\mathbf{X}$ and $\mathbf{U}$ in the experiment, as any predicates can be the predicate with missing information. We just use separate notations for model description. We initialize a fixed, unique embedding for each predicate, whether observable or missing. For example, these embeddings can be instantiated as one-hot vectors within an embedding space of dimension d. We denote the collection of embeddings for observable predicates as $\mathbf{K}_X$ and incomplete predicates as $\mathbf{K}_U$. These predicate embeddings remain frozen throughout the rule learning phase and serve as a foundational dictionary, enabling the interpretation of the composition of learned rules by relating rule components back to specific predicates.

With the predicate representations defined, we next describe the core of our model: the representation of logical rules and the mechanism by which inferences are drawn.

4.2 Model Backbone: Rule Representation and Inference

In our NS-FCN framework, logical rules are materialized as learnable rule embeddings, which are the primary trainable parameters. Our model employs an asynchronous coordinate descent learning process. This learning scheme is particularly well-suited for discovering complex logical structures such as chained dependencies (where one latent predicate forms part of the definition of another) and disjunctive rules (where a latent predicate can be satisfied by one of several distinct conditions).

4.2.1 Specification of Rule Embeddings Θ

Let $\mathcal{F}$ be the set of rules/clauses, and let $\Theta = \{\Theta_f\}_{f \in \mathcal{F}}$ be their embeddings. Each Θ_f encodes one rule with head predicate U_j, and a single head predicate U_j may be associated with multiple rules (OR-of-ANDs).

Conjunctive Rule Embedding. For a latent predicate U_j that is defined by a single conjunctive rule (e.g., $U_j = X_a \wedge X_b$), its corresponding rule embedding $\Theta_f = [\theta_1, \ldots, \theta_h] \in \mathbb{R}^{d \times h}$. Here, h represents the number of predicates forming the body of the conjunctive rule (the arity of the conjunction, e.g., $h = 2$ for $X_a \wedge X_b$), and d is the dimensionality of the predicate embeddings. Each of the h rows in this matrix is learned to align with the embedding of one of the constituent predicates in the rule's body.

Disjunctive Rule Embeddings. If a latent predicate U_k is defined by a disjunction of R_k distinct conjunctive clauses (e.g., $U_k = \bigvee_{r=1}^{R_k}(\text{clause}_r)$), it will be associated with a set of R_k distinct rule embeddings, denoted $\{\Theta_{k,1}, \ldots, \Theta_{k,R_k}\}$. Each individual rule embedding $\Theta_{k,r}$ is itself an $h_r \times d$ matrix, representing the r-th conjunctive clause, where h_r is the arity of that specific clause.

All rule embeddings are initialized randomly prior to training and are subsequently optimized as described in Section 5. Given these rule embeddings, the model infers the truth values (or continuous approximations thereof) of latent predicates through a carefully defined inference mechanism.

Parameters for Continuous Predicates. For each continuous feature $f \in \mathcal{F}_C$, where $\mathcal{F}_C$ is the set of continuous features, the model learns two additional scalar parameters: a threshold θ_f and a slope β_f. These parameters are used to define a learnable soft predicate function that maps the continuous feature value to a probabilistic truth value, as explained next.

4.2.2 Inferring Predicate Values

The latent predicates is inferred based on the current state of observable predicates, any previously inferred latent predicate values, and the learned rule embeddings Θ.

Predicate Matching. Each column $\theta_j (j = 1, \ldots, h)$ in the rule embedding Θ_f is matched with a corresponding predicate embedding. This matching is achieved by finding the predicate embedding most similar to θ_j using cosine similarity:

$$K_j^* = \operatorname*{argmax}_{K \in \boldsymbol{K}} \cos\left(K, \theta_j\right), \quad j = 1, \ldots, h \tag{2}$$

where $\boldsymbol{K} = \boldsymbol{K}_X \cup \boldsymbol{K}_U$ represents the set of all available predicate embeddings. The inverse mapping $I(K)$ maps a predicate embedding $K \in \mathbb{R}^d$ back to its corresponding index. Thus, indices $1, \ldots, (n+m)$ correspond to $n+m$ predicate embeddings.

Predicate Truth Values. Once the best matching predicate K_j^* is identified for a rule component θ_j, we determine its truth value, denoted as t_j. The calculation depends on whether the corresponding feature is binary or continuous:

1) For a *binary feature* (e.g., from one-hot encoding), its truth value is its current value in the data tensor: $t_j = \boldsymbol{v}^t\left(I\left(K_j^*\right)\right)$.

2) For a *continuous feature*, its truth value is computed using a learnable **soft predicate** function (a sigmoid): $t_j = \sigma(\beta_{f_j} \cdot (v_{f_j} - \epsilon_{f_j}))$ where v_{f_j} is the value of the feature corresponding to K_j^* (i.e. $v_{f_j} = \boldsymbol{v}^t(I(K_j^*)))$, ϵ_{f_j} and β_{f_j} are its learned parameters, and $\sigma(\cdot)$ is the sigmoid function. This allows learning soft boundaries like "$v_{f_j} > \epsilon_{f_j}$".

Conjunctive Clause Inference (Soft-AND). The value for a conjunctive clause is then computed by aggregating the contributions of all its components, modeling a Soft-AND operation. The contribution of each component j is the product of its similarity score and its truth value. The aggregated value is:

$$v = \prod_{j=1,\ldots,h} \cos\left(K_j^*, \theta_j\right) \cdot t_j, \tag{3}$$

where $\boldsymbol{v}^t$ is the current value for observable predicates or any previously imputed values. At the beginning, $\boldsymbol{v}^t$ is all from observable predicates. With the optimization steps of coordinate descent, $\boldsymbol{v}^t$ is updated based on the refined Θ.

To address the potential issue of diminishing values, we can use the min function instead: $v = \min_{j=1,\ldots,h}\left\{\cos\left(K_j^*, \theta_j\right), t_j\right\}$.

However, to make this function *differentiable*, we approximate the min function using the softmin function. For each component j, there are two terms: the similarity score $\cos\left(K_j^*, \theta_j\right)$ and the truth value t_j. The softmin is applied to the set of all $2h$ such terms:

$$\operatorname{softmin}\left(x_1, \ldots, x_{2h}; \Theta\right) = -\frac{1}{\tau} \log\left(\frac{1}{2h} \sum_{i=1}^{2h} e^{-x_i/\tau}\right) \tag{4}$$

where each x_i represents one of the $2h$ terms (all similarity scores and all truth values), and τ is a temperature parameter controlling the smoothness of the approximation. As τ approaches 0, the softmin function approximates the behavior of the hard min function.

Disjunctive Rule Inference (Soft-OR). When a latent predicate U_k is defined by a disjunction of multiple conjunctive clauses, $U_k = \bigvee_{r=1}^{R_k} \text{clause}_{k,r}$, its final inferred value v_{U_k} is determined by aggregating the values of its individual clauses $\{v_{\text{clause}_{k,1}}, \dots, v_{\text{clause}_{k,R_k}}\}$. This aggregation is performed using the LogSumExp (LSE) function, which serves as a differentiable soft-OR operator:

$$v_{U_k} = \frac{1}{\beta} \log \sum_{r=1}^{R_k} \exp(\beta \cdot v_{\text{clause}_{k,r}}), \tag{5}$$

where β is a temperature parameter. As $\beta \to \infty$, the LSE function increasingly approximates the true $\max$ operator, thereby hardening the OR logic. Conversely, smaller values of β yield a softer aggregation. The model's ability to discover meaningful rules and infer latent predicate states accurately hinges on an effective learning procedure. We now outline the training methodology employed to optimize the rule embeddings Θ.

5 Model Learning

The core of our model learning process involves training the rule embeddings Θ by minimizing a loss function that quantifies the discrepancy between the inferred values of latent predicates and their partially observed truth values. Our approach leverages a sequential and staged optimization strategy, drawing parallels with coordinate descent and incorporating elements of rule covering, particularly for disjunctive rules. This is typically followed by a joint fine-tuning phase for rules involving disjunctions.

5.1 Coordinate Gradient Descent for Rule Optimization

We employ a block coordinate gradient descent approach, iteratively optimizing the embedding Θ_j for each predicate U_j (treated as a disjoint parameter block) while holding the embeddings of other predicates fixed. The order in which predicates U_j are selected for optimization is randomized in each complete pass (cycle) through all learnable latent predicates. Such optimization progress is similar to human thinking strategy, as we humans usually draw conclusions step by step.

During the optimization step for a specific predicate U_j within a cycle, the inferred value v_{U_j} is obtained by Eq. 3 or Eq. 4 as mentioned in the previous Section. The Mean Squared Error (MSE) loss is computed between the inferred value v_{U_j} and its observed value $U_{j,\text{obs}}$, exclusively for instances where U_j is observed, which can be viewed as a **weak supervision** setting:

$$\mathcal{L}_{U_j} = \text{mean}((v_{U_j} \odot \text{mask}_j - U_{j,\text{obs}} \odot \text{mask}_j)^2), \tag{6}$$

where mask_j is a binary vector indicating observed instances of U_j ($\text{mask}_j = 1$ indicates the observation). The rule embedding Θ_j is then updated using gradients from this loss while all other blocks are kept fixed, which implements a Gauss–Seidel block coordinate gradient method on the smooth objective $\mathcal{L}(\Theta) = \sum_j \mathcal{L}_{U_j}(\Theta)$. A brief convergence discussion is provided in Appendix B.

After its training epochs within a cycle, if Θ_j meets the criteria for a "perfect rule" (i.e., the imputation accuracy of missing variables is larger than 0.99 and a marginal loss drop is less than 10^{-3}), the parameters of Θ_j will be frozen for efficient computing in subsequent cycles.

5.2 Sequential Covering and Fine-tuning of Disjunctive Rules

Sequential Covering. When a latent predicate U_k is hypothesized to be formed by a disjunction of multiple clauses (e.g., $U_k = \text{clause}_{k,1} \vee \text{clause}_{k,2} \vee \cdots \vee \text{clause}_{k,R_k}$), its constituent rule embeddings $(\Theta_{k,1}, \Theta_{k,2}, \dots, \Theta_{k,R_k})$ are learned in a sequential manner. This iterative procedure—training a rule embedding for a clause and then conceptually "covering" the samples it explains—is repeated for all R_k rule clauses intended for the disjunctive predicate U_k.

The process begins by training the first rule embedding, $\Theta_{k,1}$, to capture one set of conditions that satisfy U_k. The inferred value $v_{\text{clause}_{k,1}}$ is computed, and the loss $\mathcal{L}_{\text{clause}_{k,1}}$ (as per Eq. 6) is minimized against the partially observed $U_{k,\text{obs}}$.

The learning of multiple rule clauses for a predicate U_k proceeds sequentially. After an initial clause, $\Theta_{k,1}$, is trained to a point where it effectively explains a subset of positive instances for U_k, a hard covering step is employed. Specifically, training instances are considered "well-explained" if the output of $\Theta_{k,1}$ (i.e., $v_{\text{clause}_{k,1}}$) for these instances exceeds a high confidence threshold (e.g., 0.99). These "well-explained" instances are then removed from the active training set. The training of $\Theta_{k,1}$

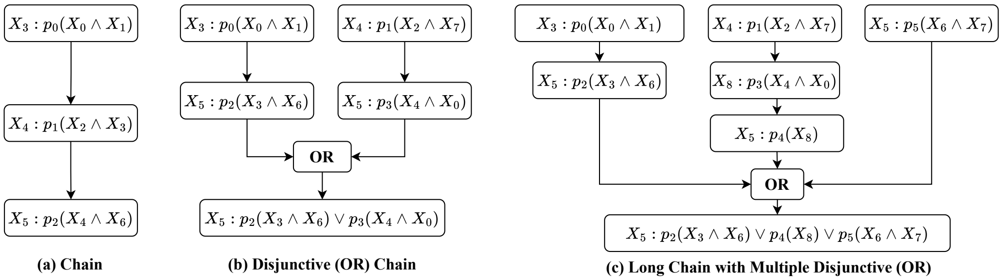
concludes at this stage, and the subsequent rule clause $\Theta_{k,2}$ is then trained on the remaining, unexplained instances of U_k. This iterative hard covering approach encourages further clause discovery of distinct rules that satisfy U_k.

Joint Fine-tuning of Disjunctive Rules. After the individual rule clauses for a disjunctive predicate U_k have been initialized through the sequential training and covering strategy, a joint fine-tuning phase is employed to refine these rules collectively. In this phase, the optimizer simultaneously updates all associated rule embeddings $\{\Theta_{k,1}, \ldots, \Theta_{k,R_k}\}$ for U_k. The MSE loss is computed between the combined soft-OR output v_{U_k} (obtained using Eq. 5, which aggregates the evidence from all R_k clauses) and the observed values $U_{k,\text{obs}}$. Given that latent predicates are, by definition, not always directly measurable, this MSE is calculated based on the small fraction of instances where the true state of the hidden predicate U_k is actually observed in the training data, which is a weakly supervised scenario: $\mathcal{L}_{U_k,\text{finetune}} = \text{mean}((v_{U_k} \odot \text{mask}_k - U_{k,\text{obs}} \odot \text{mask}_k)^2)$.

The optimization details, including Adam optimizer parameters and rule embedding normalizations, are illustrated in the Appendix C.1.

6 Experiments

6.1 Synthetic Data Experiments


$X_3 : p_0(X_0 \wedge X_1)$
$X_4 : p_1(X_2 \wedge X_3)$
$X_5 : p_2(X_4 \wedge X_6)$
$X_3 : p_0(X_0 \wedge X_1)$
$X_4 : p_1(X_2 \wedge X_7)$
$X_5 : p_2(X_3 \wedge X_6)$
$X_5 : p_3(X_4 \wedge X_0)$
OR
$X_5 : p_2(X_3 \wedge X_6) \vee p_3(X_4 \wedge X_0)$
$X_3 : p_0(X_0 \wedge X_1)$
$X_4 : p_1(X_2 \wedge X_7)$
$X_5 : p_5(X_6 \wedge X_7)$
$X_5 : p_2(X_3 \wedge X_6)$
$X_8 : p_3(X_4 \wedge X_0)$
$X_5 : p_4(X_8)$
OR
$X_5 : p_2(X_3 \wedge X_6) \vee p_4(X_8) \vee p_5(X_6 \wedge X_7)$

(a) Chain

(b) Disjunctive (OR) Chain

(c) Long Chain with Multiple Disjunctive (OR)

Figure 3: Example rule structures of synthetic experiments.

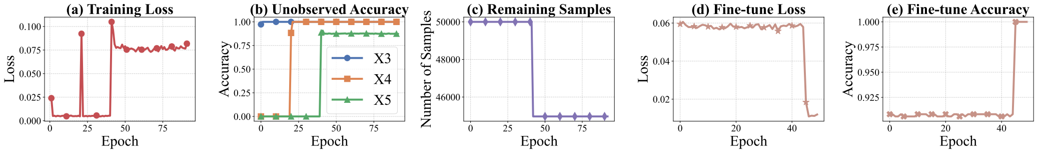


Figure 4: An example of loss and imputation accuracy during coordinate optimization (Obs. Ratio = 0.2, seed = 42). We assume the training order is X_3, X_4, X_5. Epochs 0–19 correspond to rule learning for X_3; epochs 20–39 for X_4; and epochs 40-end for X_5. Remaining samples identified how many samples are "well-explained" during the hard covering phase.

Table 1: Results for synthetic data example Figre 3(b) with an observation ratio of 0.2. Metrics are averaged over 20 random seeds on a dataset of 50,000 samples. Ground truth rules are underlined.

	Imp. Acc. (Before FT)	**Imp. Acc. (After FT)**	**Train Loss (Before FT)**	**Train Loss (After FT)**	**Learned Rules**	**Rule Acc.**
X_3	1.00 ± 0.000	/	0.005 ± 0.000	/	$\underline{X_0 \wedge X_1}$	1.00
X_4	0.95 ± 0.010	/	0.041 ± 0.005	/	$\underline{X_2 \wedge X_7}$, $X_0 \wedge X_7$, X_2	0.80
X_5	0.93 ± 0.003	0.96 ± 0.002	0.063 ± 0.003	0.067 ± 0.001	$\underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}$ $\underline{(X_3 \wedge X_4) \vee (X_3 \wedge X_6)}$ $(X_0 \wedge X_1) \vee (X_0 \wedge X_4)$	0.40

We use synthetic datasets to evaluate our model's ability to learn chained and disjunctive rules under partial observability (Figure 3). Each dataset is built from observable Bernoulli variables, with missing predicates defined by ground truth rules and made partially available (10%-30% observability) under an MCAR setting. The task is to learn rule embeddings that capture the ground truth logic, evaluated by *Rule Discovery Accuracy* (i.e. the proportion of runs which learn the truth rules) and *Imputation Accuracy*. Our method is also robust to MAR and MNAR mechanisms (Appendix E).

Table 2: Ablation Study: Effect of Fine-tuning on X_5 (Disjunctive Rule) Learning

Metric for X_5	Before Fine-tuning	After Fine-tuning
Recovered Rule Structure	$(X_0 \wedge X_2) \vee (X_0 \wedge X_4)$	$(X_3 \wedge X_6) \vee (X_0 \wedge X_4)$
Imputation Accuracy for X_5 (Unobserved)	0.8729	1.0

Results and Analysis. We analyze example (b) from Figure 3 (full results of observation ratio at 0.1 and 0.3 are in Appendix J.1). Table 1 shows that our model achieves near-perfect recovery for simple conjunctive rules (X_3, X_4) and high imputation accuracy for the complex disjunctive rule (X_5). Figure 4(a)-(b) illustrates stable training dynamics. For X_5, the model uses sequential covering (Figure 4(c)), with "well-explained" examples reducing the remaining set. The fine-tuning (FT) phase is followed, which corrects the rule structure and boosts accuracy (Figure 4(d)-(e)). The corresponding ablation study (Table 2) confirms that fine-tuning is critical for disjunctive rules, increasing unobserved imputation accuracy for X_5 from 0.87 to 1.00.

Table 3: Impact of **rule optimization order** on learning progress. Use the example (a) of Figure 3. Note: ✓ denotes successful learning for the respective predicate.

Cycle	Metric	Run 1	Run 2	Run 3
Cycle 1	Optimization Order	$[X_5, X_4, X_3]$	$[X_3, X_5, X_4]$	$[X_3, X_4, X_5]$
	Rule Accu.	X_3 ✓, X_4, X_5	X_3 ✓, X_4 ✓, X_5	X_3 ✓, X_4 ✓, X_5 ✓
	Imputation Accu., Train Loss	$X_3 : 1.00, 0.005$ $X_4 : 0.87, 0.074$ $X_5 : 0.94, 0.053$	$X_3 : 1.00, 0.005$ $X_4 : 1.00, 0.004$ $X_5 : 0.94, 0.035$	$X_3 : 1.00, 0.005$ $X_4 : 1.00, 0.004$ $X_5 : 1.00, 0.003$
Cycle 2	Optimization Order	$[X_3, X_5, X_4]$	$[X_5, X_3, X_4]$	—
	Rule Accu.	X_3 ✓, X_4 ✓, X_5	X_3 ✓, X_4 ✓, X_5 ✓	—
	Imputation Accu., Train Loss	$X_3 : 1.00, 0.005$ $X_4 : 1.00, 0.004$ $X_5 : 0.94, 0.035$	$X_3 : 1.00, 0.005$ $X_4 : 1.00, 0.004$ $X_5 : 1.00, 0.003$	—
Cycle 3	Optimization Order	$[X_3, X_4, X_5]$	—	—
	Rule Accu.	X_3 ✓, X_4 ✓, X_5 ✓	—	—
	Imputation Accu., Train Loss	$X_3 : 1.00, 0.005$ $X_4 : 1.00, 0.004$ $X_5 : 1.00, 0.003$	—	—

Our asynchronous coordinate descent is robust to different rule optimization orders (Table 3, Appendix Figures 9-11) and is data-efficient, recovering complex rules with as few as 4,000 samples (Appendix Figure 8). While coordinate descent requires different cycle numbers, Appendix Table 9 demonstrate minimal time and memory costs.

Convergence Analysis of Asynchronous Coordinate Descent. Exact rule-set induction reduces to the minimum-set-cover problem (*NP-hard*), so like any practical rule learner, we do not claim global optimality. Instead, we frame search as asynchronous block-coordinate descent on a smooth surrogate loss: at each step, we update a single rule embedding in closed form, which guarantees the loss never increases yet keeps each move computationally cheap. To guard against poor local minima, we (i) freeze a rule only after this rule is perfectly learned, and (ii) launch diverse initializations. Across 20 runs on synthetic datasets (Tables 23-28), this strategy delivers $< 1.3\%$ imputation performance variance, and the top-ranked learned rules consistently match ground truth rules. More theoretical discussions are provided in Appendix B.

6.2 Real-World Data Experiments

We validate our approach on three real-world datasets, comparing it with *(i) statistical models* (**MICE**(van Buuren & Groothuis-Oudshoorn, 2011), **MissForest**(Stekhoven & Bühlmann, 2012)), *(ii) deep generative models* (**MLP**, **GAIN**(Yoon et al., 2018), **MissDiff**(Ouyang et al., 2023),

mDAE(Dupuy et al., 2024), **VAE**(Veldkamp et al., 2025)) and *(iii) rule-based interpretable models* (**BRCG**(Dash et al., 2018), **RRL**(Wang et al., 2021), **DR-NET**(Qiao et al., 2021), **LEN**(Barbiero et al., 2022)). For each dataset, we randomly miss some features. We then evaluated the models on their ability to impute these missing values, as well as their performance on a downstream target classification task. Preprocessing and baselines details are provided in Appendix D.2 and D.3.

Table 4: Comparison of imputation accuracy and learned rules on the Birds dataset.

Method	**Imp Acc.**	**Learned Rules**
LEN	0.57	$\mathit{abnormal\_bird} \leftarrow (\mathit{ostrich} \wedge \neg \mathit{wounded}) \vee (\mathit{bird} \wedge \mathit{wounded})$
	0.55	$\mathit{can\_fly} \leftarrow (\mathit{bird} \wedge \neg \mathit{ostrich}) \vee (\neg \mathit{ostrich} \wedge \neg \mathit{wounded})$
RRL	0.53	$\mathit{abnormal\_bird} \leftarrow (\mathit{bird} \wedge \neg \mathit{wounded}) \vee (\mathit{bird} \wedge \mathit{ostrich})$
	0.51	$\mathit{can\_fly} \leftarrow (\neg \mathit{ostrich} \wedge \neg \mathit{wounded}) \vee (\mathit{bird} \wedge \neg \mathit{ostrich})$
BRCG	0.50	$\mathit{abnormal\_bird} \leftarrow \mathit{bird} \wedge \mathit{ostrich}$
	0.47	$\mathit{can\_fly} \leftarrow \mathit{bird} \wedge \neg \mathit{abnormal\_bird}$
DR-NET	0.56	$\mathit{abnormal\_bird} \leftarrow (\mathit{bird} \wedge \neg \mathit{ostrich} \wedge \mathit{wounded}) \vee (\mathit{bird} \wedge \mathit{ostrich} \wedge \neg \mathit{wounded})$
	0.53	$\mathit{can\_fly} \leftarrow (\mathit{bird} \wedge \neg \mathit{ostrich} \wedge \neg \mathit{abnormal\_bird}) \vee (\mathit{bird} \wedge \neg \mathit{ostrich} \wedge \neg \mathit{wounded})$
NS-FCN	**1.00**	$\mathit{abnormal\_bird} \leftarrow \mathit{ostrich} \vee (\mathit{bird} \wedge \mathit{wounded})$
	1.00	$\mathit{can\_fly} \leftarrow \mathit{bird} \wedge \neg \mathit{abnormal\_bird}$

Table 5: Comparison of imputation accuracy and learned rules on the Heart Disease dataset.

Method	**Imp Acc.**	**Learned Rules**
LEN	0.65	$\mathit{trestbps\_high} \leftarrow (\neg \mathit{st\_mild} \wedge \mathit{cp\_atypical\_angina}) \vee (\mathit{chol\_low} \wedge \mathit{cp\_asymptomatic})$
	0.53	$\mathit{chol\_high} \leftarrow (\mathit{sex\_female} \wedge \mathit{ca\_2}) \vee (\mathit{bp\_normal} \wedge \mathit{cp\_asymptomatic})$
	0.62	$\mathit{hr\_high} \leftarrow (\mathit{cp\_asymptomatic} \wedge \mathit{target}) \vee (\mathit{chol\_low} \wedge \mathit{ca\_1})$
	0.70	$\mathit{st\_severe} \leftarrow (\mathit{cp\_non\_anginal} \wedge \neg \mathit{fbs\_normal}) \vee (\mathit{age\_old} \wedge \mathit{chol\_low})$
RRL	0.28	$\mathit{trestbps\_high} \leftarrow (\mathit{sex\_female} \wedge \neg \mathit{cp\_typical\_angina}) \vee (\mathit{exang\_yes} \wedge \neg \mathit{thal\_normal})$
	0.33	$\mathit{hr\_high} \leftarrow (\mathit{age\_middle} \wedge \mathit{sex\_male}) \vee (\neg \mathit{restecg\_stt\_abnormality} \wedge \mathit{slope\_upsloping})$
	0.33	$\mathit{thalach} \leftarrow (\text{age} < 60) \wedge (\text{restecg} = 0)$
	0.32	$\mathit{st\_severe} \leftarrow (\neg \mathit{exang\_yes} \wedge \neg \mathit{slope\_flat}) \vee (\mathit{chol\_low} \wedge \mathit{cp\_asymptomatic})$
BRCG	0.53	$\mathit{trestbps\_high} \leftarrow \neg \mathit{age\_young} \wedge \neg \mathit{ca\_4}$
	0.35	$\mathit{chol\_high} \leftarrow \neg \mathit{age\_young} \wedge \neg \mathit{restecg\_hypertrophy}$
	0.33	$\mathit{hr\_high} \leftarrow \neg \mathit{cp\_typical\_angina} \wedge \neg \mathit{ca\_4}$
	0.32	$\mathit{st\_severe} \leftarrow \neg \mathit{age\_young} \wedge \neg \mathit{slope\_upsloping}$
DR-NET	0.53	$\mathit{trestbps\_high} \leftarrow (\mathit{chol\_low} \wedge \neg \mathit{hr\_low} \wedge \neg \mathit{fbs\_high}) \vee (\mathit{slope\_flat} \wedge \mathit{ca\_1} \wedge \mathit{thal\_normal})$
	0.33	$\mathit{chol\_high} \leftarrow \mathit{sex\_male} \wedge \mathit{slope\_upsloping} \wedge \mathit{ca\_3}$
	0.33	$\mathit{hr\_high} \leftarrow \neg \mathit{age\_old} \wedge \neg \mathit{cp\_typical\_angina} \wedge \mathit{fbs\_high}$
	0.32	$\mathit{st\_severe} \leftarrow \mathit{hr\_high} \wedge \neg \mathit{sex\_male} \wedge \neg \mathit{fbs\_normal}$
NS-FCN	**0.86**	$\mathit{trestbps\_high} \leftarrow (\text{age} > 60) \wedge (\text{chol} > 250)$
	0.85	$\mathit{chol\_high} \leftarrow (\text{sex} = 1 \wedge \text{age} > 55) \vee (\text{trestbps} > 150)$
	0.90	$\mathit{hr\_high} \leftarrow (\text{trestbps} > 145) \vee (\text{age} > 57 \wedge \text{cp} = 3)$
	0.76	$\mathit{st\_severe} \leftarrow (\text{slope} = 2) \wedge (\text{thalach} < 150)$

For *logical reasoning*, we used the Birds dataset (Tafjord et al., 2021) with a 90% missing ratio for two key predicates. As shown in Table 4, under some random seeds, NS-FCN achieves perfect imputation accuracy (1.00) and, crucially, **perfectly recovers the ground truth logical rules**, highlighting its superior capability in deciphering underlying logical structures. Table 6 compares our approach with non-interpretable baselines. While a MLP achieve optimal performance given the simplicity of the Birds dataset, our model remains highly competitive; more importantly, it demonstrates robustness across diverse random initializations, successfully recovering the correct ground-truth rules in the majority of cases. Table 20 further show that the a few hundred samples are sufficient for the model to converge to the correct logical truth.

In *medical diagnosis*, we use Heart Disease (Detrano et al., 1989) and SPECT Heart (Kurgan et al., 2001) datasets, introducing 30% missingness. We also vary the observation ratio from 0.3 to 0.9, and the results in Tables 18 and 19 shows comparable performance with only 30% of the data observed.

On the Heart Disease dataset, with its mix of continuous and categorical features, NS-FCN's direct handling of continuous values led to superior imputation (e.g., 90% accuracy for `thalach`) and the discovery of **clinically relevant rules with numerical thresholds** (e.g., *age* > 60, *chol* > 250), as shown in Tables 5 and 32. NS-FCN attains imputation accuracy comparable to the advanced statistical and generative baselines, yet distinguishes itself by offering full interpretability, a critical advantage over these black-box approaches. Compared with rule-based models, our evaluation highlights NS-FCN's unique ability to handle heterogeneous data types. A key distinction is that NS-FCN directly models continuous features, whereas **baseline methods are restricted to binary inputs**, forcing discretization (e.g., for `trestbps`, binning values into $< 120, 120 - 140, > 140$ mmHg as low, normal, and high).

On the binary SPECT dataset, we randomly miss all 22 features, thus we report the diagnosis accuracy after imputation. When the imputed features are used for diagnosis, NS-FCN outperforms all baselines on both Heart Disease and SPECT, as shown in Table 7. Unlike baseline models that train a classifier on previously imputed samples, where imputation errors inevitably propagate to the downstream task, our method jointly optimizes rule discovery and target inference. Furthermore, our use of soft-logic relaxation prevents the model from overfitting to noise (such as incorrect features), enabling it to capture dominant logical structures. This robustness is further supported by the comprehensive noise sensitivity analysis in Appendix I.1 (Tables 16 and 17), which demonstrates that the model learns valid rule approximations (e.g. capturing one correct clause) and maintains strong predictive performance even as noise levels increase.

Detailed rules and LLM assessments are in Appendix Tables 30, 31, and 32.

Table 6: Imputation accuracy of missing feature value comparison across Heart Disease and Bird datasets on non-interpretable baselines. Results are over 10 random seeds.

Method	**Heart Disease**				**Birds**	
	trestbps	*chol*	*thalach*	*oldpeak*	*abnormal_bird*	*can_fly*
MICE	0.84±0.016	0.83±0.014	0.88±0.011	0.87±0.015	0.88±0.006	0.86±0.011
MissForest	**0.88±0.015**	0.84±0.012	**0.91±0.004**	0.88±0.016	0.38±0.123	0.68±0.086
MLP	**0.88±0.009**	**0.85±0.016**	0.88±0.014	0.80±0.025	**0.96±0.059**	**0.99±0.003**
GAIN	0.85±0.022	0.84±0.011	0.90±0.014	**0.89±0.014**	0.83±0.102	0.82±0.083
MissDiff	0.82±0.017	0.83±0.019	0.89±0.018	0.84±0.030	0.83±0.020	0.86±0.007
mDAE	**0.88±0.011**	0.84±0.012	0.90±0.015	0.87±0.015	0.87±0.002	0.87±0.004
VAE-based	0.85±0.015	0.84±0.021	0.90±0.015	0.86±0.015	0.62±0.006	0.87±0.004
NS-FCN	0.87±0.025	**0.85±0.017**	0.88±0.014	0.78±0.020	0.95±0.064	0.95±0.064

Table 7: Medical diagnosis after missing value imputation. Results are over 10 random seeds.

Method	**Heart Disease**		**SPECT**	
	Accuracy	*F1*	*Accuracy*	*F1*
MICE(van Buuren & Groothuis-Oudshoorn, 2011)	0.83±0.010	0.81±0.012	0.78±0.019	0.87±0.013
MissForest(Stekhoven & Bühlmann, 2012)	0.83±0.013	0.81±0.014	0.79±0.012	0.87±0.008
MLP	0.84±0.010	0.82±0.012	**0.92±0.007**	0.90±0.005
GAIN(Yoon et al., 2018)	0.84±0.004	0.82±0.006	0.76±0.019	0.85±0.013
MissDiff(Ouyang et al., 2023)	0.84±0.010	0.82±0.011	0.77±0.023	0.86±0.016
mDAE(Dupuy et al., 2024)	0.84±0.009	0.82±0.010	0.80±0.013	0.88±0.009
VAE-based(Veldkamp et al., 2025)	0.83±0.009	0.81±0.009	0.75±0.016	0.85±0.011
BRCG(Dash et al., 2018)	0.77±0.006	0.74±0.034	0.85±0.046	0.90±0.035
RRL(Wang et al., 2021)	0.78±0.002	0.80±0.003	0.90±0.005	0.94±0.005
DR-NET(Qiao et al., 2021)	0.85±0.005	0.82±0.005	0.89±0.025	0.92±0.017
LEN(Barbiero et al., 2022)	0.69±0.007	0.80±0.000	0.76±0.035	0.85±0.017
NS-FCN	**0.91±0.009**	**0.91±0.009**	**0.92±0.009**	**0.96±0.009**

7 Conclusion

Our NS-FCN framework effectively learns interpretable rules for missing value imputation, demonstrating strong performance across a diverse range of synthetic and real-world datasets. A key strength is its ability to seamlessly reason over heterogeneous data, handling both binary predicates (e.g., Birds) and continuous features in complex domains like medical diagnosis (SPECT, Heart Disease). It successfully handles missing data and learns hierarchical rule structures, offering significant potential for trustworthy diagnostics and transparent decision-making.

Reproducibility Statement

We have made extensive efforts to ensure the reproducibility of our results. The complete description of both synthetic dataset generation and real-world dataset preprocessing methods are illustrated in Appendix E and D.2. Details of the computational setup, including hardware configuration and software environment, as well as the choice of hyper-parameters are documented in Appendix J.4 and K.3. We will release our code in the camera-ready stage to facilitate replication and further research.

References

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Pietro Lió, Marco Gori, and Stefano Melacci. Entropy-based logic explanations of neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6046–6054, 2022.

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3): 334–334, 1997.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Andres Campero, Aldo Pareja, Tim Klinger, Josh Tenenbaum, and Sebastian Riedel. Logical rule induction and theory learning using neural theorem proving. *arXiv preprint arXiv:1809.02193*, 2018.

Zhi Chen, Sarah Tan, Urszula Chajewska, Cynthia Rudin, and Rich Caruna. Missing values and imputation in healthcare data: Can interpretable machine learning help? In *Conference on Health, Inference, and Learning*, pp. 86–99. PMLR, 2023.

Xuening Feng Paul Weng Matthieu Zimmer Dong Li Wulong Liu Jianye Hao Claire Glanois, Zhaohui Jiang. Neuro-symbolic hierarchical rule induction. In *International Conference on Machine Learning (ICML)*, pp. 7583–7615. PMLR, 2022.

William W Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pp. 115–123. Elsevier, 1995.

Andrew Cropper and Rolf Morel. Learning programs by learning from failures. *Machine Learning*, 110(4):801–856, 2021.

Sanjeeb Dash, Oktay Gunluk, and Dennis Wei. Boolean decision rules via column generation. *Advances in neural information processing systems*, 31, 2018.

Robert Detrano, Andras Janosi, Walter Steinbrunn, Matthias Pfisterer, Johann-Jakob Schmid, Sarbjit Sandhu, Kern H Guppy, Stella Lee, and Victor Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *The American journal of cardiology*, 64(5):304–310, 1989.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. *arXiv preprint arXiv:1904.11694*, 2019.

Mariette Dupuy, Marie Chavent, and Remi Dubois. mdae: modified denoising autoencoder for missing data imputation. *arXiv preprint arXiv:2411.12847*, 2024.

Bradley Efron. Missing data, imputation, and the bootstrap. *Journal of the American Statistical Association*, 89(426):463–475, 1994.

Khaled M Fouad, Mahmoud M Ismail, Ahmad Taher Azar, and Mona M Arafa. Advanced methods for missing values imputation based on similarity learning. *PeerJ Computer Science*, 7:e619, 2021.

Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*, 2015.

Pascal Hitzler and Md Kamruzzaman Sarker. Neuro-symbolic artificial intelligence: The state of the art. 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lukasz A Kurgan, Krzysztof J Cios, Ryszard Tadeusiewicz, Marek Ogiela, and Lucy S Goodenday. Knowledge discovery approach to automated cardiac spect diagnosis. *Artificial intelligence in medicine*, 23(2):149–169, 2001.

Linchao Li, Bowen Du, Yonggang Wang, Lingqiao Qin, and Huachun Tan. Estimation of missing values in heterogeneous traffic data: Application of multimodal deep learning model. *Knowledge-Based Systems*, 194:105592, 2020.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Alessandro Oltramari, Jonathan Francis, Cory Henson, Kaixin Ma, and Ruwan Wickramarachchi. Neuro-symbolic architectures for context understanding. In *Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges*, pp. 143–160. IOS Press, 2020.

Yidong Ouyang, Liyan Xie, Chongxuan Li, and Guang Cheng. Missdiff: Training diffusion models on tabular data with missing values. *arXiv preprint arXiv:2307.00467*, 2023.

Leonardo Pellegrina and Fabio Vandin. Scalable rule lists learning with sampling. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2352–2363, 2024.

Litao Qiao, Weijia Wang, and Bill Lin. Learning accurate and interpretable decision rule sets from neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4303–4311, 2021.

J. Ross Quinlan. Learning logical definitions from relations. *Machine learning*, 5:239–266, 1990.

Md Geaur Rahman and Md Zahidul Islam. A decision tree-based missing value imputation technique for data pre-processing. In *The 9th Australasian Data Mining Conference: AusDM 2011*, pp. 41–50. Australian Computer Society Inc, 2011.

Md Geaur Rahman and Md Zahidul Islam. Fimus: A framework for imputing missing values using co-appearance, correlation and similarity analysis. *Knowledge-Based Systems*, 56:311–327, 2014.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62:107–136, 2006.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1):1–38, 2014.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. *Advances in neural information processing systems*, 30, 2017.

Hikaru Shindo, Masaaki Nishino, and Akihiro Yamamoto. Differentiable inductive logic programming for structured examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 5034–5041, 2021.

Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

Lena Stempfle and Fredrik Johansson. Minty: Rule-based models that minimize the need for imputing features with missing values. In *International Conference on Artificial Intelligence and Statistics*, pp. 964–972. PMLR, 2024.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pp. 2071–2080. PMLR, 2016.

Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

Stef van Buuren and Catharina Gerarda Maria Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45(3), 2011.

Karel Veldkamp, Raoul Grasman, and Dylan Molenaar. Handling missing data in variational autoencoder based item response theory. *British Journal of Mathematical and Statistical Psychology*, 78 (1):378–397, 2025.

Xinyu Wang, Isil Dillig, and Rishabh Singh. Synthesis of data completion scripts using finite tree automata. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–26, 2017.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014. doi: 10.1609/aaai.v28i1.8870. URL `https://doi.org/10.1609/aaai.v28i1.8870`.

Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Scalable rule-based representation learning for interpretable classification. *Advances in Neural Information Processing Systems*, 34:30479–30491, 2021.

Dennis Wei, Sanjeeb Dash, Tian Gao, and Oktay Gunluk. Generalized linear rule models. In *International conference on machine learning*, pp. 6687–6696. PMLR, 2019.

Stephen J Wright. Coordinate descent algorithms. *Mathematical programming*, 151(1):3–34, 2015.

Yang Yang, Chao Yang, Boyang Li, Yinghao Fu, and Shuang Li. Neuro-symbolic temporal point processes. *arXiv preprint arXiv:2406.03914*, 2024.

Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*, pp. 5689–5698. PMLR, 2018.

Chengqi Zhang, Yongsong Qin, Xiaofeng Zhu, Jilian Zhang, and Shichao Zhang. Clustering-based missing value imputation for data preprocessing. In *2006 4th IEEE International Conference on Industrial Informatics*, pp. 1081–1086. IEEE, 2006.

A Related Work Supplement

Traditional Inductive Logic Programming (ILP) Methods. Inductive Logic Programming learns logical rules from relational data. Cohen (1995) proposed RIPPER, a fast rule induction algorithm using separate-and-conquer strategy. Quinlan (1990) developed FOIL, which generates clauses iteratively. Dash et al. (2018) introduced Boolean decision rules using column generation. Wei et al. (2019) proposed GLRM integrating decision rules into linear models. Cropper & Morel (2021) presented LFF implemented in Popper. These approaches rely on heuristics but may not guarantee optimal solutions. Pellegrina & Vandin (2024) proposed SamRuLe for near-optimal rule lists via sampling.

Differentiable ILP Methods. Traditional ILP models struggle with noisy data and scalability. Differentiable approaches address these issues by integrating continuous relaxation, which allows gradient descent for optimization. Shindo et al. (2021) proposed ∂ILP, which represents logic rules in a differentiable form and combines neural networks with symbolic logic. Manhaeve et al. (2018) introduced DeepProbLog, extending ProbLog with neural predicates. Neural Logic Machines (NLMs) (Dong et al., 2019) combine MLPs with logic programming to improve computational efficiency but reduce interpretability.

Broader Missing Data Imputation Methods. Missing data imputation methods range from global model-based techniques to localized and hybrid strategies, extending to deep and ensemble frameworks.

At the global end, nonparametric bootstrap methods (Efron, 1994) provide bias-corrected estimates via repeated sampling, while spectral regularization approaches like SOFT-IMPUTE (Mazumder et al., 2010) solve a nuclear-norm minimization through iterative soft-thresholded SVD. Classical multivariate imputation schemes such as MICE (van Buuren & Groothuis-Oudshoorn, 2011) construct a sequence of conditional models for each variable with missingness and iteratively sample from these chained regressions until convergence, thereby approximating draws from the joint posterior and naturally propagating uncertainty across multiple imputations. Tree-based ensemble methods such as MissForest (Stekhoven & Bühlmann, 2012) adopt an iterative refinement strategy in which random forests are trained per variable using the currently imputed data as predictors, updating missing entries via out-of-bag predictions until changes stabilize, thus capturing complex nonlinearities and high-order interactions without requiring parametric distributional assumptions.

Moving toward local adaptation, decision tree–based EM (DMI) (Rahman & Islam, 2011) partitions complete cases via C4.5 and imputes within each leaf, and clustering-based random imputation (CRI) (Zhang et al., 2006) applies kernel-weighted estimation in the nearest k-means cluster. Hybrid similarity learners, such as KI and its fuzzy extension FCKI (Fouad et al., 2021), refine this idea by dynamically selecting neighborhood sizes before multivariate imputation. For high-dimensional or heterogeneous data, deep architectures like GAIN (Yoon et al., 2018) cast imputation as a generative adversarial game where a generator proposes imputations conditioned on an observed-mask vector and a discriminator learns to distinguish observed from imputed components, while VAE-based imputers (Veldkamp et al., 2025) treat the complete feature matrix as generated from low-dimensional latent variables and learn to reconstruct missing entries via amortized variational inference under a probabilistic encoder–decoder architecture. Building on denoising autoencoders, mDAE (Dupuy et al., 2024) modifies the reconstruction loss to ignore pre-imputed values at missing positions and couples this with an overcomplete hidden representation, which empirically improves RMSE over standard DAEs and several classical imputers across multiple UCI datasets (Dupuy et al., 2024). In the same spirit of generative modeling, MissDiff (Ouyang et al., 2023) trains a diffusion model on tabular data with missing values by injecting noise along a forward stochastic process and learning a reverse denoising process that is explicitly conditioned on the observed-mask pattern, thereby producing imputations through iterative refinement from pure noise. Models such as MMDL (Li et al., 2020) align stacked autoencoder embeddings across modalities to exploit cross-view correlations. Ensemble schemes like FIMUS (Rahman & Islam, 2014) combine co-appearance, correlation, and similarity in a weighted-voting framework. Despite their varied focuses—ranging from global inference to localized and multimodal learning—these methods uniformly rely on statistical patterns and *do not leverage explicit logical rules to govern inter-variable relationships.*

B Convergence Analysis of Coordinate Gradient Descent

For clarity, we analyze an simplified version of our learning algorithm in which each head predicate U_j is associated with a single parameter block Θ_j. Let $\Theta = (\Theta_1, \ldots, \Theta_m)$ collect all parameters. The global training objective is

$$\mathcal{L}(\Theta) = \sum_{j=1}^{m} \mathcal{L}_{U_j}(\Theta), \qquad \mathcal{L}_{U_j}(\Theta) = \mathrm{mean}\big((v_{U_j}(\Theta) \odot \mathrm{mask}_j - U_{j,\mathrm{obs}} \odot \mathrm{mask}_j)^2\big), \tag{7}$$

where $v_{U_j}(\Theta)$ is computed by forward chaining using the differentiable operators introduced in the main text (e.g., Eq. 4).

B.1 Assumptions

We make the following standard assumptions for smooth block coordinate descent (e.g., (Tseng, 2001; Bertsekas, 1997; Nesterov, 2013))

Assumption 1 *The objective $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is*

1. *bounded below: $\inf_\Theta \mathcal{L}(\Theta) > -\infty$,*
2. *continuously differentiable in Θ, and*
3. *has block-wise Lipschitz-continuous gradients: for each j there exists $L_j < \infty$ such that, for all Θ and all h_j,*

$$\big\|\nabla_{\Theta_j}\mathcal{L}(\Theta + e_j h_j) - \nabla_{\Theta_j}\mathcal{L}(\Theta)\big\| \;\le\; L_j \, \|h_j\|, \tag{8}$$

where $e_j h_j$ denotes the vector obtained by changing only block j.

These conditions hold in our setting because $\mathcal{L}$ is built from smooth operations (e.g., linear maps, sigmoid, softmin, log-sum-exp) composed with a squared loss, and training is restricted to bounded level sets.

B.2 Idealized Full-Batch Block Coordinate Gradient Descent

Consider the following idealized algorithm. At iteration t we pick a block index $j_t \in \{1, \ldots, m\}$ (e.g., by cycling through $\{1, \ldots, m\}$) and perform a gradient step on that block only:

$$\Theta_{j_t}^{t+1} = \Theta_{j_t}^{t} - \eta \, \nabla_{\Theta_{j_t}} \mathcal{L}(\Theta^t), \tag{9}$$

$$\Theta_{\ell}^{t+1} = \Theta_{\ell}^{t} \qquad \text{for all } \ell \neq j_t, \tag{10}$$

where $\eta > 0$ is a step size. This matches the idealized version of the rule update in Section 5.1: when we update U_{j_t}, all other predicates U_ℓ are kept fixed.

Lemma 1 (Monotone decrease for small steps) *Suppose Assumption 1 holds. If the step size satisfies $0 < \eta \le 1/L_{j_t}$ at iteration t, then*

$$\mathcal{L}(\Theta^{t+1}) \;\le\; \mathcal{L}(\Theta^t) - \frac{\eta}{2} \big\|\nabla_{\Theta_{j_t}} \mathcal{L}(\Theta^t)\big\|^2. \tag{11}$$

In particular, the sequence $\{\mathcal{L}(\Theta^t)\}_{t \ge 0}$ is monotonically non-increasing and convergent.

Proof 1 (Proof sketch) *By block-wise Lipschitz continuity of $\nabla_{\Theta_{j_t}} \mathcal{L}$,*

$$\mathcal{L}(\Theta^{t+1}) = \mathcal{L}\big(\Theta^t + e_{j_t}(\Theta_{j_t}^{t+1} - \Theta_{j_t}^{t})\big) \tag{12}$$

$$\le \mathcal{L}(\Theta^t) + \big\langle \nabla_{\Theta_{j_t}} \mathcal{L}(\Theta^t), \Theta_{j_t}^{t+1} - \Theta_{j_t}^{t} \big\rangle + \frac{L_{j_t}}{2} \|\Theta_{j_t}^{t+1} - \Theta_{j_t}^{t}\|^2. \tag{13}$$

Substituting the update $\Theta_{j_t}^{t+1} - \Theta_{j_t}^{t} = -\eta \, \nabla_{\Theta_{j_t}} \mathcal{L}(\Theta^t)$ and rearranging gives

$$\mathcal{L}(\Theta^{t+1}) \le \mathcal{L}(\Theta^t) - \eta\Big(1 - \frac{\eta L_{j_t}}{2}\Big) \big\|\nabla_{\Theta_{j_t}} \mathcal{L}(\Theta^t)\big\|^2. \tag{14}$$

If $\eta \le 1/L_{j_t}$, then $1 - \eta L_{j_t}/2 \ge 1/2$, yielding the claimed inequality.

Lemma 1 implies that the loss decreases at every iteration and the gradients on updated blocks cannot stay large forever. Combined with a mild assumption that each block is selected infinitely often, we obtain convergence to a block-stationary point.

Proposition 1 (Convergence to a block-stationary point) *Assume 1 holds, the level set $\{\Theta : \mathcal{L}(\Theta) \leq \mathcal{L}(\Theta^0)\}$ is bounded, each block j is selected infinitely often, and the step sizes satisfy $0 < \eta \leq \min_j 1/L_j$. Then any limit point $\Theta^\star$ of the sequence $\{\Theta^t\}$ generated by the above block coordinate gradient method is* block-stationary*:*

$$\nabla_{\Theta_j}\mathcal{L}(\Theta^\star) = 0 \quad \textit{for all } j = 1, \ldots, m. \tag{15}$$

Equivalently, no single block Θ_j can be perturbed to decrease $\mathcal{L}$ while all other blocks are fixed.

Proof 2 (Proof sketch) *Summing the inequality from Lemma 1 over t shows that*

$$\sum_{t=0}^{\infty} \left\| \nabla_{\Theta_{j_t}} \mathcal{L}(\Theta^t) \right\|^2 < \infty,$$

so the block gradients must tend to zero along the subsequence where a given block j is updated. Since each block is selected infinitely often and the iterates remain in a bounded level set, standard arguments for block coordinate descent (Tseng, 2001) imply that any limit point has zero gradient in every block.

Thus, in the ideal full-batch setting with sufficiently small steps, our predicate-wise coordinate updates produce a non-increasing loss sequence $\{\mathcal{L}(\Theta^t)\}$ and converge to a point where no single predicate block Θ_j can further reduce the global objective.

B.3 Stochastic Mini-Batch Variant and Adam

In practice, our implementation uses mini-batches and the Adam optimizer for each block update (as described in Section 5.1). In this case, the gradient $\nabla_{\Theta_j}\mathcal{L}$ is replaced by a stochastic estimate computed on a mini-batch, and the step uses Adam's adaptive preconditioning. This yields a *stochastic* block-coordinate gradient scheme: the loss is no longer guaranteed to decrease at every single update, but under standard assumptions stochastic block-coordinate methods are known to approach a neighborhood of a stationary point in expectation (see, e.g., (Richtárik & Takáč, 2014; Wright, 2015)).

C Model Supplement Description

C.1 Optimization Details

Throughout all training stages, each rule embedding (or set of embeddings during joint fine-tuning) is optimized using the Adam optimizer. A crucial step following each gradient update is the normalization of the rule embeddings. This involves applying a Rectified Linear Unit (ReLU) activation to the embedding data (ensuring non-negative values, which can aid interpretability for positive predicate contributions) followed by L_2 normalization of each row vector within the rule embedding matrix. This normalization helps stabilize the training process and maintains consistent magnitudes for the embedding components.

D Datasets and Baselines

D.1 Datasets

Heart Disease. We use the widely-cited Cleveland Clinic dataset from the UCI Heart Disease database (Detrano et al., 1989). This dataset contains 303 patient records, each with 13 features—a mix of continuous and categorical variables—such as age, cholesterol level, and resting blood pressure. The task is to predict the presence of heart disease, which is indicated by the target variable on a scale from 0 (absence) to 4 (severe). Following standard practice, we simplify this into a binary classification problem: predicting presence (values 1-4) versus absence (value 0).

SPECT. The SPECT (Single Proton Emission Computed Tomography) dataset presents a binary classification task to diagnose cardiac conditions (normal/abnormal) based on 22 binary patient features. The dataset describes the diagnosis of cardiac SPECT images. Each of the patients is classified into two categories: normal and abnormal. The 267 SPECT image sets (patients) database were processed to extract features that summarize the original SPECT images. As a result, 44 continuous feature patterns were created for each patient. The pattern was further processed to obtain 22 binary feature patterns. The CLIP3 algorithm was used to generate classification rules from these patterns (Kurgan et al., 2001). The CLIP3 algorithm generated rules that were 84.0% accurate (as compared with cardiologists' diagnoses). A key challenge in this domain is the prevalence of missing data, making it an ideal testbed for our model's imputation and rule-learning capabilities.

Birds. Bird's Rulebase is a well-known logic problem designed to assess an AI's ability to learn and reason with hierarchical logical rules that mimic common-sense knowledge (Tafjord et al., 2021). It has the ground truth single theory of six rules [1] as follows.

$$\begin{aligned}
\text{can\_fly}(X) &\leftarrow \text{bird}(X), \text{not abnormal\_bird}(X)\\
\text{bird}(X) &\leftarrow \text{ostrich}(X)\\
\text{abnormal\_bird}(X) &\leftarrow \text{ostrich}(X)\\
\text{not can\_fly}(X) &\leftarrow \text{ostrich}(X)\\
\text{abnormal\_bird}(X) &\leftarrow \text{bird}(X), \text{wounded}(X)\\
\text{not can\_fly}(X) &\leftarrow \text{wounded}(X)
\end{aligned}$$

Figure 5 further illustrates the structure of these rules.

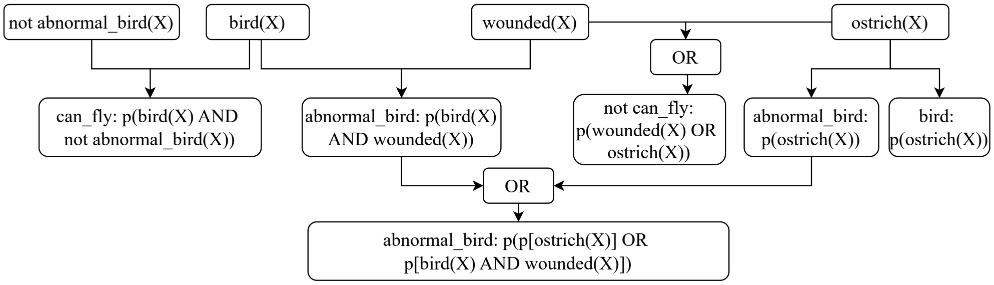

not abnormal_bird(X)
bird(X)
wounded(X)
ostrich(X)
OR
can_fly: p(bird(X) AND not abnormal_bird(X))
abnormal_bird: p(bird(X) AND wounded(X))
not can_fly: p(wounded(X) OR ostrich(X))
abnormal_bird: p(ostrich(X))
bird: p(ostrich(X))
OR
abnormal_bird: p(p[ostrich(X)] OR p[bird(X) AND wounded(X)])

Figure 5: Ground truth rules for Bird dataset.

D.2 PREPROCESSING OF DATASETS

Heart Disease. The UCI Heart Disease dataset contains a mix of 13 continuous and categorical features with 303 samples. To create a challenging imputation task, we introduced a 30% missing ratio independently into four key continuous variables: resting blood pressure (`trestbps`), cholesterol (`chol`), maximum heart rate (`thalach`), and ST depression (`oldpeak`). Following the protocol in MissDiff(Ouyang et al., 2023), we generate missing values under a Missing Completely At Random (MCAR) mechanism. Let $\mathbf{x} \in \mathbb{R}^d$ denote the complete data vector. We generate a binary mask vector $\mathbf{m} \in \{0, 1\}^d$, where $m_i = 1$ indicates that x_i is observed, and $m_i = 0$ indicates it is missing. The observed data is represented as $\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{m} + \text{na} \odot (1 - \mathbf{m})$, where $\odot$ denotes element-wise multiplication.

For our NS-FCN framework, the task is to directly impute these missing continuous values. For deep learning baselines, continuous features are standardized using Z-score normalization, and categorical features are one-hot encoded. For tree-based and statistical baselines (MissForest, MICE), categorical variables are treated as factors. However, to accommodate the baseline models which only support binary inputs, we first discretized these four variables into three categorical bins based on clinical thresholds: blood pressure ($< 120, 120 - 140, > 140$), cholesterol ($< 200, 200 - 240, \geq 240$), max heart rate ($< 100, 100 - 160, \geq$160), and ST depression ($\leq 1.0, 1.0 - 2.0, > 2.0$). The baselines were then tasked with imputing the correct category. Consequently, we evaluate the imputation accuracy on the discretized bins.

SPECT Heart. The dataset's 22 binary features were randomly masked with a 30% probability to simulate missing data. Our framework was then applied to a two-stage task: first, to impute the missing features, and second, to perform the final patient diagnosis based on the completed feature set. The diagnostic performance is compared against five baseline methods, including four rule-based approaches and an MLP.

Birds. Following the ground truth logical rules, we generated a dataset of 1,500 samples. To create a difficult logical reasoning challenge, we introduced a 90% missing ratio for two crucial latent

[1] https://www.doc.ic.ac.uk/ mjs/teaching/KnowledgeRep491/ExtendedLP 491-2x1.pdf, p5

predicates: `can_fly` and `abnormal_bird`. The task for all models was to impute these missing binary values based on the observed predicates. The imputation accuracy is compared against the same set of baselines.

D.3 Baseline Models

To rigorously evaluate performance, we compare our method against 11 established baselines, ranging from classical statistical methods, to advanced deep generative models, and interpretable models.

Statistical Models.

- **MICE** (van Buuren & Groothuis-Oudshoorn, 2011)

  Multivariate Imputation by Chained Equations (MICE) is a widely used statistical method based on Fully Conditional Specification (FCS). It iteratively imputes missing values by modeling each feature with missing data as a function of other features using linear regression (for continuous variables) or logistic regression (for categorical variables). We generate $m = 5$ imputed datasets and report results from the first completion.

- **MissForest** (Stekhoven & Bühlmann, 2012)

  MissForest is a non-parametric method that handles mixed-type data using an iterative Random Forest approach. It treats the missing data problem as a prediction task, training a random forest on the observed parts of the data to predict the missing values. It is particularly effective at capturing non-linear interactions without explicit distributional assumptions.

Deep Generative Models.

- **MLP (Multilayer Perceptron)**

  We use a simple feed-forward neural network with fully connected layers and ReLU activations as a deterministic imputation baseline. Given an input vector $\mathbf{x} \in \mathbb{R}^d$ and a binary mask $\mathbf{m} \in \{0,1\}^d$ indicating observed entries ($m_j = 1$ if x_j is observed, 0 otherwise), we first obtain $\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{m} + \mathrm{na} \odot (1 - \mathbf{m})$, and use the observed mask for input gating:

  $$\mathbf{h}_0 = \tilde{\mathbf{x}} \odot \mathbf{m}.$$

  The network f_θ takes $\mathbf{h}_0$ as input and outputs a reconstruction $\hat{\mathbf{x}} = f_\theta(\mathbf{h}_0)$. Training is performed under weak supervision by minimizing the Mean Squared Error (MSE) *only* on observed entries:

  $$\mathcal{L}_{\text{MLP}} = \|(\hat{\mathbf{x}} - \mathbf{x}) \odot \mathbf{m}\|_2^2\,,$$

  so that gradients are propagated only through coordinates with ground-truth observations; at test time, the missing entries ($m_j = 0$) are imputed using the corresponding components of $\hat{\mathbf{x}}$.

- **VAE (Variational Autoencoder)**

  Our VAE-based imputer follows the amortized inference framework of Kingma & Welling (2013), adapted to incomplete tabular data as in recent work on VAE with missingness (e.g. Veldkamp et al. (2025)). Given $(\mathbf{x}, \mathbf{m})$, we construct a gated and masked input

  $$\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{m} + \mathrm{na} \odot (1 - \mathbf{m}), \qquad \mathbf{h}_0 = \tilde{\mathbf{x}} \odot \mathbf{m},$$

  and feed the concatenated vector $[\mathbf{h}_0,\ \mathbf{1} - \mathbf{m}]$ into the encoder to obtain a Gaussian posterior

  $$q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{m}) = \mathcal{N}(\boldsymbol{\mu}_\phi, \mathrm{diag}(\boldsymbol{\sigma}_\phi^2)).$$

  A latent sample $\mathbf{z}$ is drawn via the reparameterization trick and passed through a decoder $p_\theta(\mathbf{x} \mid \mathbf{z})$ to produce $\hat{\mathbf{x}}_\theta(\mathbf{z})$. The model is trained by maximizing the Evidence Lower Bound (ELBO), where the reconstruction term only involves *observed* entries:

  $$\mathcal{L}_{\text{VAE}} = \underbrace{\big\|\big(\hat{\mathbf{x}}_\theta(\mathbf{z}) - \mathbf{x}\big) \odot \mathbf{m}\big\|_2^2}_{\text{reconstruction on observed data}} + \underbrace{\mathrm{KL}\big(q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{m}) \,\|\, p(\mathbf{z})\big)}_{\text{KL regularization}}.$$

  At inference time, missing values are imputed by the decoder output $\hat{\mathbf{x}}_\theta(\mathbf{z})$ at coordinates where $m_j = 0$.

- **DAE / mDAE (modified Denoising Autoencoder)**

  For the autoencoder baseline, we adopt a denoising autoencoder architecture with a modification of the loss function proposed in the mDAE Dupuy et al. (2024). Given $(\mathbf{x}, \mathbf{m})$, we first perform a simple pre-imputation to obtain a complete input $\tilde{\mathbf{x}}$, and then apply masking noise with rate ρ *only* on originally observed entries:

  $$\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{m} + \mathrm{na} \odot (1 - \mathbf{m}), \quad \mathbf{c} \sim \mathrm{Ber}(\rho)^d, \quad \tilde{\mathbf{x}}^{(\text{noisy})} = (\tilde{\mathbf{x}} \odot \mathbf{m}) \odot \big(\mathbf{1} - \mathbf{c} \odot \mathbf{m}\big).$$

The corrupted input $\tilde{\mathbf{x}}^{(\text{noisy})}$ is fed into an encoder–decoder network g_ψ that outputs a reconstruction $\hat{\mathbf{x}} = g_\psi(\tilde{\mathbf{x}}^{(\text{noisy})})$. Crucially, following the modified-loss idea of mDAE (Dupuy et al., 2024), the reconstruction loss is computed *only on truly observed entries*, and pre-imputed missing values are ignored:

$$\mathcal{L}_{\text{mDAE}} = \left\| (\hat{\mathbf{x}} - \mathbf{x}) \odot \mathbf{m} \right\|_2^2 .$$

This prevents the autoencoder from overfitting arbitrary pre-imputed values at missing positions while still benefiting from denoising training; at test time, imputations for missing entries ($m_j = 0$) are taken from the corresponding components of $\hat{\mathbf{x}}$.

- **GAIN (Generative Adversarial Imputation Nets) (Yoon et al., 2018)**

  GAIN adapts the Generative Adversarial Network framework for imputation. The generator G imputes missing components, while the discriminator D attempts to distinguish between observed and imputed components. A hint mechanism is introduced to provide D with partial information about the mask distribution, forcing G to learn the true underlying data distribution. We utilize a hybrid loss function combining adversarial loss with MSE for continuous features and cross-entropy for categorical features.

- **MissDiff (Diffusion Imputation Nets)(Ouyang et al., 2023)**

  We employ a diffusion probabilistic model specifically adapted for tabular missing data. The model is trained to reverse a noise-adding process. During inference (imputation), we utilize the *guided sampling* or *conditioning* strategy: at each denoising step t, the known observed values $\mathbf{x}^{obs}$ ($\mathbf{x}^{obs} = \mathbf{x} \odot \mathbf{m} + \text{na} \odot (1 - \mathbf{m})$) are re-injected into the sample to ensure consistency with the ground truth. The model effectively samples $\mathbf{x}^{imp}$ from the conditional distribution $p(\mathbf{x}^{miss}|\mathbf{x}^{obs})$.

Interpretable Models.

- **BRCG** (Dash et al., 2018) is an integer program designed to trade classification accuracy for rule simplicity. It uses column generation to search over an exponential number of candidate clauses efficiently.
- **LEN** (Barbiero et al., 2022) is an end-to-end differentiable method for extracting logical explanations from neural networks using First-Order Logic.
- **DR-NET** (Qiao et al., 2021) is a method for learning independent logical rules in disjunctive standard form as an interpretable model for classification.
- **RRL** (Wang et al., 2021) learns interpretable non-fuzzy rules for data representation and classification using a novel training method called Gradient Grafting.

E Performance under different missingness mechanisms

We compare three general missingness mechanisms for dataset generation:

- **MCR (Missing Completely at Random)**: The probability of being missing is the same for all cases, which is the missingness mechanism in other experiments on our paper.
- **MAR (Missing at Random)**: Missingness depends on observed variables. We can indicate which observed variable to use for missingness; the default is X_0. Then, we set a higher probability of missing when the dependency variable is 1.
- **MNAR (Missing Not at Random)**: Missingness depends on unobserved variables or the missing values themselves. Take X_3 for example, we set it is more likely to be missing when $X_3 = 1$ (positive values are harder to observe).

We show an observation ratio = 0.2 and a sample size = 50,000 as a representative case in Table 8. We run 20 random seeds. Since the seeds are different from those used in Tables 23 and 24, the results are slightly different.

Table 8: Comparison of inference accuracy and rule accuracy under different missing mechanisms.

	MCAR		MAR		MNAR	
	Imputation Accu.	Rule Accu.	Imputation Accu.	Learned Rules	Imputation Accu.	Rule Accu.
X_3	1.00 ± 0.00	1.0	1.00 ± 0.00	1.0000	1.00 ± 0.00	1.0000
X_4	1.00 ± 0.00	1.0	1.00 ± 0.00	1.0000	1.00 ± 0.00	1.0000
X_5	0.95 ± 0.07	0.6	0.95 ± 0.07	0.6000	0.93 ± 0.05	0.4000

The results show that MAR and MNAR show comparable results to MCAR, which demonstrates our method's effectiveness across the full spectrum of missing data scenarios.

F Running Time and Memory Cost Analysis

F.1 Synthetic Dataset

While coordinate descent requires different cycle numbers (Table 3), our method demonstrates efficient performance on standard CPU configurations. We conducted experiments using an Apple M4 chip with 10 cores and 16GB memory, taking observation ratio = 0.2 as an example. Results over 20 runs on setting (b) of Figure 3.

Table 9: Running time and memory cost of our model with varying sample sizes. Results over 20 seeds on the example (b) of Figure 3.

Sample size	**2500**	**5000**	**10,000**	**25,000**	**50,000**	**100,000**
Running time (s)	15.66±3.48	30.17±2.12	54.49±15.98	130.36±45.80	194.59±98.53	493.99±152.81
Memory cost (MB)	64.84±10.72	71.92±0.82	78.55±1.96	95.81±12.13	126.99±26.97	175.64±33.93

Overall, we observe **minimal time and memory costs**. Time complexity scales near-linearly with increasing sample size, while memory requirements remain modest even for large datasets. Processing 100,000 samples in under 9 minutes demonstrates strong efficiency for CPU-based execution.

F.2 Real-world Dataset

Table 10: Comparison of running time and memory cost across different methods in SPECT dataset.

Method	**Running time (s)**	**Memory cost (MB)**
MLP	0.16 ± 0.02	158.60 ± 0.10
LEN	0.20 ± 0.00	102.78 ± 0.01
RRL	16.23 ± 0.01	132.59 ± 0.01
BRCG	2.65 ± 0.27	135.11 ± 0.08
DR-NET	89.01 ± 0.06	45.93 ± 0.30
NS-FCN (Ours)	10.34 ± 0.30	61.42 ± 1.02

We conducted a comparative analysis of our proposed NS-FCN model against baseline methods, focusing on computational efficiency. We take SPECT dataset as an example. The results in Table 10 demonstrate that NS-FCN achieves a competitive balance between performance and resource consumption. While methods like MLP and LEN offer the fastest execution times, they use higher memory costs. Our NS-FCN, though not the fastest, maintains a considerably minimal memory cost and running time.

G Assessment of Rule Quality

G.1 Structural Stability.

To quantify the structural stability and reliability of the learned rules, we measure the consistency of rule predicates across different random seeds using the Jaccard index. For each rule, we treat the set of instances that satisfy its predicates in a given run as a binary mask, and compute pairwise Jaccard indices between runs obtained under different random seeds and observation probabilities. The **Jaccard index**, defined as the intersection over union of two predicate sets, provides a natural measure of similarity between rule structures learned across independent runs. High mean Jaccard scores (close to 1.0) indicate that the learned rules are structurally stable and robust to stochasticity in training and sampling, whereas lower scores reveal predicates whose semantics are more sensitive to noise or initialization.

Synthetic Dataset (Figure 3 (b)). As shown in Table 11, rules X_3 and X_4 achieve perfect Jaccard indices of 1.0 across all observation probabilities, demonstrating complete structural stability. In contrast, the aggregated X_5 rule exhibits more variability (ranging from 0.60 to 0.76), reflecting the increased complexity of learning disjunctive rule structures. In this way, structural stability—measured via the Jaccard index of predicates across runs—provides a complementary notion of reliability that focuses on the consistency of the learned logical structure rather than solely on predictive performance.

Table 11: Jaccard index of learned rule predicates on synthetic data under different observation probabilities. Example (b) of Figure 3 with 50,000 samples over 20 seeds.

Obs. Ratio	X_3	X_4	X_5
0.1	1.0000 ± 0.0000	1.0000 ± 0.0000	0.7572 ± 0.2526
0.2	1.0000 ± 0.0000	1.0000 ± 0.0000	0.5987 ± 0.2837
0.3	1.0000 ± 0.0000	1.0000 ± 0.0000	0.6726 ± 0.2385

Table 12: Jaccard Index of learned predicates across different sample sizes on the Birds dataset. Results over 10 seeds.

Sample Size	abnormal_clause1	abnormal_clause2	can_fly
	(*ostrich*)	(*bird* $\wedge$ *wounded*)	(*bird* $\wedge \neg$*abnormal_bird*)
100	0.8000 ± 0.2449	0.5000 ± 0.3162	0.5000 ± 0.3162
500	0.8000 ± 0.2449	0.8000 ± 0.2449	0.7000 ± 0.2449
1000	0.8000 ± 0.2449	0.6000 ± 0.3000	0.6000 ± 0.3000
1500	0.8187 ± 0.2404	0.6868 ± 0.3024	0.7967 ± 0.2670
2000	0.8000 ± 0.2449	0.6000 ± 0.3000	0.6000 ± 0.3000

Birds Dataset. We analyze the consistency of learned rule structures in Birds Dataset (Figure 5). Table 12 presents the Jaccard indices across all pairwise comparisons between seeds for different sample sizes, where *abnormal_clause1* and *abnormal_clause2* correspond to the two conjunctive clauses in the disjunctive rule for *abnormal_bird*: *abnormal_bird* $\leftarrow$ *ostrich* $\vee$ (*bird* $\wedge$ *wounded*). The results demonstrate that, with the exception of $n = 100$ where the sample size is insufficient, the model achieves good consistency (Jaccard index > 0.60) across all rules and sample sizes. Overall, $n = 1500$ yields the best consistency, with *abnormal_clause1* reaching 0.8187 and *can_fly* reaching 0.7967, indicating that this sample size provides an optimal balance between data availability and model stability.

Table 13: Structural stability of learned prediction rules on the Heart Disease dataset.

Metric	**Value**
Mean Pairwise Jaccard Index	0.4151 ± 0.0994
Most Frequently Selected Features	
restecg_1.0 (ST-T wave abnormality)	$9/10$ runs
thal_3.0 (normal thalassemia)	$8/10$ runs
ca_3.0 (3 major vessels colored)	$8/10$ runs
thalach (maximum heart rate achieved)	$7/10$ runs

Heart Disease dataset. For this real-world dataset, where ground-truth rules are unknown, we evaluate structural stability by computing the Jaccard index of selected features across all prediction rules learned under different random seeds. Table 13 shows that the model achieves moderate consistency (Jaccard index 0.4151 ± 0.0994), indicating that while different seeds may select varying feature combinations, there is substantial overlap in the most important features. The most frequently selected features include *restecg_1.0* (ST-T wave abnormality on resting electrocardiogram), *thal_3.0* (normal thalassemia, a blood disorder), *ca_3.0* (three major vessels colored by fluoroscopy, indicating severe coronary artery disease), and *thalach* (maximum heart rate achieved during exercise). These features align with established clinical risk factors for heart disease, suggesting that the model successfully identifies medically relevant features despite the lack of explicit rule supervision.

G.2 Rule Length Analysis

To understand the sensitivity of our framework to the rule structure hyperparameters, we conduct ablation studies on the Heart Disease dataset, systematically varying the arity of conjunction (h) and the number of conjunctive clauses (R_k).

We find that both h and R_k show optimal performance in a wide range. For instance, $h \in [3, 9]$ and $R_k \in [5, 20]$, showing that except for very small h and R_k, our model is able to capture the logic structure within the dataset. Besides, the number of disjunctive clauses is more critical than the arity of individual conjunctions for this dataset. This aligns with the intuition that complex real-world decision boundaries often require multiple alternative rules rather than highly complex single rules.

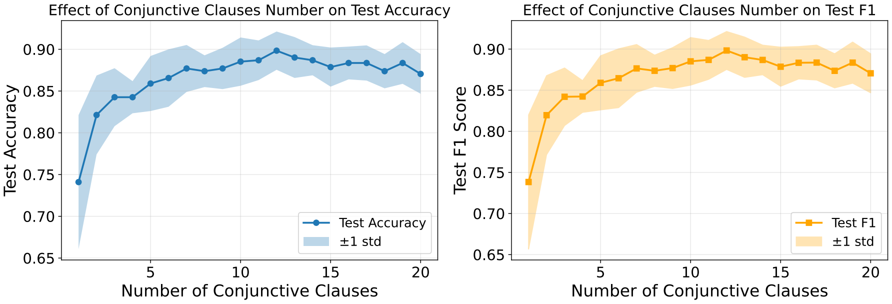


Figure 6: Classification accuracy for heart disease risk under the effect of the number of conjunction arity (h). Results are over 10 seeds.

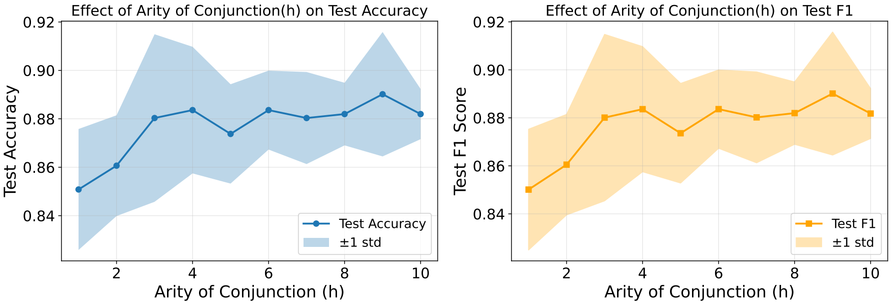


Figure 7: Classification accuracy for heart disease risk under the effect of the number of conjunctive Clauses (R_k). Results are over 10 seeds.

H Analysis of Temperature in Soft Operators at Equations 4, 5

To validate robustness, we conducted a sensitivity analysis on Figure 3(b) with 20,000 samples and 0.2 observation ratio.

Soft-AND (τ). Table 14 shows that the model maintains high accuracy when τ is small (e.g., $\tau \in [0.01, 0.20]$). This is expected because as $\tau \to 0$, Softmin approximates the hard $\min$ logic required for strict conjunctions. Performance degrades only when τ becomes too large ($\tau \geq 1.0$), where the operator becomes too "soft" to capture the decisive logical boundaries. Thus, a small constant temperature (e.g., $\tau = 0.1$) is a safe and effective default.

Soft-OR (β). Table 15 reports the imputation accuracy under varying constant β values. The results demonstrate that our model is highly robust to β: it achieves near-perfect accuracy for all latent predicates (X_3, X_4, X_5) across a wide range, specifically for $\beta \geq 3$. This aligns with the theoretical property that as $\beta \to \infty$, the LogSumExp function approximates the hard $\max$ operator. In practice, any sufficiently large provides a strong gradient signal for discrimination while maintaining differentiability.

Table 14: Imputation accuracy for latent predicates X_3, X_4, X_5 under different softmin temperatures (with fixed $\beta = 10$ over 20 random seeds).

τ in Equation 4	Imputation Acc. X3	Imputation Acc. X4	Imputation Acc. X5
0.01	1.000 ± 0.000	1.000 ± 0.000	0.965 ± 0.063
0.02	1.000 ± 0.000	1.000 ± 0.000	0.987 ± 0.034
0.05	1.000 ± 0.000	1.000 ± 0.000	0.939 ± 0.061
0.10	1.000 ± 0.000	1.000 ± 0.000	0.958 ± 0.076
0.20	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
0.50	1.000 ± 0.000	1.000 ± 0.000	0.858 ± 0.038
1.00	0.928 ± 0.122	0.892 ± 0.134	0.776 ± 0.027
2.00	0.751 ± 0.002	0.839 ± 0.157	0.786 ± 0.008
5.00	0.769 ± 0.111	0.750 ± 0.003	0.772 ± 0.083
10.00	0.752 ± 0.003	0.770 ± 0.111	0.802 ± 0.064
20.00	0.697 ± 0.098	0.733 ± 0.046	0.799 ± 0.064
50.00	0.733 ± 0.049	0.733 ± 0.046	0.798 ± 0.077
100.00	0.750 ± 0.004	0.698 ± 0.064	0.803 ± 0.069

Table 15: Imputation accuracy for latent predicates X_3, X_4, X_5 under different constant temperature values β (with fixed $\tau = 0.1$ over 20 random seeds).

β of Equation 5	Imputation Acc. X3	Imputation Acc. X4	Imputation Acc. X5
0.1	1.000 ± 0.000	1.000 ± 0.000	0.218 ± 0.002
0.2	1.000 ± 0.000	1.000 ± 0.000	0.218 ± 0.002
0.5	1.000 ± 0.000	1.000 ± 0.000	0.218 ± 0.002
1	1.000 ± 0.000	1.000 ± 0.000	0.217 ± 0.002
2	1.000 ± 0.000	1.000 ± 0.000	0.870 ± 0.117
3	1.000 ± 0.000	1.000 ± 0.000	0.942 ± 0.056
4	1.000 ± 0.000	1.000 ± 0.000	0.866 ± 0.057
5	0.965 ± 0.093	1.000 ± 0.000	0.894 ± 0.056
10	0.965 ± 0.093	1.000 ± 0.000	0.915 ± 0.040
15	0.965 ± 0.092	1.000 ± 0.000	0.899 ± 0.072
20	0.966 ± 0.091	1.000 ± 0.000	0.928 ± 0.057
25	0.965 ± 0.092	1.000 ± 0.000	0.886 ± 0.060
30	1.000 ± 0.000	1.000 ± 0.000	0.899 ± 0.088
35	0.965 ± 0.093	1.000 ± 0.000	0.880 ± 0.076
40	1.000 ± 0.000	1.000 ± 0.000	0.889 ± 0.075
45	0.964 ± 0.094	0.965 ± 0.093	0.909 ± 0.051
50	1.000 ± 0.000	1.000 ± 0.000	0.917 ± 0.064
100	0.965 ± 0.094	1.000 ± 0.000	0.916 ± 0.064
200	1.000 ± 0.000	1.000 ± 0.000	0.908 ± 0.070

Conclusion. Our framework does not rely on carefully hyperparameter tuning. A moderate to large β for Soft-OR and a small τ for Soft-AND consistently yield optimal results. Thus, we use $\tau = 0.1$ and $\beta = 10$ as temperature parameters for all our experiments. Furthermore, complex scheduling strategies like cosine annealing can be employed if constant temperature are not good enough.

I Sensitivity Analysis with Label Noise and Missing ratio

I.1 Robustness Analysis with Label Noise

To assess the robustness of our framework against data inconsistencies and imperfect logical dependencies, we conducted experiments by injecting label noise into the latent predicates.

Specifically, we first generate the ground-truth latent predicates X_3, X_4, X_5 following the perfect logical rules (e.g., $X_3 = X_0 \wedge X_1$). Then, we introduce stochasticity by flipping the binary labels of these latent predicates with a probability $p_{noise} \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. **This setup simulates real-world scenarios where logical rules may have exceptions or where the observed data contains errors, directly challenging the model's ability to distill consistent symbolic rules from**

noisy supervision. Tables 16 and 17 present the learned rule structures and their corresponding imputation accuracies under varying noise ratios. We use Figure 3 (b) as the representative example with an observation ratio of 0.3 and sample sizes of 20,000.

In the noise-free setting ($p_{noise} = 0.0$), our model perfectly recovers the ground-truth rules for the simpler conjunctive predicates X_3 and X_4 (with rule accuracy 1.00), achieving perfect imputation accuracy (1.000). For the more complex disjunctive rule X_5, the model achieves a rule accuracy of 0.50 and an imputation accuracy of 0.955 after fine-tuning, indicating that while the exact ground-truth structure is harder to isolate, the learned approximations maintain strong predictive performance.

Remarkably, the model demonstrates strong robustness at low-to-moderate noise levels ($p_{noise} \leq 0.3$). At $p_{noise} = 0.1$ and 0.2, the ground-truth rules (underlined in the table) for X_3 and X_4 are perfectly recovered (rule accuracy 1.00) with near-perfect imputation accuracies; for the complex multi-hop rules of X_5, the ground-truth rules frequently emerge as the dominant learned structures (with rule accuracy above 0.5). Even at $p_{noise} = 0.3$, the model maintains high rule accuracy (0.85) for both X_3 and X_4, with imputation accuracies above 0.95; for X_5, the rule accuracy decreases to 0.2 at $p_{noise} = 0.3$, but the imputation accuracy remains at 0.828, suggesting that **the model learns valid approximations (e.g., capturing one correct disjunctive branch) that preserve predictive power.**

As noise increases beyond 0.3, the performance degrades more significantly. At $p_{noise} = 0.4$, rule accuracies drop to 0.85 and 0.6 for X_3 and X_4 respectively, while X_5 fails to recover the correct structure (rule accuracy 0.00). At $p_{noise} = 0.5$, the model struggles to learn meaningful rules, with rule accuracies in $[0.0, 0.1]$ for all predicates. However, the imputation accuracies remain above 0.70 even at these high noise levels, indicating that the learned approximations, while not perfectly matching the ground-truth rules, still provide useful predictive signals.

The imputation accuracy degrades gracefully as noise increases, rather than collapsing abruptly, indicating that **the soft-logic relaxation effectively prevents the model from overfitting to noise**, allowing it to capture the dominant logical signals within the data. The fine-tuning step for X_5 consistently improves imputation accuracy across all noise levels, demonstrating the effectiveness of the iterative refinement process.

Table 16: Impact of label noise on rule learning and missing value imputation performance. Results are over 20 random seeds.

Noise Ratio	**Avg. Imputation Accu. (Before Fine-tune)**	**Avg. Imputation Accu. (After Fine-tune)**	**Train Loss (Before Fine-tune)**	**Train Loss (After Fine-tune)**
0.0	$X_3 : 1.000 \pm 0.000$	/	$X_3 : 0.001 \pm 0.000$	/
	$X_4 : 1.000 \pm 0.000$	/	$X_4 : 0.001 \pm 0.000$	/
	$X_5 : 0.907 \pm 0.050$	$X_5 : 0.955 \pm 0.049$	$X_5 : 0.089 \pm 0.035$	$X_5 : 0.067 \pm 0.031$
0.1	$X_3 : 1.000 \pm 0.000$	/	$X_3 : 0.098 \pm 0.003$	/
	$X_4 : 1.000 \pm 0.000$	/	$X_4 : 0.099 \pm 0.004$	/
	$X_5 : 0.948 \pm 0.042$	$X_5 : 0.946 \pm 0.038$	$X_5 : 0.168 \pm 0.026$	$X_5 : 0.123 \pm 0.021$
0.2	$X_3 : 0.975 \pm 0.076$	/	$X_3 : 0.193 \pm 0.005$	/
	$X_4 : 0.987 \pm 0.057$	/	$X_4 : 0.193 \pm 0.007$	/
	$X_5 : 0.894 \pm 0.050$	$X_5 : 0.902 \pm 0.046$	$X_5 : 0.260 \pm 0.008$	$X_5 : 0.204 \pm 0.012$
0.3	$X_3 : 0.950 \pm 0.103$	/	$X_3 : 0.282 \pm 0.006$	/
	$X_4 : 0.987 \pm 0.056$	/	$X_4 : 0.282 \pm 0.008$	/
	$X_5 : 0.822 \pm 0.046$	$X_5 : 0.824 \pm 0.066$	$X_5 : 0.320 \pm 0.006$	$X_5 : 0.266 \pm 0.006$
0.4	$X_3 : 0.863 \pm 0.127$	/	$X_3 : 0.360 \pm 0.009$	/
	$X_4 : 0.862 \pm 0.128$	/	$X_4 : 0.357 \pm 0.008$	/
	$X_5 : 0.792 \pm 0.078$	$X_5 : 0.786 \pm 0.065$	$X_5 : 0.370 \pm 0.006$	$X_5 : 0.311 \pm 0.006$
0.5	$X_3 : 0.745 \pm 0.085$	/	$X_3 : 0.418 \pm 0.007$	/
	$X_4 : 0.725 \pm 0.077$	/	$X_4 : 0.421 \pm 0.007$	/
	$X_5 : 0.761 \pm 0.057$	$X_5 : 0.767 \pm 0.072$	$X_5 : 0.421 \pm 0.007$	$X_5 : 0.349 \pm 0.007$

I.2 Missing ratio

In three synthetic datasets, we have varied the missing ratio in $\{0.7, 0.8, 0.9\}$ in the above results.

Table 17: Learned rule structures under label noise. Ground truth rules are indicated with underlines. Results are over 20 random seeds.

Noise Ratio	Learned Rule Structure	Rule Accu.
0.0	$X_3 : \underline{X_0 \wedge X_1}$ $X_4 : \underline{X_2 \wedge X_7}$ $X_5 : \underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}, (X_3 \wedge X_4) \vee (X_3 \wedge X_6), (X_0 \wedge X_4) \vee (X_0 \wedge X_4), (X_0 \wedge X_4) \vee (X_1 \wedge X_6), (X_1 \wedge X_3) \vee (X_3 \wedge X_6)$	$X_3 : 1.00$ $X_4 : 1.00$ $X_5 : 0.50$
0.1	$X_3 : \underline{X_0 \wedge X_1}$ $X_4 : \underline{X_2 \wedge X_7}$ $X_5 : \underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}, (X_3 \wedge X_4) \vee (X_3 \wedge X_6), (X_0 \wedge X_4) \vee (X_0 \wedge X_4), (X_0 \wedge X_4) \vee (X_2 \wedge X_3), (X_0 \wedge X_2) \vee (X_3 \wedge X_6)$	$X_3 : 1.00$ $X_4 : 1.00$ $X_5 : 0.55$
0.2	$X_3 : \underline{X_0 \wedge X_1}$ $X_4 : \underline{X_2 \wedge X_7}$ $X_5 : \underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}, (X_0 \wedge X_4) \vee (X_3 \wedge X_4), (X_0 \wedge X_1) \vee (X_3 \wedge X_6), (X_0 \wedge X_4) \vee (X_0 \wedge X_7), (X_0 \wedge X_4) \vee (X_0 \wedge X_6)$	$X_3 : 1.00$ $X_4 : 1.00$ $X_5 : 0.60$
0.3	$X_3 : \underline{X_0 \wedge X_1}, X_1 \wedge X_2, X_1 \wedge X_1, X_1 \wedge X_7$ $X_4 : \underline{X_2 \wedge X_7}, X_2 \wedge X_6, X_2 \wedge X_2$ $X_5 : \underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}, (X_0 \wedge X_7) \vee (X_3 \wedge X_6), (X_0 \wedge X_3) \vee (X_0 \wedge X_4), (X_3 \wedge X_6) \vee (X_4 \wedge X_7), (X_0 \wedge X_1) \vee (X_0 \wedge X_6)$	$X_3 : 0.85$ $X_4 : 0.85$ $X_5 : 0.20$
0.4	$X_3 : \underline{X_0 \wedge X_1}, X_1 \wedge X_1, X_1 \wedge X_2, X_0 \wedge X_6, X_0 \wedge X_0$ $X_4 : \underline{X_2 \wedge X_7}, X_7 \wedge X_7, X_0 \wedge X_2, X_0 \wedge X_0, X_2 \wedge X_6$ $X_5 : (X_0 \wedge X_2) \vee (X_4 \wedge X_6), (X_0 \wedge X_6) \vee (X_3 \wedge X_3), (X_0 \wedge X_2) \vee (X_2 \wedge X_3), (X_0 \wedge X_6) \vee (X_1 \wedge X_7), (X_1 \wedge X_4) \vee (X_4 \wedge X_4)$	$X_3 : 0.60$ $X_4 : 0.60$ $X_5 : 0.00$
0.5	$X_3 : X_1 \wedge X_7, X_2 \wedge X_2, X_6 \wedge X_6, X_0 \wedge X_2, \underline{X_0 \wedge X_1}$ $X_4 : X_6 \wedge X_6, X_0 \wedge X_6, X_0 \wedge X_7, X_6 \wedge X_7, X_1 \wedge X_2$ $X_5 : (X_0 \wedge X_3) \vee (X_1 \wedge X_6), (X_0 \wedge X_2) \vee (X_2 \wedge X_7), (X_0 \wedge X_0) \vee (X_2 \wedge X_7), (X_0 \wedge X_3) \vee (X_4 \wedge X_7), (X_2 \wedge X_7) \vee (X_6 \wedge X_7)$	$X_3 : 0.10$ $X_4 : 0.05$ $X_5 : 0.00$

In real-world datasets, to assess the model's robustness under different levels of data scarcity, we evaluated its performance on the SPECT and Heart Disease dataset while varying the observation ratio from 0.3 to 0.9 (i.e. missing ratio from 0.1 to 0.7).

As shown in Tables 18 and 19, the model's accuracy remains acceptable and improves consistently as more data becomes available. Notably, in SPECT, even with only 30% of the data observed (a 70% missing ratio), the model maintains a high F1 score of 0.751, demonstrating its capability to learn meaningful diagnostic rules from highly incomplete datasets.

For the Birds Dataset, we fix the observation ratio as 0.1 (i.e. 90% missingness) and show results over different number of training samples. Results in Table 20 show that a few hundred samples are sufficient for the model to converge to the correct logical truth.

Table 18: Performance on the SPECT dataset with varying observation ratios.

Observation Ratio	Imputation Acc.	Diagnosis Acc.	Diagnosis F1
0.3	0.501	0.679	0.751
0.5	0.630	0.765	0.808
0.7	0.763	0.920	0.958
0.9	0.791	0.929	0.960

Table 19: Imputation accuracy for Heart Disease under different observation ratios.

Observation Ratio	Overall	trestbps	chol	thalach	oldpeak
0.3	0.6444	0.7129	0.6304	0.7393	0.4950
0.5	0.7434	0.8053	0.7558	0.7954	0.6172
0.7	0.8432	0.8647	0.8482	0.9043	0.7558
0.9	0.9439	0.9439	0.9307	0.9769	0.9241

J Additional Synthetic Experiments Results

J.1 Main Results Supplement of Example (b) of Figure 3.

Dataset Generation. The base variables $\{X_0, X_1, X_2, X_6, X_7\}$ are independently generated from a Bernoulli distribution, each with $p = 0.5$. Subsequently, the values for $\{X_3, X_4, X_5\}$ are deterministically derived using the ground truth logical rules depicted in Figure 3. Specifically, these rules are:

$$X_3 \leftarrow X_0 \wedge X_1$$
$$X_4 \leftarrow X_2 \wedge X_7$$
$$X_5 \leftarrow (X_3 \wedge X_6) \vee (X_4 \wedge X_0)$$

Table 20: Impact of training sample size on the imputation accuracy of latent predicates (`abnormal`, `fly`) in the Birds domain. Results are reported as mean ± std over 10 random seeds, evaluated with 10% observation probability.

# Samples	Acc. Abnormal Bird	Acc. Can Fly
100	0.896 ± 0.058	0.845 ± 0.148
500	0.976 ± 0.054	0.928 ± 0.066
1000	0.951 ± 0.067	0.928 ± 0.066
1500	0.949 ± 0.070	0.952 ± 0.066
2000	0.951 ± 0.067	0.928 ± 0.066

Finally, to introduce missing information, a portion of the values for X_3, X_4, and X_5 are randomly masked. These masked variables become the targets for imputation. In our experiments, we vary the level of missingness, applying masking probabilities of 70%, 80%, and 90% to these target variables (corresponding to observation ratios of 30%, 20%, and 10%, respectively).

Main Results. As demonstrated in a previous case study (Table 3, which shows three runs using the same seed but different internal rule optimization orders), variations in the rule optimization sequence within a single seed can affect training efficiency. We thus show the coordinate descent training progress under a different random optimization order from Figure 4 here in Figure 9. In this run, the optimization order is $[X_5, X_4, X_3]$ for cycle 1 and $[X_4, X_5, X_3]$ for cycle 2. Given such different learning trajectories, our model still discovers the correct rules successfully.

Furthermore, random initialization across different seeds can lead to the discovery of varied rule sets, and occasionally, the model might converge to a local optimum. However, as the analysis of convergence before, performing multiple runs with different initializations enhances the probability of identifying the global optimal solution. Our findings indicate that the model can finds global optima several times within 20 random seeds (Tables 23 and 24).

Learning Efficiency. As the observation ratio decreases, the guidance signal becomes less informative, reducing both rule structure recovery and missing value imputation. We also evaluated the model's performance with a smaller training set of 10,000 samples. The results, detailed in Tables 21 and 22, demonstrate that our model maintains high accuracy for simple AND rule learning and predicate inference. Even for challenging OR rule learning, the model successfully identifies most body predicates. We further investigated the impact of dataset sample size, varying it from 1,000 to 20,000 samples. As shown in Figure 8, the most efficient setting we can recover the OR rule for X_5 is to use an observation ratio of 0.1 and a dataset of 4,000 samples. For the simpler AND rules governing X_3 and X_4, correct rule structures could be learned with 1,000 or even smaller samples and a 0.1 observation ratio.

Table 21: Summary of synthetic data experiment results for example (b) of the Figure 3. Each observation ratio is evaluated using 10,000 samples and results are averaged over 20 random seeds.

Obs. Ratio	**Avg. Imputation Accu. (Before Fine-tune)**	**Avg. Imputation Accu. (After Fine-tune)**	**Train Loss (Before Fine-tune)**	**Train Loss (After Fine-tune)**
0.3	$X_3 : 0.91 \pm 0.014$ $X_4 : 0.93 \pm 0.013$ $X_5 : 0.87 \pm 0.003$	/ / $X_5 : 0.88 \pm 0.002$	$X_3 : 0.069 \pm 0.007$ $X_4 : 0.054 \pm 0.006$ $X_5 : 0.110 \pm 0.002$	/ / $X_5 : 0.103 \pm 0.000$
0.2	$X_3 : 0.91 \pm 0.014$ $X_4 : 0.90 \pm 0.015$ $X_5 : 0.86 \pm 0.003$	/ / $X_5 : 0.87 \pm 0.003$	$X_3 : 0.067 \pm 0.007$ $X_4 : 0.072 \pm 0.007$ $X_5 : 0.116 \pm 0.002$	/ / $X_5 : 0.105 \pm 0.001$
0.1	$X_3 : 0.90 \pm 0.015$ $X_4 : 0.91 \pm 0.014$ $X_5 : 0.85 \pm 0.003$	/ / $X_5 : 0.88 \pm 0.002$	$X_3 : 0.075 \pm 0.007$ $X_4 : 0.063 \pm 0.006$ $X_5 : 0.124 \pm 0.002$	/ / $X_5 : 0.107 \pm 0.001$

Table 22: Summary of learned rule structures and accuracy for example (b) of Figure 3. Each observation ratio is evaluated using 10,000 samples, with results averaged over 20 random seeds. We present the top 3 learned rule structures in order of discovery accuracy. Rule accuracy indicates the percentage of 20 runs in which a rule was learned completely correctly.

Obs. Ratio	Learned Rule Structure	Rule Accu.
0.3	$X_3 : \underline{X_0 \wedge X_1}, X_0 \wedge X_6, X_1 \wedge X_7$	$X_3 : 0.65$
	$X_4 : \underline{X_2 \wedge X_7}, X_6 \wedge X_7, X_7$	$X_4 : 0.70$
	$X_5 : \underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}, (X_3 \wedge X_4) \vee (X_3 \wedge X_6), (X_0 \wedge X_1) \vee (X_0 \wedge X_4)$	$X_5 : 0.10$
0.2	$X_3 : \underline{X_0 \wedge X_1}, X_0, X_0 \wedge X_2$	$X_3 : 0.65$
	$X_4 : \underline{X_2 \wedge X_7}, X_2, X_6 \wedge X_7$	$X_4 : 0.60$
	$X_5 : \underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}, (X_3 \wedge X_4) \vee (X_3 \wedge X_6), (X_0) \vee (X_1 \wedge X_6)$	$X_5 : 0.10$
0.1	$X_3 : \underline{X_0 \wedge X_1}, X_0 \wedge X_2, X_0 \wedge X_6$	$X_3 : 0.60$
	$X_4 : \underline{X_2 \wedge X_7}, X_2, X_6 \wedge X_7$	$X_4 : 0.65$
	$X_5 : (X_3 \wedge X_4) \vee (X_3 \wedge X_6), (X_0 \wedge X_1) \vee (X_0 \wedge X_4), \underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}$	$X_5 : 0.10$

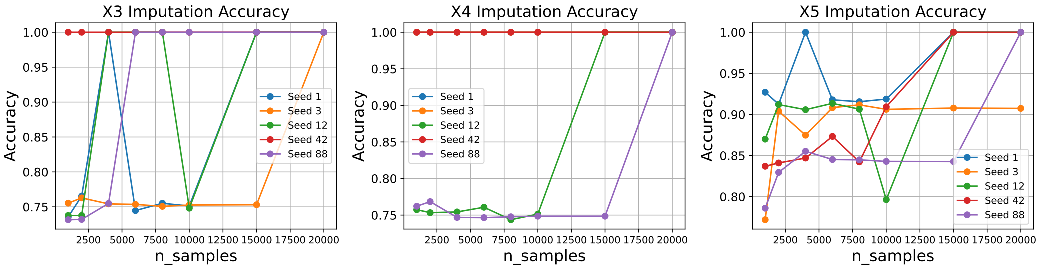


Figure 8: Imputation accuracy versus dataset sample size for Figure 3 (b). For these experiments, 10% of the data was observed (i.e., a 90% missing ratio) for predicates in X_3, X_4, and X_5.

J.2 Results of Example (a) of Figure 3

Dataset Generation. The base variables $\{X_0, X_1, X_2, X_6\}$ are independently generated from a Bernoulli distribution, each with $p = 0.5$. Subsequently, the values for $\{X_3, X_4, X_5\}$ are deterministically derived using the ground truth logical rules depicted in Figure 3. Specifically, these rules are:

$$X_3 \leftarrow X_0 \wedge X_1$$
$$X_4 \leftarrow X_2 \wedge X_3$$
$$X_5 \leftarrow X_4 \wedge X_6$$

Finally, to introduce missing information, a portion of the values for X_3, X_4, and X_5 are randomly masked. These masked variables become the targets for imputation. In our experiments, we vary the

Table 23: Summary of synthetic data experiment results for example (b) of the Figure 3. Evaluated on 50,000 samples and results are averaged over 20 random seeds.

Obs. Ratio	Avg. Imputation Accu. (Before Fine-tune)	Avg. Imputation Accu. (After Fine-tune)	Train Loss (Before Fine-tune)	Train Loss (After Fine-tune)
0.3	$X_3 : 0.98 \pm 0.006$	/	$X_3 : 0.024 \pm 0.003$	/
	$X_4 : 1.00 \pm 0.000$	/	$X_4 : 0.005 \pm 0.000$	/
	$X_5 : 0.94 \pm 0.003$	$X_5 : 0.96 \pm 0.003$	$X_5 : 0.054 \pm 0.002$	$X_5 : 0.065 \pm 0.001$
0.2	$X_3 : 1.00 \pm 0.000$	/	$X_3 : 0.005 \pm 0.000$	/
	$X_4 : 0.95 \pm 0.010$	/	$X_4 : 0.041 \pm 0.005$	/
	$X_5 : 0.93 \pm 0.003$	$X_5 : 0.96 \pm 0.002$	$X_5 : 0.063 \pm 0.003$	$X_5 : 0.067 \pm 0.001$
0.1	$X_3 : 1.00 \pm 0.000$	/	$X_3 : 0.005 \pm 0.000$	/
	$X_4 : 1.00 \pm 0.000$	/	$X_4 : 0.005 \pm 0.000$	/
	$X_5 : 0.93 \pm 0.003$	$X_5 : 0.94 \pm 0.002$	$X_5 : 0.056 \pm 0.002$	$X_5 : 0.073 \pm 0.001$

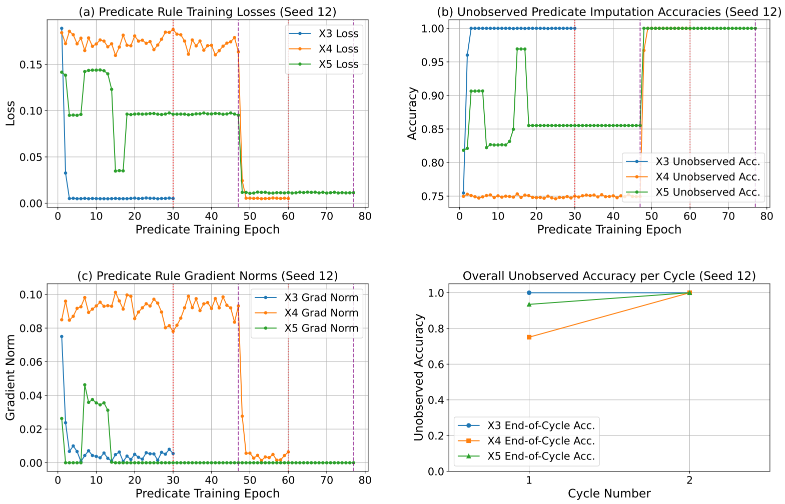


Figure 9: Training dynamics for a representative run (Obs. Ratio = 0.3) of Figure 3 (b). The optimization order: [X_5, X_4, X_3] for Cycle 1; [X_4, X_5, X_3] for Cycle 2. Subplots display: (a) training losses, (b) unobserved imputation accuracies, and (c) gradient norms for rule embeddings; (d) overall imputation accuracies each cycle. Red dashed lines indicate the conclusion of training blocks for X_3 or X_4 (each allocated 30 epochs when active within a cycle). Purple dashed lines delineate training phases for X_5 (Rule 1, Rule 2, and Fine-tune); the epoch count for these X_5 phases can vary per cycle due to the dynamic nature of the hard covering mechanism. Correct rule structures were learned for X_3 by the end of Cycle 1, and for X_4 and X_5 by the end of Cycle 2.

Table 24: Summary of learned rule structures and accuracy for example (b) of Figure 3. Each observation ratio was evaluated using 50,000 samples, with results averaged over 20 random seeds. We present the top 3 learned rule structures in order of discovery accuracy. Rule accuracy indicates the percentage of 20 runs in which a rule was learned completely correctly.

Obs.Ratio	**Learned Rule Structure**	**Rule Accu.**
0.3	$X_3 : \underline{X_0 \wedge X_1}, X_0 \wedge X_2$	$X_3 : 0.90$
	$X_4 : \underline{X_2 \wedge X_7}$	$X_4 : 1.00$
	$X_5 : \underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}, (X_3 \wedge X_4) \vee (X_3 \wedge X_6), (X_0 \wedge X_4) \vee (X_1 \wedge X_3)$	$X_5 : 0.50$
0.2	$X_3 : \underline{X_0 \wedge X_1}$	$X_3 : 1.00$
	$X_4 : \underline{X_2 \wedge X_7}, X_0 \wedge X_7, X_2$	$X_4 : 0.80$
	$X_5 : \underline{(X_0 \wedge X_4) \vee (X_3 \wedge X_6)}, (X_3 \wedge X_4) \vee (X_3 \wedge X_6), (X_0 \wedge X_1) \vee (X_0 \wedge X_4)$	$X_5 : 0.40$
0.1	$X_3 : X_0 \wedge X_1$	$X_3 : 1.00$
	$X_4 : X_2 \wedge X_7$	$X_4 : 1.00$
	$X_5 : (X_0 \wedge X_4) \vee (X_3 \wedge X_6), (X_3 \wedge X_4) \vee (X_3 \wedge X_6), (X_0 \wedge X_1) \vee (X_0 \wedge X_4)$	$X_5 : 0.30$

level of missingness, applying masking probabilities of 70%, 80%, and 90% to these target variables (corresponding to observation ratios of 30%, 20%, and 10%, respectively).

Main Results. We show the coordinate descent training progress under different random optimization order. Figure 10 demonstrates the convergence in two cycles, while Figure 11 requires three cycles to complete training.

We summarize the results for example (a) of the Figure 3 in Tables 25 and 26, which demonstrate both the effectiveness of our rule discovery approach and the precision of missing variables imputation. Our analysis reveals that learning the multi-step chain structure presents significant challenges, primarily because the algorithm uses inferred predicate values $\boldsymbol{v}^t$ from previous steps to update the current values by Eq. 3. This creates a dependency chain where suboptimal rule embeddings learned at earlier optimization steps can propagate errors to subsequent steps, potentially degrading overall performance. Despite these challenges, our model successfully identifies the correct rules in the majority of experimental runs. This robustness indicates that with multiple random initializations, the

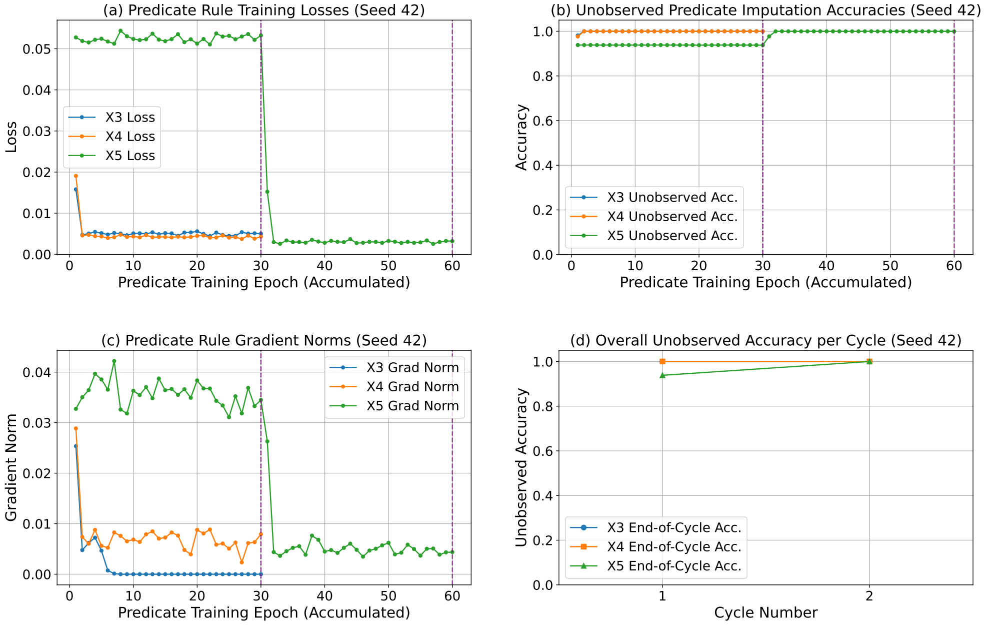


Figure 10: Training dynamics for a representative run (Observation Ratio = 0.2) of Figure 3 (a). Subplots display: (a) training losses, (b) unobserved imputation accuracies, and (c) gradient norms for rule embeddings; (d) overall imputation accuracies each cycle. Purple dashed lines indicate the conclusion of training blocks for one cycle (each allocated 30 epochs). The rule embedding optimization order: [X_5, X_3, X_4] for Cycle 1; [X_5, X_4, X_3] for Cycle 2. Correct rule structures were learned for X_3 and X_4 by the end of Cycle 1, for X_5 by the end of Cycle 2. The learned rules: $X_3 \leftarrow X_0 \wedge X_1, X_4 \leftarrow X_2 \wedge X_3, X_5 \leftarrow X_4 \wedge X_6$.

algorithm reliably converges to the optimal rule structures like the results from Figures 10 and 11, which effectively overcome the inherent difficulties of sequential dependency learning in chain-like logical structures.

Table 25: Summary of synthetic data experiment results for example (a) of the Figure 3. Each observation ratio is evaluated using 50,000 samples and results are averaged over 20 random seeds. No fine-tune phase since we assume no disjunctive rules.

Obs. Ratio	Avg. Imputation Accu.	Train Loss
0.3	$X_3 : 0.86 \pm 0.13$ $X_4 : 0.91 \pm 0.06$ $X_5 : 0.95 \pm 0.03$	$X_3 : 0.09 \pm 0.08$ $X_4 : 0.07 \pm 0.04$ $X_5 : 0.04 \pm 0.02$
0.2	$X_3 : 0.85 \pm 0.13$ $X_4 : 0.90 \pm 0.05$ $X_5 : 0.94 \pm 0.02$	$X_3 : 0.10 \pm 0.08$ $X_4 : 0.08 \pm 0.04$ $X_5 : 0.04 \pm 0.02$
0.1	$X_3 : 0.82 \pm 0.12$ $X_4 : 0.90 \pm 0.05$ $X_5 : 0.94 \pm 0.02$	$X_3 : 0.10 \pm 0.07$ $X_4 : 0.08 \pm 0.04$ $X_5 : 0.05 \pm 0.02$

J.3 Results of Example (c) of Figure 3

Dataset Generation. The base variables $\{X_0, X_1, X_2, X_6, X_7\}$ are independently generated from a Bernoulli distribution, each with $p = 0.5$. Subsequently, the values for $\{X_3, X_4, X_5, X_8\}$ are deterministically derived using the ground truth logical rules depicted in Figure 3. Specifically,

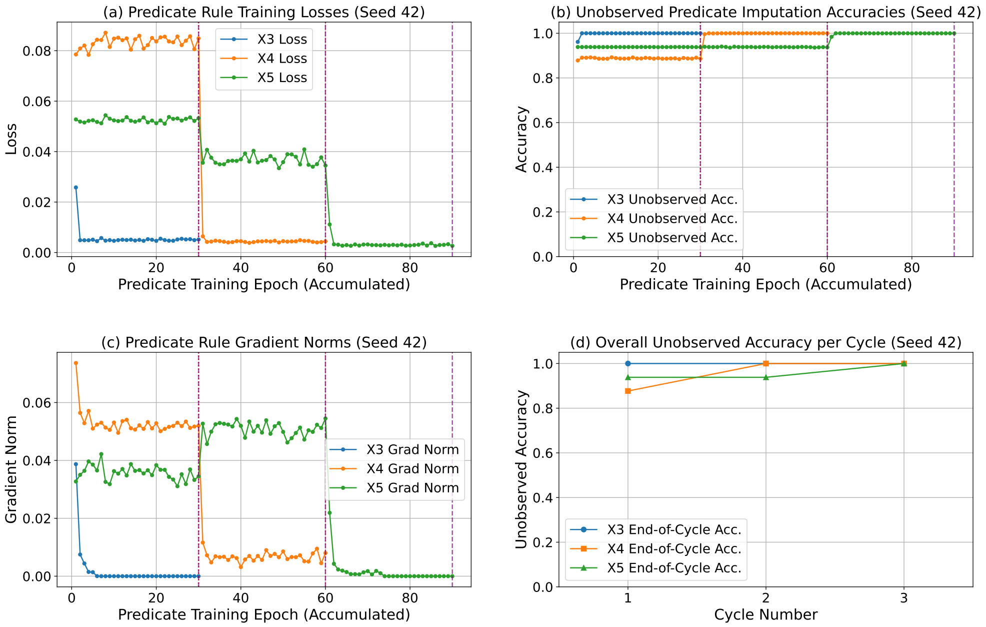


Figure 11: Training dynamics for a representative run (Observation Ratio = 0.2) of Figure 3 (a). Subplots display: (a) training losses, (b) unobserved imputation accuracies, and (c) gradient norms for rule embeddings; (d) overall imputation accuracies each cycle. Purple dashed lines indicate the conclusion of training blocks for one cycle (each allocated 30 epochs). The rule embedding optimization order: [X_5, X_4, X_3] for Cycle 1,2; [X_3, X_5, X_4] for Cycle 3. Correct rule structures were learned for X_3 by the end of Cycle 1, for X_4 by the end of Cycle 2, and for X_5 by the end of Cycle 3. The learned rules: $X_3 \leftarrow X_0 \wedge X_1, X_4 \leftarrow X_2 \wedge X_3, X_5 \leftarrow X_4 \wedge X_6$.

Table 26: Summary of learned rule structures and accuracy for example (a) of Figure 3. Each observation ratio is evaluated using 50,000 samples, with results averaged over 20 random seeds. We present the top 3 learned rule structures in order of discovery accuracy. The rules that are truth rules are indicated by underline. Rule accuracy indicates the percentage of 20 runs in which a rule was learned completely correctly.

Obs. Ratio	Learned Rule Structure	Rule Accuracy
0.3	$X_3 : \underline{X_0 \wedge X_1}, X_0 \wedge X_4, X_5$ $X_4 : \underline{X_2 \wedge X_3}, X_3 \wedge X_5, X_0 \wedge X_5$ $X_5 : \underline{X_4 \wedge X_6}, X_3 \wedge X_4, X_1 \wedge X_4$	$X_3 : 0.60$ $X_4 : 0.40$ $X_5 : 0.40$
0.2	$X_3 : \underline{X_0 \wedge X_1}, X_5, X_0 \wedge X_5$ $X_4 : \underline{X_2 \wedge X_3}, X_5 \wedge X_3, X_5$ $X_5 : \underline{X_4 \wedge X_6}, X_4, X_3 \wedge X_4$	$X_3 : 0.40$ $X_4 : 0.30$ $X_5 : 0.20$
0.1	$X_3 : \underline{X_0 \wedge X_1}, X_0 \wedge X_4, X_0 \wedge X_5$ $X_4 : \underline{X_2 \wedge X_3}, X_5, X_2 \wedge X_5$ $X_5 : \underline{X_4 \wedge X_6}, X_4 \wedge X_2, X_3 \wedge X_6$	$X_3 : 0.40$ $X_4 : 0.40$ $X_5 : 0.10$

these rules are:

$$X_3 \leftarrow X_0 \wedge X_1$$
$$X_4 \leftarrow X_2 \wedge X_7$$
$$X_8 \leftarrow X_4 \wedge X_0$$
$$X_5 \leftarrow (X_3 \wedge X_6) \vee (X_8) \vee (X_6 \wedge X_7)$$

Finally, to introduce missing information, a portion of the values for X_3, X_4, X_8 and X_5 are randomly masked. These masked variables become the targets for imputation. In our experiments, we vary the level of missingness, applying masking probabilities of 70%, 80%, and 90% to these target variables (corresponding to observation ratios of 30%, 20%, and 10%, respectively).

Main Results. We summarize the results for example (c) of the Figure 3 in the Tables 27 and 28, showcasing the effectiveness of rule discovery and the precision of missing variables imputation. We have random coordinate descent training order for rule optimization.

This task is more challenging due to the chain-like structure of the disjunctive rules, particularly with three clauses for X_5, resulting in lower learning accuracy than in example (b). Nonetheless, our method achieves the highest rule discovery accuracy for the ground-truth rules while maintaining acceptable imputation accuracy. For the most difficult prediction task (X_5), we obtain over 80% accuracy across all three observation ratios. Other predicate predictions reach $\sim$ 90% accuracy, including the chain-derived predicate X_8. For the learned rules in Table 28, we can find most body predicates are correct even for the complex three-clause rules governing X_5, which include the chain-derived predicate X_8. We also show the loss plot for one run in Figure 12.

Table 27: Summary of synthetic data experiment results for example (c) of the Figure 3. Each observation ratio is evaluated using 50,000 samples and results are averaged over 20 random seeds.

Obs. Ratio	Avg. Imputation Accu. (Before Fine-tune)	Avg. Imputation Accu. (After Fine-tune)	Train Loss (Before Fine-tune)	Train Loss (After Fine-tune)
0.3	$X_3 : 0.86 \pm 0.12$	/	$X_3 : 0.102 \pm 0.09$	/
	$X_4 : 0.87 \pm 0.13$	/	$X_4 : 0.093 \pm 0.09$	/
	$X_5 : 0.79 \pm 0.09$	$X_5 : 0.84 \pm 0.08$	$X_5 : 0.111 \pm 0.06$	$X_5 : 0.147 \pm 0.08$
	$X_8 : 0.91 \pm 0.06$	/	$X_8 : 0.060 \pm 0.04$	/
0.2	$X_3 : 0.89 \pm 0.12$	/	$X_3 : 0.083 \pm 0.09$	/
	$X_4 : 0.88 \pm 0.12$	/	$X_4 : 0.087 \pm 0.08$	/
	$X_5 : 0.78 \pm 0.08$	$X_5 : 0.82 \pm 0.09$	$X_5 : 0.119 \pm 0.06$	$X_5 : 0.159 \pm 0.08$
	$X_8 : 0.93 \pm 0.06$	/	$X_8 : 0.046 \pm 0.04$	/
0.1	$X_3 : 0.89 \pm 0.12$	/	$X_3 : 0.084 \pm 0.09$	/
	$X_4 : 0.89 \pm 0.12$	/	$X_4 : 0.084 \pm 0.09$	/
	$X_5 : 0.78 \pm 0.10$	$X_5 : 0.80 \pm 0.11$	$X_5 : 0.133 \pm 0.07$	$X_5 : 0.169 \pm 0.10$
	$X_8 : 0.93 \pm 0.06$	/	$X_8 : 0.048 \pm 0.04$	/

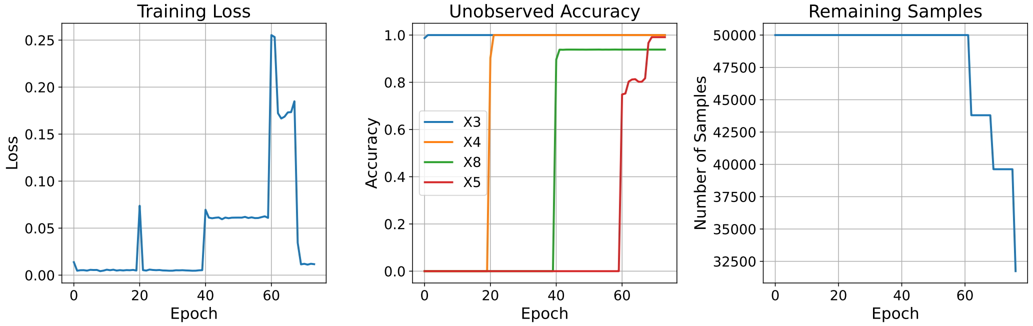


Figure 12: An example of loss and imputation accuracy during coordinate optimization (Obs. Ratio = 0.1, seed = 88, from example (c) of Figure 3). The training order is $[X_3, X_4, X_8, X_5]$. Epochs 0–19 correspond to rule learning for X_3; epochs 20–39 for X_4; epochs 40–59 for X_8, and epochs 60-end for X_5. Remaining samples identified how many samples are "well-explained" during hard covering phase. As the imputation accuracy for missing X_5 is 1.00, we do not go to the fine-tune phase. The learned rules: $X_3 \leftarrow X_0 \wedge X_1, X_4 \leftarrow X_2 \wedge X_7, X_5 \leftarrow (X_3 \wedge X_6) \vee X_8 \vee (X_6 \wedge X_7), X_8 \leftarrow X_3 \wedge X_4$.

J.4 Hyper-parameters Setting and Computing Resource

Our model operates efficiently in a CPU environment utilizing the PyTorch library. The hyperparameters are configured as follows:

Table 28: Summary of learned rule structures and accuracy for example (c) of Figure 3. Each observation ratio is evaluated using 50,000 samples, with results averaged over 20 random seeds. We present the top 3 learned rule structures in order of discovery accuracy. The rules that are truth rules are indicated by underline. Rule accuracy indicates the percentage of 20 runs in which a rule was learned completely correctly.

Obs. Ratio	Learned Rule Structure	Rule Accu.
0.3	$X_3 : \underline{X_0 \wedge X_1}, X_0 \wedge X_2, X_1 \wedge X_2$	$X_3 : 0.45$
	$X_4 : \underline{X_2 \wedge X_7}, X_1 \wedge X_2, X_2$	$X_4 : 0.5$
	$X_5 : \underline{(X_3 \wedge X_6) \vee X_8 \vee (X_6 \wedge X_7)}, X_4 \vee (X_3 \wedge X_6) \vee (X_6 \wedge X_7),$	$X_5 : 0.15$
	$(X_3 \wedge X_4) \vee X_3 \vee (X_3 \wedge X_7)$	
	$X_8 : \underline{X_4 \wedge X_0}, X_3 \wedge X_7, X_2 \wedge X_0$	$X_8 : 0.3$
0.2	$X_3 : \underline{X_0 \wedge X_1}, X_1 \wedge X_2, X_0 \wedge X_2$	$X_3 : 0.55$
	$X_4 : \underline{X_2 \wedge X_7}, X_0 \wedge X_2, X_1 \wedge X_2$	$X_4 : 0.55$
	$X_5 : \underline{(X_3 \wedge X_6) \vee X_8 \vee (X_6 \wedge X_7)},(X_2 \wedge X_3) \vee X_4 \vee (X_6 \wedge X_7),$	$X_5 : 0.10$
	$X_4 \vee X_8 \vee (X_3 \wedge X_6)$	
	$X_8 : \underline{X_4 \wedge X_0}, X_3 \wedge X_4, X_3 \wedge X_7$	$X_8 : 0.25$
0.1	$X_3 : \underline{X_0 \wedge X_1}, X_0, X_0 \wedge X_7$	$X_3 : 0.55$
	$X_4 : \underline{X_2 \wedge X_7}, X_0 \wedge X_2, X_1 \wedge X_7$	$X_4 : 0.55$
	$X_5 : X_4 \vee X_8 \vee (X_6 \wedge X_7), \underline{(X_3 \wedge X_6) \vee X_8 \vee (X_6 \wedge X_7)},$	$X_5 : 0.10$
	$X_3 \vee X_4 \vee (X_6 \wedge X_7)$	
	$X_8 : \underline{X_4 \wedge X_0}, X_2 \wedge X_4, X_3 \wedge X_7$	$X_8 : 0.2$

- Rule Embedding and Fine-tuning Optimizer: Adam, learning rate: 0.01.
- Temperature of softmin and softmax: 0.1 (for Eq. 4) and 10.0 for (Eq. 5).
- "Well-explained" Threshold: 0.99 (for sequential hard covering in disjunctive rule learning).
- Batch Size: 64 .

K Additional Real World Data Experiments Results

K.1 SPECT

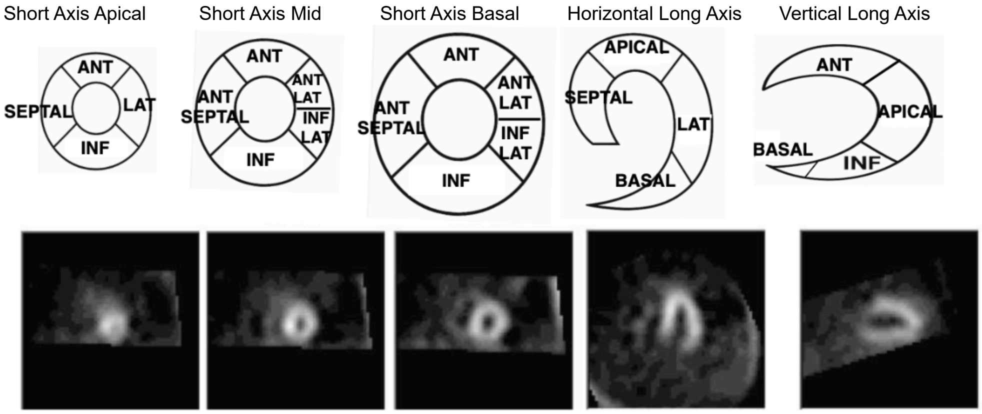


Figure 13: The five slices consists of 22 regions of interest (ROI) for SPECT Diagnosis. The slices are chosen according to the following: Three slices for short axis view-one slice near heart's apex, one in middle of the LV and one near the heart base; One slice corresponds to the center of the LV cavity for horizontal long axis view; One slice corresponds to the center of the LV cavity for vertical long axis view (Kurgan et al., 2001).

We ask for an expertise from cardiovascular surgery of a hospital to give us domain knowledge, and then we try to explain the learned rules. We select several meaningful rules to demonstrate.

The domain knowledge are as follows.

- R1: The anterior wall and the septum of the left ventricle are adjacent and often simultaneously affected by the Left Anterior Descending artery (LAD). If both anterior wall and septum show infarction, it strongly suggests an issue with the LAD. If both apical anterior and mid-anterior show infarction, it indicates a more extensive problem within the LAD territory, affecting both the apical and mid-portions of the anterior wall.
- R2: The apical lateral wall (typically LCX territory) and the apical inferior wall (typically RCA or LCX territory) are adjacent. Infarction in both suggests a problem in this combined region.
- R3: If both apical septal and apical septal show infarction, it indicates a more extensive problem in the LAD territory, involving ischemia in multiple myocardial segments.
- R4: If apical lateral and apical lateral show infarction, it indicates a more extensive ischemic problem in the Left Circumflex artery (LCX) territory.
- R5: The apical anterior (ANT) and apical septal (SEPTAL) regions are primarily supplied by the Left Anterior Descending artery (LAD); the apical lateral (LAT) region is primarily supplied by the Left Circumflex artery (LCX); the apical inferior (INF) region is primarily supplied by the Right Coronary Artery (RCA), but can sometimes be supplied by the LCX, depending on the coronary artery dominance pattern.

Refer to Figure 13, we can give some explanations of rules learned in Table 29 based on the domain knowledge R1 to R5. For example,

- $F_5 \leftarrow F_1 \wedge F_2$: F_1 and F_2 are features from the first slice near the heart's apex, while F_5 is from the second slice at the middle of the left ventricle (LV). According to clinical knowledge R1 and R5, the anterior and septal regions are primarily supplied by the Left Anterior Descending (LAD) artery. Therefore, this rule is clinically plausible: if partial diagnosis (labeled as 1) is present in both F_1 and F_2, it strongly suggests an LAD artery problem. Since F_5 is in the mid-anterior region, also supplied by the LAD, it has a high probability of being affected as well.
- $Diagnosis \leftarrow F_5 \wedge F_6$: From R1, we know that the anterior wall and the septum of the left ventricle are adjacent. F_5 and F_6 both from middle of the LV (left ventricular), and they are adjacent. Thus, if both these adjacent mid-ventricular regions (F_5 and F_6) show signs of infarction, it significantly increases the likelihood of an overall positive diagnosis.

Table 29: Example rules learned by NS-FCN for SPECT feature imputation and diagnosis.

Selected Feature Imputation Rules Learned by NS-FCN
$F_5 \leftarrow F_1 \wedge F_2$: partial diagnosis of segment 1 and 2 causes the partial diagnosis of segment 5.
$F_6 \leftarrow F_{11} \wedge F_{19}$
$F_{13} \leftarrow F_{22} \wedge F_{12}$
Learned Diagnosis Rule Structure
$Diagnosis \leftarrow (F_5 \wedge F_6) \vee (F_2 \wedge F_{11}) \vee (F_4 \wedge F_{13})$

As detailed in Table 30, the learned rules for diagnosing cardiac abnormalities correspond closely with established domain knowledge from cardiovascular surgery experts. For instance, the model identified that infarcts in adjacent regions like F_1 and F_2 are indicative of an issue in the Left Anterior Descending (LAD) artery territory. Furthermore, the model learned a composite rule for the final diagnosis, logically aggregating signals from multiple infarcted regions across different coronary artery territories (LAD, LCX, RCA). This ability to synthesize information from disparate features into a coherent diagnostic rule highlights the model's capacity for complex reasoning. The clinical relevance of these rules was further validated by a Large Language Model (LLM), which confirmed their consistency with expert knowledge on ischemia propagation patterns.

Performance with varying missing ratios. To assess the model's robustness under different levels of data scarcity, we evaluated its performance on the SPECT dataset while varying the observation ratio from 0.3 to 0.9. As shown in Table 18, the model's accuracy remains strong and improves consistently as more data becomes available. Notably, even with only 30% of the data observed (a 70% missing ratio), the model maintains a high F1 score of 0.751, demonstrating its capability to learn meaningful diagnostic rules from highly incomplete datasets.

Table 30: Analysis of learned rules for the SPECT dataset, evaluated by human experts and LLM.

Rules	**Evaluation with Human Expert Knowledge**	**LLM Evaluation**
$F_6 \leftarrow F_1 \wedge F_2$	Matches R1 & R5: F_1 and F_2 are in LAD territory. Infarction in both suggests LAD issue affecting apical and mid-anterior LV.	**Plausible:** Both regions are LAD-supplied and adjacent; mid-anterior (F_3) likely also affected if F_1 & F_2 show infarction. Clinically consistent.
$F_0 \leftarrow F_{11} \wedge F_{19}$	Related to R2 & R4: F_{11} and F_{19} are adjacent. Infarction implies LCX or RCA/LCX combined territory issue.	**Valid:** Matches adjacency and vascular territory logic (LCX-lateral, RCA-inferior). Supports ischemia propagation in midventricular slices.
$F_{13} \leftarrow F_{22} \wedge F_{12}$	Partial link to R3 & R5: Likely involves basal/apical septal (F_{22}) and adjacent basal regions. Indicates LAD or multi-segment ischemia.	**Reasonable:** Suggests ischemia spread in basal-septal regions (LAD) adjacent to basal/anterior. Fits multi-segment LAD pathology.
Diagnosis $\leftarrow (F_1 \wedge F_0) \vee (F_2 \wedge F_{11}) \vee (F_6 \wedge F_{13})$	Consistent with R1 & R4. Combines LAD (F_0), LCX/RCA (F_6), and adjacent mixed regions. Multiple adjacent infarct pairs increase diagnosis likelihood.	**Strong:** Logical aggregation of adjacent infarcted regions across LAD, LCX, RCA territories. Matches expert ischemia propagation patterns.

K.2 Heart Disease

K.2.1 Assessment of Learned Rules

For feature imputation, as shown in Table 31, our model discovers rules with clinically relevant numerical thresholds by directly modeling continuous data. For instance, it learns to impute resting blood pressure (`trestbps`) based on conditions like age > 60 and chol > 250. Similarly, it links high cholesterol to factors like age > 55 in males or very high blood pressure (trestbps > 150). The learned rule for ST depression (`oldpeak`) combines the slope of the ST segment with a maximum heart rate threshold (thalach < 150), demonstrating the model's ability to capture complex, non-linear relationships within the data.

Beyond imputation, NS-FCN learns interpretable rules for the final diagnosis, classifying patients into low-risk or high-risk categories.

Table 32 presents several of these diagnostic rules. For example, the model learns that a combination of factors such as an upsloping ST segment (slope_upsloping), a fixed thallium defect (thal_fixed_defect), and exercise-induced angina (exang_yes) is strongly indicative of high risk. Conversely, it identifies that factors like the absence of exercise-induced angina (exang_no) and a flat ST slope (slope_flat) in female patients suggest a low risk of coronary artery disease. These diagnostic rules were also evaluated by an LLM and deemed ”Excellent” or ”Strong,” underscoring their consistency with clinical practice.

K.3 Hyper-parameters Setting and Computing Resource

For NS-FCN (Ours):

- Rule embedding optimizer: Adam with learning rate of 0.01.
- Fine-tune optimizer: Adam with learning rate of 0.01.
- Temperature of softmin and softmax: 0.1 (for Eq. 4) and 10.0 for (Eq. 5).
- Our model can run efficiently on a CPU environment with the PyTorch package.

Baselines:

- **BRCG** (Dash et al., 2018), **LEN** (Barbiero et al., 2022), **DR-NET** (Qiao et al., 2021), **RRL** (Wang et al., 2021) are trained with the default hyperparameter settings specified in the original paper.
- **MICE** (van Buuren & Groothuis-Oudshoorn, 2011): We use $m = 5$ imputations and $maxit = 5$ iterations with the default imputation methods in the `mice` R package.

Table 31: Learned rules for feature imputation on the Heart dataset, with LLM assessments.

Feature	Imputation Acc.	Learned Rule	LLM Assessment
trestbps	0.86	$\textit{trestbps\_high} \leftarrow (\text{age} > 60) \wedge (\text{chol} > 250)$	**Excellent:** This rule captures the well-established link between age, high cholesterol, and hypertension. Both are primary risk factors for cardiovascular disease and often co-occur.
chol	0.85	$\textit{chol\_high} \leftarrow (\text{sex} = 1 \wedge \text{age} > 55) \vee (\text{trestbps} > 150)$	**Excellent:** The rule correctly identifies two key risk profiles for high cholesterol: middle-aged to elderly males, and individuals with significant hypertension. This aligns perfectly with clinical understanding of metabolic syndrome.
thalach	0.90	$\textit{hr\_high} \leftarrow (\text{trestbps} > 145) \vee (\text{age} > 57 \wedge \text{cp} = 3)$	**Strong:** This rule insightfully links factors that limit exercise capacity to the maximum heart rate achieved. Both hypertension and severe asymptomatic coronary disease can prevent a patient from reaching a higher peak heart rate.
oldpeak	0.76	$\textit{st\_severe} \leftarrow (\text{slope} = 2) \wedge (\text{thalach} < 150)$	**Excellent:** This rule identifies a classic high-risk pattern. A downsloping ST segment is a strong positive finding, and its occurrence at a sub-maximal heart rate indicates ischemia at a low workload, a sign of severe coronary artery disease.

Table 32: Learned rules for disease prediction on the Heart dataset, with LLM assessments.

Learned Rule	LLM Assessment
$\textit{high\_risk} \leftarrow \textit{restecg\_stt\_abnormality} \wedge \textit{ca} = 3 \wedge \textit{oldpeak} > 1.49$	**Excellent:** This rule identifies a high-risk profile by combining three critical indicators of severe coronary artery disease: significant ST depression, an abnormal resting ECG, and extensive vessel blockage.
$\textit{high\_risk} \leftarrow \textit{slope\_downsloping} \wedge \textit{restecg\_normal} \wedge \textit{trestbps} > 145.68$	**Strong:** A downsloping ST segment is a powerful predictor of ischemia. Combining this with hypertension identifies patients at high risk, even if their resting ECG appears normal, highlighting the importance of stress-test indicators.
$\textit{high\_risk} \leftarrow \textit{slope\_flat} \wedge \textit{oldpeak} > 1.49 \wedge \textit{restecg\_hypertrophy}$	**Excellent:** This rule effectively combines signs of acute ischemia (a flat ST slope with significant depression) with evidence of chronic cardiac stress (left ventricular hypertrophy). This profile is strongly indicative of advanced coronary artery disease.

- **MissForest** (Stekhoven & Bühlmann, 2012): We use the default hyperparameter settings in the `missForest` R package.
- **MLP**: We train a 3-layer fully connected network (input-128-128-output) with batch size 32, learning rate 0.001, and 100 epochs using Adam optimizer.
- **VAE(Veldkamp et al., 2025)**: We use a variational autoencoder with latent dimension 16, encoder architecture (input×2-128-64-latent), decoder architecture (latent-64-128-output), batch size 32, learning rate 0.001, and 100 epochs.
- **DAE (mDAE) (Dupuy et al., 2024)**: We use a denoising autoencoder with bottleneck dimension 16, encoder architecture (input-128-64-bottleneck), decoder architecture

(bottleneck-64-128-output), corruption rate $\rho = 0.2$, batch size 32, learning rate 0.001, and 100 epochs.

- **GAIN** (Yoon et al., 2018): We use mini-batch size 128, hint rate $p_{hint} = 0.9$, MSE loss weight $\alpha = 100.0$, cross-entropy loss weight $\beta = 100.0$, learning rate 0.001, and 1000 epochs.
- **MissDiff** (Ouyang et al., 2023): We use 1000 diffusion timesteps with $\beta_{start} = 10^{-4}$ and $\beta_{end} = 0.02$, batch size 32, learning rate 0.001, and 100 epochs.
- All baseline models can run efficiently on CPU environment with PyTorch package (for deep learning methods) or R packages (for statistical methods).

L LIMITATION

While our model shows promising performance, the ethical implications, such as potential over-reliance or misuse for inferring sensitive information, require careful consideration.

Despite its strengths, NS-FCN has limitations. While effective, the asynchronous coordinate gradient descent optimization can be computationally intensive. Besides, the negative predicates are not well explored (we consider negative predicates as an independent predicate from positive predicates). Furthermore, while our model can derive predicates from continuous features, the current implementation learns a single threshold per feature, which may not capture more complex relationships (e.g., intervals). Extending the framework to learn more expressive predicates from continuous data is a promising direction for future work.

M USE OF LLMS

In this paper, LLMs were used solely for writing polishing. The key idea, the model design, research study, and all substantive writing are completed by human authors.

In the assessment of discovered rules, we use LLM to write the evaluation of rule quality, which we have mentioned in the paper.