PACE: Learning Effective Task Decomposition for Human-in-the-loop Healthcare Delivery

Kaiping Zheng
National University of Singapore
kaiping@comp.nus.edu.sg

Gang Chen
Zhejiang University
cg@zju.edu.cn

Melanie Herschel
Universität Stuttgart
Melanie.Herschel@ipvs.uni-stuttgart.de

Kee Yuan Ngiam
National University Health System
kee_yuan_ngiam@nuhs.edu.sg

Beng Chin Ooi
National University of Singapore
ooibc@comp.nus.edu.sg

Jinyang Gao
Alibaba Group
jinyang.gjy@alibaba-inc.com

ABSTRACT

Human-in-the-loop data analysis involves both machine learning models and humans in analytic tasks. In healthcare applications, human-in-the-loop data analysis is crucial in that the model can handle "easy" tasks and hand over "hard" ones to medical experts for assistance and medical judgment, where easy tasks are the ones for which the model can provide high accuracy and hard tasks vice versa. In this process, how to decompose tasks in an effective manner is an important stage. To achieve task decomposition, classification with a reject option is a solution. However, existing studies either directly implement a reject option or dive into the theoretical details of the rejection mechanism. Different from such studies, we aim to optimize general classifiers with a reject option and hence, optimize task decomposition for healthcare applications.

To this end, we first introduce task decomposition for healthcare applications, which is a crucial stage in human-in-the-loop healthcare delivery. We then devise a framework PACE to learn effective task decomposition concentrating on delivering high performance on the easy tasks. PACE is two-level: on the macro level, PACE employs the Self-Paced Learning method to select easy tasks for each training iteration; on the micro level, PACE adapts the weights of selected tasks through its weighted loss revision strategy. Experimental results in two real-world healthcare datasets show that PACE outperforms baselines in terms of their performance on the easy tasks which are expected to be solved by the learning model.

CCS CONCEPTS

• **Applied computing** → *Health informatics.*

KEYWORDS

Healthcare; Task decomposition; Human-in-the-loop

ACM Reference Format:

Kaiping Zheng, Gang Chen, Melanie Herschel, Kee Yuan Ngiam, Beng Chin Ooi, and Jinyang Gao. 2021. PACE: Learning Effective Task Decomposition for Human-in-the-loop Healthcare Delivery. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3448016.3457281

SIGMOD '21, June 20–25, 2021, Virtual Event, China.

ACM ISBN 978-1-4503-8343-1/21/06.
https://doi.org/10.1145/3448016.3457281

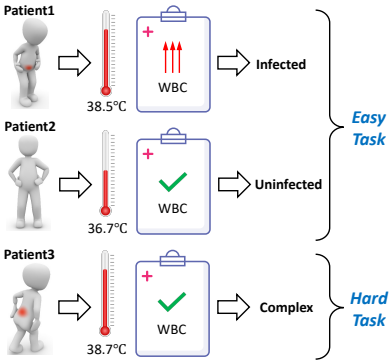


Figure 1: An example of easy tasks and hard tasks.

1 INTRODUCTION

Human-in-the-loop data analysis [1, 17] refers to the data management tasks that require the participation and involvement of both machine learning models and humans, e.g., employing crowdsourcing techniques to harness the humans' expertise for the tasks that are complex and challenging to solely models [37]. Human-in-the-loop data analysis is of vital importance to healthcare delivery. Let's take the disease progression modeling (DPM) [38] application as an example in which we aim to predict each patient's future disease progression trajectory. We first train a DPM model to achieve satisfactory performance based on the training data available. Then we deploy the trained model for inference service, i.e., provide prediction results for newly coming unlabeled tasks. In this inference phase, we let the model handle only the easy tasks that it can well predict, and ask doctors for assistance on the hard ones. After doctors provide their judgment to these hard tasks based on their domain expertise, such tasks become highly valuable labeled ones with doctors' medical knowledge incorporated and should be utilized as new training tasks.

In the human-in-the-loop healthcare delivery discussed above, **task decomposition** is a necessary and important stage as it can differentiate easy tasks from hard tasks and facilitate the process that the model handles easy tasks and hands over hard ones to domain experts. We consider an example scenario in everyday life to demonstrate what are easy tasks and hard tasks. As shown in Figure 1, Patient1 exhibits an elevated temperature above the normal range and an obvious increase in white blood cell (WBC) [43]. Based on such conditions, Patient1 is likely to have some common infections such as those caused by bacteria or viruses [51].

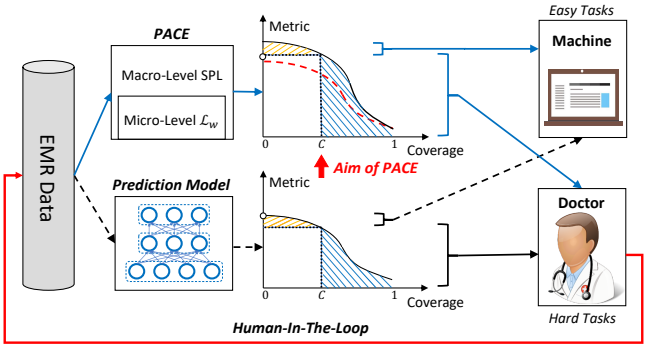


Figure 2: Overview of PACE. PACE learns to optimize task decomposition for human-in-the-loop healthcare delivery, where electronic medical records (EMR) are a major data source. Compared with standard prediction models, the aim of PACE is to improve the performance on easy tasks for which the model can provide high accuracy.

However, Patient2 who has a normal temperature and WBC level, tends to be diagnosed as uninfected. These two cases are relatively common and hence, are easy tasks to be predicted well by the model. Nonetheless, for Patient3 with an abnormal temperature but without a boosted WBC level, the condition is more complex. Patient3 may develop some diseases of the immune system such as HIV [42, 52], or may exhibit some other symptoms closely related to an increased temperature such as hyperthermia [53], rather than have any infectious diseases at all. In this sense, Patient3's conditions are more complex and uncommon; therefore, Patient3 is a hard task that should be handled by doctors.

In such healthcare applications, backed up by doctors, when it is not practical to well predict all tasks based on the current technologies, an AI assistant system that can solve part of the easy tasks with high accuracy can be highly usable. In this situation, we hope to conduct task decomposition, i.e., decompose the given patients into easy tasks and hard tasks. For the easy tasks, we can rely on the model for providing prediction results, while for the hard tasks, we need to turn to medical experts for advice and assistance.

To achieve task decomposition, classification with a reject option is a solution. It is different from standard classification models in that instead of always providing a hard decision, a reject option is allowed for some tasks. Its key idea is to reject to provide a prediction for some tasks in order to achieve better performance on the tasks accepted. There are some existing studies on classification with a reject option [26, 47, 48, 50]. However, they either implement a reject option directly [26, 47] or focus on studying the theoretical characteristics of the rejection mechanism [48, 50].

Different from these existing studies, in this paper, we propose a general framework PACE that aims to o**P**timize t**A**sk de**C**omposition for h**E**althcare applications, where neural networks and deep hierarchical models are often the prevailing choices. The overview of PACE is illustrated in Figure 2. For evaluating the performance of task decomposition, we use a Metric-Coverage plot in which the y-axis denotes the evaluation metric we are interested in and the x-axis denotes the percentage of tasks (ordered from easy to hard[1]) we incorporate into the model. This Metric-Coverage plot can reflect the model's prediction performance clearly. As illustrated in Figure 2, given all input tasks, models provide prediction results shown in the Metric-Coverage plot. Based on the plot, given a coverage C, we can differentiate which tasks are easy for the model to handle and which ones are hard, hence in need of doctors' assistance. As shown in Figure 2, compared with traditional prediction models, the aim of PACE is to optimize general classifiers with a reject option, and boost the generalization performance on easy tasks, i.e., raise the front part of the Metric-Coverage plot[2].

Specifically, PACE can re-weight the given tasks both on the macro level and on the micro level. On the macro level, PACE employs the Self-Paced Learning (SPL) [32] method to select the easy tasks and incorporate them in each training iteration. Drilling down, for the selected tasks, PACE devises a weighted loss revision $\mathcal{L}_w$ to adapt the weights of different tasks on the micro level, specifically, to give more weight to either correctly predicted tasks or confidently predicted tasks.

To summarize, our contributions are as follows:

- We identify task decomposition, which is a vitally necessary and important stage to human-in-the-loop healthcare delivery.
- We devise a general two-level framework PACE to optimize task decomposition for healthcare applications. Specifically, PACE re-weights the task distribution by selecting easy tasks adaptively during the training process on the macro level and re-weighting the loss for each task based on its difficulty on the micro level.
- We evaluate the effectiveness of PACE in two real-world healthcare datasets, for intensive care unit (ICU) patient in-hospital mortality prediction and chronic kidney disease (CKD) patient deterioration prediction respectively. Extensive experiments show that compared with baseline methods, PACE can substantially improve the generalization performance on easy tasks.

The rest of this paper is organized as follows. In Section 2, we review related work. In Section 3, we discuss some preliminaries and in Section 4, we formalize the problem. Section 5 describes our proposed two-level framework PACE to optimize task decomposition for healthcare applications. In Section 6, we demonstrate the experimental results. Finally, we conclude in Section 7.

2 RELATED WORK

2.1 Classification with a Reject Option

Classification with a reject option or selective classification, is an attractive method for improving classification performance in practical applications where a standard model cannot achieve a desired sufficiently low error. Specifically, a reject option is allowed for certain tasks instead of a hard decision and then, these tasks are further fed to exceptional handling such as manual inspection. Hence, classification with a reject option can guarantee the classification performance through filtering out some tasks to reject, and is of vital importance in some mission-critical classification tasks such as medical diagnosis and bioinformatics [26, 47].

[1] In Figure 2, given a pre-defined coverage value C in the x-axis, the tasks in the range $[0, C]$ are considered as easy tasks and the tasks in the range $(C, 1]$ as hard tasks.
[2] This is the case where the evaluation metric has a positive correlation with the model's performance, e.g., area under the ROC curve (AUC).

There exists a long stream of research on classification with a reject option [4, 11, 15, 18, 22, 23, 25–27, 47, 48, 50]. Among these existing studies, some implement a reject option within a specific learning scheme directly [26, 47]. Others focus on the theoretical analysis of rejection mechanisms [4, 11, 15, 18, 22, 23, 25, 27, 48, 50]. Different from them, we propose a general framework PACE for applications in which neural networks and deep hierarchical models are prevailingly employed.

2.2 Task-based Re-weighting Methods

Transfer learning [44] between task domains is proposed to alleviate the problem that training data and testing data may be drawn from different distributions. Curriculum learning [7] can be viewed as a special form of transfer learning in that the chosen initial tasks guide the model to perform better in the final analytics [46]. Its key idea is to start training the model with some easy tasks and then gradually incorporate hard tasks into the training set. One deficiency of curriculum learning lies in that it needs a predefined criterion on the easiness of a task, which is difficult to derive in many real-world applications. To alleviate this, an improved method, self-paced learning (SPL) [32] is proposed, the key idea of which is to define a task's easiness in terms of the model's ability to predict its true output. Specifically, SPL selects easy tasks in each iteration, and updates the parameters repeatedly until convergence.

From another perspective on whether to re-weight the given tasks, there also exists a close relationship between SPL, curriculum learning, and transfer learning. One widely applied approach in transfer learning is instance-transfer, which aims to re-weight the tasks in the source domain for contributing more to the target domain [29, 44]. From this point of view, we can consider curriculum learning and SPL in the instance-transfer approach category as they both need to re-weight the tasks according to a certain criterion. However, the difference lies in that existing studies in instance-transfer focus on the problem that training data and testing data are not under the same distribution, and try to solve it to improve the performance on testing data. Our aim is different as we hope to re-weight the tasks in the training data for improving the model's capability of differentiating easy tasks from hard ones and hence, improve the model's generalization performance on easy tasks.

Another related study is [34], in which a novel Focal Loss based on a revision of the standard cross-entropy loss is proposed. As a result, the proposal can solve the class imbalance problem in the datasets through down-weighting the easy negative tasks and thus, avoid such tasks overwhelming the model training.

2.3 Healthcare Data Analytics

Healthcare data analytics makes use of healthcare data such as EMR data to derive healthcare insights and provide better patient management, more accurate diagnosis, prognosis, etc [5, 8–10, 13, 14, 35, 36, 56–58]. However, most existing studies generally optimize the "average" performance for all tasks, which is often not satisfactory, as in healthcare applications, the accuracy of predictions should be considered as a more crucial requirement than giving predictions to all tasks. The model should not have to predict for all tasks. Instead, it should be trained to obtain the differentiating capability of easy/hard tasks, and only cope with the tasks they are capable of learning and predicting. For the easy tasks which can be predicted by the model accurately, the model can provide predictions, whereas the remaining tasks can be handed over to medical experts for further assistance.

3 PRELIMINARIES

To simplify the notations, we introduce the classification with a reject option in the case of binary classification. Let $\mathcal{X} \in \mathbb{R}^{d_x}$ and $\mathcal{Y} = \{+1, -1\}$ be a feature space and a label space, denoting d_x-dimensional feature vectors and the corresponding label, respectively. A standard binary classifier corresponds to $f : \mathcal{X} \to \mathcal{Y}$, which is learned based on a finite set of M training tasks $S_M = \{(\mathbf{x_i}, y_i)\}_{i=1}^{M}$. In each task $(\mathbf{x_i}, y_i)$, $\mathbf{x_i} \in \mathcal{X}$ is the input feature vector and $y_i \in \mathcal{Y}$ is the corresponding label. All M tasks are assumed to be sampled i.i.d. from an unknown probability distribution $P(X, Y)$ over $\mathcal{X} \times \mathcal{Y}$.

For classification with a reject option, the output is (f, r), where f is a standard binary classifier and $r : \mathcal{X} \to \{0, 1\}$ is a selection function denoting whether to reject a task or not. If $r(\mathbf{x}) = 0$, the classifier will reject this task; otherwise, the classifier will accept it. The value of $r(\mathbf{x})$ is determined by both a function $h : \mathcal{X} \to [0, 1]$ and a predefined rejection threshold τ:

$$r(\mathbf{x}) = \begin{cases} 0 & \text{if } h(\mathbf{x}) \le \tau, \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

Two representative characteristics of a classifier with a reject option (f, r) are *Coverage* and *Risk*.

Definition 3.1. Coverage $C(f, r)$. The coverage of (f, r) is the percentage of the accepted tasks out of all tasks:

$$C(f, r) \triangleq \frac{\sum_{i=1}^{M} I(r(\mathbf{x_i}) = 1)}{M} \tag{2}$$

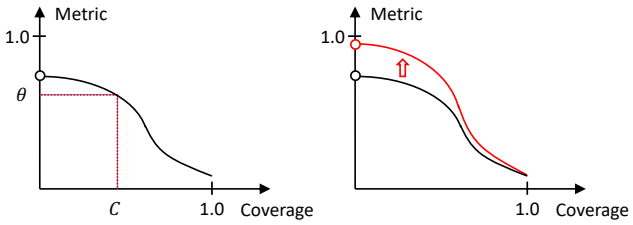
Note that $I(\cdot)$ is the indicator function, which assumes 1 if the argument is true, and 0 otherwise. The risk of a classifier with a reject option is defined as follows.

Definition 3.2. Risk $\mathcal{R}(f, r)$. For a bounded loss function $l : \mathcal{Y} \times \mathcal{Y} \to [a, b]$, the risk of (f, r) is the average loss on the tasks that are accepted:

$$\mathcal{R}(f, r) \triangleq \frac{\sum_{i=1}^{M} l(f(\mathbf{x_i}), y_i) \cdot r(\mathbf{x_i})}{\sum_{i=1}^{M} I(r(\mathbf{x_i}) = 1)} \tag{3}$$

The essence of classification with a reject option is to trade off coverage for better performance, i.e., lower risk in this case. The theoretical characteristics of the Risk-Coverage trade-off are studied in some existing studies such as [48, 50]. Different from them, our proposed framework PACE targets at optimizing the performance of general classifiers with a reject option. To clearly illustrate the improvement of the model's performance, we introduce a Metric-Coverage plot to illustrate the performance of a classifier with a reject option (f, r).

Definition 3.3. Metric-Coverage Plot. In the Metric-Coverage plot, for each coverage C in the x-axis, the y-axis value is the corresponding metric value achieved when the C tasks are accepted as input by (f, r).



(a) Metric-Coverage plot. (b) Aim of PACE.

Figure 3: A Metric-Coverage plot and the aim of PACE.

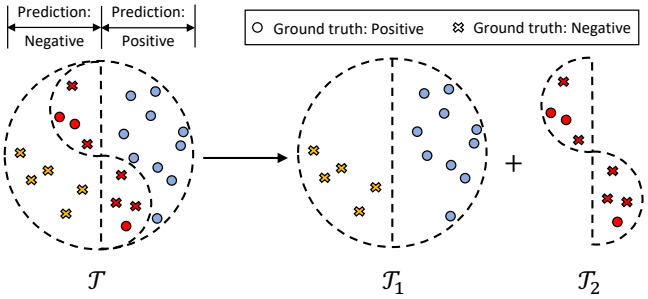


Figure 4: Task decomposition is to decompose the given tasks $\mathcal{T}$ into $\mathcal{T}_1$ (i.e., the tasks that are easy to classify) and $\mathcal{T}_2$ (i.e., the tasks that are hard to classify).

An example Metric-Coverage plot is shown in Figure 3(a). In practice, we can choose the evaluation metrics we are interested in, such as AUC, etc. The aim of PACE is to optimize the classifier with a reject option, i.e., to improve the performance on easy tasks and hence, to raise the front part of the Metric-Coverage plot as illustrated in Figure 3(b).

4 PROBLEM FORMULATION

The key idea of task decomposition is demonstrated in Figure 4, in which the negative/positive sample distribution in both predictions and ground truths are illustrated. Specifically, given a set of tasks $\mathcal{T}$ to classify, task decomposition is to decompose $\mathcal{T}$ into two sets of tasks $\mathcal{T}_1$ and $\mathcal{T}_2$, where $\mathcal{T}_1$ contains tasks easy to classify, and $\mathcal{T}_2$ contains tasks hard to classify.

To achieve task decomposition, we propose to cover a subset of tasks out of all tasks available via a classifier with a reject option in order to achieve the desired Risk-Coverage trade-off based on the specific domain requirements.

For our classifier with a reject option (f, r), when defining the selection function $r(\mathbf{x})$ based on the function $h(\mathbf{x})$ as discussed in Section 3, we set $h(\mathbf{x})$ as the probability of the predicted class. We propose a framework PACE to achieve the optimization of task decomposition as in Definition 4.1. Specifically, PACE adapts the weights of tasks during training in order to differentiate easy tasks from hard ones.

Definition 4.1. Optimization of Task Decomposition. Optimization of task decomposition is to assign different weights to $\forall t_i \in \mathcal{T}$ such that an easy task $t_{i1} \in \mathcal{T}_1$ and a hard task $t_{i2} \in \mathcal{T}_2$ can be differentiated via their corresponding new weights w^t_{i1} and w^t_{i2}.

Table 1: Notations

Notation	Description
$\mathcal{X}$, $\mathcal{Y}$	Feature space, label space
S_M, $(\mathbf{x_i}, y_i)$	M tasks for training, i-th task
(f, r)	Classifier with a reject option
$C(f, r)$	Coverage of (f, r)
$\mathcal{R}(f, r)$	Risk of (f, r)
K	Number of iterations to run for SPL warm-up
λ	A hyperparameter for SPL threshold updating
$\mathbf{W}$	Parameters in neural networks
m_i	Indicator of easiness for i-th task
p	Model's prediction of the class $y = 1$
p_{gt}	Model's prediction of the ground truth class
u_{gt}	Model's computation before activation to p_{gt}
$\mathcal{L}_{CE}$	Standard cross-entropy loss
$\mathcal{L}_{w_1}$	Loss revision in Strategy 1
$p^{w_1}_{gt}$	p_{gt} revision in Strategy 1
$\mathcal{L}_{\bar{w}_1}$	Loss revision in Strategy 1's opposite design
$p^{\bar{w}_1}_{gt}$	p_{gt} revision in Strategy 1's opposite design
γ	A hyperparameter in Strategy 1
$\mathcal{L}_{w_2}$	Loss revision in Strategy 2
$\mathcal{L}_{\bar{w}_2}$	Loss revision in Strategy 2's opposite design
Γ	Number of time steps in GRU
T	A hyperparameter in temperature-based methods

With the differentiating capability of easy/hard tasks, PACE can improve the model's generalization performance on easy tasks in testing data. The notations used in the remaining sections of this paper are summarized in Table 1.

5 THE PACE FRAMEWORK

In this section, we introduce our two-level framework PACE. We start with the macro-level SPL-based training to choose easy tasks to incorporate in each iteration, and then introduce our proposed two strategies for the micro-level weighted loss revision. Finally, we discuss the combination of these two levels.

5.1 Macro-level SPL-based Training

For neural networks, the parameter updating process can be considered as to optimize an objective function involving parameters $\mathbf{W}$, and the objective function is generally composed of a regularization function $\Omega(\cdot)$ and a loss function, e.g., cross-entropy loss ($\mathcal{L}_{CE}$) for classification. The update of $\mathbf{W}$ for optimizing the objective function is:

$$\mathbf{W_{t+1}} = \underset{\mathbf{W}}{\operatorname{argmin}} \left(\Omega(\mathbf{W}) + \sum_{i=1}^{M} \mathcal{L}_{CE}(\mathbf{x_i}, y_i; \mathbf{W})\right) \tag{4}$$

When taking into account SPL-based training, we introduce the indicator variable m_i for each task $(\mathbf{x_i}, y_i)$, denoting whether the

Algorithm 1: SPL-based training with a weighted loss revision $\mathcal{L}_w$ for updating parameters in a neural network

Input: M tasks $D = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_M}, y_M)\}$ where $\mathbf{x_i}$ is the feature vector of the i-th task and y_i is the corresponding label, initial parameters $\mathbf{W_0}$, N_0, λ_0 and tolerance ε.

1: Initialize $t \leftarrow 0$, $N \leftarrow N_0$, $\lambda \leftarrow \lambda_0$.
2: **repeat**
3: Given parameters $\mathbf{W_t}$, obtain the optimum $\mathbf{m_t}$ as $m_i = I(\mathcal{L}_w(\mathbf{x_i}, y_i; \mathbf{W_t}) < \frac{1}{N})$, where $I(\cdot)$ is the indicator function.
4: Denote the tasks $\{(\mathbf{x_i}, y_i)\}$ with $m_i = 1$ as X^s, Y^s, compute $\widehat{Y}^s \leftarrow NeuralNetwork(X^s, \mathbf{W_t})$.
5: Optimize $\mathbf{W_{t+1}}$, $\mathbf{b_{t+1}}$ to minimize $\mathcal{L}_w(\widehat{Y}^s, Y^s) = \Omega(\mathbf{W}) + \sum_{i=1}^{M} m_i \mathcal{L}_w(\mathbf{x_i}, y_i; \mathbf{W}) - \frac{1}{N}\sum_{i=1}^{M} m_i$.
6: $t \leftarrow t + 1$, $N \leftarrow \frac{N}{\lambda}$.
7: **until** $m_i = 1$ for $\forall m_i$, and the objective function cannot be decreased further by tolerance ε.

task is easy or not. Then the revised parameter updating in each iteration is defined below:

$$\begin{aligned}\mathbf{W_{t+1}} &= \underset{\substack{\mathbf{W}\\ \mathbf{m}\in\{0,1\}^M}}{\operatorname{argmin}} \ (\Omega(\mathbf{W}) + \sum_{i=1}^{M} m_i(\mathcal{L}_{CE}(\mathbf{x_i}, y_i; \mathbf{W}) - \frac{1}{N})) \\ &= \underset{\substack{\mathbf{W}\\ \mathbf{m}\in\{0,1\}^M}}{\operatorname{argmin}} \ (\Omega(\mathbf{W}) + \sum_{i=1}^{M} m_i \mathcal{L}_{CE}(\mathbf{x_i}, y_i; \mathbf{W}) - \frac{1}{N}\sum_{i=1}^{M} m_i)\end{aligned} \tag{5}$$

where only the tasks with $m_i = 1$ (i.e., easy tasks) are taken into account, and $1/N$ is the loss threshold for determining whether to incorporate a task in each training iteration or not. In the parameter updating process, we gradually decrease the value of N such that the threshold becomes larger and more tasks will be included in the training. Finally, all tasks will be included. We consider the variables $\mathbf{W}$ and $\mathbf{m}$ as two sets of variables, and employ the alternative convex search (ACS) [6] method to optimize $\mathbf{W}$ and $\mathbf{m}$ alternatively. Therefore, we can obtain an efficient approximate solution to Equation 5.

The algorithm for SPL-based training to update parameters in a neural network is demonstrated in Algorithm 1. As shown, the input includes the tasks with both the feature vectors and the labels. The parameters $\mathbf{W}$ are initialized to $\mathbf{W_0}$. We note that $\mathbf{W_0}$ can be set via SPL warm-up in which we set $m_i = 1$ for all tasks and update the parameters in the neural network for a small yet fixed number of iterations K. The values of N_0, λ_0 and ε are also given. In Line 1, we initialize t that is the iteration number, N that is related to the threshold for selecting tasks in each iteration and λ ($\lambda > 1$ is a constant) that is used to iteratively decrease N.

In Lines 2-6, we iteratively update the parameters $\mathbf{W}$ in the neural network until the stopping criterion is met. In Line 3, given the current parameters, we compute the optimum $\mathbf{m_t}$ based on the indicator function. Alternatively, in Line 4, we select easy tasks X^s, Y^s based on the computed $\mathbf{m_t}$ and further employ $NeuralNetwork(X^s, \mathbf{W_t})$ to predict the labels Y^s as $\widehat{Y}^s$. Next, in Line 5, we update $\mathbf{W}$ according to Equation 5 with the selected tasks, but with a weighted loss revision $\mathcal{L}_w$, which will be further explained in Section 5.2. At the end of this iteration, we update the iteration number t, and decrease N with λ as shown in Line 6.

This iterative process will be ended if all tasks are incorporated in the training and the model converges such that the loss function cannot be decreased further by a pre-defined tolerance ε.

5.2 Micro-level Weighted Loss Revision $\mathcal{L}_w$

We first introduce the standard $\mathcal{L}_{CE}$ for binary classification. Given the ground truth class $y \in \{+1, -1\}$, $\mathcal{L}_{CE}$ is defined in Equation 6, where p is the model's predicted probability of the class $y = 1$.

$$\mathcal{L}_{CE}(p, y) = \begin{cases} -\log p & \text{if } y = 1, \\ -\log(1-p) & \text{otherwise.} \end{cases} \tag{6}$$

For notational simplicity, we define p_{gt} as the model's predicted probability of the ground truth class below.

$$p_{gt} = \begin{cases} p & \text{if } y = 1, \\ 1-p & \text{otherwise.} \end{cases}$$

Hence, the standard $\mathcal{L}_{CE}$ can be expressed as:

$$\mathcal{L}_{CE}(p, y) = \mathcal{L}_{CE}(p_{gt}) = -\log(p_{gt}) \tag{7}$$

We define a new variable u_{gt}, representing the model's computed result before the *sigmoid* activation function ($\sigma(\cdot)$), i.e., $p_{gt} = \sigma(u_{gt})$. When $u_{gt} > 0$, we know $p_{gt} > 0.5$ denoting that the model's prediction is correct.

With the aforementioned definitions, we next elaborate on our two proposed weighted loss revision strategies and explain the underlying rationale for each design.

5.2.1 Strategy 1: Assign More Weight to Correctly Predicted Tasks. The key idea of this strategy is to assign more weight to the tasks that are correctly predicted by the model. This task distribution re-weighting is achieved via modifying the derivative of the loss function w.r.t. u_{gt}, i.e., $d\mathcal{L}/du_{gt}$ and we denote this strategy as $\mathcal{L}_{w_1}$.

Specifically, we revise the original p_{gt} to $p_{gt}^{w_1}$, and revise the standard $\mathcal{L}_{CE}$ to $\mathcal{L}_{w_1}$ as follows.

$$p_{gt}^{w_1} = \sigma(\gamma \cdot u_{gt})$$

$$\mathcal{L}_{w_1}(p_{gt}) = -\frac{1}{\gamma}\log p_{gt}^{w_1}$$

To explain the underlying rationale for this revision strategy more clearly, we derive $d\mathcal{L}_{w_1}/du_{gt}$:

$$\frac{d\mathcal{L}_{w_1}}{du_{gt}} = \sigma(\gamma \cdot u_{gt}) - 1 \tag{8}$$

We set $\gamma = 1/2$ in this strategy. For illustration, we plot this function together with $\mathcal{L}_{CE}$'s derivative in Figure 5. As shown, as soon as $u_{gt} > 0$, which means the task is correctly predicted, the absolute value of $d\mathcal{L}_{CE}/du_{gt}$ is small. However, in this case, our proposed $\mathcal{L}_{w_1}$ still assigns a relatively large absolute derivative value (i.e.,

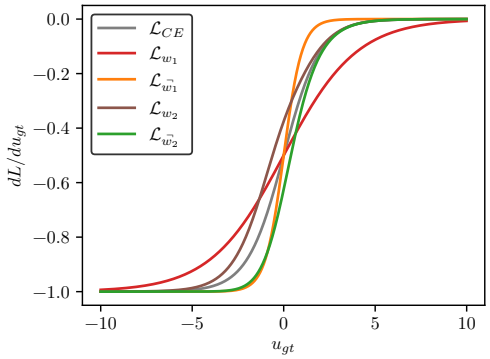


Figure 5: Derivative function $d\mathcal{L}/du_{gt}$ of the standard cross-entropy loss $\mathcal{L}_{CE}$ and four different weighted loss revisions. $\mathcal{L}_{w_1}$ assigns more weight to correctly predicted tasks via a larger $|d\mathcal{L}_{w_1}/du_{gt}|$ when $u_{gt} > 0$, while $\mathcal{L}_{\bar{w_1}}$ is its opposite design. $\mathcal{L}_{w_2}$ assigns more weight to confidently predicted tasks, i.e., less weight to unconfident ones via a smaller $|d\mathcal{L}_{w_2}/du_{gt}|$ when u_{gt} is near 0, while $\mathcal{L}_{\bar{w_2}}$ is its opposite design.

$|d\mathcal{L}_{w_1}/du_{gt}|$) to such tasks, because we think such correctly predicted tasks are informative to optimize the loss function.

For comparison, we propose an opposite weighted loss revision $\mathcal{L}_{\bar{w_1}}$, which assigns less weight to correctly predicted tasks than $\mathcal{L}_{CE}$. The corresponding $p_{gt}^{\bar{w_1}}$ and $\mathcal{L}_{\bar{w_1}}(p_{gt})$ also follow the design of Strategy 1 but with a γ larger than 1. We set $\gamma = 2$ in our experiments. We then derive $d\mathcal{L}_{\bar{w_1}}/du_{gt} = \sigma(2u_{gt}) - 1$ and also plot this derivative function in Figure 5. Compared with the standard $\mathcal{L}_{CE}$, $\mathcal{L}_{\bar{w_1}}$ tends to assign less weight to the tasks with $u_{gt} > 0$, i.e., the correctly predicted tasks.

5.2.2 Strategy 2: Assign More Weight to Confidently Predicted Tasks. In this weighted loss revision, our strategy is to give more weight to the tasks that are predicted more confidently regardless of the correctness of the prediction. To achieve this goal, we multiply a weight function $w(p_{gt})$ to $d\mathcal{L}_{CE}/dp_{gt}$, such that we assign less weight when p_{gt} is near 0.5 denoting an unconfident prediction, but assign more weight when $p_{gt} \to 0$ or $p_{gt} \to 1$ representing a confident prediction.

In our strategy, we use $w(p_{gt}) = 1 - {p_{gt}}^a(1-p_{gt})^a$ and we set $a = 1$ in our experiments. Then we can derive $d\mathcal{L}_{w_2}/dp_{gt}$ and integrate it to $\mathcal{L}_{w_2}(p_{gt})$ as follows. The constant c_1 is added to meet the constraint that when $p_{gt} = 1$, the corresponding loss is 0.

$$\frac{d\mathcal{L}_{w_2}}{dp_{gt}} = \frac{d\mathcal{L}_{CE}}{dp_{gt}} \cdot w(p_{gt}) = -\frac{1}{p_{gt}} + 1 - p_{gt}$$

$$\mathcal{L}_{w_2}(p_{gt}) = -\log p_{gt} + p_{gt} - \frac{1}{2}p_{gt}^2 + c_1$$

We further derive $d\mathcal{L}_{w_2}/du_{gt}$ as follows:

$$\frac{d\mathcal{L}_{w_2}}{du_{gt}} = -\frac{e^{-u_{gt}}}{1+e^{-u_{gt}}} + \frac{e^{-u_{gt}}}{(1+e^{-u_{gt}})^2} - \frac{e^{-u_{gt}}}{(1+e^{-u_{gt}})^3} \tag{9}$$

We also plot this function in Figure 5 for explaining the underlying rationale. As shown, the difference between the standard $\mathcal{L}_{CE}$'s derivative and $\mathcal{L}_{w_2}$'s derivative lies in that when u_{gt} is near 0 (i.e., p_{gt} is near 0.5), $\mathcal{L}_{w_2}$ assigns less weight to such unconfidently predicted tasks. As a result, $\mathcal{L}_{w_2}$ relatively assigns more weight to confident tasks and less weight to unconfident tasks.

To better illustrate our strategy's effectiveness, we propose an opposite weighted loss revision with a weight function $\bar{w}(p_{gt}) = 1 + {p_{gt}}^a(1-p_{gt})^a$ and set $a = 1$ in our experiments. We then derive $d\mathcal{L}_{\bar{w_2}}/dp_{gt}$ and integrate it to $\mathcal{L}_{\bar{w_2}}(p_{gt})$ shown below. c_2 is added for the same reason as in c_1 of $\mathcal{L}_{w_2}$.

$$\frac{d\mathcal{L}_{\bar{w_2}}}{dp_{gt}} = \frac{d\mathcal{L}_{CE}}{dp_{gt}} \cdot \bar{w}(p_{gt}) = -\frac{1}{p_{gt}} - 1 + p_{gt}$$

$$\mathcal{L}_{\bar{w_2}}(p_{gt}) = -\log p_{gt} - p_{gt} + \frac{1}{2}p_{gt}^2 + c_2$$

We further derive $d\mathcal{L}_{\bar{w_2}}/du_{gt}$ as follows:

$$\frac{d\mathcal{L}_{\bar{w_2}}}{du_{gt}} = -\frac{e^{-u_{gt}}}{1+e^{-u_{gt}}} - \frac{e^{-u_{gt}}}{(1+e^{-u_{gt}})^2} + \frac{e^{-u_{gt}}}{(1+e^{-u_{gt}})^3} \tag{10}$$

As illustrated in Figure 5, when u_{gt} is near zero, the absolute derivative value of $\mathcal{L}_{\bar{w_2}}$ is larger than that in $\mathcal{L}_{CE}$. Therefore, the unconfident tasks will pose a larger decrease in the loss, whereas the confident ones pose a smaller decrease in the loss. As a result, this weighted loss revision $\mathcal{L}_{\bar{w_2}}$ will guide the model to assign less weight to the confidently predicted tasks.

5.3 SPL-based Training with $\mathcal{L}_w$

We combine the proposed techniques as shown in Algorithm 1, such that we employ SPL-based training on the macro level and re-weight the task distribution via the weighted loss revision $\mathcal{L}_w$ on the micro level.

We adopt this combined algorithm to train a deep learning model for binary classification. In this model, we employ the gated recurrent unit (GRU) [12], a state-of-the-art recurrent neural network (RNN) model to capture the dynamic information in time-series EMR data.

Specifically, we feed the input time-series EMR data $\mathbf{x}^{(t)}$ ($t \in \{1, 2, \ldots, \Gamma\}$) representing input medical features in consecutive time steps to go through GRU to calculate the hidden state $\mathbf{h}^{(t)}$ for each time step t. Then we obtain the last time step's computation result $\mathbf{h}^{(\Gamma)}$ as the summary of all the previous information. We conduct an affine transformation on $\mathbf{h}^{(\Gamma)}$:

$$\mathbf{u} = \mathbf{W}^{(\mathbf{u})}\mathbf{h}^{(\Gamma)} + \mathbf{b}^{(\mathbf{u})}$$

Then the output $\mathbf{u}$ is fed to the *sigmoid* activation function, and the derived probability p is then used to optimize the loss function, chosen from the standard $\mathcal{L}_{CE}$ or $\mathcal{L}_{w_1}$, $\mathcal{L}_{\bar{w_1}}$, $\mathcal{L}_{w_2}$, $\mathcal{L}_{\bar{w_2}}$ as proposed in Section 5.2.

6 EXPERIMENTAL EVALUATION

6.1 Experimental Set-up

We conduct the experimental evaluation in two real-world datasets.

MIMIC-III Dataset is a public dataset [30] recording EMR data for more than 40,000 patients admitted to the critical care units between 2001 and 2012. In this dataset, each admission denotes a patient's one visit to the ICU. For each admission, an in-hospital mortality label is recorded representing if the patient passes away in the hospital. We extract the admissions longer than 48 hours as tasks, partition each task's first 48 hours' data into two-hour time windows and aggregate the features within each time window

Table 2: Dataset Statistics

Statistics	MIMIC-III	NUH-CKD
# of Features	710	279
# of Tasks	52665	10289
# of Positive Tasks	4299	3268
# of Negative Tasks	48366	7021
Positive Rate	8.16%	31.76%
Time Window Length	2 hours	1 week
# of Time Windows	24	28

as input. Then we model this **ICU patient in-hospital mortality prediction** as a binary classification problem.

NUH Dataset is a real-world longitudinal EMR dataset from National University Hospital (NUH) in Singapore, with which we have close cooperation on healthcare analytics. The NUH dataset records the data of more than 100,000 patients admitted in 2012, and the recorded data includes patients' diagnoses, lab tests, procedures, medications, etc. In our collaborative project with doctors in NUH, we focus on studying the patients with CKD; hence, we choose Stage 3 (or latter stages) [2] CKD patients as our cohort and extract this cohort as the **NUH-CKD** dataset.

According to doctors' advice and medical references [3], it is medically important to monitor the situation of CKD Stage 3 (or latter stages) patients and check if they will develop more serious kidney damage. Therefore, we predict if the patients in our cohort will deteriorate in the future, and model this **CKD patient deterioration prediction** as a binary classification problem. We incorporate each patient's 28-week lab test data as input to predict if his/her condition will deteriorate in the future. We measure the CKD patients' severity based on a lab test Glomerular Filtration Rate (GFR) [41] and a smaller GFR value indicates a more severe state of CKD patients.

Some statistics of these two datasets are shown in Table 2. In this table, "Positive Tasks" represent the admissions in which the patient passes away in hospital in the MIMIC-III dataset and the tasks in which the patient deteriorates in the NUH-CKD dataset. "Positive Rate" is computed as the ratio between "# of Positive Tasks" and "# of Tasks". We find that the MIMIC-III dataset has a more severe class imbalance problem, so we conduct oversampling in the MIMIC-III dataset. Moreover, we note that "# of Time Windows" is the same as the number of time steps Γ in Section 5.3.

We randomly partition all data into 80% training data, 10% validation data, and 10% testing data. During training, we choose the hyperparameters which can achieve the best performance in validation data (when coverage = 1.0), and then apply the trained model on testing data for reporting performance.

For both applications, we use the AUC value as the performance metric and illustrate the AUC-Coverage plot in testing data for evaluation. Each AUC-Coverage plot is drawn 10 times corresponding to 10 repeats and then averaged for illustration. In each plot, we draw lines parallel to the y-axis when the coverage is 0.1, 0.2, 0.3, 0.4, and 1.0 for a clearer comparison.

Specifically, the RNN dimension, i.e., the dimension of $\mathbf{h}^{(t)}$ in Section 5.3, is 32 in both datasets, and the learning rate is 0.001 and 0.002 in the MIMIC-III dataset and the NUH-CKD dataset respectively. Both hyperparameters are tuned via grid search. Other hyperparameters include the batch size of 32, and the epoch number of 100 with early stopping.

6.2 Main Results

As shown later in the ablation study (Section 6.3), our proposed SPL-based training outperforms the standard $\mathcal{L}_{CE}$ (Section 6.3.1) and our proposed $\mathcal{L}_{w_1}$ achieves the best performance among all weighted loss revisions (Section 6.3.2) with $\gamma = 1/2$ (Section 6.3.5). Further, PACE achieves the best performance when the hyperparameter λ that influences the threshold for task selection is set to 1.3 (Section 6.3.4). Therefore, we compare PACE, which is the combination of SPL and $\mathcal{L}_{w_1}$, i.e., our best-performing proposed method with the aforementioned hyperparameter settings, with other methods as the main results.

In this section, we conduct three experiments. First, we compare PACE with several baseline classifiers that are widely used in practice in Section 6.2.1. Second, we compare PACE with temperature-based methods which also revise the derivative function $d\mathcal{L}/du_{gt}$ in Section 6.2.2. Third, we compare PACE with temperature-based methods with SPL-based training in Section 6.2.3.

6.2.1 Comparison with Baseline Classifiers. In this experiment, we select several widely used baselines for comparison as follows.

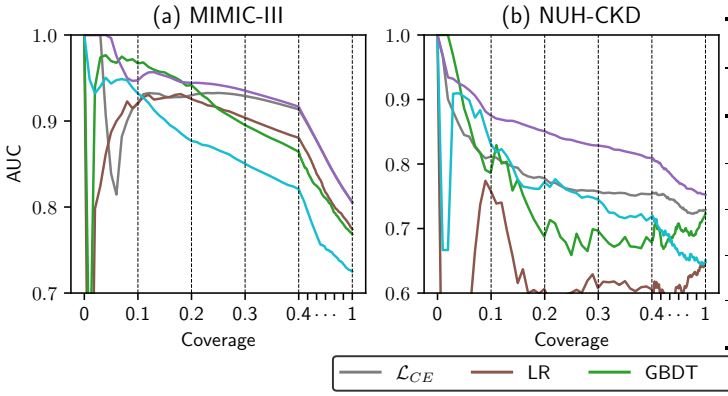
- **LR**, short for Logistic Regression, is a classifier widely applied in various applications [19]. We set $\varphi = 0.001$ in the MIMIC-III dataset and $\varphi = 1$ in the NUH-CKD dataset where the parameter φ controls the regularization strength, and use "liblinear" solver in both datasets.
- **AdaBoost**, short for Adaptive Boosting, is a weighted combination of "weak learners" (i.e., decision trees in this case). AdaBoost is adaptive, as the tasks that are not correctly predicted by previous weak learners will be assigned more weight in the subsequent weak learners [20]. We set *n_estimators* to 50 in the MIMIC-III dataset and 500 in the NUH-CKD dataset.
- **GBDT** is short for Gradient Boosting Decision Tree. GBDT builds an ensemble model composed of "weak learners" (i.e., decision trees in this case) in a forward stage-wise fashion in order to optimize the loss function [21]. We set *n_estimators* = 100 and *max_depth* = 3 in both datasets.

For these three baseline classifiers, we concatenate the time-series features in different time windows as input. Besides, we also add the standard $\mathcal{L}_{CE}$ as a baseline.

- $\mathcal{L}_{CE}$ employs the standard cross-entropy loss function for classification (without SPL-based training). Specifically, $\mathcal{L}_{CE}$ is based on the GRU model to incorporate time-series features as discussed in Section 5.3.

The experimental results of the baseline classifiers against our proposed PACE are demonstrated in Figure 6. For the MIMIC-III dataset, when the coverage is smaller than 0.1, there are too few tasks incorporated in the model, which causes a severe fluctuation in AUC values. Thus, we focus our analysis specific to the coverage range [0.1, 1.0] in the MIMIC-III dataset, and the whole coverage range in the NUH-CKD dataset.

There are some common findings in both datasets. First, our proposed PACE can achieve higher AUC values than the standard $\mathcal{L}_{CE}$. In the MIMIC-III dataset, PACE can effectively raise the front



Dataset	MIMIC-III					NUH-CKD				
Coverage	0.1	0.2	0.3	0.4	1	0.1	0.2	0.3	0.4	1
$\mathcal{L}_{CE}$	0.920	0.930	0.929	0.914	**0.806**	0.811	0.778	0.758	0.753	0.729
LR	0.921	0.926	0.904	0.880	0.774	0.759	0.606	0.608	0.606	0.648
GBDT	**0.967**	0.942	0.895	0.864	0.769	0.786	0.688	0.697	0.679	0.723
AdaBoost	0.932	0.877	0.850	0.821	0.725	0.832	0.772	0.745	0.718	0.650
PACE	0.947	**0.944**	**0.935**	**0.917**	0.805	**0.875**	**0.851**	**0.829**	**0.809**	**0.752**

Figure 6: Comparison results between PACE and baseline classifiers $\mathcal{L}_{CE}$, LR, GBDT and AdaBoost.

part of the AUC-Coverage plot compared with $\mathcal{L}_{CE}$, and in the NUH-CKD dataset, PACE outperforms $\mathcal{L}_{CE}$ in the whole coverage range. This validates the performance improvement from PACE in both datasets.

Second, compared with LR and AdaBoost, PACE achieves better performance in terms of the AUC-Coverage plot in the studied coverage range of both datasets. Compared with GBDT, PACE is slightly worse when the coverage is within $[0.1, 0.15]$ in the MIMIC-III dataset and when the coverage is within $[0, 0.05]$ in the NUH-CKD dataset. After this coverage range in the very beginning, PACE exhibits an obvious advantage over GBDT. We think a possible reason is that boosting-based methods such as GBDT can more quickly recover from the initial fluctuation problem, which is caused by the small number of tasks incorporated in the model, due to the robustness advantage of ensemble models[3].

Third, when the coverage approaches 1.0, we find that PACE and $\mathcal{L}_{CE}$ achieve a higher AUC value than LR, AdaBoost and GBDT. This is because both PACE and $\mathcal{L}_{CE}$ are based on RNN, which can effectively exploit the time-series information in the data and hence, these two models can provide more accurate predictions.

6.2.2 Comparison with Temperature-based Methods. In neural networks, the $softmax$ function is typically used to produce the probability $p_i^{w_T}$ of class i through comparing with other classes j, as shown below [28]:

$$p_i^{w_T} = \frac{exp(u_i/T)}{\sum_j exp(u_j/T)}$$

where the hyperparameter temperature T is normally set to 1 as in the standard $\mathcal{L}_{CE}$. Specifically, temperature T controls the sensitivity to classes with a low probability and a larger T corresponds to a softer probability distribution over classes. In the case of binary classification, $p_i^{w_T}$ is updated as follows:

$$p_i^{w_T} = \frac{1}{1 + exp(-u_i/T)} = \sigma(u_i/T)$$

For notational simplicity, we show the predicted probability of the ground truth class $p_{gt}^{w_T}$ and the corresponding loss function

[3] AdaBoost is also a boosting-based method and hence, should also recover from the initial fluctuation problem. However, AdaBoost does not perform well. The reason may be twofold: (1) AdaBoost is sensitive to outliers and noisy tasks; (2) The way that GBDT creates weak learners based on "pseudo-residuals" is more effective than that of AdaBoost in our two datasets.

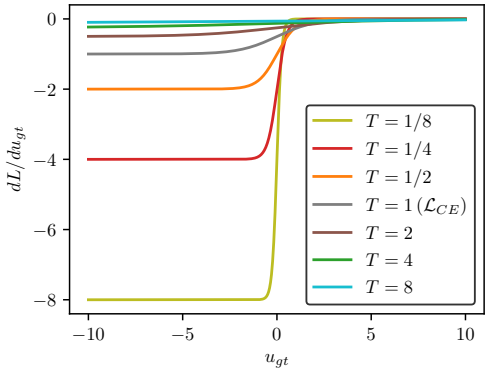


Figure 7: Derivative function $d\mathcal{L}/du_{gt}$ for different T settings. Compared with $\mathcal{L}_{CE}$ (i.e., $T = 1$), different T settings deform the derivative function curve in both directions and hence, adapt the weights of tasks.

$\mathcal{L}_{w_T}(p_{gt})$ below for analysis:

$$p_{gt}^{w_T} = \sigma(u_{gt}/T)$$

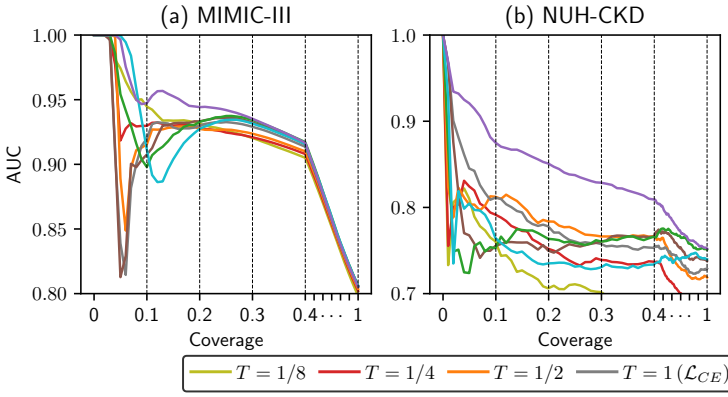
$$\mathcal{L}_{w_T}(p_{gt}) = -\log p_{gt}^{w_T}$$

To investigate the effect of T, we derive $d\mathcal{L}_{w_T}/du_{gt}$:

$$\frac{d\mathcal{L}_{w_T}}{du_{gt}} = \frac{\sigma(u_{gt}/T) - 1}{T} \tag{11}$$

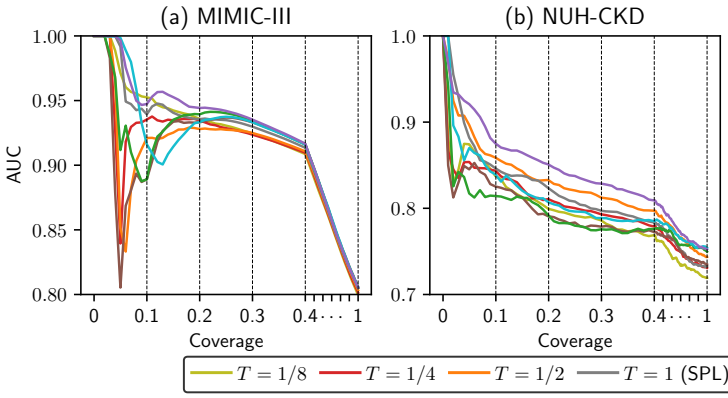
We specify $T \in \{1/8, 1/4, 1/2, 1, 2, 4, 8\}$ and plot the corresponding $d\mathcal{L}_{w_T}/du_{gt}$ with different T settings in Figure 7. As illustrated, compared with $T = 1$ ($\mathcal{L}_{CE}$), different T settings will deform the curve in both the u_{gt} direction and the $d\mathcal{L}_{w_T}/du_{gt}$ direction.

We compare PACE with such temperature-based methods with different temperature settings, and the experimental results are demonstrated in Figure 8. In both datasets, different temperature settings will pose different effects on models' performance. In the MIMIC-III dataset, among various temperature settings, when the coverage is smaller than 0.2, $T = 1/8$ performs best in terms of the AUC-Coverage plot, whereas when the coverage is larger than 0.2, $T = 4$ is the best-performing temperature setting. In the NUH-CKD dataset, we find that $T = 1$, $T = 1/2$, $T = 4$ become the best-performing temperature setting in the coverage range of $[0, 0.1]$, $[0.1, 0.4]$ and $[0.4, 1]$ respectively. This phenomenon confirms the effect of temperature on models' performance, which indicates that



Dataset	MIMIC-III					NUH-CKD				
Coverage	0.1	0.2	0.3	0.4	1	0.1	0.2	0.3	0.4	1
$T = 1/8$	0.945	0.931	0.920	0.905	0.798	0.761	0.706	0.702	0.698	0.677
$T = 1/4$	0.930	0.929	0.921	0.908	0.802	0.792	0.752	0.739	0.737	0.695
$T = 1/2$	0.915	0.927	0.924	0.910	0.804	0.812	0.784	0.766	0.767	0.719
$T = 1$	0.920	0.930	0.929	0.914	**0.806**	0.811	0.778	0.758	0.753	0.729
$T = 2$	0.908	0.933	0.932	0.916	**0.806**	0.755	0.750	0.760	0.766	0.738
$T = 4$	0.898	0.933	0.934	**0.917**	**0.806**	0.753	0.764	0.760	0.766	0.751
$T = 8$	0.911	0.927	0.932	0.916	**0.806**	0.765	0.735	0.731	0.734	0.740
PACE	**0.947**	**0.944**	**0.935**	**0.917**	0.805	**0.875**	**0.851**	**0.829**	**0.809**	**0.752**

Figure 8: Comparison results between PACE and temperature-based methods.



Dataset	MIMIC-III					NUH-CKD				
Coverage	0.1	0.2	0.3	0.4	1	0.1	0.2	0.3	0.4	1
$T = 1/8$	**0.952**	0.936	0.924	0.909	0.800	0.837	0.799	0.786	0.769	0.719
$T = 1/4$	0.935	0.935	0.924	0.910	0.802	0.844	0.811	0.793	0.779	0.731
$T = 1/2$	0.921	0.928	0.925	0.911	0.803	0.859	0.833	0.812	0.797	0.744
$T = 1$	0.939	0.936	0.930	0.914	0.805	0.847	0.823	0.797	0.784	0.733
$T = 2$	0.889	0.933	0.933	0.916	**0.806**	0.825	0.791	0.779	0.773	0.735
$T = 4$	0.889	0.939	0.934	0.916	0.805	0.814	0.792	0.776	0.776	0.750
$T = 8$	0.916	0.933	0.933	0.916	**0.806**	0.839	0.807	0.789	0.785	**0.754**
PACE	0.947	**0.944**	**0.935**	**0.917**	0.805	**0.875**	**0.851**	**0.829**	**0.809**	0.752

Figure 9: Comparison results between PACE and temperature-based methods with SPL-based training.

for each specific application and dataset, this hyperparameter T should be tuned. However, in both datasets, PACE outperforms all different temperature settings. This validates the effectiveness of PACE in improving the performance on easy tasks.

6.2.3 Comparison with Temperature-based Methods with SPL-based Training. We further compare PACE with temperature-based methods with SPL-based training. Specifically, we use $\mathcal{L}_{w_T}$ discussed in Section 6.2.2 as the loss function where $T \in \{1/8, 1/4, 1/2, 1, 2, 4, 8\}$, with SPL-based training.

The experimental results are illustrated in Figure 9, where $T = 1$ corresponds to the standard SPL method. Based on the results, our findings are threefold. First, different temperature settings exhibit varied performance in terms of the AUC-Coverage plot. Specifically, in the MIMIC-III dataset, when the coverage is smaller than 0.2, $T = 1/8$ tends to outperform other settings, whereas when the coverage falls in $[0.2, 1]$, $T = 4$ tends to be the best-performing setting. In the NUH-CKD dataset, when the coverage is smaller than 0.05, $T = 1$ tends to achieve the best performance out of all the temperature settings; however, when the coverage exceeds 0.05, $T = 1/2$ becomes the best-performing setting. In addition, when the coverage is near 1, $T = 4$ and $T = 8$ yield better performance than other settings. Such an advantage over $T = 1$ (i.e., only SPL-based training on the macro level) demonstrates that the temperature-based weighting methods are effective in improving the performance.

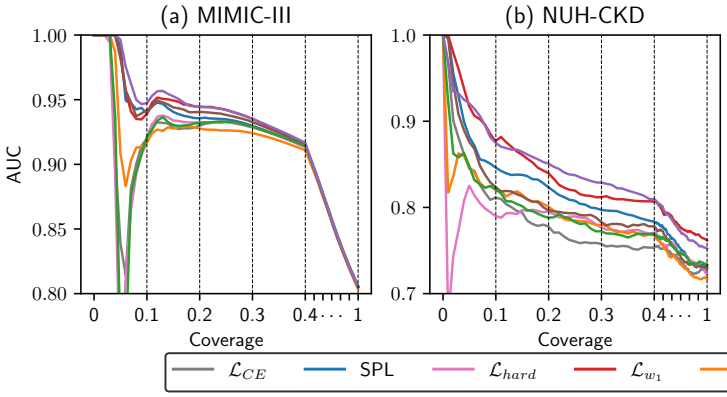
Second, compared with the results of the temperature-based methods in Figure 8, the corresponding methods with SPL-based training exhibit boosted performance on the whole as shown in Figure 9. This confirms the effectiveness of our proposed macro-level SPL-based training.

Third, as illustrated in Figure 9, PACE as the combination of SPL and our proposed weighted loss revision, generally achieves higher AUC values than all the temperature-based methods with SPL-based training. This further validates the advantage of our proposed weighted loss revision over temperature-based weighting methods in improving the performance on easy tasks.

6.3 Ablation Study

6.3.1 PACE's Macro-level SPL-based Training. We evaluate the effectiveness of PACE's macro-level SPL-based training with the results shown in Figure 10. For the SPL warm-up mentioned in Section 5.1, we set K as 1 in the MIMIC-III dataset and 2 in the NUH-CKD dataset.

We observe that SPL-based training outperforms the standard $\mathcal{L}_{CE}$ in both datasets. One reason could be that the hard tasks in healthcare applications may carry some intrinsic noise, such that SPL-based training which utilizes easy tasks for building the model first, can reduce the influence of noisy tasks and thus, can benefit the model's performance on easy tasks. As a result, SPL-based training is effective in improving the prediction performance on easy tasks.



Dataset	MIMIC-III					NUH-CKD				
Coverage	0.1	0.2	0.3	0.4	1	0.1	0.2	0.3	0.4	1
$\mathcal{L}_{CE}$	0.920	0.930	0.929	0.914	**0.806**	0.811	0.778	0.758	0.753	0.729
SPL	0.939	0.936	0.930	0.914	0.805	0.847	0.823	0.797	0.784	0.733
$\mathcal{L}_{hard}$	0.917	0.933	0.928	0.913	0.805	0.790	0.795	0.777	0.770	0.723
$\mathcal{L}_{w_1}$	0.939	**0.945**	**0.935**	0.915	0.805	**0.877**	0.840	0.812	0.807	**0.762**
$\mathcal{L}_{\bar{w_1}}$	0.917	0.928	0.924	0.911	0.803	0.821	0.801	0.780	0.768	0.717
$\mathcal{L}_{w_2}$	0.942	0.941	0.933	0.915	0.805	0.824	0.797	0.783	0.778	0.730
$\mathcal{L}_{\bar{w_2}}$	0.921	0.932	0.929	0.914	**0.806**	0.821	0.788	0.772	0.769	0.734
PACE	**0.947**	0.944	**0.935**	**0.917**	0.805	0.875	**0.851**	**0.829**	**0.809**	0.752

Figure 10: Experimental results of $\mathcal{L}_{CE}$, SPL, $\mathcal{L}_{hard}$, different weighted loss revisions and PACE.

In the MIMIC-III dataset, when the coverage is within $[0.1, 0.3]$, we find that SPL-based training obtains a higher AUC value than $\mathcal{L}_{CE}$, but after 0.3, the two methods tend to behave similarly. However, in the NUH-CKD dataset, SPL-based training shows an advantage in terms of AUC values over $\mathcal{L}_{CE}$ in the whole coverage range, especially when the coverage is around 0.2. After 0.2, the advantage becomes smaller with the coverage increasing. Finally, when the coverage is 1.0, SPL-based training slightly outperforms $\mathcal{L}_{CE}$. The reason for such differences in two datasets may be that there are more hard tasks with more noise in the NUH-CKD dataset than in the MIMIC-III dataset. As a result, the performance improvement on easy tasks brought by PACE's macro-level SPL-based training is more significant in the NUH-CKD dataset, in terms of both the coverage range with a higher AUC value and the improvement in the overall AUC value when the coverage is 1.0.

6.3.2 PACE*'s Micro-level Weighted Loss Revision.* We next compare different weighted loss revisions $\mathcal{L}_{w_1}$, $\mathcal{L}_{\bar{w_1}}$, $\mathcal{L}_{w_2}$ and $\mathcal{L}_{\bar{w_2}}$ on the micro level, and the results are shown in Figure 10.

We observe some common findings in both datasets. First, the performance on easy tasks of $\mathcal{L}_{w_1}$ is better than that of $\mathcal{L}_{\bar{w_1}}$. This means that our proposed Strategy 1 ($\mathcal{L}_{w_1}$) that assigns more weight to correctly predicted tasks is effective, whereas the opposite $\mathcal{L}_{\bar{w_1}}$ is not. The main reason is that hard tasks tend to be tasks that are noisy and hence, difficult to be predicted correctly. In contrast, easy tasks tend to carry more useful information for correct predictions. Therefore, when we re-weight the task distribution by assigning more weight to easy tasks as in $\mathcal{L}_{w_1}$, we manage to reduce the effect caused by the noise from hard tasks and make better use of easy tasks. As a consequence, we can improve the performance on easy tasks.

Second, in Figure 10(a), we observe that $\mathcal{L}_{w_2}$ outperforms $\mathcal{L}_{\bar{w_2}}$ in the MIMIC-III dataset. In Figure 10(b), $\mathcal{L}_{w_2}$ achieves better performance than $\mathcal{L}_{\bar{w_2}}$ when the coverage is in $[0, 0.6]$ and $\mathcal{L}_{w_2}$'s advantage slightly drops when the coverage is in $[0.6, 1.0]$ in the NUH-CKD dataset. Such results demonstrate that our proposed Strategy 2 ($\mathcal{L}_{w_2}$), which assigns more weight to confidently predicted tasks is more effective than the opposite $\mathcal{L}_{\bar{w_2}}$. This may be because the confidently predicted tasks are less noisy and thus, easier to predict. However, the tasks with p_{gt} near 0.5, i.e., the tasks which are predicted unconfidently, carry more noise to the prediction model. Therefore, when we adapt the weights of tasks by assigning more weight to confidently predicted tasks as in $\mathcal{L}_{w_2}$, we can avoid the harmful influence from the noisy tasks, and build a more accurate prediction model.

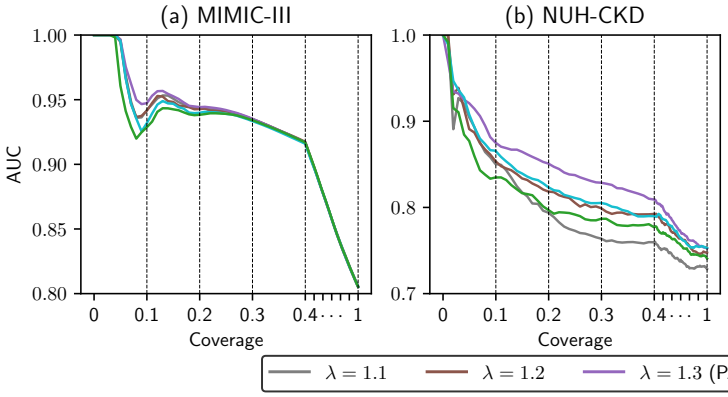
Third, in both datasets, we observe that although both strategies are effective, $\mathcal{L}_{w_1}$ is able to outperform $\mathcal{L}_{w_2}$. These two strategies in nature correspond to two definitions of "easiness", either from whether the prediction is provided correctly or confidently (no matter correctly or not). Therefore, in $\mathcal{L}_{w_2}$, for the tasks which are predicted incorrectly yet confidently, we still assign more weight to them. In this case, such tasks may exert a bad influence on the model's performance and hence, degrade the effectiveness of $\mathcal{L}_{w_2}$.

Moreover, in the MIMIC-III dataset, PACE achieves the best performance in the whole studied coverage range $[0.1, 1.0]$, and outperforms the second best performing method $\mathcal{L}_{w_1}$ when the coverage is in $[0.1, 0.2]$. In the NUH-CKD dataset, the second best performing method is also $\mathcal{L}_{w_1}$, and PACE exhibits an obvious advantage over it when the coverage is about $[0.15, 0.4]$, with a slight drop in the performance later. Such findings confirm that PACE has the capability of differentiating easy tasks from hard ones and hence, is effective in optimizing task decomposition and providing satisfactory prediction performance on easy tasks.

6.3.3 Comparison with $\mathcal{L}_{hard}$. We further compare PACE with the baseline $\mathcal{L}_{hard}$, in which we set a hard cutoff threshold *thres* such that we first filter out the tasks with $p_{gt} \in (thres, 1 - thres)$, and only incorporate the rest, i.e., the tasks with p_{gt} close to 0 or 1. Then we apply SPL-based training on these incorporated tasks, with the weights derived from the *sigmoid* activation function. The corresponding experimental results are illustrated in Figure 10.

Specifically, we vary the hyperparameter *thres* within $\{0.1, 0.2, 0.3, 0.4\}$ ($thres = 0.5$ corresponds to SPL without any tasks filtered out), and find that $thres = 0.4$ and $thres = 0.3$ yield the best performance in the MIMIC-III dataset and the NUH-CKD dataset respectively. Hence, such *thres* settings are adopted in $\mathcal{L}_{hard}$ in the results shown in Figure 10. From the figure, we observe that our proposed PACE outperforms $\mathcal{L}_{hard}$ by a large margin, which confirms the effectiveness of our proposed weighted loss revision.

6.3.4 Effects of Hyperparameter λ in PACE. In this experiment, we investigate the effect of λ on the performance of PACE by affecting the threshold $1/N$ for selecting tasks.



Dataset	MIMIC-III					NUH-CKD				
Coverage	0.1	0.2	0.3	0.4	1	0.1	0.2	0.3	0.4	1
$\lambda = 1.1$	0.942	**0.944**	**0.935**	**0.917**	0.805	0.850	0.794	0.764	0.760	0.729
$\lambda = 1.2$	0.942	0.943	**0.935**	**0.917**	**0.806**	0.854	0.818	0.799	0.792	0.747
$\lambda = 1.3$	**0.947**	**0.944**	**0.935**	**0.917**	0.805	**0.875**	**0.851**	**0.829**	**0.809**	0.752
$\lambda = 1.4$	0.932	0.940	0.933	0.916	0.805	0.866	0.823	0.805	0.790	**0.753**
$\lambda = 1.5$	0.929	0.938	0.934	**0.917**	0.805	0.835	0.797	0.787	0.777	0.741

Figure 11: Experimental results of PACE with different λ settings.

As described in Algorithm 1, N is used to calculate the threshold $1/N$ for selecting tasks in each iteration. Specifically, N is initialized as N_0 and then divided by λ in each iteration. Therefore, λ is a hyperparameter that directly influences the threshold $1/N$ and hence, the speed of SPL-based training in PACE.

We set N_0 as 16, so that $1/N_0$ is sufficiently small and hence, no tasks are selected in the beginning. We then vary λ within $\{1.1, 1.2, 1.3, 1.4, 1.5\}$ to study the effect of λ on the performance of PACE and the experimental results are illustrated in Figure 11. From the results, we find that $\lambda = 1.3$ is the hyperparameter setting that yields the best performance in terms of the AUC-Coverage plot in both datasets.

Specifically, in the MIMIC-III dataset, $\lambda = 1.3$ produces better performance than $\lambda = 1.4$, and even more significant than $\lambda = 1.5$. Moreover, $\lambda = 1.1$ and $\lambda = 1.2$ exhibit similar performance, but both are inferior to $\lambda = 1.3$. In the NUH-CKD dataset, $\lambda = 1.4$ achieves better performance than $\lambda = 1.5$, and $\lambda = 1.2$ yields better performance than $\lambda = 1.1$. The reason for such phenomena can be explained as follows. When λ decreases from 1.5 to 1.3, the threshold $1/N$ increases more slowly and hence, more iterations are generally needed before convergence. Therefore, the easy tasks tend to participate in the training with more iterations and due to the macro-level SPL-based training, the performance of PACE on easy tasks tends to improve accordingly. However, when λ further drops from 1.3 to 1.1, the easy tasks may already be incorporated for many iterations, which tends to cause the overfitting problem on easy tasks. As a result, the performance of PACE degrades.

6.3.5 Effects of Hyperparameter γ in $\mathcal{L}_{w_1}$. Based on the experimental analysis in Section 6.3.2, our proposed Strategy 1 ($\mathcal{L}_{w_1}$) achieves the best performance out of four different weighted loss revisions and hence, is a better-performing micro-level re-weighting strategy in both datasets. In this experiment, we explore the effect of the hyperparameter γ on $\mathcal{L}_{w_1}$'s performance on easy tasks.

Specifically, we vary γ within the range $\{1, 1/2, 1/4, 1/8, 1/16\}$ where $\gamma = 1$ corresponds to the standard $\mathcal{L}_{CE}$, and $\gamma = 1/2$ is the setting we choose for $\mathcal{L}_{w_1}$ when exploring the benefits of the weighted loss revision in Section 6.3.2. We plot the function $d\mathcal{L}/du_{gt}$ with different γ settings in Figure 12. We note that the smaller γ is, the more weight (in terms of the absolute derivative value) $\mathcal{L}_{w_1}$ assigns to correctly predicted tasks.

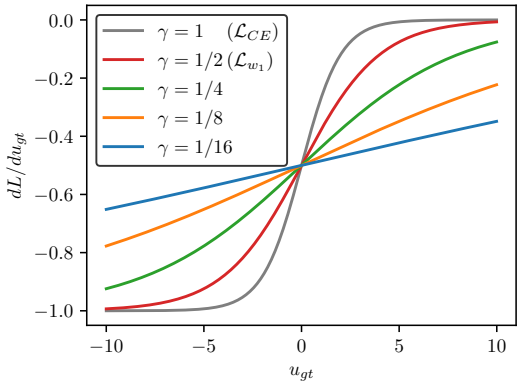
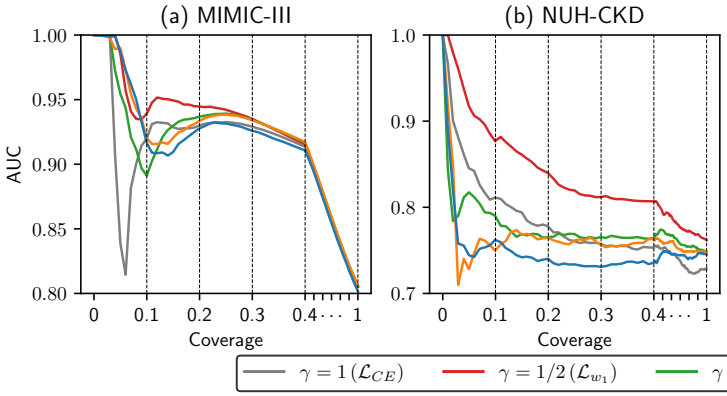


Figure 12: Derivative function $d\mathcal{L}/du_{gt}$ for different γ settings. The smaller γ is, the more weight $\mathcal{L}_{w_1}$ assigns to correctly predicted tasks in terms of $|d\mathcal{L}/du_{gt}|$.

The experimental results are shown in Figure 13. First, we can observe that $\mathcal{L}_{w_1}$ is effective, i.e., $\gamma = 1/2$ ($\mathcal{L}_{w_1}$) outperforms $\gamma = 1$ ($\mathcal{L}_{CE}$), which agrees with the findings in Section 6.3.2. Moreover, a common finding in both datasets is that when γ gradually decreases from 1/2 to 1/4, further to 1/8 and finally to 1/16, $\mathcal{L}_{w_1}$'s performance on easy tasks in terms of the AUC-Coverage plot drops. Such a phenomenon indicates that when γ is set too small, we give too much weight to the correctly predicted tasks such that our model may suffer from the overfitting problem on easy tasks of the training data. Therefore, the model's generalization performance on easy tasks of the testing data degrades. In addition, when γ is set too small, we also decrease the weights of incorrectly predicted tasks to a large extent and might lose some important information for prediction. In a nutshell, we need to adjust the value of γ appropriately to provide satisfactory generalization performance on easy tasks and in our two healthcare datasets, γ should be set to 1/2.

6.4 Effects of Post-hoc Calibration Methods

We have thus far demonstrated how PACE facilitates the learning of effective task decomposition and hence, contributes to accurate and reliable analytics in healthcare applications. However, as a modern neural network, PACE may not be well calibrated, i.e., the calculated probability estimates may not well represent the true correctness



Dataset	MIMIC-III					NUH-CKD				
Coverage	0.1	0.2	0.3	0.4	1	0.1	0.2	0.3	0.4	1
$\gamma = 1$	0.920	0.930	0.929	0.914	**0.806**	0.811	0.778	0.758	0.753	0.729
$\gamma = 1/2$	**0.939**	**0.945**	**0.935**	0.915	0.805	**0.877**	**0.840**	**0.812**	**0.807**	**0.762**
$\gamma = 1/4$	0.891	0.937	0.934	**0.917**	**0.806**	0.790	0.765	0.766	0.764	0.748
$\gamma = 1/8$	0.920	0.934	0.934	**0.917**	**0.806**	0.750	0.765	0.756	0.763	0.748
$\gamma = 1/16$	0.917	0.928	0.926	0.911	0.802	0.762	0.740	0.731	0.736	0.745

Figure 13: Experimental results of $\mathcal{L}_{w_1}$ with different γ settings.

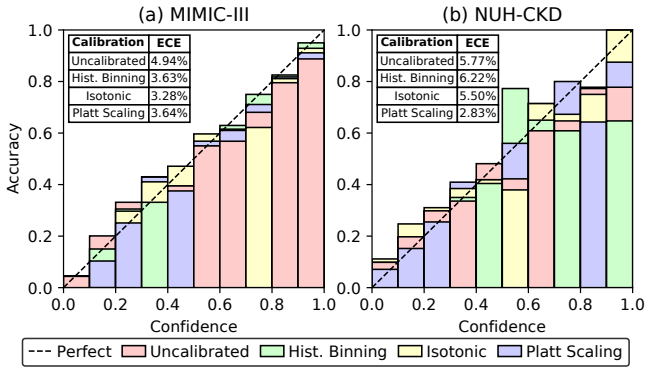


Figure 14: Reliability diagrams of PACE before and after calibration via three methods: histogram binning, isotonic regression, and Platt scaling.

likelihood [24]. To alleviate this issue, some post-processing confidence calibration methods are proposed and widely applied to supplement the models' prediction results with a calibrated confidence measure [24, 31, 49]. In this section, we discuss how these post-hoc calibration methods [33] can supplement useful auxiliary information to PACE in healthcare applications.

We illustrate the reliability diagrams [16, 40] (i.e., diagrams showing accuracy as a function of confidence) of PACE before and after calibration in Figure 14. We note that if a model is perfectly calibrated, meaning that confidence is exactly the expected accuracy, then the reliability diagram should plot the identity function (shown as the diagonal in Figure 14). Hence, any deviation from the diagonal denotes the presence of miscalibration. Besides the reliability diagrams, another metric Expected Calibration Error (ECE) [39] is also a widely adopted empirical metric for measuring the performance of the model calibration. The smaller ECE is, the better the calibration method performs. We show the ECE metrics on the top of each diagram in Figure 14 corresponding to each dataset.

We next evaluate the performance of several representative post-hoc calibration methods including histogram binning [54], isotonic regression [55] and Platt scaling [45]. As shown in Figure 14, the calibration methods generally manage to improve the performance of PACE in terms of ECE. Specifically, isotonic regression and Platt scaling are the best-performing one in the MIMIC-III dataset and the NUH-CKD dataset respectively.

These findings confirm the effectiveness of the post-processing methods in calibrating the predictions of PACE. Such calibrated confidence estimates are highly helpful to clinical decision making, in that PACE can hence better differentiate easy tasks and hard tasks, which can be handled separately by either machines or humans. As a consequence, both machines and humans participate in the analytics and PACE can help render unto human/machine on what is due in a more reliable manner. This has a far-reaching meaning for more reliable analytics in healthcare applications and even in some other critical applications.

7 CONCLUSIONS

In this paper, we identify the vital necessity and importance of task decomposition, which is a critical stage of human-in-the-loop healthcare delivery. We then propose a two-level framework PACE to learn effective task decomposition for healthcare applications. Specifically, PACE employs SPL-based training on the macro level and a weighted loss revision on the micro level to assign more weight to easy tasks.

We conduct extensive experiments in two real-world healthcare datasets. The results show that both PACE's macro-level SPL-based training and PACE's micro-level weighted loss revision strategies (especially $\mathcal{L}_{w_1}$) can improve the prediction performance on easy tasks compared with the baselines. Furthermore, by combining these two levels, PACE achieves a larger improvement as shown in the AUC-Coverage plot, and achieves a better performance on easy tasks than all the baselines, including the standard $\mathcal{L}_{CE}$, baseline classifiers, temperature-based methods and temperature-based methods with SPL-based training.

8 ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful comments. This research is supported by the National Research Foundation Singapore under its AI Singapore Programme (Award Number: AISG-100E-2018-007). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Daniel J. Abadi, Rakesh Agrawal, Anastasia Ailamaki, Magdalena Balazinska, Philip A. Bernstein, Michael J. Carey, Surajit Chaudhuri, Jeffrey Dean, AnHai Doan, Michael J. Franklin, Johannes Gehrke, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, H. V. Jagadish, Donald Kossmann, Samuel Madden, Sharad Mehrotra, Tova Milo, Jeffrey F. Naughton, Raghu Ramakrishnan, Volker Markl, Christopher Olston, Beng Chin Ooi, Christopher Ré, Dan Suciu, Michael Stonebraker, Todd Walter, and Jennifer Widom. 2014. The Beckman Report on Database Research. *SIGMOD Rec.* 43, 3 (2014), 61–70.

[2] The Renal Association. 2020. *CKD stages.* Retrieved Dec 1, 2020 from https://renal.org/health-professionals/information-resources/uk-eckd-guide/ckd-stages

[3] The Renal Association. 2020. *Stage 3 CKD.* Retrieved Dec 1, 2020 from https://renal.org/health-professionals/information-resources/uk-eckd-guide/ckd-stage-g3

[4] Peter L. Bartlett and Marten H. Wegkamp. 2008. Classification with a Reject Option using a Hinge Loss. *J. Mach. Learn. Res.* 9 (2008), 1823–1840.

[5] Inci M. Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K. Jain, and Jiayu Zhou. 2017. Patient Subtyping via Time-Aware LSTM Networks. In *KDD*. ACM, 65–74.

[6] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. 2005. *Nonlinear Programming - Theory and Algorithms, Third Edition.* Wiley.

[7] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML (ACM International Conference Proceeding Series)*, Vol. 382. ACM, 41–48.

[8] Shaofeng Cai, Kaiping Zheng, Gang Chen, H. V. Jagadish, Beng Chin Ooi, and Meihui Zhang. 2021. ARM-Net: Adaptive Relation Modeling Network for Structured Data. In *SIGMOD Conference*. ACM.

[9] Zhengping Che, David C. Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. 2015. Deep Computational Phenotyping. In *KDD*. ACM, 507–516.

[10] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* 8, 1 (2018), 1–12.

[11] Hong Chen, Luoqing Li, and Yuan Yan Tang. 2009. Analysis of Classification with a Reject Option. *Int. J. Wavelets Multiresolution Inf. Process.* 7, 3 (2009), 375–385.

[12] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*. ACL, 1724–1734.

[13] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. 2017. GRAM: Graph-based Attention Model for Healthcare Representation Learning. In *KDD*. ACM, 787–795.

[14] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter F. Stewart. 2016. RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism. In *NIPS*. 3504–3512.

[15] C. K. Chow. 1970. On optimum recognition error and reject tradeoff. *IEEE Trans. Inf. Theory* 16, 1 (1970), 41–46.

[16] Morris H DeGroot and Stephen E Fienberg. 1983. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)* 32, 1-2 (1983), 12–22.

[17] AnHai Doan. 2018. Human-in-the-Loop Data Analysis: A Personal Perspective. In *HILDA@SIGMOD*. ACM, 1:1–1:6.

[18] Ran El-Yaniv and Yair Wiener. 2010. On the Foundations of Noise-free Selective Classification. *J. Mach. Learn. Res.* 11 (2010), 1605–1641.

[19] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.* 9 (2008), 1871–1874.

[20] Yoav Freund and Robert E. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* 55, 1 (1997), 119–139.

[21] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[22] Giorgio Fumera, Fabio Roli, and Giorgio Giacinto. 2000. Reject option with multiple thresholds. *Pattern Recognit.* 33, 12 (2000), 2099–2101.

[23] Yves Grandvalet, Alain Rakotomamonjy, Joseph Keshet, and Stéphane Canu. 2008. Support Vector Machines with a Reject Option. In *NIPS*. Curran Associates, Inc., 537–544.

[24] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *ICML (Proceedings of Machine Learning Research)*, Vol. 70. PMLR, 1321–1330.

[25] L Gyorfi, Z Gyorfi, and I Vajda. 1979. Bayesian decision with rejection. *Problems of control and information theory* 8, 5-6 (1979), 445–452.

[26] Blaise Hanczar and Edward R. Dougherty. 2008. Classification with reject option in gene expression data. *Bioinform.* 24, 17 (2008), 1889–1895.

[27] Lars Kai Hansen, Christian Liisberg, and Peter Salamon. 1997. The error-reject tradeoff. *Open Systems & Information Dynamics* 4, 2 (1997), 159–184.

[28] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015).

[29] Jing Jiang and ChengXiang Zhai. 2007. Instance Weighting for Domain Adaptation in NLP. In *ACL*. The Association for Computational Linguistics.

[30] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3 (2016).

[31] Ananya Kumar, Percy Liang, and Tengyu Ma. 2019. Verified Uncertainty Calibration. In *NeurIPS*. 3787–3798.

[32] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-Paced Learning for Latent Variable Models. In *NIPS*. Curran Associates, Inc., 1189–1197.

[33] Fabian Küppers, Jan Kronenberger, Amirhossein Shantia, and Anselm Haselhoff. 2020. Multivariate Confidence Calibration for Object Detection. In *CVPR Workshops*. IEEE, 1322–1330.

[34] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017. Focal Loss for Dense Object Detection. In *ICCV*. IEEE Computer Society, 2999–3007.

[35] Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wetzel. 2016. Learning to Diagnose with LSTM Recurrent Neural Networks. In *ICLR (Poster)*.

[36] Zachary C. Lipton, David C. Kale, and Randall C. Wetzel. 2016. Directly Modeling Missing Data in Sequences with RNNs: Improved Classification of Clinical Time Series. In *MLHC (JMLR Workshop and Conference Proceedings)*, Vol. 56. JMLR.org, 253–270.

[37] Xuan Liu, Meiyu Lu, Beng Chin Ooi, Yanyan Shen, Sai Wu, and Meihui Zhang. 2012. CDAS: A Crowdsourcing Data Analytics System. *Proc. VLDB Endow.* 5, 10 (2012), 1040–1051.

[38] DR Mould. 2012. Models for disease progression: new approaches and uses. *Clinical Pharmacology & Therapeutics* 92, 1 (2012), 125–131.

[39] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. 2015. Obtaining Well Calibrated Probabilities Using Bayesian Binning. In *AAAI*. AAAI Press, 2901–2907.

[40] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *ICML (ACM International Conference Proceeding Series)*, Vol. 119. ACM, 625–632.

[41] Lab Tests Online. 2020. *Glomerular Filtration Rate.* Retrieved Dec 1, 2020 from https://labtestsonline.org/tests/estimated-glomerular-filtration-rate-egfr

[42] Lab Tests Online. 2020. *HIV Infection and AIDS.* Retrieved Dec 1, 2020 from https://labtestsonline.org/conditions/hiv-infection-and-aids

[43] Lab Tests Online. 2020. *White Blood Cell Count (WBC).* Retrieved Dec 1, 2020 from https://labtestsonline.org/tests/white-blood-cell-count-wbc

[44] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359.

[45] John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* 10, 3 (1999), 61–74.

[46] Sebastian Thrun. 2012. *Explanation-based neural network learning: A lifelong learning approach.* Vol. 357. Springer Science & Business Media.

[47] Daniel Shu Wei Ting, Carol Yim-Lui Cheung, Gilbert Lim, Gavin Siew Wei Tan, Nguyen D Quang, Alfred Gan, Haslina Hamzah, Renata Garcia-Franco, Ian Yew San Yeo, Shu Yen Lee, et al. 2017. Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. *Jama* 318, 22 (2017), 2211–2223.

[48] Marten Wegkamp, Ming Yuan, et al. 2011. Support vector machines with a reject option. *Bernoulli* 17, 4 (2011), 1368–1385.

[49] Jonathan Wenger, Hedvig Kjellström, and Rudolph Triebel. 2020. Non-Parametric Calibration for Classification. In *AISTATS (Proceedings of Machine Learning Research)*, Vol. 108. PMLR, 178–190.

[50] Yair Wiener and Ran El-Yaniv. 2011. Agnostic Selective Classification. In *NIPS*. 1665–1673.

[51] Wikipedia. 2020. *Fever.* Retrieved Dec 1, 2020 from https://en.wikipedia.org/wiki/Fever

[52] Wikipedia. 2020. *HIV.* Retrieved Dec 1, 2020 from https://en.wikipedia.org/wiki/HIV

[53] Wikipedia. 2020. *Hyperthermia.* Retrieved Dec 1, 2020 from https://en.wikipedia.org/wiki/Hyperthermia

[54] Bianca Zadrozny and Charles Elkan. 2001. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *ICML*. Morgan Kaufmann, 609–616.

[55] Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *KDD*. ACM, 694–699.

[56] Kaiping Zheng, Shaofeng Cai, Horng Ruey Chua, Wei Wang, Kee Yuan Ngiam, and Beng Chin Ooi. 2020. TRACER: A Framework for Facilitating Accurate and Interpretable Analytics for High Stakes Applications. In *SIGMOD Conference*. ACM, 1747–1763.

[57] Kaiping Zheng, Jinyang Gao, Kee Yuan Ngiam, Beng Chin Ooi, and James Wei Luen Yip. 2017. Resolving the Bias in Electronic Medical Records. In *KDD*. ACM, 2171–2180.

[58] Kaiping Zheng, Wei Wang, Jinyang Gao, Kee Yuan Ngiam, Beng Chin Ooi, and James Wei Luen Yip. 2017. Capturing Feature-Level Irregularity in Disease Progression Modeling. In *CIKM*. ACM, 1579–1588.