AnyLoss: Transforming Classification Metrics into Loss Functions

Doheon Han
University of Notre Dame
Department of Computer Science & Engineering and Lucy Family Institute for Data & Society
Notre Dame, Indiana, USA
dhan6@nd.edu

Nuno Moniz
University of Notre Dame
Lucy Family Institute for Data & Society
Notre Dame, Indiana, USA
nmoniz2@nd.edu

Nitesh V. Chawla
University of Notre Dame
Department of Computer Science & Engineering and Lucy Family Institute for Data & Society
Notre Dame, Indiana, USA
nchawla@nd.edu

ABSTRACT

Many evaluation metrics can be used to assess the performance of models in binary classification tasks. However, most of them are derived from a confusion matrix in a non-differentiable form, making it very difficult to generate a differentiable loss function that could directly optimize them. The lack of solutions to bridge this challenge not only hinders our ability to solve difficult tasks, such as imbalanced learning, but also requires the deployment of computationally expensive hyperparameter search processes in model selection. In this paper, we propose a general-purpose approach that transforms any confusion matrix-based metric into a loss function, *AnyLoss*, that is available in optimization processes. To this end, we use an approximation function to make a confusion matrix represented in a differentiable form, and this approach enables any confusion matrix-based metric to be directly used as a loss function. The mechanism of the approximation function is provided to ensure its operability and the differentiability of our loss functions is proved by suggesting their derivatives. We conduct extensive experiments under diverse neural networks with many datasets, and we demonstrate their general availability to target any confusion matrix-based metrics. Our method, especially, shows outstanding achievements in dealing with imbalanced datasets, and its competitive learning speed, compared to multiple baseline models, underscores its efficiency.

CCS CONCEPTS

• **Computing methodologies** → *Neural networks.*

KEYWORDS

Neural Network; Binary Classification; Loss Function; Evaluation Metrics

ACM Reference Format:
Doheon Han, Nuno Moniz, and Nitesh V. Chawla. 2024. AnyLoss: Transforming Classification Metrics into Loss Functions. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3672017

KDD '24, August 25–29, 2024, Barcelona, Spain
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3672017

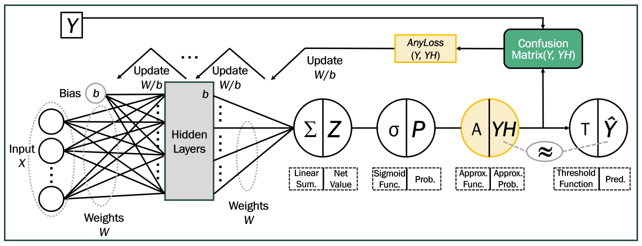

Figure 1: Our method in the multi-layer perceptron (MLP) structure. Input X and weights W generate the net value Z, and the sigmoid function σ transform the net value into the class probability P. The approximation function A generates YH by amplifying the probability P. The confusion matrix is constructed in a differentiable form by the ground truth Y and the YH. Consequently, our loss function, *AnyLoss*, aimed at any confusion matrix-based metric, is available in a differentiable form.

1 INTRODUCTION

Model evaluation is a vital aspect of machine learning, and selecting an appropriate evaluation metric is a challenging step due to the availability of multiple metrics [26]. This selection process can be considered a goal in model development since it is closely related to the task objectives defined by users and the characteristics of the data domains. Most of them, such as accuracy or F_β scores, are derived from a confusion matrix [20], obtained from operations over discrete values, i.e., labels. Such labels are generated from a threshold function that transforms continuous values, i.e., class probabilities, into discrete values, meaning that the confusion matrix is in a non-differentiable form. Importantly, this implies that confusion matrix-based metrics cannot be used as a goal, i.e., a loss function, in model development, even though the ultimate goal of the learning is to achieve a better score for the selected metric.

Mean Squared Error (MSE), L2 loss [6], and Binary Cross Entropy (BCE) are generally used as loss functions in neural networks [13]. However, they cannot directly target the desired metric, although indirectly aiming at a higher accuracy, i.e., a lower error rate. Therefore, several strategies have been suggested to meet the unique needs of classification tasks. The thresholding strategy finds the optimal threshold value to get a desired score of a chosen evaluation metric [21, 30], the data pre-processing strategy deals with raw data issues such as inconsistencies, imbalance, etc. [3], and the surrogate loss function strategy provides a new loss function that aimed at the evaluation metric scores [2, 5, 11, 19, 22]. However, the thresholding strategies have some issues like the precision-recall trade-off [28, 30], and the data pre-processing, such as data resampling, has some issues like overfitting or data losing [8, 29]. The surrogate loss function strategy that is comparatively free from the aforementioned issues has been studied, and they are discussed more in the related work section.

In this paper, we introduce a general-purpose method to generate a surrogate loss function, *AnyLoss*, directly aimed at any desired confusion matrix-based evaluation metric. Our method contains an approximation function that amplifies the class probability to build a confusion matrix in a differentiable form. This allows the estimated metric scores to be calculated before updating weights. In a neural network for binary classification, class probability is generated from the sigmoid function after the output layer, i.e., the probability of being classified as a positive case. The approximation function amplifies the probability to close to 0 or 1 to be regarded as the predicted label. The approximated probability can be used with the ground truth label to build a confusion matrix, and the evaluation metrics calculated from the matrix are represented in a differentiable form. The metric scores from the confusion matrix are similar to those of an actual confusion matrix when the approximated probability is close enough to the predicted label. Consequently, our method can use any confusion matrix-based evaluation metric as a loss function to optimize learning processes.

To show the operability of our method in neural networks, we provide mathematical and experimental analysis for the approximation function and prove the differentiability of loss functions by suggesting their derivatives. Extensive experiments using several datasets are conducted to assess the availability of our method in single- and multi-layer perceptron structures. The performance of the experiments shows our method's ability for generalization, indicating that our method can indeed be used with any confusion matrix-based evaluation metric. Our method, especially, shows considerable performance with imbalanced datasets since our method is available for any desired metric. In addition, our method's learning speed is competitive with the baseline models, as demonstrated by experimental results.

Section 2 introduces related works, and Section 3 explains our method in detail, including the form of the approximation function and the derivatives of the selected loss functions. In Section 4, we demonstrate the effectiveness of our method through experimental results using various datasets. Section 5 summarizes the strengths and limitations of our method and concludes this paper.

2 RELATED WORKS

There have been approaches to optimize the evaluation metric scores in classification by adopting surrogate loss functions. Some introduce a method available for a certain metric [2, 5, 19]; in this case, a unique surrogate loss is suggested for the targeted evaluation metric. Others provide methods that can generally be used for any evaluation metric [11, 22], and ours falls into this category. Regardless of strategies, methods such as approximating the gradient of the score, assuming the score in a stochastic setting, or constructing a temporary confusion matrix have mostly been considered.

2.1 Specific Metric

Most works targeting a specific metric have been pursued optimizing the F_β score. Lee at el. [19] propose a surrogate loss for the F_β score inspired by the BCE gradient conditions and the mean absolute error loss. It is derived from the derivative condition of the F_β score by approximating its gradient path during learning, and its form is similar to the BCE loss. Busa-Fekete et al. [5] introduce a method to optimize F_1 score in an online learning environment, and the method can maximize the score on the population level with the partially observed data when the i.i.d. setting is assumed. They consider the problem in a stochastic setting and represent the F_1 score as a function of a variable threshold. Benedict et al. [2] introduce a surrogate loss for the F_1 score in multi-label classification. They applied the method to their study's text and image datasets, but it is also available for general classification tasks. They suggest a surrogate loss function for the F_1 score using the smoothed probability obtained from the sigmoid function. The works are uniquely designed for each desired metric, so the approach's availability is limited in the area targeting a specific metric. The next shows more general work available for most evaluation metrics.

2.2 General Metrics

Huang et al. [11] suggest the 'MetricOpt' method, which learns the approximate metric gradient for a given surrogate loss function. This method was devised for computer vision tasks, such as image classification, but also applies to non-image data. It shows improved results in diverse metrics such as the miss-classification rate, Jaccard index, and F_1 score but slower than standard loss optimization since it includes finetuning steps. This method is generally more intricate because it operates through additional processes, such as meta-learning a value function and pre-training a main model. Marchetti et al. [22] present 'Score-Oriented Loss' functions for binary classification tasks. They try probabilistic methodology, which sets the threshold used for prediction to a continuous random variable. Based on the assumption, a probabilistic confusion matrix is constructed, and any desired metric can be derived from it. This method is similar to ours in constructing a confusion matrix to generate any desired evaluation metric. However, it has additional processes, such as selecting parameters: the probability distribution, the uniform or cosine-raised distribution, and two parameters μ and δ have to be chosen for the threshold. This method performs well with their optimized settings compared to the Cross-Entropy and the Kullback-Leibler divergence. This method has a more complicated process than ours, which can increase learning time. The appropriate distribution and corresponding parameters must be found for each task since performance depends on them. Compared to our simpler approach, this method has a slower learning speed and includes a relatively time-consuming process of deciding on a setting for every task, corroborated by experimental results.

3 METHOD

Our method in a multi-layer perception (MLP) structure is shown in Fig. 1. Input data $\mathbf{X} = \{\mathbf{x}_i | \mathbf{x}_i = \{x_{ij} | x_{ij} \in \mathbb{R}, j = 1, \cdots, m\}, i = 1, \cdots, n\}$ with the number of samples n and the number of features m, and the weights $\mathbf{W} = \{w_j | w_j \in \mathbb{R}, j = 0, \cdots, m\}$ generate the net value $\mathbf{Z} = \{z_i | z_i = w_0 + \sum_{j=1}^{m} x_{ij} \cdot w_j, i = 1, \cdots, n\}$. The class probability $\mathbf{P} = \{p_i | p_i = \sigma(z_i),\ i = 1, \cdots, n\}$ is generated from the sigmoid function σ. The approximated probability $\mathbf{YH} = \{yh_i | yh_i = A(p_i),\ i = 1, \cdots, n\}$ is calculated from the approximation function A. Each value of **YH** is very close to 0 or 1 so that it can be considered as the predicted labels $\widehat{\mathbf{Y}} = \{\widehat{y}_i | \widehat{y}_i = T(p_i),\ i = 1, \cdots, n\}$. The **YH** and the ground truths $\mathbf{Y} = \{y_i | y_i \in \{0, 1\},\ i = 1, \cdots, n\}$ can build a confusion matrix, and any evaluation metrics can be generated. When a desired evaluation metric is set as a loss function, *AnyLoss*, the weights will be updated to optimize the loss function.

3.1 Approximation

The role of the approximation function is, in brief, **"To make the *A(p_i)* close to 0 or 1 but not converged to exact 0 or 1 with the given p_i"**. The mathematical form of the approximation function is shown as (1). The amplifying scale L is a positive real number, and p_i is the given class probability after the sigmoid function. For its operation, the function *A(p_i)* has to meet the two conditions, which decide the amplifying scale L. The two conditions are given in mathematical forms with examples, followed by the analysis for the proper range of the amplifying scale L.

$$A(p_i) = \frac{1}{1 + e^{-L(p_i - 0.5)}} \tag{1}$$

1st Condition: Amplifier. The approximation function should be able to make the *A(p_i)* closer to 1, indicating a positive label, if the given p_i is closer to 1 than 0, and vice versa. This process amplifies the class probability p_i so that the approximated probability *A(p_i)* can be regarded as actual labels. The mathematical definition of the first condition is as (2).

$$|A(p_i) - 0.5| \geq |p_i - 0.5| \quad OR \quad \begin{cases} A(p_i) \geq p_i, & p_i \geq 0.5 \\ A(p_i) \leq p_i, & p_i \leq 0.5 \end{cases} \tag{2}$$

If the class probability p_i is larger than 0.5, the *A(p_i)* should be larger than the p_i and close to 1. For instance, if p_i is 0.7, the approximation function should generate the *A(p_i)* larger than 0.7 and close to 1 so that we can assume it as an actual label '1'. Reversely, if p_i is 0.2, the approximation function should generate the *A(p_i)* smaller than 0.2 and close to 0 so that we can assume it as an actual label '0'. However, the approximation function with a small value of L cannot amplify the p_i in some areas. Fig. 2 shows how a small L violates the condition. The X-axis and Y-axis represent each p_i and corresponding *A(p_i)*. On the left graph, the L is five, and the slope is too gentle, so the function does not work as an amplifier in some areas. For example, the *A(p_i)* is rather larger than the corresponding p_i on the point (0.1, 0.12) although p_i is smaller than 0.5. The approximation function should generate a value smaller than 0.1, but it does not. A larger L makes the function shape steeper. Therefore, this issue will be solved with a larger L. This shows that there exists a minimum value of L to ensure the operation of the approximation function.

2nd Condition: No 0/1. Each value of *A(p_i)* should be between 0 and 1 as described in (3).

$$0 < A(p_i) < 1 \tag{3}$$

If *A(p_i)* values become 0 or 1, of course, it would guarantee a more accurate calculation of evaluation metric scores since they are the same as the actual labels. However, it can cause a serious problem,

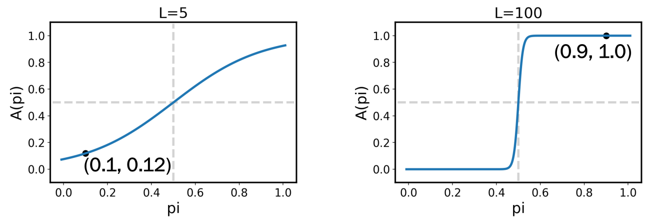

Figure 2: The approximation function *A(p_i)* with 2 different *L* values. On the left, a smaller *L*, the *A(p_i=0.1)* should be smaller than 0.1, but it generates 0.12. On the right, a larger *L*, the *A(p_i=0.9)* converges to 1.0.

'No More Update', because of the form of the partial derivative of *A(p_i)* as shown in (4). The function *A(p_i)* is part of a loss function because the loss function is made with the confusion matrix entries, and it includes ***YH*** which is produced by the function *A(p_i)*. So the partial derivative of the loss function includes the partial derivative of *A(p_i)*. The derivative contains both *A(p_i)* and *(1-A(p_i))* terms, so any case of *A(p_i)* = 0 or 1 will make the partial derivative 0. The right graph in Fig. 2 shows that the value of *A(p_i)* can be converged to 0 or 1. The L is 100, and the slope is steep. The point (0.9, 1.0) shows the *A(p_i=0.9)* value has converged to 1. Mathematically, the approximation function is not converged to exact 0 or 1, but it does in a machine [9]. Therefore, a smaller L should be considered in this case, which implies that a maximum value of L should exist to ensure the operation of the approximation function.

$$\frac{\partial A(p_i)}{\partial p_i} = L \cdot A(p_i) \cdot (1 - A(p_i)) \tag{4}$$

The valid *L*. The two conditions imply the existence of a suitable range for L. The first condition shows a minimum L, and the second shows a maximum L. Fig. 3 shows the four learning curves with four different L values. The X-axis and Y-axis represent each iteration number and loss value. When L is 1, the model is not learning well, and its loss is not small enough. But it gets a smaller loss when L is 5. When L is 10, the learning curve becomes steeper, and the final loss is smaller, indicating it learns faster and better with a larger L. However, when L is 100, the model abruptly stops learning because the 'No More Update' happens by violating the second condition.

More detailed experimental results are needed to determine the proper range of L. We have the two sure conditions in a mathematical form (2, 3) and the range of the input p_i is set to [0, 1] since it is the probability. The approximation function *A(p_i)* should be able to work with any input values when with the valid L. The possible p_i values are infinite because the probability is continuous, therefore we adopt the boundary value analysis [27] to cover all possible p_i values. So the two boundaries, i.e., the two extreme values of the given range, are chosen for the test. Even for the input values very close to 0 or 1, the approximation function should amplify them but not make them converge to exact 0 or 1.

Table 1 shows valid ranges of L with different accuracy levels. For instance, with the accuracy level of 0.001 in the third row, the two extreme values 0.001 and 0.999 will not violate the two conditions if L is between 13.85 and 73.62. As such, the approximation function works with any input values [0.001, 0.999] when L is in the

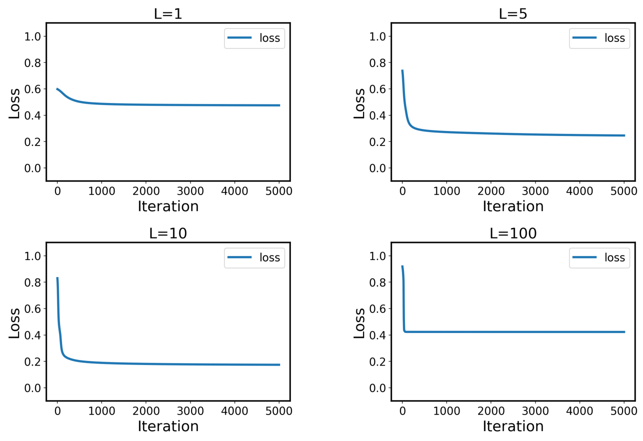

Figure 3: Learning curves with 4 different L values. A model is not learning with a small L but is better with a larger L. However, a model stops learning if a L is too large.

Table 1: The valid range of L. All values between the two p_i values are well approximated with the given L.

Accuracy Level t	Probability $p_i = t$, 1-t	Range of L MIN $\leq L \leq$ MAX
0.1	0.1, 0.9	$5.50 \leq L \leq 91.84$
0.01	0.01, 0.99	$9.38 \leq L \leq 74.97$
0.001	0.001, 0.999	$13.85 \leq L \leq 73.62$
...	...	...
1e-14	1e-14, 1-(1e-14)	$64.48 \leq L \leq 73.47$
1e-15	1e-15, 1-(1e-15)	$\mathbf{69.08} \leq L \leq \mathbf{73.47}$
1e-16	1e-16, 1-(1e-16)	$73.69 \leq L \leq 73.47$

given valid range. If L is selected 13.84, then it will violate the first condition, *Amplifier*, and if L is selected 73.63, then it will violate the second condition, *No 0/1*. The first and the second conditions regulate each the minimum and the maximum value of L. There is no valid range with an accuracy level of $1e^{-16}$, so the highest accuracy level is $1e^{-15}$, and we consider an integer within the range just for convenience. The candidates are 70, 71, 72, and 73, and we chose 73 since a larger L is learning faster and better according to the previous observation in Fig. 3.

3.2 AnyLoss

Loss functions in our method consist of the entries of a confusion matrix, which are True Negative (TN), False Negative (FN), False Positive (FP), and True Positive (TP). Diverse loss functions can be generated since they are in a differentiable form. Our loss function, *AnyLoss*, is defined as (5), where the *f(TN,FN,FP,TP)* is a function of an evaluation metric score represented with a confusion matrix entries. The score range is $[0, 1]$, and the form is *'1-score'* so that the score can be maximized. To demonstrate the general availability of our method in diverse metrics, accuracy, F$_\beta$ scores, geometric mean, and balanced accuracy [4] are chosen. The confusion matrix in a differentiable form and the derivatives of each loss function are provided to prove its availability in neural networks.

$$AnyLoss = 1 - f(TN, FN, FP, TP) \tag{5}$$

Confusion Matrix. Our method constructs the confusion matrix with the approximated probability $\mathbf{YH}$ and the ground truth $\mathbf{Y}$. We assume that the approximated probability $\mathbf{YH}$ is close enough to the predicted label $\widehat{\mathbf{Y}}$, and this enables the evaluation metric scores to be estimated before updating weights and used as the goal of optimization. The four entries of the confusion matrix are described as (6). This shows that the estimated entries can be approximated to the actual entries.

$$\begin{aligned}
TN(\mathbf{Y}, \widehat{\mathbf{Y}} \approx \mathbf{YH}) &= \sum_{i=1}^{n}(1-y_i)(1-\widehat{y_i}) \approx \sum_{i=1}^{n}(1-y_i)(1-yh_i) \\
FN(\mathbf{Y}, \widehat{\mathbf{Y}} \approx \mathbf{YH}) &= \sum_{i=1}^{n} y_i \cdot (1-\widehat{y_i}) \approx \sum_{i=1}^{n} y_i \cdot (1-yh_i) \\
FP(\mathbf{Y}, \widehat{\mathbf{Y}} \approx \mathbf{YH}) &= \sum_{i=1}^{n}(1-y_i) \cdot \widehat{y_i} \approx \sum_{i=1}^{n}(1-y_i) \cdot yh_i \\
TP(\mathbf{Y}, \widehat{\mathbf{Y}} \approx \mathbf{YH}) &= \sum_{i=1}^{n} y_i \cdot \widehat{y_i} \approx \sum_{i=1}^{n} y_i \cdot yh_i
\end{aligned} \tag{6}$$

Chain Rule. The entries are represented with composite functions through the process described in Fig. 1. Therefore, the functional form of *AnyLoss* is shown as in (7). The chain rule [12] is needed as shown in (8), the first term, $\frac{\partial AnyLoss}{\partial yh_i}$, depends on each loss function, and the rest are calculated as (9).

$$AnyLoss = f(yh_i(p_i(z_i(x_i, \mathbf{W})))) \tag{7}$$

$$\frac{\partial AnyLoss}{\partial \mathbf{W}} = \frac{\partial AnyLoss}{\partial yh_i} \times \frac{\partial yh_i}{\partial p_i} \times \frac{\partial p_i}{\partial z_i} \times \frac{\partial z_i}{\partial \mathbf{W}} \tag{8}$$

$$\frac{\partial yh_i}{\partial \mathbf{W}} = [L \cdot yh_i \cdot (1-yh_i)] \times [p_i \cdot (1-p_i)] \times [\mathbf{x}_i] \tag{9}$$

Diverse Loss Functions. *AnyLoss* indicates loss functions that can target any confusion matrix-based metric, so diverse loss functions exist according to the targeted metric as depicted in (10).

$$\begin{aligned}
L_A &= 1 - Accuracy = 1 - \frac{TP+TN}{TP+TN+FP+FN} \\
L_F &= 1 - F_\beta score = 1 - \frac{(1+\beta^2)TP}{(1+\beta^2)TP+FP+\beta^2 FN} \\
L_G &= 1 - GeometricMean = 1 - \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}} \\
L_B &= 1 - BalancedAcc. = 1 - \frac{1}{2} \times (\frac{TP}{TP+FN} + \frac{TN}{TN+FP})
\end{aligned} \tag{10}$$

Partial Derivative of AnyLoss. How to calculate the partial derivative of the loss function aiming at accuracy, L_A, is addressed as an example. Still, the calculations for other loss functions that follow the same process are explained in Appendix A. The L_A is represented as (11) with the entries (6), and the first term of (8) for L_A, $\frac{\partial L_A}{\partial yh_i}$, is derived as (12), so the partial derivative of L_A is calculated as (13).

$$L_A = 1 - \frac{\sum_{i=1}^{n} 1 - \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} yh_i + 2(\sum_{i=1}^{n} y_i \cdot yh_i)}{n} \tag{11}$$

$$\frac{\partial L_A}{\partial yh_i} = \frac{\sum_{i=1}^{n} yh'_i - 2(\sum_{i=1}^{n} y_i \cdot yh'_i)}{n} \tag{12}$$

$$\frac{\partial L_A}{\partial \mathbf{W}} = \frac{\sum_{i=1}^{n} \frac{\partial yh_i}{\partial \mathbf{W}} - 2(\sum_{i=1}^{n} y_i \cdot \frac{\partial yh_i}{\partial \mathbf{W}})}{n} \tag{13}$$

4 EXPERIMENTS

Extensive experiments demonstrate the availability of our method and its effectiveness by suggesting improved scores against the baseline models. In addition, the performance with imbalanced datasets and the analysis of the learning speed of our method strengthens its efficiency. The experiments are executed in the SLP structure, which is a feed-forward network, and the MLP structure, which uses a back-propagation algorithm so that we can show that our method applies to any type of neural network. The MLP network contains one hidden layer with two nodes, and batch normalization is used at each layer. We set the two baselines, the first is MSE which equals the sum of squared error, the initial perceptron's loss function with the delta rule [23]. Another is the BCE, a representative loss function widely used for classification [11] and has a solid theoretical background in neural networks [22]. The 10-fold cross-validation is used for the robust results instead of splitting train and test data, so the scores indicate the mean of validation scores. In addition, only the results of accuracy, F_1 score, and balanced accuracy are shown here for conciseness, and the full results, including all metrics, are shown in Appendix B. We have research questions for our method as follows:

- Can *AnyLoss* generate optimized scores for any targeted metric regardless of datasets? (4.1)
- Is *AnyLoss* more efficient with imbalanced datasets? (4.1, 4.2)
- Is the learning speed of *AnyLoss* competitive against the baselins? (4.3)

The two groups of datasets are selected to discover the research questions. The first group contains 102 general datasets with diverse properties to see if our method can work regardless of the datasets. The second group contains four imbalanced datasets, and the results will show how well our method works with imbalanced datasets by providing more specific results. As common settings, 1,000 epochs for the SLP and 100 epochs for the MLP are applied. The learning rates and the batch sizes in the MLP are different for each loss function. The detailed settings for experiments are included in the full results in Appendix B.

4.1 Performance on 102 Diverse Datasets

We conduct experiments with 102 diverse datasets and the datasets were initially collected for the imbalanced classification work [24]. The datasets are depicted as having multiple domains and diverse characteristics by the author, and their metadata is shown in Table 2. The description of each dataset is shown in Table 14 in the Appendix. They can provide the diversity required for this experiment since they have different properties regarding the dataset size, the number of features, and imbalance ratios. The imbalance ratio, the 4th column, shows the ratio of major cases versus minor cases, which are considered positive. The experiments with these datasets can demonstrate our method's availability in diverse datasets.

Table 2: Metadata of 102 Diverse Datasets. Datasets have different properties, such as the dataset size, feature numbers, and imbalance ratios.

Metrics	# Samples	# Features	Imb. Rat.
mean	1930.89	37.55	5.35:1
std	2209.16	52.88	3.62:1
min	250.00	2.00	1.54:1
max	9961.00	299.00	16.43:1

Experimental Results in both SLP and MLP structures are shown in Table 3. The numbers indicate each winning number; for instance, for Acc, our *AnyLoss* L_A in the SLP structure achieves a better accuracy score than MSE and BCE in 69 datasets out of a total of 102 datasets. For other metrics, our loss functions show larger winning numbers, meaning our loss functions perform better against baseline models in more datasets for the corresponding evaluation metric. The specific settings for each loss function are shown in Table 15 and Table 16 in the Appendix.

Table 3: Statistical results on 102 datasets. The numbers indicate the datasets with each loss function winning with the corresponding metric.

Evaluation	SLP			MLP		
Metrics	MSE	BCE	OURS	MSE	BCE	OURS
Acc	13	20	**69**	16	20	**66**
F-1	2	10	**90**	5	3	**94**
B-Acc	4	9	**89**	4	4	**94**

AnyLoss mostly dominates the two baselines, but it is not easy to define quantitatively their effectiveness. Therefore, we adopt the Bayesian Sign Test [1], which measures the probability that one model is better than another based on the experiment results. This test provides the probabilities of *AnyLoss* winning, drawing, or losing against each baseline model. The graphical results of the test are shown in Fig. 4. The first two figures represent the results in the SLP structure, and the rest show the results in the MLP structure. The *r* represents the *rope* that decides a region of practical equivalence [1]. The test captures a little difference between the two models when the *r* is small and distinguishes them to declare which one is better. Reversely, when with a larger *r*, the test does not distinguish the two models without a large difference, and it more likely judges them as similar in performance. The authors in [1, 18] address that it is reasonable that two models whose mean difference of scores is less than 1% are considered practically equivalent. Therefore, we provide the results with r=0.01 and r=0.05 for further analysis. For each metric on the X-axis, M and B indicate each MSE and BCE. The Y-axis shows the probability of *AnyLoss* winning, drawing, or losing against the corresponding

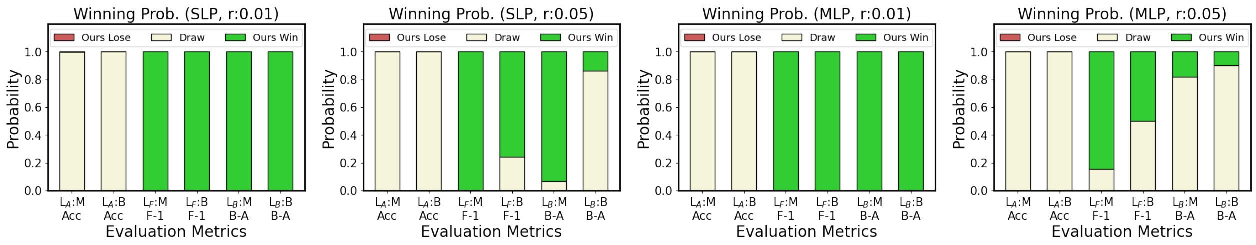

Figure 4: The winning probability of *AnyLoss* against baseline models in stacked bar graphs, with the bottom for losing, the middle for drawing, and the top for winning. M and B represent, respectively, MSE and BCE. In the SLP, *AnyLoss* mostly has a larger winning probability against baseline models. In some cases, a larger drawing probability is observed. In the MLP, the results are similar to those in the SLP, but more cases with a larger drawing probability are observed. And there are no red colors, meaning no cases of *AnyLoss* losing.

baseline model. They are stacked with the bottom for losing, the middle for drawing, and the top for winning. The sum of the three probabilities, the height of all bars, is 1.00. The first bar in the first figure (SLP, r:0.01) shows that *AnyLoss*, L_A, has both probabilities of winning and drawing. However, the area for drawing is much larger, which indicates L_A has a larger probability of drawing and a little probability of winning against MSE in terms of the accuracy score. In the second figure, when with a larger r=0.05, *AnyLoss* generally has a larger drawing probability compared to when r=0.01 since a little difference is ignored. In the SLP structure, our loss functions mostly have a larger winning probability against the two baseline models except for accuracy. The results in the MLP structure are very similar to those in the SLP. The red color is not observed in any graph, which means that the probability of MSE or BCE being better than *AnyLoss* is 0 for all metrics. Consequently, *AnyLoss* shows mostly better performance of all metrics except for accuracy on 102 general datasets. Considering the objective of the two baselines, pursuing higher accuracy, our method shows good performance, which was our intention. The numerical results of the test are shown in Table 15 and Table 16 in the Appendix.

To observe if our method works better with imbalanced datasets, we classify 102 datasets into four groups based on their imbalance ratios. Whether better or not has been shown in the previous results, now we provide how much better *AnyLoss* against each baseline model in Table 4. The symbol Δ means the difference, therefore the results indicate '*AnyLoss* score - baseline score'. For example, the 31 datasets in the first group have imbalance ratios between 60:40 and 70:30, and *AnyLoss* accuracy score is better than MSE as 0.017 of mean ± 0.035 of standard deviation. Accuracies show irregular patterns. However, there is a tendency for the F-1 score and balanced accuracy, a larger difference in the group of highly imbalanced datasets. From top to bottom, little imbalanced to highly imbalanced, differences become larger, which means our method works better with more imbalanced datasets. This result shows that *AnyLoss* can be more suitable than the two baseline models in dealing with highly imbalanced datasets.

4.2 Performance on 4 Imbalanced Datasets

We conduct experiments with 4imbalanced datasets to observe our method's availability with imbalanced datasets, and their descriptions are shown in Table 5. The first dataset is generated with the

Table 4: The effect of our method in different groups based on imbalance ratios. The symbol Δ indicates the difference, meaning how much our method is better than each model. The numbers mean '*AnyLoss* score - baseline score'. From top to bottom, datasets are more imbalanced, and larger scores are observed, which indicates our method performs better in more imbalanced datasets.

Eva.	SLP		MLP	
Met.	Δ MSE	Δ BCE	Δ MSE	Δ BCE
	Imbalance Ratio: 60:40 ∼ 70:30 | 31 Datasets			
Acc	0.017±0.035	0.007±0.029	0.004±0.025	0.009±0.016
F-1	0.081±0.095	0.057±0.092	0.040±0.100	0.040±0.091
B-A	0.043±0.052	0.026±0.049	0.007±0.046	0.009±0.044
	Imbalance Ratio: 70:30 ∼ 80:20 | 19 Datasets			
Acc	0.021±0.029	0.017±0.027	-0.058±0.207	-0.062±0.207
F-1	0.186±0.156	0.142±0.132	0.114±0.117	0.096±0.124
B-A	0.086±0.065	0.067±0.057	0.067±0.061	0.055±0.056
	Imbalance Ratio: 80:20 ∼ 90:10 | 37 Datasets			
Acc	0.008±0.021	0.002±0.009	0.002±0.006	0.002±0.007
F-1	0.208±0.209	0.122±0.131	0.138±0.144	0.137±0.151
B-A	0.128±0.120	0.084±0.086	0.084±0.080	0.079±0.082
	Imbalance Ratio: 90:10 ∼ | 15 Datasets			
Acc	0.004±0.008	0.001±0.005	0.002±0.009	0.004±0.009
F-1	0.304±0.144	0.167±0.127	0.253±0.179	0.222±0.127
B-A	0.226±0.097	0.170±0.088	0.152±0.076	0.128±0.076

score: mean ± standard deviation

Scikitlearn library *datasets.make_classification*, and others are collected from *Kaggle*. For *Credit Card* [16] and *Breast Cancer* [14] datasets, we intentionally make them more imbalanced using the Python library *sample*. In addition, we adopt one of the SOTA surrogate approaches, the Score-Oriented Loss (SOL) [22] for comparison. SOL proposes a similar approach to ours, constructing a confusion matrix to generate any metric, therefore we compare it with ours.

Experimental Results in both SLP and MLP structures are shown in Table 6. The table provides only one corresponding metric that each *AnyLoss* aims at. For example, L_A is made for higher accuracy,

Table 5: Description of 4 Imbalanced Datasets

Datasets	# Samples	# Features	Imb. Rat.
#1. Random	10,000	2	9:1
#2. Credit Card [16]	298,531	29	20:1
#3. Breast Cancer [14]	393	30	10:1
#4. Diabetes [17]	100,000	8	11:1

so we only present the accuracy score of it. In the first row, 0.922 is the accuracy score of L_A, and 0.632 in the second row is the F-1 score of L_{F_1}. The scores of our *AnyLoss* are similar to SOL in the SLP and mostly better than SOL in the MLP structure but the differences are not large. Baseline models generally show good performance only for accuracy and large differences for other metrics. The full experiment results including other metric scores of each *AnyLoss* are shown in Table 17 and Table 18 in the Appendix. The results in Table 6 show that it is possible to get an optimized score of the desired metric with *AnyLoss* in imbalanced datasets.

Table 6: Scores on four imbalanced datasets. *AnyLoss* and SOL show similar performance and are better than the two baseline models. In particular, large differences are observed in metrics other than accuracy.

Eva	SLP				MLP			
Met	MSE	BCE	SOL	OUR	MSE	BCE	SOL	OUR
	#1. Random							
Acc	0.898	0.921	**0.922**	**0.922**	0.920	0.924	0.923	**0.925**
F-1	0.075	0.546	**0.634**	0.632	0.527	0.590	0.636	**0.637**
B-A	0.519	0.714	**0.875**	**0.875**	0.711	0.748	0.858	**0.862**
	#2. Credit Card							
Acc	0.989	0.990	**0.991**	0.990	0.991	**0.994**	0.992	**0.994**
F-1	0.880	0.884	0.896	**0.903**	0.889	0.929	0.944	**0.946**
B-A	0.894	0.897	**0.957**	**0.957**	0.904	0.942	0.950	**0.957**
	#3. Breast Cancer							
Acc	0.979	0.984	0.980	**0.987**	0.982	**0.995**	0.985	**0.995**
F-1	0.863	0.901	**0.922**	0.901	0.833	0.966	0.955	**1.000**
B-A	0.887	0.927	**0.965**	0.963	0.900	0.971	0.975	**0.983**
	#4. Diabetes							
Acc	0.939	0.959	0.898	**0.960**	0.956	0.960	**0.961**	**0.961**
F-1	0.453	0.710	0.670	**0.732**	0.655	0.717	**0.736**	**0.736**
B-A	0.647	0.790	0.880	**0.885**	0.766	0.793	0.865	**0.877**

Table 7 compares our method with resampling strategies on the four datasets in the SLP structure. The resampling strategies are applied only to BCE since it is better than MSE in the previous experiments. BSO indicates BCE with SMOTE [7], and BRU means BCE with the random under-sampling. As for a resampling rate, we try multiple (major : minor) rates from (1:1) to (1:0.1) with 0.1 intervals, and the result with the highest F-1 score is chosen from the results from all different rates. Over-sampling and under-sampling strategies show improved results w.r.t. basic BCE, with scores similar to those of *AnyLoss*. On the other hand, our method does not distort data distribution and achieves a goal at once without multiple times of experiments to choose the best.

Table 7: Comparison with resampling strategies. BSO and BRU indicate BCE with SMOTE and BCE with random under-sampling. Both strategies improve performance, but *AnyLoss* still shows competitive results.

Eva.	SLP					
Met.	BSO	BRU	OURS	BSO	BRU	OURS
	#1. Random			*#3. Breast Cancer*		
Acc	0.915	0.916	**0.922**	**0.987**	0.984	**0.987**
F-1	**0.637**	**0.637**	0.632	**0.930**	0.914	0.901
B-A	0.823	0.821	**0.875**	**0.970**	0.965	0.963
	#2. Credit Card			*#4. Diabetes*		
Acc	**0.992**	**0.992**	0.990	0.959	0.959	**0.960**
F-1	0.912	**0.913**	0.903	0.727	0.727	**0.732**
B-A	0.932	0.932	**0.957**	0.809	0.809	**0.885**

4.3 Learning Time

We measure the learning time of *AnyLoss*, SOL, and the two baseline models in the same epochs, and this shows the pure learning time per epoch, i.e., learning speed. We also analyze the loss curve to conjecture the required epochs for each method. This gives us inspiration about the practical learning time of each method.

Learning Speed. Under the same number of epochs, we measure the learning time of *AnyLoss*, SOL, and the two baseline models with the four imbalanced datasets in both SLP and MLP structures. The same condition is required for all loss functions, so the same settings, learning rates, and batch sizes are applied instead of the already used settings. The results are as Table 8. The ratios are

Table 8: Ratios of learning time based on BCE's learning time with the four datasets. L_{mean} and S_{mean} mean each the average ratio of all *Anyloss* and SOL. In both SLP and MLP structures, MSE is faster than BCE, *AnyLoss* is similar to BCE, and the SOL is the slowest.

Data	SLP			MLP		
sets	MSE	S_{mean}	L_{mean}	MSE	S_{mean}	L_{mean}
#1	1.001	1.120	1.005	0.888	1.158	1.031
#2	0.893	1.175	0.968	0.783	1.090	0.993
#3	0.821	1.103	0.901	0.994	1.156	1.011
#4	0.905	1.151	1.041	0.804	1.082	0.987

The standard is BCE's learning time, i.e., BCE: 1.00.

calculated based on the learning time of BCE, therefore the ratios in the table are calculated as its learning time divided by the learning time of BCE. The differences among learning times of all *AnyLoss* are small, so we used their average value, L_{mean}. In the same sense, S_{mean} indicates the average value of all SOL functions. The ratios are learning speed because the result times are based on the same number of epochs. The results are similar in both the SLP and MLP structures. MSE is faster than BCE, *AnyLoss* is similar to BCE, and SOL is the slowest. Our method shows a competitive learning speed against BCE since our approach includes one more step, the approximation step. Its calculation is simple, so it rarely affects the learning time. On the other hand, SOL shows slower speed due to its complicated process. This result shows the competitiveness of our method in terms of pure learning speed in neural networks.

Practical Learning Time. In reality, the more important figure about learning time is the practically required time for learning, not pure speed. We measured the pure learning time in the previous step; however, the amount of time needed for learning is not explained by the unit learning speed because it largely depends on the required epochs. Therefore, we analyze the loss learning curves to discover each method's practical required number of epochs. We choose BCE and *AnyLoss*, the fastest learning speed for this experiment. Different learning rates for each method have to be applied since they decide the slope of the curve and performance. For example, if we increase the learning rate for a certain loss function, it may learn faster, but it does not guarantee the best performance. We already have the appropriate learning rates used for the previous experiments, so they are used for this experiment. Fig. 5 shows how fast each method learns and converges to its minimum loss value. The X-axis and Y-axis represent each the number of epochs

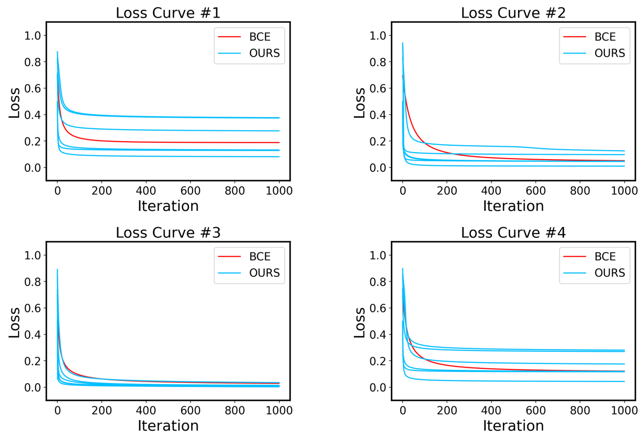

Figure 5: The learning curves BCE vs. *AnyLoss*. They show similar slopes in dataset 1, and *AnyLoss* shows a steeper slope in other datasets, meaning it learns faster and needs fewer epochs to achieve its optimal point.

and loss value. The red line represents BCE's learning curve, and the blue lines show all *AnyLoss*, i.e. L_A, L_{F1}, etc. The minimum loss value of each function is different since they have their own goal defined by different loss functions. The first graph, 'Loss Curve #1' shows the results on the first dataset, and the slopes of all lines look similar. Before the 100 epochs, they seem to have mostly achieved the minimum values. However, in the other three graphs, the slopes of the blue lines look steeper than those of BCE. For a more detailed analysis, we calculate the achievement rate, described as (14).

$$AchievementRate = \frac{InitialLoss - CurrentLoss}{InitialLoss - FinalLoss} \tag{14}$$

This rate indicates how much loss decreased by a certain period of epochs. For instance, if the initial loss value is 1.0, the loss value is 0.7 when the epoch is 100, and the final loss value is 0.4, then it decreases a total of 0.6 for a whole period of epochs. It decreases by 0.3 for the first 100 epochs. Therefore, the achievement rate at epoch 100 is 0.5, and the rate at the last epoch is always 1.0. Table 9 shows the results of the achievement rates, and L_{mean} is the average of the achievement rates of all *AnyLoss*. In the first dataset, the two achievement rates are similar, and the difference is not large. In other datasets, the average achievement rate of *AnyLoss* is larger than BCE at all epochs, which indicates *AnyLoss* generally learns faster than BCE. Consequently, *AnyLoss* normally needs a smaller number of epochs leading to a shorter learning time.

Table 9: Achievement rates on four datasets

Data sets	Loss Fun.	Epoch 100	200	300	400	500	1,000
#1	BCE	0.930	0.974	0.987	0.993	0.996	1.0
	L_{mean}	0.951	0.973	0.982	0.987	0.991	1.0
#2	BCE	0.790	0.904	0.945	0.965	0.977	1.0
	L_{mean}	0.970	0.980	0.984	0.987	0.989	1.0
#3	BCE	0.904	0.949	0.967	0.978	0.984	1.0
	L_{mean}	0.961	0.978	0.985	0.990	0.993	1.0
#4	BCE	0.838	0.922	0.953	0.969	0.979	1.0
	L_{mean}	0.959	0.978	0.985	0.990	0.993	1.0

5 ADDITIONAL EXPERIMENTAL RESULTS

To reinforce our argument on the availability of *AnyLoss*, we provide more experimental results with an advanced MLP architecture and a larger dataset. In addition, we present the performance change results along with the change of the amplifying scale L to support our approach suggested in the previous section to determine a good L value of the approximation function.

5.1 Advanced Architecture

The SLP and MLP structures were chosen for the analysis from the perspective of neural networks, but they may not be enough to prove the effectiveness of our method convincingly. Therefore, we provide more experimental results with an advanced MLP architecture. We did an image classification task with the ResNet50 [10] and the MNIST_784 [25] dataset. The dataset originally had 10 class labels, so we chose one class label as a positive case to make a binary classification task and others were given a negative label. The results support that *AnyLoss* still works with more complex architecture as shown in Table 10.

5.2 Large Dataset

In terms of dataset size, a new loss function should properly work with any size of datasets, therefore, we provide experiment results with a large dataset, *Huge Titanic* [15] (#samples: 799,603, #features: 7, imbalance ratio: 0.62:0.38). The original number of samples is a million but the dataset contains many missing data, so we used 799,603 after pre-processing. Table 11 shows that *AnyLoss* works well in both structures with any metrics.

Table 10: Scores of different loss functions in the ResNet50 architecture with the MNIST_784 dataset. *AnyLoss* shows better performance in each evaluation metric.

Metrics	MSE	BCE	OURS
Acc	0.994	0.994	**0.998**
F-1	0.965	0.965	**0.980**
B-A	0.972	0.970	**0.980**

Table 11: Scores on the Huge Titanic dataset. *AnyLoss* is better than the two baseline models in both SLP and MLP structures.

Eva	SLP			MLP		
Met	MSE	BCE	OURS	MSE	BCE	OURS
Acc	0.800	0.806	**0.814**	0.807	0.814	**0.822**
F-1	0.726	0.734	**0.739**	0.730	0.729	**0.746**
B-A	0.780	0.786	**0.788**	0.783	0.784	**0.796**

5.3 Determination of L

Determination of the amplifying scale L is essential for the approximation function since it decides how to amplify the given probability from the Sigmoid function. As discussed earlier, an invalid value of L can cause malfunctioning of the approximation function, and then it will not provide the desired results even if it looks working properly. So, we executed experiments to observe how performances change along with the change of the amplifying scale L. The experiment results in the MLP structure with different L values on different datasets are shown in Table 12 and Table 13. We used 5 different L values including 73 which has been used as a value of L in this paper. The 3 evaluation metrics are calculated on the 4 imbalanced datasets and 102 diverse datasets as done earlier. In the results, the value of 73 shows better performance than others but the difference is not large. Therefore, this topic, determining a good L, can be considered as future work.

6 CONCLUSION

In this paper, we introduce *AnyLoss*, which can aim at a specific confusion matrix-based evaluation metric with the approximation function. Our method enables any confusion matrix-based metric to be directly set as a goal of learning, i.e., loss function, in a neural network architecture to achieve a unique goal of the binary classification task. With mathematical approaches, we prove that *AnyLoss* are differentiable and provide their derivatives. Analysis of the approximation function provided with experiments helps users use it more efficiently in practice. Extensive experiment results demonstrate that our method achieves a goal in both SLP and MLP structures with diverse datasets while maintaining a competitive learning time. It, especially, shows considerable performance in imbalanced datasets although it is available with any type of dataset. A similar approach to handling multi-class classification tasks or determining a good value of L to improve its efficiency can be interesting research topics for future research. All the datasets and codes used for the experiments are available at https://github.com/doheonhan/anyloss.

Table 12: Scores on 4 imbalanced datasets with different L. The value 73 which we chose earlier mostly shows the best score in each dataset and evaluation metric even though big differences are not observed.

Dataset	Metric	L=30	L=50	L=73	L=90	L=110
Random Generated	Acc	0.915	0.923	**0.924**	0.895	0.922
	F-1	0.636	0.632	**0.637**	0.634	0.635
	B-A	0.840	0.834	**0.852**	0.826	0.825
Credit Card	Acc	0.952	0.994	**0.995**	0.994	0.994
	F-1	0.935	0.942	**0.945**	**0.945**	0.944
	B-A	0.950	0.915	**0.962**	0.956	0.871
Breast Cancer	Acc	0.998	0.995	**1.000**	0.997	0.997
	F-1	0.980	0.980	**0.989**	0.980	0.969
	B-A	0.949	0.950	**1.000**	0.989	0.996
Diabetes Prediction	Acc	0.948	**0.961**	**0.961**	**0.961**	**0.961**
	F-1	0.728	0.733	**0.736**	0.735	0.734
	B-A	0.837	0.859	**0.882**	0.832	0.826

Table 13: Scores on 102 diverse datasets with different L. Each number in the table indicates the winning number, i.e., a model with L=73 shows the best score in 31 out of 102 datasets. The value 73 shows the best winning number.

Metric	L=30	L=50	L=73	L=90	L=110
Acc	21	18	**31**	21	24
F-1	21	19	**27**	21	16
B-A	17	20	**26**	17	23

REFERENCES

[1] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. 2017. Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. *The Journal of Machine Learning Research* 18, 1 (2017), 2653–2688.

[2] Gabriel Bénédict, Vincent Koops, Daan Odijk, and Maarten de Rijke. 2021. SigmoidF1: A smooth F1 score surrogate loss for multilabel classification. *arXiv preprint arXiv:2108.10566* (2021).

[3] Houda Benhar, Ali Idri, and JL Fernández-Alemán. 2020. Data preprocessing for heart disease classification: A systematic literature review. *Computer Methods and Programs in Biomedicine* 195 (2020), 105635.

[4] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. 2010. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*. IEEE, 3121–3124.

[5] Róbert Busa-Fekete, Balázs Szörényi, Krzysztof Dembczynski, and Eyke Hüllermeier. 2015. Online f-measure optimization. *Advances in Neural Information Processing Systems* 28 (2015).

[6] Tianfeng Chai and Roland R Draxler. 2014. Root mean square error (RMSE) or mean absolute error (MAE). *Geoscientific model development discussions* 7, 1 (2014), 1525–1534.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[8] Vaishali Ganganwar. 2012. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering* 2, 4 (2012), 42–47.

[9] David Goldberg. 1991. What every computer scientist should know about floating-point arithmetic. *ACM computing surveys (CSUR)* 23, 1 (1991), 5–48.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.

[11] Chen Huang, Shuangfei Zhai, Pengsheng Guo, and Josh Susskind. 2021. MetricOpt: Learning To Optimize Black-Box Evaluation Metrics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 174–183.

[12] HN Huang, SAM Marcantognini, and NJ Young. 2006. Chain rules for higher derivatives. *The Mathematical Intelligencer* 28, 2 (2006), 61–69.

[13] Katarzyna Janocha and Wojciech Marian Czarnecki. 2017. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659* (2017).

[14] Kaggle. 2021. *Breast Cancer Dataset.* Retrieved October 2, 2023 from https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset

[15] Kaggle. 2022. *Titanic Huge Dataset - 1M Passengers.* Retrieved May 21, 2024 from https://www.kaggle.com/datasets/marcpaulo/titanic-huge-dataset-1m-passengers

[16] Kaggle. 2023. *Credit Card Fraud Detection Dataset 2023.* Retrieved October 2, 2023 from https://www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023

[17] Kaggle. 2023. *Diabetes prediction dataset.* Retrieved October 2, 2023 from https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset

[18] John K Kruschke and Torrin M Liddell. 2015. The Bayesian new statistics: Two historical trends converge. *SSRN Electronic Journal* 2606016 (2015), 26.

[19] Namgil Lee, Heejung Yang, and Hojin Yoo. 2021. A surrogate loss function for optimization of F$_\beta$ score in binary classification with imbalanced data. *arXiv preprint arXiv:2104.01459* (2021).

[20] Jake Lever. 2016. Classification evaluation: It is important to understand both what a classification metric expresses and what it hides. *Nature methods* 13, 8 (2016), 603–605.

[21] Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. 2014. Optimal thresholding of classifiers to maximize F1 measure. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II 14.* Springer, 225–239.

[22] F. Marchetti, S. Guastavino, M. Piana, and C. Campi. 2022. Score-Oriented Loss (SOL) functions. *Pattern Recognition* 132 (2022), 108913. https://doi.org/10.1016/j.patcog.2022.108913

[23] Tom M Mitchell. 1997. Machine learning.

[24] Nuno Moniz and Vitor Cerqueira. 2021. Automated imbalanced classification via meta-learning. *Expert Systems with Applications* 178 (2021), 115011.

[25] OpenML. 2014. *mnist_784.* Retrieved May 21, 2024 from https://openml.org/search?type=data&status=active&id=554&sort=runs

[26] Sebastian Raschka. 2018. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808* (2018).

[27] Stuart C Reid. 1997. An empirical analysis of equivalence partitioning, boundary value analysis and random testing. In *Proceedings Fourth International Software Metrics Symposium.* IEEE, 64–73.

[28] Yiming Yang. 2001. A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval.* 137–145.

[29] Yage Yuan, Jianan Wei, Haisong Huang, Weidong Jiao, Jiaxin Wang, and Hualin Chen. 2023. Review of resampling techniques for the treatment of imbalanced industrial data classification in equipment condition monitoring. *Engineering Applications of Artificial Intelligence* 126 (2023), 106911.

[30] Quan Zou, Sifa Xie, Ziyu Lin, Meihong Wu, and Ying Ju. 2016. Finding the best classification threshold in imbalanced classification. *Big Data Research* 5 (2016), 2–8.

A CALCULATION OF PARTIAL DERIVATIVES

A.1 F-beta scores

The loss function aiming at F-β, L_F, is represented as (15) with the confusion matrix (6). The first term of (8) for L_F is as (16). The partial derivative of L_F is as (17).

$$L_F = 1 - \frac{(1+\beta^2)(\sum_{i=1}^n y_i \cdot yh_i)}{\beta^2 \sum_{i=1}^n y_i + \sum_{i=1}^n yh_i} \tag{15}$$

$$\frac{\partial L_F}{\partial yh_i} = -(1+\beta^2)\times \frac{(\sum_{i=1}^n y_i \cdot yh_i')(\beta^2 \sum_{i=1}^n y_i + \sum_{i=1}^n yh_i) - (\sum_{i=1}^n yh_i')(\sum_{i=1}^n y_i \cdot yh_i)}{(\beta^2 \sum_{i=1}^n y_i + \sum_{i=1}^n yh_i)^2} \tag{16}$$

$$\frac{\partial L_F}{\partial \mathbf{W}} = -(1+\beta^2)\times \frac{(\sum_{i=1}^n y_i \cdot \frac{dyh_i}{d\mathbf{W}})(\beta^2 \sum_{i=1}^n y_i + \sum_{i=1}^n yh_i) - (\sum_{i=1}^n \frac{dyh_i}{d\mathbf{W}})(\sum_{i=1}^n y_i \cdot yh_i)}{(\beta^2 \sum_{i=1}^n y_i + \sum_{i=1}^n yh_i)^2} \tag{17}$$

A.2 Geometric Mean

The loss function aiming at geometric mean, L_G, is represented as (18) with the confusion matrix (6). The first term of (8) for L_G is as (19). The partial derivative of L_G is as (20).

$$L_G = 1 - \sqrt{\frac{(\sum_{i=1}^n y_i \cdot yh_i)(n - \sum_{i=1}^n y_i - \sum_{i=1}^n yh_i + \sum_{i=1}^n y_i \cdot yh_i)}{(\sum_{i=1}^n y_i)(n - \sum_{i=1}^n y_i)}} \tag{18}$$

$$\frac{\partial L_G}{\partial yh_i} = \frac{-2}{\sqrt{(\sum_{i=1}^n y_i)(n - \sum_{i=1}^n y_i)}}\times \left[\frac{(\sum_{i=1}^n y_i \cdot yh_i')(n - \sum_{i=1}^n y_i - \sum_{i=1}^n yh_i + \sum_{i=1}^n y_i \cdot yh_i)}{\sqrt{(\sum_{i=1}^n y_i \cdot yh_i)(n - \sum_{i=1}^n y_i - \sum_{i=1}^n yh_i + \sum_{i=1}^n y_i \cdot yh_i)}} + \frac{(-\sum_{i=1}^n yh_i' + \sum_{i=1}^n y_i \cdot yh_i')(\sum_{i=1}^n y_i \cdot yh_i)}{\sqrt{(\sum_{i=1}^n y_i \cdot yh_i)(n - \sum_{i=1}^n y_i - \sum_{i=1}^n yh_i + \sum_{i=1}^n y_i \cdot yh_i)}}\right] \tag{19}$$

$$\frac{\partial L_G}{\partial \mathbf{W}} = \frac{-2}{\sqrt{(\sum_{i=1}^n y_i)(n - \sum_{i=1}^n y_i)}}\times \left[\frac{(\sum_{i=1}^n y_i \cdot \frac{dyh_i}{d\mathbf{W}})(n - \sum_{i=1}^n y_i - \sum_{i=1}^n yh_i + \sum_{i=1}^n y_i \cdot yh_i)}{\sqrt{(\sum_{i=1}^n y_i \cdot yh_i)(n - \sum_{i=1}^n y_i - \sum_{i=1}^n yh_i + \sum_{i=1}^n y_i \cdot yh_i)}} + \frac{(-\sum_{i=1}^n \frac{dyh_i}{d\mathbf{W}} + \sum_{i=1}^n y_i \cdot \frac{dyh_i}{d\mathbf{W}})(\sum_{i=1}^n y_i \cdot yh_i)}{\sqrt{(\sum_{i=1}^n y_i \cdot yh_i)(n - \sum_{i=1}^n y_i - \sum_{i=1}^n yh_i + \sum_{i=1}^n y_i \cdot yh_i)}}\right] \tag{20}$$

A.3 Balanced Accuracy

The loss function aiming at balanced accuracy, L_B, is represented as (21) with the confusion matrix (6). The first term of (8) for L_B is as (22). The partial derivative of L_B is as (23).

$$L_B = 1 - \frac{n(\sum_{i=1}^n y_i \cdot yh_i) + n\sum_{i=1}^n y_i - \sum_{i=1}^n y_i \cdot \sum_{i=1}^n yh_i - (\sum_{i=1}^n y_i)^2}{2(\sum_{i=1}^n y_i)(n - \sum_{i=1}^n y_i)} \tag{21}$$

$$\frac{\partial L_B}{\partial yh_i} = \frac{-n(\sum_{i=1}^n y_i \cdot yh_i') + \sum_{i=1}^n y_i \cdot \sum_{i=1}^n yh_i'}{2(\sum_{i=1}^n y_i)(n - \sum_{i=1}^n y_i)} \tag{22}$$

$$\frac{\partial L_B}{\partial \mathbf{W}} = \frac{-n(\sum_{i=1}^n y_i \cdot \frac{dyh_i}{d\mathbf{W}}) + \sum_{i=1}^n y_i \cdot \sum_{i=1}^n \frac{dyh_i}{d\mathbf{W}}}{2(\sum_{i=1}^n y_i)(n - \sum_{i=1}^n y_i)} \tag{23}$$

B RESULTS WITH THE DETAILS

Table 14 shows the description of 102 datasets. Table 15 and Table 16 show the winning number and probability with the 102 datasets. Table 17 and Table 18 show the scores in the 4 datasets.

Table 14: Description of 102 Diverse Datasets. The number of samples, the number of features, and the imbalance ratios.

Data	#Samp.	#Feat.	Imb.	Data	#Samp.	#Feat.	Imb.	Data	#Samp.	#Feat.	Imb.	Data	#Samp.	#Feat.	Imb.
1	250	12	0.64:0.36	27	504	19	0.91:0.09	53	1000	19	0.70:0.30	78	2201	2	0.68:0.32
2	250	9	0.62:0.38	28	522	20	0.80:0.20	54	1000	20	0.74:0.26	79	2310	17	0.86:0.14
3	306	3	0.74:0.26	29	531	101	0.90:0.10	55	1043	37	0.88:0.12	80	2407	299	0.82:0.18
4	310	6	0.68:0.32	30	540	20	0.91:0.09	56	1055	32	0.66:0.34	81	2417	115	0.74:0.26
5	320	6	0.67:0.33	31	559	4	0.86:0.14	57	1066	7	0.83:0.17	82	2534	71	0.94:0.06
6	327	37	0.87:0.13	32	562	21	0.84:0.16	58	1074	16	0.68:0.32	83	3103	12	0.93:0.07
7	328	32	0.69:0.31	33	569	30	0.63:0.37	59	1077	37	0.88:0.12	84	3103	12	0.92:0.08
8	335	3	0.85:0.15	34	583	10	0.71:0.29	60	1109	21	0.93:0.07	85	4052	5	0.76:0.24
9	336	14	0.76:0.24	35	593	77	0.68:0.32	61	1156	5	0.78:0.22	86	4474	11	0.75:0.25
10	349	31	0.62:0.38	36	600	61	0.83:0.17	62	1320	17	0.91:0.09	87	4521	14	0.88:0.12
11	351	33	0.64:0.36	37	609	7	0.63:0.37	63	1324	10	0.78:0.22	88	4601	7	0.61:0.39
12	358	31	0.69:0.31	38	641	19	0.68:0.32	64	1458	37	0.88:0.12	89	4859	120	0.83:0.17
13	363	8	0.77:0.23	39	645	168	0.94:0.06	65	1563	37	0.90:0.10	90	5000	40	0.66:0.34
14	365	5	0.92:0.08	40	661	37	0.92:0.08	66	1728	6	0.70:0.30	91	5000	19	0.86:0.14
15	381	38	0.85:0.15	41	683	9	0.65:0.35	67	1941	31	0.65:0.35	92	5404	5	0.71:0.29
16	392	8	0.62:0.38	42	705	37	0.91:0.09	68	2000	6	0.90:0.10	93	5473	10	0.90:0.10
17	400	5	0.78:0.22	43	748	4	0.76:0.24	69	2000	76	0.90:0.10	94	5620	48	0.90:0.10
18	403	35	0.92:0.08	44	768	8	0.65:0.35	70	2000	216	0.90:0.10	95	6598	169	0.85:0.15
19	450	3	0.88:0.12	45	797	4	0.81:0.19	71	2000	47	0.90:0.10	96	6598	168	0.85:0.15
20	458	38	0.91:0.09	46	812	6	0.77:0.23	72	2000	64	0.90:0.10	97	7032	36	0.89:0.11
21	462	9	0.65:0.35	47	841	70	0.62:0.38	73	2000	239	0.90:0.10	98	7970	39	0.93:0.07
22	470	13	0.85:0.15	48	846	18	0.74:0.26	74	2000	139	0.72:0.28	99	8192	12	0.70:0.30
23	475	3	0.87:0.13	49	958	9	0.65:0.35	75	2000	140	0.87:0.13	100	8192	19	0.70:0.30
24	475	3	0.87:0.13	50	959	40	0.64:0.36	76	2001	2	0.76:0.24	101	8192	32	0.69:0.31
25	500	25	0.61:0.39	51	973	9	0.67:0.33	77	2109	20	0.85:0.15	102	9961	14	0.84:0.16
26	500	22	0.84:0.16	52	990	13	0.91:0.09								

Table 15: Results in SLP for the 102 datasets.

Winning Numbers				Winning Probability			
				Win/Draw/Lose (r:0.01)		Win/ Draw/Lose (r:0.05)	
Eval. Metrics	MSE lr:1e-2	BCE lr:1e-1	OURS L_{Any}(lr)	OURS vs. MSE	OURS vs. BCE	OURS vs. MSE	OURS vs. BCE
Acc	13	20	**69** | L_A(5e-3)	0.005/0.995/0.000	0.000/1.000/0.000	0.000/1.000/0.000	0.000/1.000/0.000
F-1	2	10	**90** | L_{F_1}(1e-2)	1.000/0.000/0.000	1.000/0.000/0.000	0.999/0.001/0.000	0.759/0.241/0.000
G-Mean	2	5	**95** | L_G(5e-3)	1.000/0.000/0.000	1.000/0.000/0.000	1.000/0.000/0.000	0.932/0.068/0.000
B-Acc	4	9	**89** | L_B(5e-3)	1.000/0.000/0.000	1.000/0.000/0.000	0.931/0.069/0.000	0.139/0.861/0.000

Epochs: 1,000

Table 16: Results in MLP for the 102 datasets.

Winning Numbers				Winning Probability			
				Win/Draw/Lose (r:0.01)		Win/ Draw/Lose (r:0.05)	
Eval. Metrics	MSE lr:5e-3	BCE lr:3e-3	OURS L_{Any}(lr)	OURS vs. MSE	OURS vs. BCE	OURS vs. MSE	OURS vs. BCE
Acc	16	20	**66** | L_A(5e-3)	0.000/1.000/0.000	0.000/1.000/0.000	0.000/1.000/0.000	0.000/1.000/0.000
F-1	5	3	**94** | L_{F_1}(1e-3)	1.000/0.000/0.000	1.000/0.000/0.000	0.844/0.156/0.000	0.501/0.499/0.000
G-Mean	4	8	**90** | L_G(1e-2)	1.000/0.000/0.000	1.000/0.000/0.000	0.696/0.304/0.000	0.689/0.311/0.000
B-Acc	4	4	**94** | L_B(5e-3)	1.000/0.000/0.000	1.000/0.000/0.000	0.181/0.819/0.000	0.095/0.905/0.000

1 Hidden Layer (2 nodes) / Activation: Sigmoid / Epochs: 100 / Batch Size: train data size × 5e-2 (5e-1 for L_G & L_B)

Table 17: Results in SLP for the 4 datasets.

Data	Eval.	MSE	BCE	L_A	L_{F_1}	$L_{F.5}$	L_{F_2}	L_G	L_B	S_A	S_F	S_B
#1	Metrics	lr:1e-2	lr:2e-1	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-2	lr:5e-3	lr:5e-3
	Acc	0.898000	0.921500	0.922000	0.913900	0.923500	0.872600	0.859400	0.853300	0.922400	0.913000	0.853300
Ran.	F-1	0.075428	0.546856	0.540208	0.632383	0.572610	0.589266	0.571078	0.563108	0.533386	0.634453	0.562872
Gen.	G-Mean	0.194819	0.664798	0.654791	0.814538	0.690798	0.872564	0.874335	0.874540	0.646546	0.821146	0.874122
	B-Acc	0.519203	0.714945	0.708910	0.823336	0.732524	0.872757	0.874674	0.875064	0.705226	0.828751	0.874640
#2	Metrics	lr:5e-3	lr:1e-2	lr:1e-3	lr:5e-3	lr:5e-3	lr:5e-4	lr:1e-3	lr:5e-3	lr:5e-3	lr:1e-3	lr:5e-3
	Acc	0.989800	0.990078	0.990430	0.991572	0.991210	0.992128	0.981580	0.982196	0.990832	0.991016	0.984491
Cre.	F-1	0.880601	0.884281	0.888906	0.903505	0.898935	0.913576	0.827887	0.832658	0.894053	0.896437	0.851334
Card	G-Mean	0.888696	0.892165	0.896594	0.910165	0.905946	0.933853	0.956730	0.956942	0.901414	0.903675	0.956256
	B-Acc	0.894874	0.897960	0.901920	0.914181	0.910349	0.935924	0.957116	0.957340	0.906273	0.908308	0.956908
#3	Metrics	lr:5e-2	lr:5e-2	lr:5e-4	lr:1e-3	lr:1e-2	lr:7e-3	lr:3e-3	lr:5e-3	lr:1e-4	lr:1e-4	lr:1e-4
	Acc	0.979615	0.984679	0.987244	0.984679	0.984679	0.989744	0.982179	0.982179	0.979808	0.982372	0.977244
Bre.	F-1	0.863810	0.901270	0.912381	0.901270	0.892381	0.932381	0.901270	0.906032	0.908045	0.922330	0.893759
Can.	G-Mean	0.875467	0.919174	0.920613	0.919174	0.902262	0.950962	0.946705	0.961198	0.964899	0.966298	0.963500
	B-Acc	0.887500	0.927738	0.929167	0.927738	0.912500	0.956944	0.952738	0.963849	0.966627	0.968016	0.965238
#4	Metrics	lr:5e-2	lr:1e-1	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-2	lr:1e-3
	Acc	0.939810	0.959400	0.960300	0.959950	0.959350	0.924600	0.878080	0.875210	0.897990	0.942200	0.871320
Dia.	F-1	0.453943	0.710881	0.708927	0.732548	0.690117	0.647293	0.555013	0.550022	0.543939	0.670434	0.541369
Pred	G-Mean	0.542504	0.764060	0.752976	0.798967	0.729499	0.872160	0.885350	0.884975	0.809091	0.813986	0.879566
	B-Acc	0.647168	0.790684	0.782798	0.817291	0.765898	0.874277	0.885407	0.885066	0.815235	0.826107	0.879686

Epochs: 1,000
Settings for SOL = S_A(cosine/mu:0.5/delta=0.1) / S_F(cosine/mu:0.5/delta=0.1) / S_B(cosine/mu:0.5/delta=0.1) / Epoch: 100

Table 18: Results in MLP for the 4 datasets.

Data	Eval.	MSE	BCE	L_A	L_{F_1}	$L_{F.5}$	L_{F_2}	L_G	L_B	S_A	S_F	S_B
#1	Metrics	lr:1e-3	lr:5e-3	lr:4e-3	lr:1e-3	lr:5e-3	lr:5e-3	lr:5e-3	lr:5e-3	lr:1e-3	lr:1e-3	lr:2e-2
	Acc	0.920300	0.923900	0.924500	0.916500	0.924900	0.877700	0.834700	0.876400	0.923100	0.914600	0.827200
Ran.	F-1	0.527334	0.590095	0.578506	0.636631	0.584536	0.594422	0.540051	0.591280	0.557660	0.636488	0.531008
Gen.	G-Mean	0.650637	0.712684	0.694489	0.811241	0.703328	0.867475	0.866778	0.859828	0.672888	0.818277	0.856774
	B-Acc	0.711984	0.747529	0.735166	0.820982	0.742470	0.868011	0.868504	0.861911	0.720869	0.826697	0.857939
#2	Metrics	lr:5e-4	lr:5e-3	lr:3e-3	lr:3e-3	lr:3e-3	lr:3e-3	lr:3e-3	lr:5e-3	lr:5e-3	lr:1e-2	lr:5e-3
	Acc	0.990567	0.993532	0.993997	0.995066	0.992155	0.994369	0.977419	0.990741	0.992095	0.994868	0.981027
Cre.	F-1	0.888571	0.928710	0.933364	0.946277	0.910775	0.940188	0.804384	0.904573	0.910032	0.944155	0.821155
Card	G-Mean	0.897839	0.940434	0.941921	0.955223	0.916897	0.962974	0.960083	0.956059	0.916242	0.954546	0.948832
	B-Acc	0.904099	0.942173	0.943752	0.956211	0.920334	0.963597	0.960347	0.956882	0.919734	0.955573	0.949542
#3	Metrics	lr:1e-3	lr:5e-3	lr:5e-3	lr:1e-2	lr:1e-2	lr:3e-3	lr:1e-2	lr:5e-3	lr:1e-3	lr:1e-2	lr:5e-3
	Acc	0.982244	0.994872	0.994936	1.000000	0.994936	1.000000	0.994872	0.969872	0.984679	0.992372	0.982179
Bre.	F-1	0.882857	0.965714	0.974603	1.000000	0.965714	1.000000	0.968889	0.910551	0.890317	0.954603	0.906984
Can.	G-Mean	0.891359	0.968252	0.985164	1.000000	0.968252	1.000000	0.980211	0.982366	0.911151	0.966813	0.973177
	B-Acc	0.900000	0.970833	0.986071	1.000000	0.970833	1.000000	0.981905	0.983294	0.923571	0.969405	0.974921
#4	Metrics	lr:1e-3	lr:5e-3	lr:5e-3	lr:3e-3	lr:5e-3	lr:3e-3	lr:1e-2	lr:7e-2	lr:1e-2	lr:2e-2	lr:3e-2
	Acc	0.955990	0.960370	0.961340	0.960310	0.960760	0.938710	0.862870	0.926500	0.961200	0.960130	0.862430
Dia.	F-1	0.654638	0.716911	0.724130	0.736366	0.703492	0.691992	0.536563	0.662707	0.724056	0.735649	0.519180
Pred	G-Mean	0.714728	0.766348	0.770847	0.803085	0.739861	0.877184	0.887109	0.874578	0.771942	0.803348	0.864478
	B-Acc	0.765876	0.792601	0.796173	0.820582	0.773552	0.880067	0.888034	0.877450	0.796950	0.820751	0.865170

1 Hidden Layer (2 nodes) / Activation: Sigmoid / Epochs: 100 / Batch Size: train data size × 5e-2 (5e-1 for L_G & L_B)
Settings for SOL = S_A(cosine/mu:0.5/delta=0.1) / S_F(cosine/mu:0.5/delta=0.1) / S_B(cosine/mu:0.5/delta=0.1)