
Detecting Data Deviation in Electronic Health Records

Appendices

Anonymous Author(s)

Affiliation

Address

email

- 1 Appendix A. Notation Table.
- 2 Appendix B. Extended Related Work.
- 3 Appendix B.1. EHR Data Analytics.
- 4 Appendix B.2. Data Valuation and Data Shapley Value.
- 5 Appendix B.3. Amortized Computation.
- 6 Appendix B.4. Knowledge Distillation.
- 7 Appendix C. Pseudocode for Core Stages of the Methodology.
- 8 Appendix C.1. Algorithm 1: Data Shapley Value Computation Per Task in \mathcal{O}_{ds} .
- 9 Appendix C.2. Algorithm 2: Knowledge Distillation from \mathcal{O}_{ds} to \mathcal{O}_{nn} .
- 10 Appendix C.3. Algorithm 3: Knowledge Distillation from \mathcal{O}_{nn} to Ψ .
- 11 Appendix D. Computational Complexity Analysis.
- 12 Appendix D.1. Computational Complexity Analysis of Algorithm 1.
- 13 Appendix D.2. Computational Complexity Analysis of Algorithm 2.
- 14 Appendix D.3. Computational Complexity Analysis of Algorithm 3.
- 15 Appendix E. Extended Experimental Set-up.
- 16 Appendix E.1. Experimental Set-up on the AKI Dataset.
- 17 Appendix E.2. Experimental Set-up on the MIMIC-III Dataset.
- 18 Appendix F. Supplementary Experimental Results.
- 19 Appendix F.1. Hyperparameter Sensitivity Study on the AKI Dataset.
- 20 Appendix F.2. Evaluation of Controlled Deviation Injection on the MIMIC-III Dataset.
- 21 Appendix F.3. Evaluation of Ψ 's Output Sensitivity on the MIMIC-III Dataset.
- 22 Appendix G. Broader Impact.

23 A Notation Table

24 We adopt the standard convention of using italic symbols (e.g., x) for scalars, bold lowercase (e.g., \mathbf{x})
 25 for vectors, and bold uppercase (e.g., \mathbf{X}) for matrices. Table 1 provides a summary of the notations
 26 used throughout the paper.

27 The table is organized into four sections: (i) general notations, (ii) notations for Data Shapley Value
 28 Computation Per Task in \mathcal{O}_{ds} (Section 3.1), (iii) notations for Knowledge Distillation from \mathcal{O}_{ds} to
 29 \mathcal{O}_{nn} (Section 3.2), and (iv) Knowledge Distillation from \mathcal{O}_{nn} to Ψ for EHR Data Deviation Detection
 30 (Section 3.3).

Table 1: Summary of notations used.

Notation	Description
\mathcal{O}_{ds}	Data Shapley oracle
\mathcal{O}_{nn}	Task-specific neural oracle
Ψ	EHR data fidelity predictor
t	Index of a task or specific application
T	Total number of tasks
$N^{(t)}$	Number of samples in task t
$\mathcal{K}^{(t)}$	EHR data for task t
$k_i^{(t)}$	The i -th EHR sample in task t
\mathbf{x}_i	Input features of $k_i^{(t)}$
$y_i^{(t)}$	Task-specific label of $k_i^{(t)}$ for task t
$\Psi(\mathbf{x}_i)$	Data fidelity of \mathbf{x}_i
$\Delta\Psi$	Decline in data fidelity
$f^{(t)}$	Prediction model used in \mathcal{O}_{ds}
\mathcal{S}	A subset of $\mathcal{K}^{(t)}$
m	Performance evaluation metric
$\eta_i^{(t)}$	Data Shapley value of $k_i^{(t)}$ for task t
Φ	Uniform distribution over all permutations of $\mathcal{K}^{(t)}$
φ	A permutation of $\mathcal{K}^{(t)}$
$\mathcal{S}_\varphi^{k_i^{(t)}}$	Samples preceding $k_i^{(t)}$ in φ
$g^{(t)}(\mathbf{x}, \theta^{(t)})$	Neural oracle \mathcal{O}_{nn} for task t
$\mathcal{L}_{ds \rightarrow nn}^{(t)}$	Knowledge distillation loss from \mathcal{O}_{ds} to \mathcal{O}_{nn} for task t
$\omega^{(t)}$	Weight assigned to $\mathcal{L}_{ds \rightarrow nn}^{(t)}$
$h^{(t)}(\mathbf{x}_i)$	Learned representation of \mathbf{x}_i in task t
$\Psi(\mathbf{x}, \theta)$	Final EHR data fidelity predictor as a neural network
$o(\mathbf{x})$	Hidden representation of \mathbf{x} learned by $\Psi(\mathbf{x}, \theta)$
\mathcal{L}_{kd}	Knowledge distillation loss from \mathcal{O}_{nn} to Ψ
$\alpha^{(t)}(\mathbf{x})$	Attention weight for task t
\mathcal{L}_{ent}	Relative entropy constraint from \mathcal{O}_{nn} to Ψ
\mathcal{L}_{sim}	Similarity constraint from \mathcal{O}_{nn} to Ψ
$\rho_{t,t'}(\mathbf{x}_i)$	Output similarity between neural oracles for tasks t and t'
τ	Temperature parameter in similarity computation
\mathcal{L}	Overall loss integrating \mathcal{L}_{kd} , \mathcal{L}_{ent} , and \mathcal{L}_{sim}
λ_{kd}	Weight of \mathcal{L}_{kd} in \mathcal{L}
λ_{ent}	Weight of \mathcal{L}_{ent} in \mathcal{L}
λ_{sim}	Weight of \mathcal{L}_{sim} in \mathcal{L}

31 **B Extended Related Work**

32 **B.1 EHR Data Analytics**

33 In EHR data analytics, heterogeneous EHR data are widely used to support applications such as risk
34 prediction, medication recommendation, and disease progression modeling [6, 20]. For example,
35 Long Short-Term Memory (LSTM) models have been applied to capture temporal patterns in time-
36 series EHR data, leading to improved diagnostic performance [24].

37 To address challenges inherent in EHR data, such as irregular visit intervals, GRU-D [4] extends
38 Gated Recurrent Units with masking and time interval mechanisms. BRITS [3] further integrates a
39 bidirectional recurrent neural network to jointly impute missing values in time series data and perform
40 classification. More recently, irregularity is tackled in both clinical notes and multivariate irregularly
41 sampled time series [40], further enhancing predictive performance in downstream clinical tasks.

42 These methodological advances improve patient management and healthcare resource allocation [31,
43 38], yielding tangible benefits for patients, clinicians, and healthcare institutions. However, such
44 progress fundamentally relies on the assumption that the underlying EHR data are of high fidelity—a
45 critical yet frequently overlooked prerequisite in real-world EHR data analytics.

46 **B.2 Data Valuation and Data Shapley Value**

47 Data valuation provides a principled framework to measure the contribution of individual data
48 samples to the performance of downstream analytic models [18, 29, 34]. Several strategies have been
49 developed for this purpose. The leave-one-out approach measures sample importance by evaluating
50 the change in model performance upon excluding each sample. Influence functions [21] estimate
51 importance by analyzing the model’s sensitivity to infinitesimal upweighting of a sample.

52 The data Shapley value [9], inspired by the Shapley value in cooperative game theory [33], has
53 emerged as a theoretically grounded and equitable method for data valuation. Building on this
54 foundation, subsequent works have sought to enhance its theoretical and practical properties. For
55 instance, the distributional Shapley framework [8] generalizes the original formulation by defining
56 the value of each data point over an underlying data distribution. Beta Shapley [22] introduces a
57 relaxation of the efficiency axiom of the Shapley value, which is not essential in machine learning
58 contexts, to achieve desirable statistical properties for efficiency. More recently, a hypothesis testing
59 framework [36] has been presented to examine the data Shapley value under different utility function
60 constraints, motivating a class of utility functions that ensure optimal data selection in such scenarios.

61 Other researchers seek to reduce the computational overhead associated with the data Shapley value.
62 For example, [17] proposes a suite of techniques to accelerate its computation by introducing specific
63 assumptions on the utility function, enabling practical estimation algorithms for machine learning
64 tasks. In contrast, a unified framework called stochastic amortization [5] is introduced to speed
65 up both feature attribution and data valuation by leveraging amortized computation, which will be
66 discussed in detail in Section B.3.

67 In addition, the practical utility of the data Shapley value has been evaluated in diverse real-world ap-
68 plications. For instance, [37] investigates scenarios where a validation set is unavailable and proposes
69 using the diversity of data samples as an intrinsic property of the dataset, therefore independent of
70 validation. In another study [35], the data Shapley value is employed to quantify the contribution of
71 each training sample to the model’s performance in pneumonia detection, using a large chest X-ray
72 medical imaging dataset.

73 **B.3 Amortized Computation**

74 Amortized computation (or optimization) [1] uses learning-based models, such as neural networks,
75 to exploit shared structure across similar problem instances, thereby enabling efficient solution
76 prediction and significantly reducing per-instance computational cost. Compared to non-amortized
77 methods, amortized approaches can achieve several orders of magnitude in speedup [1], and have
78 been widely adopted to improve efficiency across various domains.

79 In meta learning, the Model-Agnostic Meta-Learning (MAML) algorithm [7] is designed to be
80 compatible with any model trained via gradient descent, enabling the learning of model parameters
81 that can be easily adaptable. A variant that incorporates implicit differentiation [27] further separates

the computation of the meta-gradient from the specific choice of the inner-loop optimizer, allowing the proposal to tackle a greater number of gradient steps without suffering from vanishing gradients or excessive memory usage.

In explainable machine learning, INVASE [39] performs instance-wise feature selection using a selector-predictor-baseline architecture trained jointly to identify informative subsets of features, where the baseline network is devised to train the selector network. FastSHAP [16] further amortizes the estimation of Shapley values by training an explainer model that approximates them in a single forward pass, using a stochastic gradient descent objective based on weighted least squares.

In reinforcement learning, guided policy search [23] integrates trajectory optimization to direct policy learning, mitigating the risk of poor local optima encountered in direct policy search. Similarly, Stochastic Value Gradients [12] provide a unified framework that leverages backpropagation to learn continuous control policies more effectively.

Finally, in the context of feature attribution and data valuation, amortized computation is employed in [5] to address settings where exact labels are unavailable or expensive to obtain. A stochastic amortization technique is therefore proposed to train neural networks using noisy labels, while still maintaining strong performance backed by theoretical guarantees.

B.4 Knowledge Distillation

Knowledge distillation [13, 26, 10] is a widely adopted technique for model compression and acceleration. It facilitates the transfer of knowledge from a large, cumbersome model (“teacher” model) to a smaller, more efficient model (“student” model). The objectives of knowledge distillation are multifaceted [15], with two particularly relevant objectives: (i) knowledge compression and (ii) knowledge adaptation, as described below.

In knowledge compression, the goal is to preserve the predictive performance of the teacher model in a significantly more compact student model. In [13], for example, the authors distill the knowledge of an ensemble of models into a single model, achieving competitive performance. In natural language processing, DistilBERT [30] compresses the BERT model via distillation, resulting in improved inference efficiency while preserving core language understanding capabilities. FitNets [28] extend the basic distillation paradigm by training thin and deep student networks using both output predictions and intermediate representations from wide, shallower teacher networks as additional supervision.

In knowledge adaptation, the student model is trained to generalize to new or unseen target domains by leveraging knowledge from teacher models trained on related source domains. For instance, Cycle-Consistent Adversarial Domain Adaptation (CyCADA) [14] enhances domain adaptation by facilitating the alignment in both the generative image space and that in the latent representation space while preserving task-relevant semantics. Another example is the Teacher-Student Curriculum Learning framework [25], where the teacher automatically selects subtasks for the student, enabling progressive learning through a curriculum-based approach.

C Pseudocode for Core Stages of the Methodology

This section outlines the pseudocode for the three core stages of the proposed bi-level knowledge distillation approach for detecting data deviation in EHR data. The stages are as follows: (i) computing task-specific data Shapley values using the data Shapley oracle \mathcal{O}_{ds} , (ii) distilling knowledge from \mathcal{O}_{ds} to the corresponding task-specific neural oracle \mathcal{O}_{nn} , and (iii) distilling knowledge from \mathcal{O}_{nn} to the unified EHR data fidelity predictor Ψ .

C.1 Algorithm 1: Data Shapley Value Computation Per Task in \mathcal{O}_{ds}

Algorithm 1 conceptually describes the computation of data Shapley values, which serve as the ground truth supervision for the first level of knowledge distillation.

C.2 Algorithm 2: Knowledge Distillation from \mathcal{O}_{ds} to \mathcal{O}_{nn}

Algorithm 2 details the training procedure for task-specific neural networks $g^{(t)}(\mathbf{x}, \theta^{(t)})$, which serve as neural oracles to approximate the data Shapley values produced by \mathcal{O}_{ds} .

Algorithm 1 Data Shapley Value Computation Per Task in \mathcal{O}_{ds}

Input:

$\mathcal{K}^{(t)} = \{(\mathbf{x}_i, y_i^{(t)})\}_{i=0}^{N^{(t)}-1}$: Dataset for task $t \in \{0, \dots, T-1\}$
 $f^{(t)}$: Prediction model for task t
 $m(\cdot, f^{(t)})$: Performance evaluation metric for task t

Output:

$\{\eta_i^{(t)}\}_{i=0}^{N^{(t)}-1}$: Data Shapley values for each task t
1: **for** each task $t \in \{0, \dots, T-1\}$ **do**
2: **for** each sample $k_i^{(t)} = (\mathbf{x}_i, y_i^{(t)}) \in \mathcal{K}^{(t)}$ **do**
3: Let $\mathcal{S}_\varphi^{k_i^{(t)}}$ be the set of data samples in $\mathcal{K}^{(t)}$ preceding $k_i^{(t)}$ in a permutation φ
4: $\eta_i^{(t)} \leftarrow \mathbb{E}_{\varphi \sim \Phi} \left[m \left(\mathcal{S}_\varphi^{k_i^{(t)}} \cup \{k_i^{(t)}\}, f^{(t)} \right) - m \left(\mathcal{S}_\varphi^{k_i^{(t)}}, f^{(t)} \right) \right]$
5: **end for**
6: **end for**
7: **return** $\{\{\eta_i^{(t)}\}_{i=0}^{N^{(t)}-1}\}_{t=0}^{T-1}$

Algorithm 2 Knowledge Distillation from \mathcal{O}_{ds} to \mathcal{O}_{nn}

Input:

$\mathcal{D}_{\text{train}}$: Training dataloader yielding batches $(\mathbf{x}_{\text{batch}}, \{\eta_{\text{batch}}^{(t)}\}_{t=0}^{T-1})$
 \mathcal{D}_{val} : Validation dataloader
 $\{g^{(t)}(\cdot, \theta_g^{(t)})\}_{t=0}^{T-1}$: Set of task-specific neural oracle models
 opt_g : Optimizer for parameters $\{\theta_g^{(t)}\}$
 \mathcal{L}_{MSE} : Mean squared error loss function
 E_1 : Max epochs. P_1 : Early stopping patience. ϵ_1 : Stability constant for weighting

Output:

Trained models $\{g^{(t)}(\cdot, \theta_g^{(t)})\}_{t=0}^{T-1}$
1: Initialize $\{\theta_g^{(t)}\}_{t=0}^{T-1}$
2: Initialize $\{\bar{\mathcal{L}}_{\text{prev}}^{(t)} \leftarrow 1.0\}_{t=0}^{T-1}$ (for dynamic per-task loss weighting)
3: **for** epoch $e \leftarrow 1$ **to** E_1 **do**
4: **for all** models $g^{(t)}$ **do**
5: $g^{(t)}.train()$
6: **end for**
7: **for** each batch $(\mathbf{x}_{\text{batch}}, \{\eta_{\text{batch}}^{(t)}\}_{t=0}^{T-1})$ **in** $\mathcal{D}_{\text{train}}$ **do**
8: $\text{opt}_g.zero_grad()$
9: $\mathcal{L}_{\text{total_w}} \leftarrow 0$
10: **for all** tasks $t \in \{0, \dots, T-1\}$ **do**
11: $\hat{\eta}_{\text{batch}}^{(t)} \leftarrow g^{(t)}(\mathbf{x}_{\text{batch}}, \theta_g^{(t)})$
12: $\mathcal{L}_{\text{batch}}^{(t)} \leftarrow \mathcal{L}_{\text{MSE}}(\eta_{\text{batch}}^{(t)}, \hat{\eta}_{\text{batch}}^{(t)})$
13: Compute dynamic task weight $\omega^{(t)}$:
14: $\omega^{(t)} \leftarrow (e=1)?1.0 : (\mathcal{L}_{\text{batch}}^{(t)} / (\bar{\mathcal{L}}_{\text{prev}}^{(t)} + \epsilon_1))$
15: $\mathcal{L}_{\text{total_w}} \leftarrow \mathcal{L}_{\text{total_w}} + \omega^{(t)} \cdot \mathcal{L}_{\text{batch}}^{(t)}$
16: **end for**
17: $\mathcal{L}_{\text{total_w}}.backward()$
18: $\text{opt}_g.step()$
19: **end for**
20: Update $\{\bar{\mathcal{L}}_{\text{prev}}^{(t)}\}_{t=0}^{T-1}$ with current epoch's computed average task losses
21: Perform validation on \mathcal{D}_{val} ; if improvement, save $\{\theta_g^{(t)}\}$; check early stopping (P_1)
22: **end for**
23: Load best saved $\{\theta_g^{(t)}\}_{t=0}^{T-1}$
24: **return** $\{g^{(t)}(\cdot, \theta_g^{(t)})\}_{t=0}^{T-1}$

130 C.3 Algorithm 3: Knowledge Distillation from \mathcal{O}_{nn} to Ψ

131 Algorithm 3 describes the training of the final predictor Ψ by aggregating knowledge from the
 132 task-specific neural oracles $g^{(t)}(\mathbf{x}, \theta^{(t)})$ trained in Algorithm 2.

Algorithm 3 Knowledge Distillation from \mathcal{O}_{nn} to Ψ

Input:

$\mathcal{D}_{\text{train}}$: Training dataloader yielding batches $\mathbf{x}_{\text{batch}}$
 \mathcal{D}_{val} : Validation dataloader
 $\{g^{(t)}(\cdot, \theta_g^{(t)})\}_{t=0}^{T-1}$: Trained task-specific neural oracles from Algorithm 2 (frozen)
 $\Psi(\cdot, \theta_\Psi)$: Unified EHR data fidelity predictor model
 $\mathcal{A}(\cdot, \theta_{\mathcal{A}})$: Attention subnetwork
 opt_Ψ : Optimizer for $\theta_\Psi, \theta_{\mathcal{A}}$.
 τ : Temperature for \mathcal{L}_{sim} . T : Number of tasks
 E_2 : Max epochs. P_2 : Early stopping patience. ϵ_2 : Stability constant for weighting

Output:

Trained $\Psi(\cdot, \theta_\Psi)$ and $\mathcal{A}(\cdot, \theta_{\mathcal{A}})$
 1: Initialize $\theta_\Psi, \theta_{\mathcal{A}}$
 2: Initialize $\bar{\mathcal{L}}_{\text{prev,kd}}, \bar{\mathcal{L}}_{\text{prev,ent}}, \bar{\mathcal{L}}_{\text{prev,sim}} \leftarrow 1.0$ (for dynamic loss term weighting)
 3: **for** epoch $e \leftarrow 1$ **to** E_2 **do**
 4: $\Psi.\text{train}(); \mathcal{A}.\text{train}()$
 5: **for** each batch $\mathbf{x}_{\text{batch}}$ **in** $\mathcal{D}_{\text{train}}$ **do**
 6: $\text{opt}_\Psi.\text{zero_grad}()$
 7: *Perform model forward propagation:*
 8: $\{\hat{\eta}_{\text{batch}}^{(t)} \leftarrow g^{(t)}(\mathbf{x}_{\text{batch}}, \theta_g^{(t)})\}_{t=0}^{T-1}$
 9: $\{\mathbf{h}_{\text{batch}}^{(t)} \leftarrow g^{(t)}.\text{get_hidden}(\mathbf{x}_{\text{batch}})\}_{t=0}^{T-1}$
 10: $\hat{\Psi}_{\text{batch}} \leftarrow \Psi(\mathbf{x}_{\text{batch}}, \theta_\Psi); \mathbf{o}_{\text{batch}} \leftarrow \Psi.\text{get_hidden}(\mathbf{x}_{\text{batch}})$
 11: $\alpha_{\text{batch}}^{(t)} \leftarrow \mathcal{A}(\mathbf{o}_{\text{batch}} \parallel \mathbf{h}_{\text{batch}}^{(t)}, \theta_{\mathcal{A}})$
 12: *Compute loss terms:*
 13: $\mathcal{L}_{\text{kd}} \leftarrow \mathcal{L}_{\text{MSE}}(\hat{\Psi}_{\text{batch}}, \sum_t \alpha_{\text{batch}}^{(t)} \odot \text{detach}(\hat{\eta}_{\text{batch}}^{(t)}))$
 14: $\mathcal{L}_{\text{ent}} \leftarrow \text{mean}(\mathcal{D}_{\text{KL}}(\alpha_{\text{batch}} \parallel \text{Uniform}(1/T)))$
 15: $\mathcal{L}_{\text{sim}} \leftarrow \text{mean}_{\text{samples } i \in \text{batch}} (\sum_{0 \leq t < t' < T} \alpha_i^{(t)} \alpha_i^{(t')} \exp(-\text{MSE}(\hat{\eta}_i^{(t)}, \hat{\eta}_i^{(t')})/\tau))$
 16: *Compute dynamic weights for loss terms:*
 17: $\lambda_{\text{kd}} \leftarrow (e = 1)?1.0 : (\mathcal{L}_{\text{kd}}/(\bar{\mathcal{L}}_{\text{prev,kd}} + \epsilon_2))$; similar for $\lambda_{\text{ent}}, \lambda_{\text{sim}}$
 18: $\mathcal{L}_{\text{total}} \leftarrow \lambda_{\text{kd}}\mathcal{L}_{\text{kd}} + \lambda_{\text{ent}}\mathcal{L}_{\text{ent}} + \lambda_{\text{sim}}\mathcal{L}_{\text{sim}}$
 19: $\mathcal{L}_{\text{total}}.\text{backward}()$
 20: $\text{opt}_\Psi.\text{step}()$
 21: **end for**
 22: Update $\bar{\mathcal{L}}_{\text{prev,kd}}, \bar{\mathcal{L}}_{\text{prev,ent}}, \bar{\mathcal{L}}_{\text{prev,sim}}$ with current epoch's computed averages
 23: Perform validation on \mathcal{D}_{val} ; if improvement, save $\theta_\Psi, \theta_{\mathcal{A}}$; check early stopping (P_2)
 24: **end for**
 25: Load best saved $\theta_\Psi, \theta_{\mathcal{A}}$
 26: **return** $\Psi(\cdot, \theta_\Psi), \mathcal{A}(\cdot, \theta_{\mathcal{A}})$

133 D Computational Complexity Analysis

134 The computational complexity of the three proposed algorithms is analyzed below. The following
 135 notations are adopted:

- 136 • T : Total number of tasks.
- 137 • $N^{(t)}$: Number of data samples for task t .
- 138 • \mathcal{M} : Number of Monte Carlo permutations for data Shapley value approximation.
- 139 • N_1 and N_2 : Total number of training samples in Algorithm 2 and Algorithm 3, respectively.
- 140 • E_1, E_2 : Number of training epochs for Algorithm 2 and Algorithm 3, respectively.

- B_1, B_2 : Batch sizes for Algorithm 2 and Algorithm 3, respectively.
- $C_{\text{inf}}^{(t)}$: Computational cost of a single inference (prediction) using the task-specific model $f^{(t)}$ (Algorithm 1).
- m : Performance evaluation metric. Evaluating $m(\mathcal{S}, f^{(t)})$ on a subset \mathcal{S} using a fixed model $f^{(t)}$ incurs a cost of $O(|\mathcal{S}| \cdot C_{\text{inf}}^{(t)})$.
- For a generic neural network model Net with parameters θ_{Net} :
 - $C_{\text{fwd}}(Net, \text{batch_size})$: Cost of a forward pass.
 - $C_{\text{bwd}}(Net, \text{batch_size})$: Cost of a backward pass (gradient computation), typically approximated as $C_{\text{bwd}} \approx \beta \cdot C_{\text{fwd}}$ for some constant $\beta \geq 1$.
 - $C_{\text{optim}}(Net)$: Cost of updating parameters via an optimizer, typically $O(|\theta_{Net}|)$.
 - $C_{\text{train_step}}(Net, \text{batch_size})$: Total cost of a single training step, computed as:

$$C_{\text{train_step}}(Net, \text{batch_size}) = C_{\text{fwd}}(Net, \text{batch_size}) + C_{\text{bwd}}(Net, \text{batch_size}) + C_{\text{optim}}(Net)$$
- $g^{(t)}$: Task-specific neural oracle.
- Ψ : Unified EHR data fidelity predictor.
- \mathcal{A} : Attention subnetwork.

D.1 Computational Complexity Analysis of Algorithm 1

The exact computation of data Shapley values is known to be NP-hard. To address this, the proposed algorithm adopts a Monte Carlo approximation. For each task t and each of its $N^{(t)}$ samples $k_i^{(t)}$, the data Shapley value is estimated by averaging the marginal contributions across \mathcal{M} random permutations of $\mathcal{K}^{(t)}$.

For a given permutation φ , the marginal contribution of $k_i^{(t)}$ requires two evaluations of the performance metric m : namely, $m(\mathcal{S}_{\varphi}^{k_i^{(t)}} \cup k_i^{(t)}, f^{(t)})$ and $m(\mathcal{S}_{\varphi}^{k_i^{(t)}}, f^{(t)})$, where $\mathcal{S}_{\varphi}^{k_i^{(t)}}$ denotes the set of data points preceding $k_i^{(t)}$ in the permutation, with an expected size of $O(N^{(t)})$. Hence, computing one marginal contribution has a time complexity of $O(N^{(t)} \cdot C_{\text{inf}}^{(t)})$.

Aggregating across all \mathcal{M} permutations and $N^{(t)}$ samples for each task t yields a total complexity of:

$$O\left(\sum_{t=0}^{T-1} \mathcal{M} \cdot N^{(t)} \cdot (N^{(t)} C_{\text{inf}}^{(t)})\right) = O\left(\mathcal{M} \sum_{t=0}^{T-1} (N^{(t)})^2 C_{\text{inf}}^{(t)}\right)$$

Assuming uniform dataset size and inference cost across tasks, i.e., $N^{(t)} \approx N_{\text{avg}}$ and $C_{\text{inf}}^{(t)} \approx C_{\text{inf_avg}}$, this simplifies to:

$$O(T \cdot \mathcal{M} \cdot N_{\text{avg}}^2 \cdot C_{\text{inf_avg}})$$

Moreover, under the high-level asymptotic assumption that per-sample inference cost is constant, i.e., $C_{\text{inf_avg}} = O(1)$ (due to fixed model architecture and input dimensionality), the overall computational complexity becomes:

$$O(T \cdot \mathcal{M} \cdot N_{\text{avg}}^2)$$

D.2 Computational Complexity Analysis of Algorithm 2

In this stage, T task-specific neural oracle models $\{g^{(t)}\}_{t=0}^{T-1}$ are jointly trained for E_1 epochs. Each epoch processes $\lceil N_1/B_1 \rceil$ batches, where N_1 is the total number of training samples and B_1 is the batch size. During each batch, a joint training step is performed using a shared loss function $\mathcal{L}_{\text{total_w}}$, involving all T models.

Let the per-batch training cost for model $g^{(t)}$ be denoted by $C_{\text{train_step}}(g^{(t)}, B_1)$. The total training cost across all tasks is given by:

$$O\left(E_1 \cdot \frac{N_1}{B_1} \cdot \sum_{t=0}^{T-1} C_{\text{train_step}}(g^{(t)}, B_1)\right)$$

177 Assuming that all task-specific models $g^{(t)}$ have similar computational complexity, we define a
 178 representative per-batch training cost as $C_{\text{g_train_step}}(B_1)$, yielding:

$$O(E_1 \cdot \frac{N_1}{B_1} \cdot T \cdot C_{\text{g_train_step}}(B_1))$$

179 Further assuming that the per-batch training cost scales linearly with the batch size (i.e.,
 180 $C_{\text{g_train_step}}(B_1) = O(B_1)$, with a constant factor determined by a fixed model architecture), the
 181 overall time complexity simplifies to:

$$O(E_1 \cdot N_1 \cdot T)$$

182 D.3 Computational Complexity Analysis of Algorithm 3

183 In this algorithm, the unified predictor Ψ and attention subnetwork \mathcal{A} are trained for E_2 epochs using
 184 $\lceil N_2/B_2 \rceil$ batches. The computational cost per batch consists of the following components:

- 185 1. Inference from frozen models: Forward passes through the T frozen task-specific models
 186 $g^{(t)}$ to obtain hidden representations $\mathbf{h}^{(t)}$, with total cost $\sum_{t=0}^{T-1} C_{\text{fwd}}(g^{(t)}, B_2)$.
- 187 2. Ψ forward pass: Computes the prediction $\hat{\Psi}$ and hidden representation \mathbf{o} , with cost
 188 $C_{\text{fwd}}(\Psi, B_2)$.
- 189 3. \mathcal{A} forward pass: Computes attention scores with cost $C_{\text{fwd}}(\mathcal{A}, B_2)$.
- 190 4. Loss computation:
 - 191 • Knowledge distillation loss \mathcal{L}_{kd} : $O(B_2 T)$.
 - 192 • Relative entropy constraint \mathcal{L}_{ent} : $O(B_2 T)$.
 - 193 • Similarity constraint \mathcal{L}_{sim} : $O(B_2 T^2)$ (due to pairwise comparisons).

194 The dominant cost among these is:

$$C_{\text{loss_calc}} = O(B_2 T^2)$$

- 195 5. Backward pass for Ψ, \mathcal{A} : $C_{\text{bwd}}(\Psi, B_2) + C_{\text{bwd}}(\mathcal{A}, B_2)$.
- 196 6. Optimizer update for $\theta_\Psi, \theta_{\mathcal{A}}$: $C_{\text{optim}}(\Psi) + C_{\text{optim}}(\mathcal{A})$.

197 Aggregating these components, the total time is:

$$O\left(E_2 \cdot \frac{N_2}{B_2} \cdot \left[\sum_{t=0}^{T-1} C_{\text{fwd}}(g^{(t)}, B_2) + C_{\text{train_step}}(\Psi, B_2) + C_{\text{train_step}}(\mathcal{A}, B_2) + C_{\text{loss_calc}} \right]\right)$$

198 Assuming an average forward pass cost $C_{\text{g_fwd_avg}}(B_2)$ for each of the T frozen $g^{(t)}$ models, and
 199 recalling that $C_{\text{loss_calc}} = O(B_2 T^2)$, the total time complexity simplifies to:

$$O\left(E_2 \cdot \frac{N_2}{B_2} \cdot \left[T \cdot C_{\text{g_fwd_avg}}(B_2) + C_{\text{train_step}}(\Psi, B_2) + C_{\text{train_step}}(\mathcal{A}, B_2) + B_2 T^2 \right]\right)$$

200 In the high-level asymptotic case where all forward and training step costs scale linearly with batch
 201 size (i.e., $C_{\text{g_fwd_avg}}(B_2) = O(B_2)$, $C_{\text{train_step}}(\Psi, B_2) = O(B_2)$, and $C_{\text{train_step}}(\mathcal{A}, B_2) = O(B_2)$),
 202 the expression becomes:

$$O\left(E_2 \cdot \frac{N_2}{B_2} \cdot (T \cdot B_2 + B_2 + B_2 + B_2 T^2)\right) = O(E_2 \cdot N_2 (T + T^2))$$

203 Therefore, the most simplified dominant time complexity is:

$$O(E_2 \cdot N_2 \cdot T^2)$$

204 E Extended Experimental Set-up

205 This section provides additional details of the experimental set-up for both the AKI (acute kidney
206 injury) dataset from our hospital and the publicly available MIMIC-III benchmark dataset [19].

207 For both datasets, we partition the extracted samples into 85% for model development and 15% as a
208 held-out set for computing data Shapley values. Within the development set, we further divide the
209 data into 80% for training, 10% for validation, and 10% for testing. Model training is performed
210 using the Adam optimizer. Hyperparameters are selected based on the best validation performance,
211 measured by the minimum loss \mathcal{L} (Equation (12)), averaged over three independent runs. The final
212 model is then evaluated on the test set using the selected hyperparameter configuration. Additional
213 setup details specific to each dataset are provided in Sections E.1 and E.2.

214 The experiments are conducted on a server equipped with two Intel Xeon Gold 6248R CPUs, 768 GB
215 of memory, and eight NVIDIA V100 GPUs connected via NVLINK. All models are implemented
216 using PyTorch version 1.12.1.

217 To evaluate the effectiveness of our proposed bi-level knowledge distillation approach, we compute
218 the expected EHR data fidelity decline, $\Delta\Psi$, as defined in Equation (1), and use its sign to indicate
219 the presence or absence of a detected deviation. Our approach is compared against several widely
220 adopted unsupervised anomaly detection baselines, as detailed below.

- 221 • **One-Class SVM:** Constructs a decision boundary in a high-dimensional feature space using a
222 radial basis function (RBF) kernel. During training, up to 5% of the samples are allowed to be
223 treated as outliers. Anomaly scores are derived based on each sample’s distance to the learned
224 decision boundary.
- 225 • **Local Outlier Factor:** Estimates the local density of each sample based on its 20 nearest neighbors.
226 Samples exhibiting substantially lower local reachability density are identified as outliers. The
227 detection threshold is set at the 5th percentile of the Local Outlier Factor scores computed on the
228 training data.
- 229 • **Gaussian Mixture Model:** Assumes that the samples are generated from a mixture of Gaussian
230 distributions. Anomaly scores are computed based on the log-likelihood of each sample under the
231 fitted model. Samples with log-likelihoods below the 5th percentile of the training distribution are
232 flagged as outliers.
- 233 • **k-Means Distance:** Computes the Euclidean distance between each sample and its assigned cluster
234 centroid obtained via k -Means clustering. Samples with distances exceeding the 90th percentile of
235 the training distances are classified as outliers.
- 236 • **PCA Reconstruction Error:** Applies principal component analysis (PCA) to reduce dimensionality
237 while retaining 60% of the total variance. Anomaly scores are computed as the reconstruction
238 error for each sample. The 90th percentile of reconstruction errors on the training set is used as the
239 detection threshold.

240 E.1 Experimental Set-up on the AKI Dataset

241 The original cohort from our hospital’s EHR system comprises 2,888 patients diagnosed with AKI
242 (acute kidney injury) [2] between November 2015 and October 2016, with follow-up data collected
243 over a five-year period to monitor their post-AKI outcomes. We focus on four major adverse kidney
244 events (MAKE) [32], which serve as four prediction tasks in our setting and reflect long-term
245 deterioration in kidney function. These events include: (i) the development of new or progressive
246 CKD (chronic kidney disease), defined as a decline of more than 30% in baseline eGFR; (ii) the
247 onset of ESKD (end-stage kidney disease), indicated by eGFR falling below 15mL/min/1.73m²;
248 (iii) dependence on RRT (renal replacement therapy); and (iv) mortality. The prevalence of each task
249 (i.e., proportion of positive samples) in the original AKI dataset is summarized in Table 2.

250 We define a 90-day observation window following the initial AKI diagnosis and use the patients’
251 laboratory test results within this period as model input. Applying this criterion results in the exclusion
252 of 651 patients (without laboratory tests in the observation window), yielding a final cohort of 2,237

Table 2: Prevalence per task in the original AKI dataset.

Task	# Positive	%
New or Progressive CKD	941	32.58
ESKD Onset	295	10.21
Post-AKI RRT Dependence	118	4.09
Mortality	670	23.20

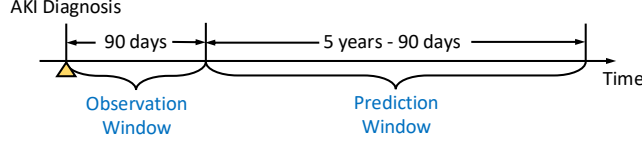


Figure 1: Relationship between observation window and prediction window in the AKI dataset.

patients. The objective is to predict the occurrence of the four target events within a subsequent prediction window. The temporal relationship between the observation and prediction windows is illustrated in Figure 1. Specifically, we extract 43 distinct types of laboratory tests recorded during the observation window, comprising a total of 130,755 test entries.

Regarding the hyperparameter settings, the task-specific neural oracle \mathcal{O}_{nn} is implemented as a multilayer perceptron (MLP) with three hidden layers of sizes 32, 16, and 8, respectively. The unified EHR data fidelity predictor Ψ is also an MLP, with hidden layers of sizes 64, 32, and 16. The representation dimension of $r^{(t)}(\mathbf{x})$ in the attention subnetwork (Equation (7)) is set to 32. We use a learning rate of 0.01 for training \mathcal{O}_{nn} and 0.0001 for Ψ . The temperature parameter τ in Equation (11) is set to 0.5. Training is conducted for a maximum of 1000 epochs with a batch size of 128. Early stopping is employed if the validation performance does not improve for 50 consecutive epochs.

E.2 Experimental Set-up on the MIMIC-III Dataset

MIMIC-III [19] (Medical Information Mart for Intensive Care) is a widely used benchmark dataset in EHR data analytics. It comprises EHR data from over forty thousand patients admitted to intensive care units (ICU) between 2001 and 2012. In this study, we adopt the multi-task learning benchmark established in [11], focusing on the phenotype classification application. This application involves predicting the presence of 25 distinct acute care conditions (i.e., phenotypes) during a given ICU stay, formulated as a multilabel classification problem.

In this dataset, a single patient may have multiple hospital admissions, and each admission can include multiple ICU stays. Following the protocol in [11], we treat each ICU stay as an individual sample, resulting in a total of 41,902 samples. The goal of phenotype classification is to predict the presence of specific acute care conditions (phenotypes) for each ICU stay. Detailed descriptions of the phenotypes and their corresponding prevalences (i.e., the proportion of positive samples) are provided in Table 3.

For each ICU stay, we extract 17 physiological variables as input features. Each variable is aggregated across seven predefined time span ranges: the first and last 10%, 25%, and 50% of the stay duration, as well as the entire time span. Within each time range, we compute six statistical measures per variable: minimum, maximum, mean, standard deviation, skewness, and the number of recorded measurements. This preprocessing procedure yields a total of 714 features per sample for subsequent analysis.

The task-specific neural oracle \mathcal{O}_{nn} is implemented as an MLP with three hidden layers of dimensions 512, 256, and 128, respectively. The unified EHR data fidelity predictor Ψ shares the same architecture, also comprising hidden layers of sizes 512, 256, and 128. The representation dimension of $r^{(t)}(\mathbf{x})$ in the attention subnetwork (Equation (7)) is set to 128. We use a learning rate of 0.01 for training \mathcal{O}_{nn} and 0.001 for Ψ . The temperature parameter τ in Equation (11) is fixed at 2.0. Models are trained for a maximum of 1000 epochs with a batch size of 512. Early stopping is employed based on validation performance, with a patience of 50 epochs.

Table 3: Prevalence per task in the MIMIC-III dataset.

Task	# Positive	%
Essential hypertension	17573	41.94
Coronary atherosclerosis and other heart disease	13540	32.31
Cardiac dysrhythmias	13458	32.12
Disorders of lipid metabolism	12162	29.02
Fluid and electrolyte disorders	11254	26.86
Congestive heart failure; nonhypertensive	11220	26.78
Acute and unspecified renal failure	8964	21.39
Complications of surgical procedures or medical care	8695	20.75
Diabetes mellitus without complication	8074	19.27
Respiratory failure; insufficiency; arrest (adult)	7566	18.06
Septicemia (except in labor)	5975	14.26
Pneumonia (except that caused by tuberculosis or sexually transmitted disease)	5815	13.88
Chronic kidney disease	5607	13.38
Hypertension with complications and secondary hypertension	5547	13.24
Chronic obstructive pulmonary disease and bronchiectasis	5455	13.02
Acute myocardial infarction	4337	10.35
Diabetes mellitus with complications	3988	9.52
Other liver diseases	3723	8.89
Pleurisy; pneumothorax; pulmonary collapse	3658	8.73
Shock	3291	7.85
Acute cerebrovascular disease	3079	7.35
Gastrointestinal hemorrhage	3067	7.32
Conduction disorders	3011	7.19
Other lower respiratory disease	2168	5.17
Other upper respiratory disease	1702	4.06

Table 4: Tuning range for hyperparameters on the AKI dataset.

Hyperparameters	Tuning Range
Dimensions of $g^{(t)}(\mathbf{x}, \theta^{(t)})$	[64, 32, 16], [32, 32, 16], [32, 16, 8]
Dimensions of $\Psi(\mathbf{x}, \theta)$	[128, 64, 32], [64, 32, 16], [32, 16, 8]
Dimension of $r^{(t)}(\mathbf{x})$	{16, 32}
Learning rate of $g^{(t)}(\mathbf{x}, \theta^{(t)})$	{0.0001, 0.001, 0.01}
Learning rate of $\Psi(\mathbf{x}, \theta)$	{0.0001, 0.001, 0.01}
Temperature τ	{0.5, 1.0, 2.0}

F Supplementary Experimental Results

F.1 Hyperparameter Sensitivity Study on the AKI Dataset

We perform a comprehensive hyperparameter sensitivity analysis on the AKI dataset collected from our hospital. The tuned hyperparameters and their corresponding search ranges are summarized in Table 4. The number of training epochs is fixed at 1000, with a batch size of 128. Early stopping is applied if the validation performance does not improve for 50 consecutive epochs. The sensitivity results are presented in Figure 2. As illustrated, the optimal validation performance is achieved when the architecture of $g^{(t)}(\mathbf{x}, \theta^{(t)})$ is configured with layer dimensions [32, 16, 8], and $\Psi(\mathbf{x}, \theta)$ with [64, 32, 16]. The representation dimension of $r^{(t)}(\mathbf{x})$ is set to 32. The learning rates for $g^{(t)}(\mathbf{x}, \theta^{(t)})$ and $\Psi(\mathbf{x}, \theta)$ are set to 0.01 and 0.0001, respectively. The temperature parameter τ is fixed at 0.5. This hyperparameter configuration is subsequently applied to the test dataset for reporting the final evaluation results.

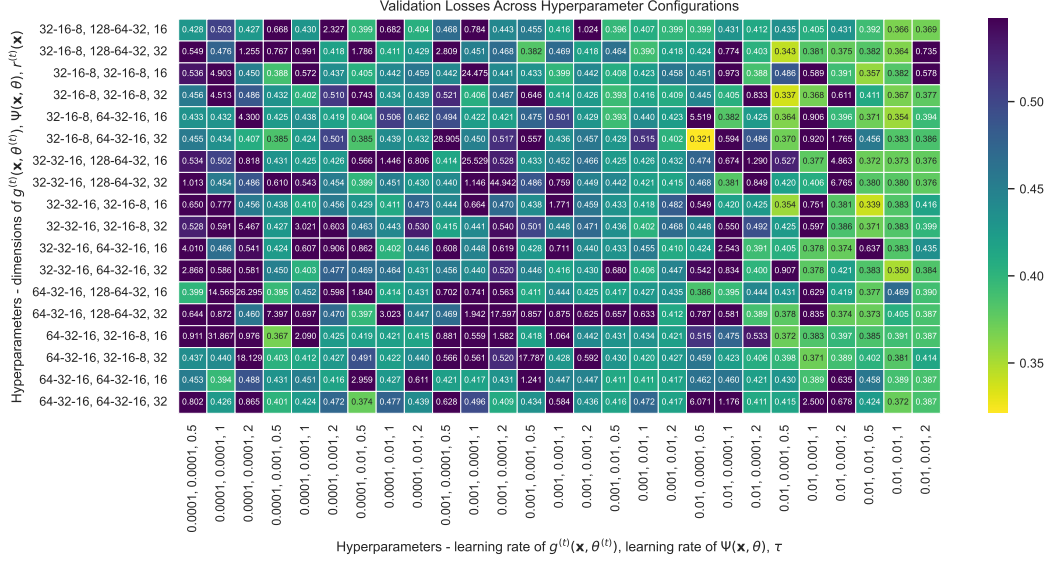


Figure 2: Hyperparameter sensitivity study results of our proposed bi-level knowledge distillation approach on the AKI dataset.

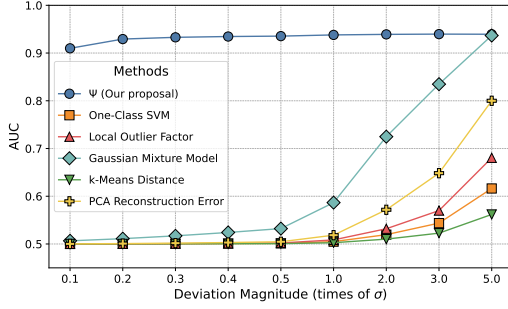


Figure 3: Performance comparison between Ψ and baselines for EHR Data deviation detection on the MIMIC-III dataset.

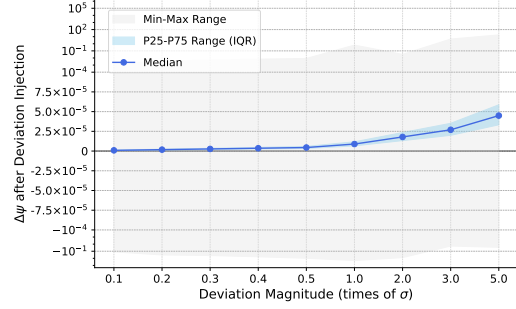


Figure 4: Impact of deviation magnitudes on $\Delta\Psi$ (data fidelity decline) after deviation injection on the MIMIC-III dataset.

302 F.2 Evaluation of Controlled Deviation Injection on the MIMIC-III Dataset

303 In this section, we evaluate the effectiveness of the proposed Ψ for detecting deviation in EHR data
 304 using the MIMIC-III dataset. We adopt a controlled deviation injection experiment similar to that
 305 conducted on the AKI dataset (Section 4.2), introducing deviation with magnitudes ranging from
 306 0.1σ to 5σ . The comparative results between Ψ and baseline methods are presented in Figure 3.

307 Consistent with the observations on the AKI dataset (Figure 3 in the main paper), both the baselines
 308 and Ψ exhibit improved AUC performance as the deviation magnitude increases, which aligns with
 309 the expectation that larger perturbations are more easily detectable. However, Ψ demonstrates superior
 310 sensitivity to small deviation. Specifically, it achieves an AUC of 0.91 when the deviation is as small
 311 as 0.1σ , and this performance further improves to 0.94 at 5σ . In contrast, most baseline methods fail
 312 to respond effectively at lower deviation levels and do not reliably distinguish between perturbed and
 313 unperturbed samples.

314 Among the baselines, One-Class SVM, Local Outlier Factor, and k -Means Distance show negligible
 315 response until the deviation exceeds 3σ , and their performance remains suboptimal even at 5σ . PCA
 316 Reconstruction Error outperforms these methods but only achieves an AUC of 0.8 at 5σ . Gaussian
 317 Mixture Model is the strongest baseline on MIMIC-III, reaching competitive AUC under 5σ , yet it
 318 remains insensitive to deviation smaller than 1σ , highlighting its limited robustness relative to Ψ .

319 Notably, in contrast to the AKI results, the baseline methods on MIMIC-III consistently fall short of
320 Ψ 's performance, even at higher deviation magnitudes. This suggests that the MIMIC-III dataset,
321 comprising 25 tasks (phenotypes to classify), poses a more complex deviation detection challenge.
322 Nevertheless, Ψ maintains high performance across the entire range of deviation magnitudes, demon-
323 strating the effectiveness of bi-level knowledge distillation from task-specific data Shapley oracles in
324 enhancing EHR deviation detection.

325 F.3 Evaluation of Ψ 's Output Sensitivity on the MIMIC-III Dataset

326 We further assess the output sensitivity of Ψ on the MIMIC-III dataset by measuring the fidelity
327 decline, denoted as $\Delta\Psi$, under varying magnitudes of injected deviation from 0.1σ to 5σ . The results
328 are summarized in Figure 4, which reports the median, interquartile range (IQR), and minimum-
329 maximum range of $\Delta\Psi$.

330 Consistent with the observations on the AKI dataset (Figure 4 in the main paper), $\Delta\Psi$ increases
331 monotonically as the deviation magnitude grows. Notably, even when $\Delta\Psi$ is close to zero, Ψ remains
332 effective at detecting subtle deviation by leveraging the sign of $\Delta\Psi$ as a reliable indicator. This is
333 corroborated by the high AUC observed in Figure 3 at small deviation levels, reinforcing the ability of
334 Ψ to identify early-stage deviation that are typically overlooked by baseline methods. This property
335 is particularly valuable for early warning applications in EHR data analytics.

336 Additionally, we observe that the absolute values of $\Delta\Psi$ on the MIMIC-III dataset are generally
337 smaller than those on the AKI dataset, suggesting that the output sensitivity of Ψ varies less markedly
338 with increasing deviation magnitude in this setting. Furthermore, $\Delta\Psi$ can be negative under small
339 deviation, indicating occasional prediction errors by Ψ in extreme cases. These findings underscore the
340 increased difficulty of the MIMIC-III scenario, consistent with the baseline performance degradation
341 reported in Section F.2. Nevertheless, Ψ continues to deliver robust deviation detection performance,
342 even in this more complex and diverse application context.

343 G Broader Impact

344 The proposed bi-level knowledge distillation approach enables reliable detection of potential data
345 deviation in EHR data arising from sources such as pre-analytical variability, documentation errors,
346 or unvalidated data sources. By assessing the EHR data fidelity, the approach enhances the reliability
347 and accuracy of downstream clinical decisions and interventions. As such, it represents a promising
348 direction for improving data acquisition, collection, and recording protocols, and may serve as a
349 foundation for future error correction and calibration mechanisms.

350 Beyond these technical contributions, it is essential to involve clinicians and healthcare professionals
351 when applying EHR deviation detection in clinical practice. In particular, a decline in data fidelity
352 may not solely stem from artifacts or errors—it may also reflect meaningful underlying physiological
353 dynamics. In such cases, additional contextual information or multimodal data may be required to
354 accurately interpret the deviation and to understand the patient's clinical condition. Addressing this
355 challenge necessitates collaborative efforts between computational researchers and domain experts,
356 and highlights an important open area for future investigation.

References

- [1] Brandon Amos. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*, 2(3):9, 2022.
- [2] Rinaldo Bellomo, John A Kellum, and Claudio Ronco. Acute kidney injury. *The Lancet*, 380(9843):756–766, 2012.
- [3] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. BRITS: bidirectional recurrent imputation for time series. In *NeurIPS*, pages 6776–6786, 2018.
- [4] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.
- [5] Ian Covert, Chanwoo Kim, Su-In Lee, James Y. Zou, and Tatsunori B. Hashimoto. Stochastic amortization: A unified approach to accelerate feature and data attribution. In *NeurIPS*, 2024.
- [6] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.
- [7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017.
- [8] Amirata Ghorbani, Michael P. Kim, and James Zou. A distributional framework for data valuation. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 3535–3544. PMLR, 2020.
- [9] Amirata Ghorbani and James Y. Zou. Data shapley: Equitable valuation of data for machine learning. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR, 2019.
- [10] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *Int. J. Comput. Vis.*, 129(6):1789–1819, 2021.
- [11] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1):96, 2019.
- [12] Nicolas Heess, Gregory Wayne, David Silver, Timothy P. Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *NIPS*, pages 2944–2952, 2015.
- [13] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.
- [14] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1994–2003. PMLR, 2018.
- [15] Chengming Hu, Xuan Li, Dan Liu, Haolun Wu, Xi Chen, Ju Wang, and Xue Liu. Teacher-student architecture for knowledge distillation: A survey. *CoRR*, abs/2308.04268, 2023.
- [16] Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. Fastshap: Real-time shapley value estimation. In *ICLR*. OpenReview.net, 2022.
- [17] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In *AISTATS*, volume 89 of *Proceedings of Machine Learning Research*, pages 1167–1176. PMLR, 2019.
- [18] Kevin Fu Jiang, Weixin Liang, James Y. Zou, and Yongchan Kwon. Opendataval: a unified benchmark for data valuation. In *NeurIPS*, 2023.

- [19] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.
- [20] Sulaiman Khan, Habib Ullah Khan, and Shah Nazir. Systematic analysis of healthcare big data analytics for efficient care and disease diagnosing. *Scientific Reports*, 12(1):22377, 2022.
- [21] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 2017.
- [22] Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *AISTATS*, volume 151 of *Proceedings of Machine Learning Research*, pages 8780–8802. PMLR, 2022.
- [23] Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1–9. JMLR.org, 2013.
- [24] Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wetzel. Learning to diagnose with LSTM recurrent neural networks. In *ICLR (Poster)*, 2016.
- [25] Tabet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *IEEE Trans. Neural Networks Learn. Syst.*, 31(9):3732–3740, 2020.
- [26] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5142–5151. PMLR, 2019.
- [27] Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *NeurIPS*, pages 113–124, 2019.
- [28] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR (Poster)*, 2015.
- [29] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Oliver Kiss, Sebastian Nilsson, and Rik Sarkar. The shapley value in machine learning. In *IJCAI*, pages 5572–5579. ijcai.org, 2022.
- [30] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.
- [31] Tabinda Sarwar, Sattar Seifollahi, Jeffrey Chan, Xiuzhen Zhang, Vural Aksakalli, Irene Lena Hudson, Karin Verspoor, and Lawrence Cavedon. The secondary use of electronic health records for data mining: Data characteristics and challenges. *ACM Comput. Surv.*, 55(2):33:1–33:40, 2023.
- [32] Emily J See, Nigel D Toussaint, Michael Bailey, David W Johnson, Kevan R Polkinghorne, Raymond Robbins, and Rinaldo Bellomo. Risk factors for major adverse kidney events in the first year after acute kidney injury. *Clinical Kidney Journal*, 14(2):556–563, 2021.
- [33] Lloyd S Shapley et al. A value for n-person games. 1953.
- [34] Rachael Hwee Ling Sim, Xinyi Xu, and Bryan Kian Hsiang Low. Data valuation in machine learning: "ingredients", strategies, and open challenges. In *IJCAI*, pages 5607–5614. ijcai.org, 2022.
- [35] Siyi Tang, Amirata Ghorbani, Rikiya Yamashita, Sameer Rehman, Jared A Dunnmon, James Zou, and Daniel L Rubin. Data valuation for medical imaging using shapley value and application to a large-scale chest x-ray dataset. *Scientific reports*, 11(1):8366, 2021.
- [36] Jiachen T. Wang, Tianji Yang, James Zou, Yongchan Kwon, and Ruoxi Jia. Rethinking data shapley for data selection tasks: Misleads and merits. In *ICML*. OpenReview.net, 2024.
- [37] Xinyi Xu, Zhaoxuan Wu, Chuan Sheng Foo, and Bryan Kian Hsiang Low. Validation free and replication robust volume-based data valuation. In *NeurIPS*, pages 10837–10848, 2021.

- 450 [38] Pranjul Yadav, Michael S. Steinbach, Vipin Kumar, and György J. Simon. Mining electronic
451 health records (ehrs): A survey. *ACM Comput. Surv.*, 50(6):85:1–85:40, 2018.
- 452 [39] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. INVASE: instance-wise variable
453 selection using neural networks. In *ICLR (Poster)*. OpenReview.net, 2019.
- 454 [40] Xinlu Zhang, Shiyang Li, Zhiyu Chen, Xifeng Yan, and Linda Ruth Petzold. Improving medical
455 predictions by irregular multimodal electronic health records modeling. In *ICML*, volume 202
456 of *Proceedings of Machine Learning Research*, pages 41300–41313. PMLR, 2023.