2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Layer-Wise Stability Optimization for Accurate and Reliable Prediction on Clinical Lab Data

Shuang Zhang [0009-0008-6894-7248], Mei Wang [0000-0001-7597-6153] and Qiao Pan [0009-0008-5962-2214]

College of Computer Science and Technology, DongHua University, Shanghai 201620, China
wangmei@dhu.edu.cn

Abstract. Clinical lab tests are essential for disease diagnosis, medical treatment and predictive modeling tasks within healthcare. Traditional learning methods often struggle with lab test data that include imprecision ranges. In this paper, we tackle the challenge of improving prediction stability without generating additional training samples nor compromising accuracy. We reformulate the learning problem as a multi-objective optimization task, where accuracy and stability are both key objectives. To accomplish this, we develop a novel approach for calculating stability loss, by decomposing stability loss into a cumulative process, propagated layer by layer. We then formulate a stability-enhanced loss (SELoss) function to control the layer-wise output errors and maintain prediction precision. In addition, we design a multi-stage learning mechanism to control instability in each layer, especially in the initial layers. These components regulate the learning process, achieving a much-improved balance between accuracy and stability. Using three real-world datasets, experimental results demonstrate that SELoss achieves more accurate and stable predictions across various tasks by reducing the instability of each layer. Also, as input perturbation increases, the rise in output instability slows down.
Keywords: Prediction stability, Neural networks, Imprecise data, Healthcare.

1 Introduction

In healthcare, it is highly desirable to evaluate the current situation of a patient and accurately predict his/her disease development. This evaluation and prediction provide a foundation for treatments such as the medication strategy, unconventional inspection, or other early active interventions. Deep neural networks (DNNs) have been increasingly applied to the prediction, prevention, diagnosis, and prognosis of diseases, showing significant potential to aid in better decision making [1,2]. Although DNNs exhibit their merits in various tasks, the performance in noise, disturbance, and imprecise data remain a challenge [3,4]. Developing accurate and stable deep learning models is an urgent problem.

Clinical lab tests play an important role in today's healthcare. From early detection of diseases to diagnosis to personalized treatment programs, lab tests guide more than 70% of medical decisions and personalized medications [5]. However, due to limitations of equipment, instruments, materials, test methods, etc., data inaccuracy always

occurs. Clinical laboratories adhere to specified quality control and process control protocols to ensure that test results fall within tolerable ranges or imprecision ranges, where values are acceptable despite being imprecise [6]. Since the values in this range are deemed acceptable, predictions should ideally remain consistent or stable within this imprecision range. However, due to the inherent vulnerability of deep learning models [7], achieving this stability is challenging.

One naive approach to address this problem is to identify all unstable samples within the imprecision range space, add them to the training dataset, and enhance training using a method similar to adversarial training [8-10]. However, identifying all unstable samples is both costly and challenging. More importantly, existing work indicates that introducing additional samples could negatively impact the prediction accuracy of the current samples [11,12].

Is it possible to enhance prediction stability without generating additional training samples nor compromising accuracy? This paper addresses this challenge by reframing the learning problem as a multi-objective optimization problem, where both accuracy and stability are key objectives. We further explore how to define and elevate stability loss. In the past, stability has been considered in relation to the final decision boundary [13,14]. However, we argue that stability is fundamentally connected to the learning process. Even with a well-learned decision boundary, instability can arise if two classes are close to each other in any layer during training.

To address this, we develop a novel approach for calculating stability loss. We begin by decomposing the stability loss into a cumulative process propagated layer by layer. Through evaluating the initial perturbation intensity and tracking its through each layer, we then derive the final perturbation output. Consequently, we design a stability-enhanced loss (SELoss) function, in which the stability loss increases stability by controlling the output errors in each layer of the DNN caused by the input. Meanwhile, accuracy loss controls the precision of the predictions. It is evident that initial instability tends to propagate and amplify layer by layer, making it crucial to control instability in each layer, especially in the initial layers. Building on this observation, we further design a multi-stage learning mechanism to focus more on the initial layers. These components collectively regulate the learning process of the model, achieve a much-improved balance between accuracy and stability.

The main contributions of this paper are as follows:

A novel SELoss function to improve stability and accuracy through a multi-objective optimization learning process.

A stability metric that relates to the learning process of the model through a cumulative process propagated layer by layer rather than the final decision boundary.

Experimental evaluation on multiple real-world medical datasets confirms that our SELoss-based approach achieves more accurate and stable predictions across various tasks. Additionally, as input perturbation increases, the rise in output instability is slower.

The paper is organized as follows. Section 2 introduces the motivations. Section 3 introduces the proposed SELoss. Section 4 presents experimental results. Section 5 discusses the related work. Section 6 concludes the paper.

2 Motivations

In this section, we first present an application example to illustrate how data imprecision could cause wrong prediction results. We then describe some of the follow-up work to improve model quality and review the notion of stability formulated to address quality of models concerning sensitivity of prediction results within allowed input data imprecision range. These discussions finally lead to the main problem in this paper: correlation of stability and precision, the two important aspects of models.

We now illustrate the impact of imprecision on deep learning models using a simple and intuitive example. In this example, a deep learning model predicts the risk of diabetes based on eight patient features, including age, DPF (the output of diabetes pedigree function), body mass, insulin level, skin thickness, blood pressure, glucose concentration, and number of pregnancies. The model's output is a binary label indicating whether the patient has diabetes.

Consider a patient X from the dataset. The top table in Fig.1 displays her 8 features, including 5 lab test results. Using these original data, the prediction result is shown in the second row of the bottom table in Fig.1 with the predicted label 0, indicating a low risk. To find out how data imprecision would impact prediction results we introduce a small margin of $\pm 1\%$ to the original lab test data. The middle table of Fig.1 lists the slightly shifted data with the imprecision interval. Using this shifted data, the prediction model would forecast the risk label 1, which means ``high risk''. Had this prediction result is used, the suggested diagnosis and treatment would be different.

The last row of the bottom table in Fig.1 provides the number of samples with label changes in the 400 test samples within $\pm 10\%$ imprecision range. We can see that the labels of 74 patients or 18.5% have changed, which is unacceptable for safety-critical applications. Note that our example only has 5-dimensional lab test data. As the number of dimensions increases and the prediction difficulty grows, the uncertainty caused by imprecision could rise higher, making this issue more crucial.

This negative impact of imprecise data on learned models led to subsequent studies. In [3,4], an alternative loss function named (IRloss) was developed, in an attempt to capture the essence of data imprecision range; Experimental results show improved precision for a few prediction problems.

A need from applications arises in our discussions with management of SHDC. SHDC serves the city government of Shanghai, China through collecting and managing all patient data from all 38 top tier hospitals. SHDC routinely evaluates learned models and make adoption recommendations to the city agency. The impact of imprecise data on models was presented to SHDC, which is then interested in objectively measuring such impact. As a preliminary study, the notion of *stability* was developed and reported in [13]: a stable model is less impacted by the data fluctuation (within the acceptable imprecision range). Experimental evaluation demonstrated that this measure reflects the impact fairly well.

However, at a fundamental level there are many questions left unanswered. This paper addresses an important one: Is it possible to tweak learning methods to correlate stability and (prediction) precision? The main result of this paper is a positive answer:

Enhancing the learning capability of the intermediate layers, especially the initial layers, in the model will improve both the stability and accuracy of the predictions.

The original lab test data x

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	DPF	Age
1	99	72	30	18	38.6	0.412	21

The data generated from the imprecision range x'

1	99.99	71.28	29.7	18.18	38.214	0.412	21

↓

The true label	0	negative
The predicted label based on x	0	negative
The predicted label based on x'	1	positive
Number of samples with label changes in 400 test samples		74

Fig. 1. Test data, shifted data, and prediction results

3 Stability Enhanced Loss

In this section, we first introduce a prediction model served as the basis in our study. We then formulate the correlation between model stability and inter-layer learning. Based on this, we develop a new loss function named SELoss, aiming to achieve combined optimization of accuracy and stability.

3.1 The Prediction Model

In this paper, the prediction model is implemented by the deep neural network. Let $f: R^d \to R^c$ represent a deep neural network, where d denotes the number of input features, and c represents the number of outputs. For an input instance $x \in R^d$, f can be written in the following form:

$$f(x) = Wh_n\left(h_{n-1}\left(\cdots h_2\left(h_1(x)\right)\cdots\right)\right) + b \tag{1}$$

where f consists of a series of fully connected layers $h_i (i = 1, \ldots, n)$ n is the number of layers in the model, W and b are the weight matrix and the bias vector of the output layer. Note that each layer includes a linear transformation followed by a nonlinear activation function.

Let a_i be the input to the i-th layer, $h_i(a_i)$ the output of the i-th layer and simultaneously as the input for the $(i+1)$-th layer.

$$a_{i+1} = h_i(a_i) = \sigma(w_i a_i + b_i), \quad i = 1, \ldots, n \tag{2}$$

where w_i's and b_i's are the weight matrix and bias vector of the i-th layer, and σ is the nonlinear activation function. We use θ to denote the parameter set of model f is always learned by minimize the following loss:

$$L(x, y, \theta) = l_{acc}\left(y, f_\theta(x)\right) \tag{3}$$

where l_{acc} is the loss function to determine how close between the model output $f_\theta(x)$ and the true output y. or regression task, $L(x, y, \theta)$ could be the mean-squared-error (MSE) loss or mean-absolute-error (MAE) loss. For classification, $L(x, y, \theta)$ could be the cross-entropy loss.

In practical applications, x is often measurement data. For example, for disease progress prediction models in medical applications, x may consist of laboratory test results from the patient. Due to the limitations of equipment, instruments, materials, test methods, etc., imprecision of data always occurs.

Let $IPR(x)$ denote the imprecision range spacefor a given data value x. We say the model is stable if $f(x)$ and $f(x')$ have the same value whenever $x' \in IPR(x)$. It is evident that our goal is for the model to be not only close to the ground truth but also as stable as possible, meaning that $f(x)$ and $f(x')$ should be as close as possible. Here we use l_{stab} to measure the difference or loss between $f(x)$and $f(x')$. Consequently, our learning process can be transformed into a multiple objective optimization problem. This means that we can formulate a new loss function as follows:

$$L(x, y, \theta) = \alpha\, l_{acc} + \beta\, l_{stab} \tag{4}$$

The loss function $L(x, y, \theta)$ incorporates stability as a learning objective, similar to accuracy, to improve the model learning process, in order for the resulting model to achieve high accuracy and stability simultaneously.

According to Equation (4), this loss function should find an optimal set of parameters that optimizes both accuracy and stability. To make our idea feasible, an important issue needs to be addressed: for each training sample, how to calculate the model's prediction stability? A naive use of $f(x') - f(x)$ is not feasible, as it requires computing $f(x')$ individually for each sample, and x' needs to be involved in the training process. Previous work [11,12] has widely demonstrated that introducing additional samples could potentially compromise the accuracy of the current sample. To address this issue, we performed layer-by-layer decomposition modeling of the perturbation caused by the input. The perturbation caused by each layer is used to determine the final output perturbation. By employing a first-order Taylor expansion to approximate instability, the final stability loss is calculable and quantifiable. The total loss is composed of accuracy loss (traditional loss) and the instability loss. These two components of the loss jointly control the learning process of the model, thereby achieving the combined optimization of accuracy and stability.

3.2 Correlation between stability and inter-layer learning of the model

Commencing from a foundation of theoretical derivation, we would like to establish the correlation between model stability and the learning dynamics across its layers. Through rigorous mathematical derivations, we can characterize more precisely how imprecision navigate through neural network layers, elucidating the relationship between output perturbation at each layer and the initial input imprecision. The goal is to leverage this theoretical groundwork to engineer more efficacious loss functions, thereby enhancing the model's stability when confronted with ambiguous input data.

This refined methodology demonstrates how imprecision propagation can be managed during model training. It paves the way for achieving more consistent performance in practical applications, further validating and ensuring the effectiveness and broad applicability of our novel approach.

For a given input instance x, the instance x' is derived from $IPR(x)$. The magnitude of their discrepancy is given by $|x - x'| = r_0$. Both x and x' are fed into the same prediction model f. The discrepancy for the pair of instance x and x' observed in the outputs after the forward pass through the i-th layer of this network is labeled as r_i, and r_{i-1} for the $i-1$ layer. r_i can be expressed as:

$$r_i = |\, h_i(a_i + r_{i-1}) - h_i(a_i)| \tag{5}$$

By leveraging the first-order Taylor expansion at a_i, we approximate $h_i(a_i + r_{i-1})$ as:

$$h_i(a_i + r_{i-1}) \approx h_i(a_i) + \frac{dh_i(a_i)}{da_i} \cdot r_{i-1} \tag{6}$$

Combining Equations (5) and (6), we have:

$$r_i = \left|\frac{dh_i(a_i)}{da_i} \cdot r_{i-1}\right| \tag{7}$$

Letting $J_i(a_i) = \frac{dh_i(a_i)}{da_i}$, Equation (7) becomes:

$$r_i = |J_i(a_i) \cdot r_{i-1}| \tag{8}$$

Similarly, we can obtain:

$$r_{i-1} = |J_{i-1}(a_{i-1}) * r_{i-2}| = |J_{i-1}(a_{i-1}) * J_{i-2}(a_{i-2}) \ldots J_1(a_1) * r_0| \tag{9}$$

Therefore, r_i can be approximated as:

$$r_i = |J_i(a_i) \cdot J_{i-1}(a_{i-1}) \cdots J_1(a_1) \cdot r_0| = \left|\prod_{j=1}^{i} J_j(a_j) \cdot r_0\right| \tag{10}$$

Let $\|\cdot\|$ represent the Euclidean norm f the vector, then we can obtain that:

$$\|r_i\| = \left\|\prod_{j=1}^{i} J_j(a_j) \cdot r_0\right\| \tag{11}$$

Considering the sub-multiplicative nature of norms,

we obtain an upper-bound for $\|r_i\|$:

$$\|r_i\| \leq \|J_i(a_i)\| \cdot \|J_{i-1}(a_{i-1})\| \cdot \ldots \cdot \|J_1(a_1)\| \cdot \|r_0\| \tag{12}$$

Let $S_i = \prod_{j=1}^{i} \|J_i(a_i)\| \cdot \|r_0\|$, we obtain the upper bound of the output error for layer i as follows:

$$\|r_i\| \leq \prod_{j=1}^{i} \|J_i(a_i)\| \cdot \|r_0\| = S_i \tag{13}$$

In summary, the analysis presented in the above investigates the propagation of small perturbations in input instances through a deep neural network and their impact on output errors. Commencing from a small difference r_0 between closely related input instances x and x', this error is incrementally analyzed and accumulated through forward propagation at each layer of the network. By utilizing first-order Taylor expansion to approximate the behavior of each network layer and integrating the norm for each layer, we establish the relationship between the output error r_i of each layer and the error r_{i-1}

of the preceding layer. According to this theoretical analysis, if earlier layers experience significant perturbations or if the decision boundaries of two classes are close on an intermediate layer, even expanding the final decision boundary through subsequent learning may not prevent the original imprecise samples from projecting onto different classes at this intermediate layer. This can cause increasing divergence in class assignments as samples propagate through layers. Therefore, regularizing intermediate layers and enhancing their stability and learning capability, especially in the initial layers, is crucial. Based on this idea, we define the output error of the entire model as:

$$J = \sum_{i=1}^{n} S_i \tag{14}$$

3.3 Stability enhanced loss function

According to Equation (14), we have determined that the upper bound on the model's output error in all of the layers is J. Thus, J can be considered the stability loss l_{stab}. By substituting J into Equation (4) and letting D denote the training set, the overall loss (L) which accounts for both accuracy and stability, can be expressed as:

$$L = \sum_{x \in D} \alpha \, l_{acc}\big(y, f_\theta(x)\big) + \beta \, J_{\theta,x} \tag{15}$$

where α and β are weights that balance the importance of accuracy and stability in the loss function. l_{acc} is the traditional loss function which measures the difference between the model output $f_\theta(x)$ and the true output y. Equation (15) could be transformed into:

$$L = \sum_{x \in D} \alpha\big(l_{\text{acc}}\big(y, f_\theta(x)\big) + \gamma \, J_{\theta,x}\big) \tag{16}$$

where $\gamma = \frac{\beta}{\alpha}$. Since α is a scaling factor, we can absorb it into the overall learning process. For simplicity, we often normalize it to 1. In this way, our stability enhanced loss (SELoss) function is defined as follows:

$$L = \sum_{x \in D} l_{\text{acc}}\big(y, f_\theta(x)\big) + \gamma \, J_{\theta,x} \tag{17}$$

In the Equation (17), the first term represents the traditional loss function, which is used to control the accuracy of the model. The second term is designed to manage the stability of the model. The parameter γ controls the relative importance of stability loss compared to the accuracy loss.

To solve the loss function defined in Equation (17), we need to compute the value of J. Recall that $J = \sum_{i=1}^{n} S_i = \sum_{i=1}^{n} \prod_{j=1}^{i} \|J_i(a_i)\| \cdot \|r_0\|$. Therefore, we need to calculate two components. The first component is $J_i(a_i)$, and the second component is r_0. For the first component, recall that $J_i(a_i) = \frac{dh_i(a_i)}{da_i}$, which is the Jacobian matrix of the i-th layer of f. Here we leverage the parallel processing function such as vmap to speed up computations of the Jacobian matrix for batched inputs.

To determine the value of r_0, we set it proportionally to the input data dimensions as follows: $r_0 = \text{noi} \cdot (\delta \odot x)$. Here noi represents a scalar noise factor, δ is a tensor of the same shape as x with elements -1 or 1, and $\odot$ denotes element-wise multiplication. This method allows the perturbation magnitude to vary with the dimensions of the input data, providing a more realistic simulation of imprecision in practical scenarios.

3.4 Multi-stage Learning

After calculating the values of $J_i(a_i)$ and r_0, we obtain the total loss. However, during model training, we observe that due to $S_i = S_{i-1} * \|J_i(a_i)\|$, when $\|J_i(a_i)\|$ is greater than 1, S_n tends to dominate the entire J, which limits the effectiveness of constraining each layer individually. To address this, we designed a multi-stage learning mechanism. First, we divide the layers in the DNN into different groups in sequence. For example, Group 1 includes the input layer and the first layer. While Group 2 includes the subsequent layers starting from the third layer. In the initial training stage, we train Group 1 using SELoss as defined in Equation 17. After training Group 1, we obtain and save the parameter values of these layers. These saved parameters are then used as initialization values for training Group 2. We then proceed to train Group 2 in the following stage. The detailed training process is illustrated in Algorithm1.

In Algorithm 1, steps 1 and 2 involve initialization, including dividing the network layers into groups and initializing parameters. Starting from step 3, multi-stage training is conducted. In each stage, a portion of the layers is trained using SELoss, allowing SELoss to more effectively regulate each layer of the network. This strategy helps reduce the impact of imprecision from the initial layers, leading to more stable final prediction results.

Algorithm 1： Multi-stage SELoss Training

Input: Initial model f_0, training dataset D
Output: model f

1： $glist$=initialize_group(f_0)；
2： Initialize para_0
3： for stage i=1 to size of($glist$)
4： initialize f_i with [$glist$[1:i], params_i−1];
5: params_i=train(f_i,D, $SELoss$);
6: end for
7: Return the final model as f

4 Evaluations

In this section, we demonstrate the effectiveness of the proposed SELoss method from the following aspects: a) Comparison with other methods that enhance model stability; b) Testing how the stability of the model trained using the new loss function varies with different levels of input perturbations; c) Comparing the changes in stability across different layers of the model trained with and without SELoss function; d) Conducting an experimental analysis of the training time for the model using the new loss function.

4.1 Datasets

In this study, we used three medical datasets for evaluation: a) MIMIC-III (Medical Information Mart for Intensive Care) dataset: This dataset, provided by Johnson et al. [17], includes data on 53,423 adult ICU patients (2001-2012) and 7,870 neonates (2001-2008) from a large tertiary hospital. It covers vital signs, medications, lab results, physician orders, and more. Following Harutyunyan et al. [18], we built a model to predict in-hospital mortality using the first 48 hours of ICU data. b) Diabetes dataset [19]: This dataset helps researchers analyze and predict the risk of diabetes. It includes eight features: age, diabetes pedigree function, body mass index, insulin level, skin thickness, blood pressure, glucose concentration, and number of pregnancies, with a binary label indicating whether the individual has diabetes. The dataset contains indicator data for 2768 patients. c) Blood_samples dataset [20]: This dataset includes 2837 patients. Each patient has 24 features for predicting whether an individual has specific diseases or other health conditions. It classifies patients into six categories: diabetes, thrombocytopenia, anemia, heart disease, thalassemia and healthy.

4.2 Instability Metrics

To evaluate accuracy, well-established metrics such as Precision, Recall, AUROC, and Accuracy are used to assess prediction accuracy. However, when it comes to evaluating stability, there is no standard metric for evaluating stability.

We adopt the idea in [13] to assess model instability. For a given input sample, if the model is stable, there will be no inconsistent predictions, that is, all labels in data imprecision space are the same. Let P_{major} to represent the proportion of samples with the majority label in the imprecision space. It can be seen that the larger P_{major} is, the more stable the model is. Let (K) denotes the number of classes, the instability measure $s(x)$for given sample x is obtained as:

$$s(x) = -\log_K\left(P_{\text{major}}\right) \tag{18}$$

For a model, more samples with lower stability $s(x)$ indicats a less stable model. The values of $s(x)$ are divided into fixed-size intervals from low to high, and the number of samples in each interval forms a distribution. Clearly, the stability of different models can be measured by the variation in the distribution. The overall instability measure $Instability$ of the model based on the distribution is defined as:

$$Instability = \frac{1}{T}\sum_{i=0}^{M} s_{ix} \cdot n_m \tag{19}$$

where m is the number of intervals. s_{ix} is the lower bound of i −th interval, n_m is the number of samples within that interval. T is the number of all samples. The value $Instability$ as well as the distribution analysis, enables a more detailed observation, understanding, and comparison of a model's predictive stability at a finer granularity.

4.3 Experimental Setting

For the MIMIC dataset, we constructed MLP and LSTM deep learning models to perform the prediction tasks. For the other two datasets, MLP models were used. The base MLP model is implemented with five layers, uses ReLU as the activation function, and adopts the cross-entropy loss function.

In the proposed new loss method, there is a parameter γ that balances accuracy and stability. For the MIMIC-III dataset, γ is set to 5e-3; for the blood_samples dataset, it is set to 2e-7; and for the diabetes dataset, it is set to 1e-5.

We compare the proposed method with IRloss [3]. The implementation of the IRloss model specifically considers two key parameters: the width of the uncertainty range Δ and the discretization parameter (s). These parameters are fine-tuned in the different datasets. On the MIMIC-III dataset, Δ and s are set to 0.1 and 0.02, respectively. In the diabetes and blood_samples datasets, a more refined setting of 0.01 for Δ and 0.001 for s.

We also compared SELoss with the randomized smoothing method [15], which involves three key parameters: the standard deviation σ of Gaussian noise, the number of samples n, and the confidence level α. For all three datasets, n and α were set to 100 and 0.01, respectively σ was set to 0.02 for the blood_samples dataset and the diabetes dataset, and to 0.05 for the MIMIC-III dataset. For brevity, we abbreviate 'randomized smoothing' as 'RS'.

Our software environment contains Ubuntu 20.04, PyTorch v1.13.0, and python 3.9.15. All experiments were conducted on a machine equipped with two GPUs (NVIDIA GeForce GTX 4090) and 64GB of memory with a batch size of 500 for MIMIC-III dataset, and a batch size of 100 for another two datasets. For the learning rate, we set the learning rate to 1e-3 for both the MIMIC-III and diabetes datasets, and to 1e-2 for the blood samples dataset. In the multi-stage training process, each of the three models is divided into two groups. The first group contains the first layer, and the second group includes the remaining layers. Training is conducted in two stages. Each experiment was repeated three times, and the average value was taken to ensure the robustness of the results.

4.4 Experimental findings

We now present four key findings in the following.

SELoss Provides more Accurate and Stable Predictions across Various Tasks We compare our proposed SELoss and SELoss with multi-stage training strategy ($SELoss_m$) methods with the base model and several previous approaches known to enhance model robustness or stability. These include IRloss [3], PGD adversarial training [10], and randomized smoothing [15].

The comparison results based on three datasets are illustrated in Table 1. From Table 1, we can observe that traditional methods designed to improve model robustness such as PGD adversarial training and randomized smoothing may lead to a decrease in model

accuracy. For instance, in the diabetes prediction task, the base method achieves an Accuracy of 0.9175, which decreases to 0.8975, 0.9125, and 0.915 with PGD, IRloss, and randomized smoothing methods respectively. For PGD and IRloss method, the AUROC declines from 0.9180 to 0.9070, 0.8978 respectively. However, the SELoss method proposed in this paper improves both model instability and all accuracy metrics. Especially, the $SELoss_m$ achieves the best performance. In Table 1, the randomized smoothing method does not have an AUROC value. This is because calculating AUROC requires the model to output the probability distribution for each class, whereas randomized smoothing directly returns the most frequently predicted class as the final result after running the base model multiple times. In the blood_samples dataset, the SELoss method decreases instability from 0.1077 to 0.0682 and increases Accuracy from 0.9027 to 0.9329. The $SELoss_m$ method furthered decreases instability to 0.0436. Similarly, in the diabetes dataset, the $SELoss_m$ method lowers instability from 0.0059 to 0.0016 and boosts Accuracy from 0.9175 to 0.9375. Precision, Recall, and AUROC have not decreased, and even Precision has shown a noticeable improvement. In the MIMIC-III dataset, $SELoss_m$ method reduces the instability of the base model from 0.0104 to 0.0038. In terms of the accuracy evaluation metric, although Recall has shown a certain degree of decline, Precision, Accuracy, and AUROC have slightly improved compared to the baseline model. It can be seen from the table that the auroc and accuracy of the improved model have not declined, and its instability is lower.

Table 1. Comparison of MLP-based methods on different datasets.

dataset	method	instability	accuracy	precision	recall	AUROC
blood_sample	Base	0.1077	0.9027	0.7575	0.7661	0.9180
	PGD	0.0679	0.9094	0.7525	0.7673	0.9070
	RS	0.0809	0.9121	0.7560	0.7786	--
	IRloss	0.0877	0.8859	0.7062	0.7678	0.8978
	SELoss	0.0682	0.9329	0.7906	0.7832	0.9207
	$SELoss_m$	0.0436	0.9295	0.7773	0.7849	0.9189
diabetes	Base	0.0059	0.9175	0.9052	0.9233	0.9595
	PGD	0.0002	0.8975	0.8845	0.9078	0.9539
	RS	0.0059	0.9150	0.9034	0.9167	--
	IRloss	0.0042	0.9125	0.9005	0.9148	0.9465
	SELoss	0.0020	0.9350	0.9272	0.9323	0.9569
	$SELoss_m$	0.0016	0.9375	0.9268	0.9420	0.9647
MIMIC III	Base	0.0104	0.8878	0.724	0.6751	0.8399
	PGD	0.0052	0.8891	0.7246	0.6595	0.8466
	RS	0.0088	0.8841	0.7109	0.6910	--
	IRloss	0.0070	0.8804	0.6908	0.6178	0.7944
	SELoss	0.0061	0.8974	0.7574	0.6595	0.8437
	$SELoss_m$	0.0014	0.8977	0.7705	0.6295	0.8549

In addition to MLPs, our method can be generalized to other types of deep neural networks. Table 2 presents the experimental results comparing the traditional LSTM method with the enhanced version using SELoss on the MIMIC dataset. As shown in the table, the accuracy and AUROC of the improved method remain largely unchanged, while precision and recall have seen modest improvements, and instability has been significantly reduced. Furthermore, both Table 1 and Table 2 demonstrate that our method consistently performs well across various deep neural networks and multiple datasets, highlighting its robustness and generalizability.

Table 2. Comparison of LSTM-based methods on MIMIC III.

method	instability	accuracy	precision	recall	AUROC
LSTM	0.0127	0.8921	0.7418	0.6171	0.8441
SELoss	0.0063	0.8967	0.7649	0.6291	0.8403

As Input Perturbation Increases, SELoss Slows Down the Rise in Output Instability To thoroughly analyze the stability of the models, we designed a rigorous perturbation test scheme for the base model and SELoss models. Specifically, we evenly divide the input perturbation range from 0.02 to 0.2 into 10 levels. Through this series of disturbance experiments, we not only quantified the instability and accuracy of the model under different disturbance intensities, but also investigated the trend of instability change of each model with increasing disturbance amplitude, so as to comprehensively evaluate and compare the performance of the model in the face of data imprecise.

Fig.2 compares the performance of the baseline model and the model using the SELoss method under gradually increasing noise factors. As depicted, with higher noise factors, all models show increased instability. However, the model employing the SELoss methods consistently exhibits lower instability and a slower growth rate compared to the baseline model.

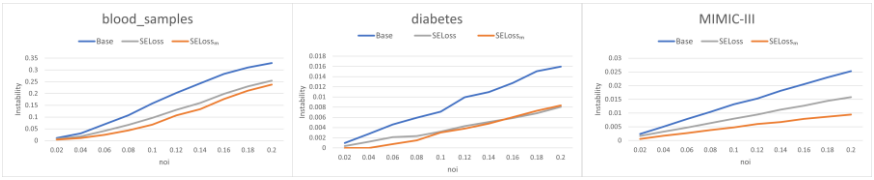

Fig. 2. Comparison of SELoss method performance with base method as noise level increases: x-axis represents the noise level (noi), while the y-axis represents the magnitude of model instability.

SELoss reduces the instability of each layer The proposed method reduces prediction instability by constraining the layer-by-layer propagation of input perturbations. To verify the effectiveness of the new loss in suppressing propagation, we present the r_i values of the predictive model for each layer across three datasets, with and without using the SELoss function in Fig.3. According to the figure, it is evident that for the base model and the model using SELoss training, r_i increases as the depth of layers increases. This is reasonable because the perturbations accumulate as the depth of the layer increases. With the SELoss, the r_i values for each

layer are smaller than those in the base model, which is particularly pronounced in the final layer. Notably, on the MIMIC-III dataset, the percentage reduction in r_i is the greatest, with an average reduction of 150% across all layers. Based on the results of three datasets, SELoss mainly reduces the value of r_i at the final layer. Compared to SELoss, $SELoss_m$ reduces the value of r_i at each layer. This indicates that the multi-stage learning strategy effectively reduces instability at each layer, starting from the initial layers, thereby mitigating the propagation of imprecision. As a result, the improvement in prediction stability is significant.

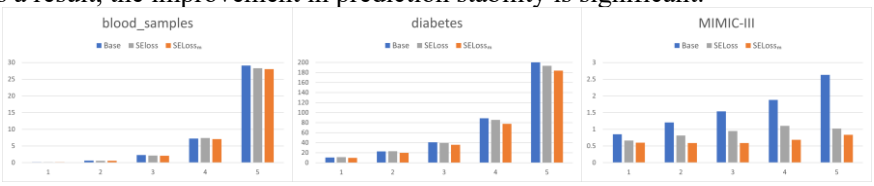

Fig. 3. Error r_i of each layer: the x-axis represents each layer of the model, and the y-axis represents r_i.

SELoss increases training time, but the increase is limited for low dimensions Due to the requirement of calculating $J_i(a_i)$ in each layer, our method incurs additional training time. Therefore, we analyzed the time overhead of the algorithm in this experiment. We first provide the training time comparison between the base model and the model with SELoss across three datasets. Since the convergence rounds of the models may be different on different datasets. In order to see the effect of the number of nodes of the models and the size of the model on the time, we uniformly set the number of training epochs to 100 for the time comparison for every model, which reduces the effect of the training time caused by the different number of epochs. Table 3 illustrates the comparison results.

Table 3. Training and testing times for different methods.

dataset	method	Train(s)	Test(s)
blood_samples	base	19.31	0.0019
	SELoss	24.24	0.0021
	Randomized Smoothing	19.41	7.1500
diabetes	base	14.35	0.0024
	SELoss	16.77	0.0024
	Randomized Smoothing	14.38	4.6267
MIMIC III	base	29.82	0.0186
	SELoss	266.58	1.4295
	Randomized Smoothing	29.65	37.3910

From the Table 3, we can see that on blood_samples and diabetes datasets, the training time increase is not significant. On the MIMIC-III dataset, the time increase is relatively larger, mainly because the input dimension of the MIMIC dataset is much higher. We used the same way as previous work to process the input data, which has 706 dimensions. Accordingly, the number of nodes in the hidden layer is higher, with

500 nodes. While on blood_samples and diabetes datasets, the number of nodes is 128 and 64. It can be seen that the method proposed in this paper adds limited time when the input dimension and the number of hidden layer nodes are within a certain range. We also compared the time overhead with the current effective method of randomized smoothing. The time for one sample inference is also provided in Table 3. From the table, we can see that different from the proposed method, randomized smoothing increases the inference time, since it requires sampling each sample within its neighborhood, resulting in much greater time overhead across all three datasets. As the input dimension increases, the number of samples in the neighborhood increases significantly, leading to greater time consumption. So, the time cost increases are particularly notable on the MIMIC dataset.

4.5 Parameter sensitivity experiment

This section explores the relationship between the parameter γ and both instability and accuracy, as well as the trend of their changes. Specifically, we analyze how variations in the γ value influence the stability of the model and its predictive performance. Through this analysis, the goal is to identify optimal settings for γ that balance the minimization of instability and the maximization of accuracy, providing valuable insights into the role of this parameter in model tuning and its impact on overall performance.

As shown in the Fig.4, increasing γ leads to a reduction in instability; however, when γ becomes too large, accuracy also significantly declines. Therefore, an appropriate γ value is crucial for achieving a balance between accuracy and instability. It can be observed that the optimal γ varies considerably across different datasets and prediction tasks.

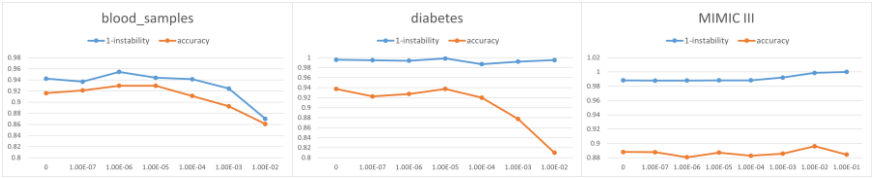

Fig. 4. The variation of instability and accuracy with different parameter settings.

5 Related Work

Deep neural networks are known for their sensitivity to input variations. Previous methods to mitigate input sensitivity can be divided into two data augmentation and model augmentation methods.

Data augmentation methods enhance the training dataset by introducing variations or adversarial samples into training process [10], thereby helping the model generalize better. Much of the research in this area focuses on generating this kind of samples. FGSM [9,][21] or PGD [10] employed min-max optimization to generate adversarial samples. In healthcare field, adversarial patients are defined and examined in [22].

Different methods [16] were provided to generate adversarial examples in medical deep learning classifiers. Although numerous adversarial example generation methods have been developed, they are known to hurt test accuracy on many datasets [11,12]. Both FGSM and PGD methods have observed a decrease in accuracy despite improving the robustness of the model.

Model augmented methods modify the model itself, by designing special neurons or learning mechanism to improve stability and robustness. Among them, certifiable models design classifiers whose prediction at any point is verifiably constant within some set around the point [23]. As well-known that the robustness of a neural network is closely related to the lipschitz continuity, Lipschitz ReLU Networks [24] use ReLU activations but enforce lipschitz continuity through constraints on the weights. Zhang et al. [25] studied the expressive power of lipschitz neural networks and then develop the SorNet by leveraging order statistic functions. Despite these advancements, neural networks that satisfy the certified properties often have limited expressive power and generalizability. Randomized Smoothing [15] is another certified method that starts with a standard classifier and then creates a smoothed classifier, which determines the final class prediction based on a majority vote among predictions made on noisy inputs. However, stochastic smoothing methods have high computational costs in prediction process and are primarily suitable for perturbations with l_2 norm.

It has been demonstrated that robustness often comes at a cost [26]. Specifically, training robust models can be more resource-intensive and may result in decreased standard accuracy. A growing body of research explores the relationship between accuracy and robustness. Some studies [12] suggest that why robustness may lead to lower accuracy is that different classes are very close together or they may even overlap. In [14], the key role of the decision boundary in the robustness of the model is emphasized, and the location and characteristics of the decision boundary are crucial to the anti-perturbation ability of the model. The TRADES training [8] is proposed to improve the robustness of the model by pushing away the decision boundary. DeepDIG [27] framework generates samples close to the decision boundary, which verifies the idea that data points near the decision boundary are vulnerable to attacks. Recent work [28] suggests that smoothing the sensitivity of features and the decision boundary can enhance robustness. However, it is easy to see that output perturbations are closely related to the learning ability of the intermediate layers. Therefore, regularizing these layers and enhancing their stability and learning capability, especially in the initial layers, is crucial.

6 Conclusions

In this paper, we focus on improving prediction stability without generating additional training samples nor compromising accuracy. The key idea is to reformulate the learning problem as a multi-objective optimization task with a new stability-enhanced loss function. With a multi-step training strategy to control instability in each layer, the experimental evaluation on three real-world datasets shows that the new method provides both accurate and stable predictions across various tasks. However, the stability problem, clearly very relevant and important to many applications including healthcare, is

far from solved. One cluster of problems concerns extension of the multi-objective optimization approach to other learning methods and models. In particular, it isn't clear if the SELoss function needs to be further adjusted, nor how the learning algorithms should be adapted. Another group of questions concerns the stability metric itself; questions such as how effective and accurate in its assessment on arbitrary models, and how efficient to determine stability deserve further exploration.

Disclosure of Interests. All authors disclosed no relevant relationships.

References

1. Priyanka R, Pooja M R. A Deep Learning Survey on Diseases Prediction and Detection in Health Care[C]//International Conference on Intelligent Systems in Computing and Communication. Cham: Springer Nature Switzerland, 2023: 119-127.
2. Xie S, Yu Z, Lv Z. Multi-disease prediction based on deep learning: a survey[J]. Computer Modeling in Engineering & Sciences, 2021, 128(2): 489-522.
3. Wang M, Lin Z, Li R, et al. Predicting disease progress with imprecise lab test results[J]. Artificial Intelligence in Medicine, 2022, 132: 102373.
4. Wang M, Su J, Lu H. Impact of medical data imprecision on learning results[J]. arXiv preprint arXiv:2007.12375, 2020.
5. American Clinical Laboratory Association. 2014. Value of Clinical Labs: The Power of Knowing. https://www.acla.com/the-power-of-knowing/
6. Zhang S, Wang W, Zhao H, et al. Status of internal quality control for thyroid hormones immunoassays from 2011 to 2016 in China[J]. Journal of clinical laboratory analysis, 2018, 32(1): e22154.
7. Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks[J]. arXiv preprint arXiv:1312.6199, 2013.
8. Zhang H, Yu Y, Jiao J, et al. Theoretically principled trade-off between robustness and accuracy[C]//International conference on machine learning. PMLR, 2019: 7472-7482
9. Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[J]. arXiv preprint arXiv:1412.6572, 2014.
10. Mądry A, Makelov A, Schmidt L, et al. Towards deep learning models resistant to adversarial attacks[J]. stat, 2017, 1050(9).
11. Raghunathan A, Xie S M, Yang F, et al. Adversarial training can hurt generalization[J]. arXiv preprint arXiv:1906.06032, 2019.
12. Yang Y Y, Rashtchian C, Zhang H, et al. A closer look at accuracy vs. robustness[J]. Advances in neural information processing systems, 2020, 33: 8588-8601.
13. Li Y, Wang M, Su J. An Evaluation Metric for Prediction Stability with Imprecise Data[C]//International Conference on Knowledge Science, Engineering and Management. Cham: Springer Nature Switzerland, 2023: 430-441.
14. Wang Y, Zou D, Yi J, et al. Improving adversarial robustness requires revisiting misclassified examples[C]//International conference on learning representations. 2019.
15. Cohen J, Rosenfeld E, Kolter Z. Certified adversarial robustness via randomized smoothing[C]//international conference on machine learning. PMLR, 2019: 1310-1320.
16. Finlayson S G, Chung H W, Kohane I S, et al. Adversarial attacks against medical deep learning systems[J]. arXiv preprint arXiv:1804.05296, 2018.
17. Johnson A E W, Pollard T J, Shen L, et al. MIMIC-III, a freely accessible critical care database[J]. Scientific data, 2016, 3(1): 1-9.

18. Harutyunyan H, Khachatrian H, Kale D C, et al. Multitask learning and benchmarking with clinical time series data[J]. Scientific data, 2019, 6(1): 96.
19. Diabetes dataset. https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset
20. Blooddataset.https://www.kaggle.com/datasets/ehababoelnaga/multiple-disease-prediction
21. Zhang Y, Zhang G, Khanduri P, et al. Revisiting and advancing fast adversarial training through the lens of bi-level optimization[C]//International Conference on Machine Learning. PMLR, 2022: 26693-26712.
22. Papangelou K, Sechidis K, Weatherall J, et al. Toward an understanding of adversarial examples in clinical trials[C]//Joint European conference on machine learning and knowledge discovery in databases. Cham: Springer International Publishing, 2018: 35-51.
23. Wong E, Kolter Z. Provable defenses against adversarial examples via the convex outer adversarial polytope[C]//International conference on machine learning. PMLR, 2018: 5286-5295.
24. Tsuzuku Y, Sato I, Sugiyama M. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks[J]. Advances in neural information processing systems, 2018, 31.
25. Zhang B, Jiang D, He D, et al. Rethinking lipschitz neural networks and certified robustness: A boolean function perspective[J]. Advances in neural information processing systems, 2022, 35: 19398-19413.
26. Nakkiran P. Adversarial robustness may be at odds with simplicity[J]. arXiv preprint arXiv:1901.00532, 2019.
27. Karimi H, Tang J. Decision boundary of deep neural networks: Challenges and opportunities[C]//Proceedings of the 13th International Conference on Web Search and Data Mining. 2020: 919-920.
28. Zhang J, Liu F, Zhou D, et al. Improving accuracy-robustness trade-off via pixel reweighted adversarial training[J]. arXiv preprint arXiv:2406.00685, 2024.