IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 16, 2018, accepted December 9, 2018, date of publication December 21, 2018,
date of current version January 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2889061

Siamese Dense Neural Network for Software Defect Prediction With Small Data

LINCHANG ZHAO[1], (Student Member, IEEE), ZHAOWEI SHANG[1], (Member, IEEE), LING ZHAO[2], (Member, IEEE), ANYONG QIN[1], (Student Member, IEEE), AND YUAN YAN TANG[3], (Fellow, IEEE)

[1]College of Computer Science, Chongqing University, Chongqing 400030, China
[2]United Imaging (Guizhou) Healthcare Co., Ltd, Guiyang 550002, China
[3]Faculty of Science and Technology, University of Macau, Macau 999078, China

Corresponding author: Zhaowei Shang (szw@cqu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 91118005, in part by the Macao Special Project of the State Ministry of Science and Technology under Grant 2015DFM10020, and in part by the Graduate Research and Innovation Foundation of Chongqing, China, under Grant CYS18006.

ABSTRACT Software defect prediction (SDP) exerts a major role in software development, concerning reducing software costs and ensuring software quality. However, developing an accurate SDP model is still a severe and challenging task with the lack of training data. Fortunately, Siamese networks are powerful for learning a few samples and have been perfectly used in other fields. This paper explores the advantages of Siamese networks to propose a novel SDP model, Siamese dense neural networks (SDNNs), which integrates similarity feature learning and distance metric learning into a unified approach. It mainly includes two phases: model building and training. To be more specific, it means building the novel SDNN for capturing the highest-level similarity features and training the model to realize prediction through the designed contrast loss function with cosine proximity. Importantly, we extensively compared the SDNN approach with the state-of-the-art SDP approaches utilizing 10 software defect datasets. The experimental results show that our SDNN is a competitive approach and is able to improve the prediction performance more significantly compared with the benchmarked approaches.

INDEX TERMS Siamese dense neural networks, deep learning, metric learning, few-shot learning, software defect prediction.

I. INTRODUCTION

Software defect prediction (SDP) is utilized to classify software defect modules into defect or non-defect [1], which can be regarded as a binary classification issue. This is a significant task in the process of software maintenance that can be utilized to improve the quality and reliability of software. However, SDP is an extremely costly task that has traditionally been executed manually by development team members [2]. In recent years, deep learning methods have been applied to develop a classification model according to historical data (trainset), and utilized this model to recognize on test data (testset).

However a conventional deep learning method generally requires that there is enough training data for establishing a prediction model. For a software project with limited data (also known as small data), we may not be able to build an effective SDP model [3]. Especially in the early stages of software testing, there were usually not enough software defect data for defect detection.

In order to build a reliable SDP model with limited data, Chen *et al.* [4] use a double transfer boosting (DTB) algorithm to extract the most alike samples from cross-company data as train-set. Their methods may bring new sample redundancy to the training data. After that, Yu *et al.* [1] use feature matching algorithm to improve the accuracy of AUC value by converting the heterogeneous features into the matched features. Although these methods are very ingenious and well thought, their limited effectiveness makes it difficult to meet the growth needs of software business.

This paper considers this to be a few-shot learning problem about the limited defect data, for more details of the few-shot learning readers can refer to [5] and [6]. In 1993, Bromley *et al.* [7] designed the Siamese networks based on human beings who are good at the observation and comparative learning from similar objects. And Koch [5] and Neculoiu *et al.* [8] verified that this Siamese networks are suitable for few-shot learning where a little data is available.

Inspired by the framework of Siamese networks, for a software project with limited data, this paper proposes a novel SDP model, called Siamese Dense neural networks (SDNN), which integrates similarity feature learning and distance metric learning into a unified approach. The designed metering function is used to learn the distance metrics between the highest-level similarity features. And the contrastive loss function with cosine-proximity is proposed to train this network in an end-to-end manner to guarantee the compatibility between learning features and distance measures [9].

To summarize the work of this paper, the key contributions are three-folds:

(1) A novel SDP model called Siamese Dense neural networks (SDNN) is proposed to recognize the defects of a software project with limited data, which classifies the software defects through learning similar or dissimilar information between sample pairs.

(2) A metering function is introduced to learn the distance between the highest-level similarity features and an end-to-end SDNN prediction is developed successfully according to the designed contrastive loss function with cosine-proximity.

(3) We compared the performance of our SDNN method with the state-of-the-art SDP methods on 10 releases of software defect datasets, and the results indicate that SDNN is an effective SDP model on the lack of sufficient software defect data.

The remainder of this paper is designed as: Section 2 is expressed the related work. Next, the proposed methodology is elaborated in detail. Section 4 gives the experiments and results. Section 5 depicts the discussion. Section 6 illustrates conclusion.

II. RELATED WORK

A. SOFTWARE DEFECT PREDICTION

SDP as a sort of technology which can expose the possibility whether a software system includes defects through analyzing the metric data of software [1]. To be more specific, the relationship between metric data and software defects is constructed through SDP. Besides, more and more software metrics are being introduced into SDP [3].

McCabe [10] used method-level metrics for SDP; change information metrics [11]; package modularization metrics [12]; lines of comment metrics [12]; code size and complexity metrics [13] and deep representations metrics [3] are some of the software metrics used for SDP.

According to literature [14], a typical SDP process usually consists of four steps. The first step is to collect entities from software modules and tag them as defect or non-defect. The second step is to extract features based on previous ones of the software metrics used for SDP. The third step is to use the extracted data to train a predictive classifier by various machine learning algorithms. Finally, new instances are fed into the trained classifier, which can predict whether the instances are defects or non-defects. Moreover, an accuracy SDP model is developed to help manager to obtain more information about software defects.

B. FEW-SHOT LEARNING

Deep learning is one of the most popular research fields at present, and it has been widely used in many domains of science and proved to be very effective, especially in the predictions and classifications of objects [15], [16]. However, conventional deep learning methods need a large number of labeled data and lots of iterations to train their great quantity parameters [17], [18]. This limits their scalability to new categories owing to exegeses cost, but more basically limits their availability to rare classes where large amounts of annotated data may not exist at all [19], [20]. Thus, these methods fail to work well on one or few examples [21], [22].

In contrast, humans are very good at identifying objects with little direct supervision or none at all, called few-shot learning [23], [24]. For example, children can sum up the concept of "cat" from a single painting in a book without any problem, or by hearing its depictions as looking like a cat [25]. Due to the inspiration from the few-shot learning ability of humans, there has been the renewed interest in machine few-shot learning recently and the Siamese networks for few-shot learning where a little data is available were proposed [7], [23], [24].

Few-shot learning aims to discriminate novel visual classes from very few tagged examples, which often resolves training into an assistant meta learning step where transferrable knowledge is achieved in the form of good initial strategies [26], [27]. The goal of few-shot learning is then learned by fine-tuning with the learned optimization strategy. And Siamese networks as a representative method of few-shot learning has been applied in many fields [8], [28].

C. SIAMESE NETWORKS

Siamese networks were primary proposed by Bromley *et al.* [7] in the 1990s to confirm signature confirmation as an image comparative learning problem. This neural network is a structure for non-linear metric learning which naturally learns expressions by two identical subnetworks. More clearly, the two sub-networks can naturally learn and extract the representation of input pairs through similarity and dissimilarity information [29].

Koch [5] and Neculoiu *et al.* [8] gave a detailed introduction to the structure and properties of the Siamese networks, and pointed out that this network is a kind of twin framework with containing two or more identical subnetworks. Every subnetwork has the same parameters and weights. The parameters of Siamese networks are updated by jointly performed on all subnetworks. Moreover, they have confirmed that the Siamese networks are good at learning with a little data is available [5], [8], [28].

Such as Neculoiu *et al.* [8] used Siamese Recurrent neural networks to learn the text similarity; Siamesed fully convolutional networks for road detection [28]; and Siamese LSTM networks were used to investigate text categorization [30]. These networks not only have the powerful ability of learning with less samples, but also score the similarity

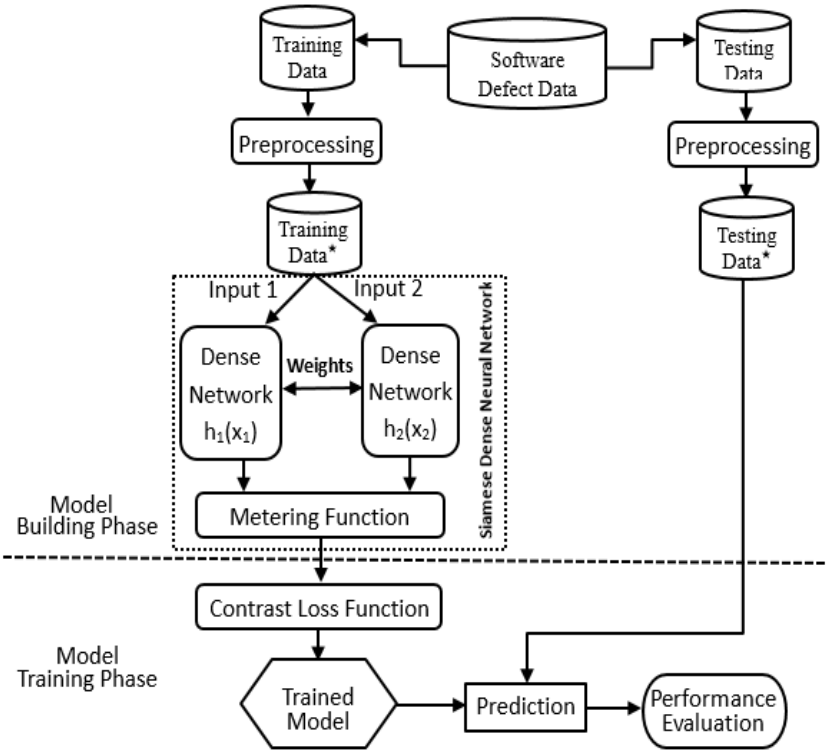


FIGURE 1. The flowchart of the proposed SDNN for SDP.

of inputting pairs according to the similarity learning. In this study, we investigated the advantage of the Siamese networks to design a novel SDP model for a software project with limited data.

III. METHODOLOGY

A. OVERALL ARCHITECTURE

According to literature [31], the NASA repository has been most popularly utilized for SDP. The paper implements experimentations on ten different baselined datasets extracted from NASA repository. However, these datasets are limited historical data, duplicated instances, the class-unbalanced distribution, and metrics of every entity having global connections. Therefore, Shepperd *et al.* [32] have pointed out the quality issues of these datasets. Inspired by their proposal to solve those problems, the data preprocessing is presented in the section B.

Meanwhile we also analyzed the structure of deep-learning models, such as RNN, CNNs, LSTM, and Dense neural network (DenseNN). CNNs is good at multi-dimensional data analysis and local feature extraction [33]. LSTM and RNN are good at sequential data with changing over time [33]. DenseNN has very high global efficiency because it combines all local features into the highest-level features and its predictor uses features of all complexity levels to calculate the score for every class [34].

For the software project with small data, the SDNN model by twin Dense neural networks is proposed on the basis of combining with the characteristics of experimental data and models. Also, some other networks such as RNN, CNNs, and LSTM are used in experiments, but results from DenseNN are the best. And then the framework of SDNN method is expressed in Figure 1.

B. DATA PREPROCESSING

1) DELETION OF REPEATED ENTITIES

Repeated entities are the same software measures and class labels in the software modules and this situation will bring about serious negative effects on machine learners [35]. To be more precise, if duplicate instances are normally divided into a portion of the test data, they will generate over-optimistic performances; otherwise, over-pessimistic [36].

2) REPLACEMENT OF MISSING VALUES

Every entity in the dataset is composed of the multiple software metrics. If there is one or more missing values in an entity, this instance cannot fulfill the input requirement of our method. We must deal with these missing values. In this paper, the average of corresponding metric is used to substitute for a missing value, and this approach is introduced detailedly in [37].

3) DATA NORMALIZATION

Owing to the fact that most software metrics have different orders of magnitude, we normalize the experimental data for convenience of experiment and operation. In this study,

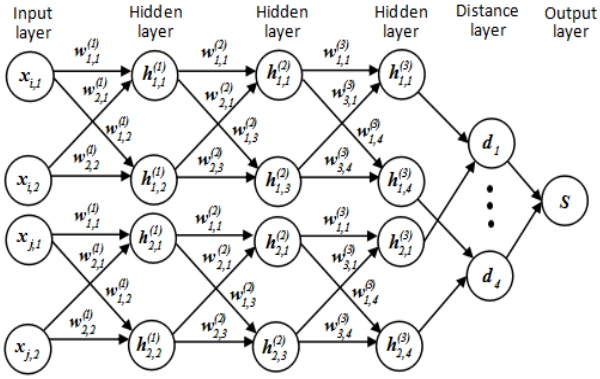


FIGURE 2. A simple 3 hidden layers Siamese networks for binary classification with metering function.

we normalize the data using the most common minimum-maximum normalization method. This approach is described in detail [38].

4) DATA OVERSAMPLING

Software defect datasets are often composed of a few defect modules and most defect-free modules, and this unbalanced distribution brings difficulties for machine learning [4]. Fortunately, this problem has been settled at data level or algorithm level. In this study, we use the Synthetic Minority Oversampling Technique with Tomek-links (SMOTETomek) proposed by Batista *et al.* [39] to process the class imbalance distribution at data level. This method can not only synthetically generate new artificial minority class instances according to feature similarities, but also remove noise data and delete redundant information based on the regional data. Also, some other resampling techniques such as random oversampling and undersampling are used in experiments, but results from SMOTETomek are the best.

C. MODEL BUILDING PHASE

1) THE SDNN MODEL

This study proposes SDNN model consisting of two identical fully-connected networks which respectively process batch-size the instances in given pairs. This model accepts distinct inputs and it is joined by a metering function at the top. This metering function measures the distance d_i between the highest-level similarity features h_1 and h_2 on each side. Figure 2 shows a simple three hidden layers Siamese networks for binary classification with a metering function $s = \sum_{i=1}^{n} d_i$, n is the number of attributes.

This SDNN has a total of 4 layers, including three hidden layers and a distance layer. The $h_{1,l}$ represents the hidden vectors in layer l for the first Siamese, and $h_{2,l}$ expresses the same for the second Siamese. The distance layer at the top contains the metering function on the learned feature space and scores the similarity between the two feature vectors.

Figure 3 shows the SDNN is learning the highest-level similar features of software metrics in the training procedure. Figure 3(a) represents the first layer training procedure. This research uses inputs of a pair of data and obtains the comparison values of similarity features. If the input $x_{i,n}$ and $x_{j,n}$ belong to the same class of data, the value is low, otherwise high. Furthermore, we can adjust the network parameters to get the best values of features. Then we echo this procedure via layer-by-layer training and acquire all the comparison values of similarity features of the three layers. Figure 3(a)-(c) express these procedures. Finally, we can obtain all trained parameters and use these information to predict the software defected by comparing similarities or dissimilarities between pairs of data. This procedure is known as end-to-end learning, and the highest-level similarity features are obtained automatically. Figure 3 shows the entire fine-tuning process.

In addition, we also tried to expand or reduce the number of hidden layers in the lab, and we get the best results from 3 hidden layers, which is the best model through experiments. Moreover, this model is symmetric so that whenever we enter the pairs of data from different category into the twin networks, the top conjoining layer will calculate the distant metric. More importantly, the Siamese architectures are tied weights, which means that all parameters between the Siamese networks are shared. Weight tying ensures that the pairs of data could not possibly be mapped to very different locations in feature space by their respective network using the same function [40].

2) METERING FUNCTION OF THE SDNN

The SDNN top is a distance layer with a metering function. When input to the Siamese networks are pairs of data and labels, these data are processed by the sub-networks, and producing the outputs through the metering function [41]. If the value of s is smaller than 0.5, it is assumed that a pair of data is of the same class; otherwise, inhomogeneity. The metering function is defined in detail below.

3) EUCLIDEAN DISTANCE

If X_1 and X_2 are a couple of input vectors expressed to the learning, w represents shared parameter vector, and the mapping of X_1 and X_2 in the feature space is represented by $H_w(X_1)$ and $H_w(X_2)$. Then the Siamese Dense neural networks can be regarded as a scalar metric function $D_w(X_1, X_2)$ to measure the compatibility between X_1, X_2, and the distance learning is defined as (1):

$$D_w(X_1, X_2) = ||H_w(X_1) - H_w(X_2)||_2 \tag{1}$$

where $D_w(X_1, X_2)$ represents the Euclidean distance to learn the distance metric of similarity features from input pairs of data.

4) SIMILARITY METRIC

Since the Euclidean distance mainly estimates the distance metric of samples, the intra-pairing changes of the samples are ignored. This paper introduces a similarity measure function expressed by cosine-proximity function [42] to strengthen the discriminative power of the learned similarity

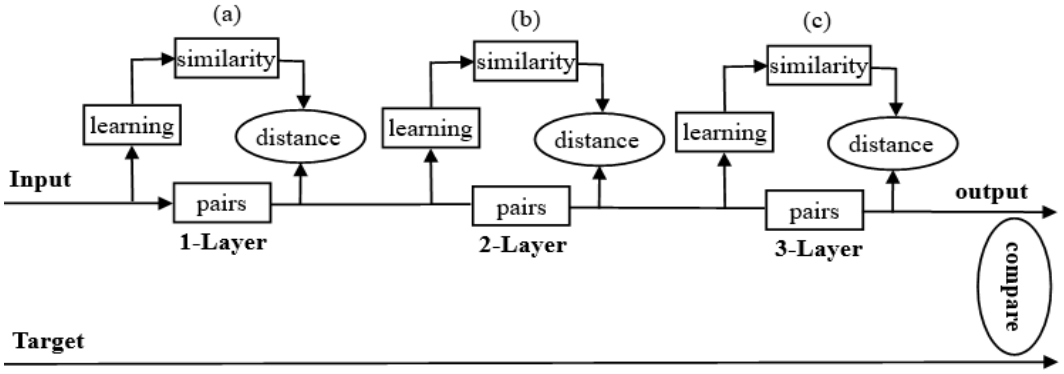


FIGURE 3. Plots of the features learn process for the SDNN.

features. Also, some other measure function such as maximum likelihood function [9] and Manhattan function [40] are used in experiments, but results from cosine-proximity are the best.

The similarity metric function is defined as follow(2):

$$L_i(w, y, X_1, X_2) = \frac{\sum_{i=1}^{N} y^{(i)} * D_w(X_1, X_2)^{(i)}}{\sqrt{\sum_{i=1}^{N}(y^{(i)})^2} * \sqrt{\sum_{i=1}^{N}(D_w(X_1, X_2)^{(i)})^2}} \tag{2}$$

The $L_i(w, y, X_1, X_2)$ ranges between [-1, 1]. If the value of $L_i(w, y, X_1, X_2)$ is closer to 1, designating that the similarity between two features is higher. But, in this study, to strengthen the distinguishable power of the learned similarity features, the opposite value of $L_i(w, y, X_1, X_2)$ is used. By applying a scalar metric function with the cosine-proximity, the final metering function of the distance learning is assigned to(3):

$$L_{end} = -\alpha L_i(w, y, X_1, X_2) + D_w(X_1, X_2) \tag{3}$$

where α is chosen to multiply $L_i(w, y, X_1, X_2)$ to a more suitable scale.

D. MODEL TRAINING PHASE

1) CONTRAST LOSS FUNCTION USED FOR TRAINING

Let y is a binary label, $y = 0$ if a pair of data(X_1,X_2) belongs to the same class and $y = 1$ if it is deemed inhomogeneity. Since the SDNN model adopts pairs of data as input in the forward propagation, we impose a contrastive loss function with cosine-proximity on the binary classifier of the following form(4):

$$L_{contr}(w, y, X_1, X_2) = \frac{1}{N}\sum_{i=1}^{N}\{(1 - y_i) * (L_{end}^{(i)})^2 + y_i * (max(m - L_{end}^{(i)}, 0))^2\} \tag{4}$$

where $m > 0$ as a pre-set threshold, L_{end} is composed of the distance metric learning by Euclidean formula and similarity penalty learning by cosine-proximity, it should be devised in such a manner that the minimum of $L_{contr}(w, y, X_1, X_2)$ will decrease L_{end} when pairs of data come from same character class and increase L_{end} when pairs of data come from different class. More concisely, the minimization of $L_{contr}(w, y, X_1, X_2)$ would emerge in low values of L_{end} for similar pairs and high values of L_{end} for dissimilar pairs.

2) PARAMETER FINE-TUNING

a: OPTIMIZATION

The SDNN model has 64 cell units in each hidden layer, and applies a rectified linear (ReLU) activation function to each layer. During training, dropout and other methods are used to prevent overfitting of the model. The parameter optimization of the SDNN model is done by using the Adam algorithm proposed by Kingma *et al.* [43]. This algorithm has the advantages of both the first-moment and second-moment gradients.

b: WEIGHT INITIALIZATION

This paper uses a normal distribution with zero-mean and a standard deviation of 10^{-2} to initialize all neural network weights, and this normal distribution is also used to initialize the value of biases. All parameters of network are corrected by fixing a mini-batch size of 32 during iterative training.

c: LEARNING RATE

We have unified the learning rate of the SDNN model in the initial training stage. A callback function is used to control the learning rate when the evaluating indicator is not improving with increasing number of iteration. More precisely, when the performance of the SDNN model is not improved, the callback function will be used to slowly attenuate the learning rate.

IV. EXPERIMENTS AND RESULTS

A. DATASETS

NASA MDP repository is commonly used as the benchmarked datasets for SDP. These software projects were developed for satellite flight control, zero gravity test software, spacecraft instrumentation, white-goods product and storage management for ground data. Ten of the most used datasets

TABLE 1. The 10 different benchmarked datasets sorted in order of the name.(Imbalanced rate: downward integer of ratio between non-defects and defects.)

Dataset	Instances	Defects	Imbalanced rate	Descriptions
AR1	114	9	11	Academic
AR4	94	20	3	Academic
AR6	87	15	4	Academic
CM1	446	49	8	Open-source
MW1	366	31	10	Proprietary
KC1	2032	326	5	Open-source
KC2	513	107	3	Open-source
PC1	1024	77	12	Open-source
PC3	1475	160	8	Open-source
PC4	1380	178	6	Open-source

TABLE 2. Defect prediction confusion matrix.

	Real true	Real false
Predicted true	True positive (TP)	False positive (FP)
Predicted false	False negative (FN)	True negative (TN)

from this repository are used in this study, they are AR1, AR4, AR6, CM1, KC1, KC2, MW1, PC1, PC3, and PC4.

Our selection criteria for these ten experimental datasets are as follows:

(1) The datasets are available from public sources so that the application or validation of model can be simpler.

(2) Each selected dataset has the same metrics, namely 21 most commonly used method-level metrics, so it can be applied easily without losing any metrics information.

Note that the instances in each database are limited, the maximum number of instances is 2032, and the minimum number of instances is 87. Conventional deep learning methods are difficult to extract highest-level similarity features effectively in such limited data. And each instance in the datasets is composed of independent code attributes and the label. The code attributes are on the basis of the count of statement lines, total number of branch count, etc. The more detailed illustration is supplied in [3]. Similar to [32], we prosseed the data from each dataset and the characteristics of these datasets after pre-processing are shown in Table 1.

B. PERFORMANCE INDEXES

The predictive model performance for binary classifier is usually evaluated by using a confusion matrix, the defect instances are regarded as positive (true) and non-defect instances as negative (false) [44]. Where *FP* shows the false positives number, *TN* represents the true negatives number, *TP* expresses the true positives number, and *FN* indicates the false negatives number. More details demonstrates in Table 2.

In this paper, four performance indexes according to the confusion matrix are utilized, which are introduced in detail as follows:

(1) Probability of detection (PD) is the percent of defects accurately classified in the defect class, which is a measure of integrity. The high value of *PD* for a good performance of model.

(2) Probability of false (PF) expresses the percent of non-defect entities which are incorrectly classified in the defect-free classes. The low value of *PF* usually shows good performance for a model.

(3) F-measure is a trade-off metric of balancing the performances of PD and precision. Both PD and precision of a good predictor should be high, and result in the high F-measure value.

(4) Matthews Correlation Coefficient (MCC) is whole measurement of binary classification, which incorporates all positives and negatives into the measure. Its value changes in the closed interval $[-1, 1]$. The large value of MCC for a perfect prediction and the low value of MCC for a poor prediction.

As known, AUC is another performance indicator for imbalanced datasets, that is statistically more discriminating than accuracy for unbalanced datasets. An excellent predictor with high-PD and low-PF, so that a high AUC value.

C. RESEARCH PROGRAMME

The proposed SDNN uses the two identical fully-connected networks to learn the highest-level similarity features and the metering function as the distance measure for between the highest-level features. And a binary classifier can be built through the designed contrastive loss function with cosine-proximity.

To comprehensively demonstrate the validity of proposed method, this study designs the following research questions (RQ) and answers them. These experimentations are performed in tensorflow, keras and matlab environments. To reduce the influence of the initialization and sampling-method on the performance of experimentations, every method is repeated 30 times in every dataset, and the results take the average of repeated experiments.

1) RQ1:CAN THE SDNN METHOD PERFORM BETTER THAN OTHER METHODS?

a: MOTIVATION

The primary goal of this paper is to improve the performance of SDP, thus, we analyze whether our SDNN can outstrip the benchmarked methods on small data.

b: APPROACH

The paper compares proposed method with several benchmarked methods, including DNN [45], LSTM [46], DBN [47], LR [16], BAG [48] and three models from [4], such as naive bayes (NB), transfer naive bayes (TNB) and double transfer boosting (DTB). To verify the effect of adding a cosine-proximity function, we construct a basic model by the contrastive loss function without cosine-proximity, its metering function is only composed of Euclidean formula as the distance metric, which denotes SDNN$^-$.

Furthermore, to certify the significant superiority of our SDNN compared with benchmarked methods, the Wilcoxon RankSum test [49] with 5% significance level is performed

TABLE 3. Comparative results for effect of sample size: averages of PD and PF on ten sub-datasets of MW1. First line appears the logograms of ten methods and Total Avg. is the overview average of every method on ten sub-datasets. The best performances are signed in boldface. #NI denotes the number of instances.

#NI	SDNN		SDNN$^-$		DNN		LSTM		DBN		NB		TNB		DTB		LR		Bag	
	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF
60	**0.906**	0.250	0.750	**0.225**	0.750	0.374	0.692	0.400	0.830	0.357	0.750	0.465	0.769	0.556	0.788	0.316	0.783	0.326	0.807	0.316
90	0.863	**0.163**	0.750	0.291	0.738	0.226	0.837	0.332	0.809	0.450	0.807	0.346	**1.000**	0.300	0.810	0.291	0.778	0.322	0.814	0.304
120	0.875	0.153	0.850	**0.150**	0.801	0.197	0.818	0.391	0.835	0.273	0.812	0.472	**0.888**	0.304	0.828	0.291	0.808	0.298	0.819	0.296
150	0.846	**0.134**	0.692	0.278	0.786	0.261	0.728	0.214	**0.851**	0.300	0.809	0.358	0.748	0.517	0.836	0.334	0.822	0.305	0.823	0.287
180	0.865	0.154	0.780	0.249	**0.867**	0.233	0.675	0.220	0.794	**0.132**	0.851	0.333	0.811	0.366	0.826	0.302	0.833	0.289	0.833	0.278
210	**0.875**	**0.168**	0.730	0.282	0.726	0.367	0.738	0.267	0.872	0.204	0.831	0.423	0.801	0.479	0.835	0.234	0.838	0.296	0.827	0.256
240	0.828	**0.117**	0.744	0.296	0.798	0.296	0.656	0.336	**0.883**	0.207	0.853	0.411	0.855	0.390	0.860	0.309	0.824	0.278	0.843	0.267
270	0.854	**0.124**	0.756	0.270	0.840	0.326	0.702	0.265	0.793	0.226	0.844	0.393	**0.862**	0.395	0.853	0.209	0.848	0.284	0.856	0.283
300	0.856	0.170	0.809	0.284	0.829	0.228	0.806	0.238	0.813	**0.168**	0.870	0.457	0.802	0.376	**0.870**	0.386	0.857	0.293	0.846	0.271
330	**0.902**	**0.120**	0.750	0.238	0.796	0.325	0.813	0.276	0.802	0.183	0.862	0.367	0.879	0.390	0.863	0.280	0.847	0.287	0.859	0.267
Total Avg.	**0.867**	**0.155**	0.761	0.256	0.793	0.283	0.747	0.294	0.828	0.250	0.829	0.403	0.842	0.407	0.837	0.295	0.824	0.298	0.833	0.283

to calculate p-value. And the Hedges' g [50] is utilized to testify the effect size. If the Hedges' g is 0.5, it expresses that two methods are equivalent and a balanced design. If the Hedges' g $>$ 0.5, it represents that our approach is better than the control method. If the Hedges' g $<$ 0.5, it means that our approach is worse than the comparative method.

The MW1 dataset is selected randomly from ten experimental datasets for illustration, and the selected dataset is divided into ten sub-datasets with the same imbalanced-rate (10%). Finally, the performance of SDP model trained on each of the sub-dataset is evaluated.

Besides the MW1 dataset, we also use the remaining datasets to do experiments, the results are presented in Appendix A.

c: RESULTS

Table 3 expresses results of PD and PF on ten sub-datasets of MW1. Figure 4 presents these results in a scatter plots. If a SDP method has the higher value of PD and fewer value of PF, it has more points distributed at bottom right.

From Table 3 and the figure 4, the SDNN has more points distributed at lower right and obtained the best PD (overall average is 0.867) and PF (overall average is 0.155). Although the SDNN$^-$ has the same Siamese architecture as the SDNN, after adding the cosine-proximity in metering function, the SDNN acquired a 10% reduction over SDNN$^-$ in term of PF on mean, and the PD of total average increased nearly 10% compared to previous SDNN$^-$.

Compared with the SDNN, the DNN, LSTM and DBN have not achieved superior results in terms of PD on average and PF on average, which proves that the Siamese architecture is more effective than the single-branch network in the performance of SDP. Moreover, the classical NB, LR and Bag have also not gained better results than the SDNN in terms of PD and PF. And these results are same as the Manjula's study [45] and GAN's research [47] utilized to other datasets.

The TNB acquires the second only result of the SDNN in terms of PD on average, but also gets the worst PF (overall average is 0.407). A predictor with high-PD and high-PF is valuable in some pivotal industrial conditions [51]. However,

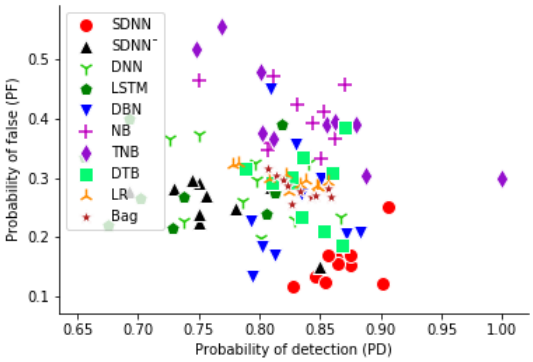


FIGURE 4. Scatter plots of ten methods on ten sub-datasets of MW1.

a high PF (PF of TNB is 40.7%, and NB is 40.3%) indicates that there is no need to allocate more attention to check the software defect modules. Besides, Menzies *et al.* [51] recommended high-PD and low-PF as the more stable performance indicators for predictive models trained on unbalanced datasets. Although the TNB method has already obtained the PD on average that is very close to the SDNN's, its PF is more than twice of the SDNN. Therefore, the SDNN method gains low-PF and high-PD, which can save time and cost to expedite SDP.

Table 4 shows the results of two balanced indicators with F-measure and MCC. The p-value is implemented to analyses the significant difference between SDNN and the benchmarked methods. The summary results are expressed in the bottom row of table, which notes how many datasets operated through the SDNN are observably different (win or lose) or no different (tie) from baselined methods.

With respect to F1, the SDNN method significantly outperforms the last five methods including DNN, LSTM, DBN, NB and LR, as it wins 10, 10, 10, 10, and 10 datasets, respectively, and loses 0 dataset; compared with the SDNN$^-$, TNB and Bag, the SDNN wins 9 datasets, ties in 1 dataset; compared with the DTB, the SDNN wins 8 datasets, ties in 1 dataset, and only loses 1 dataset. Similar observations can be gained for MCC, the only difference is that the failed dataset number is zero compared with the DTB.

TABLE 4. Comparative results for effect of sample size: averages of F-measure and MCC on ten sub-datasets of MW1. First line indicates the logograms of ten methods and Total Avg. is the overview average of every method on ten sub-datasets. Last line is a sum-up of p-value for SDNN compared with other methods. The best performances are signed in boldface. #NI denotes the number of instances.

#NI	F-measure										MCC									
	SDNN	SDNN$^-$	DNN	LSTM	DBN	NB	TNB	DTB	LR	Bag	SDNN	SDNN$^-$	DNN	LSTM	DBN	NB	TNB	DTB	LR	Bag
60	**0.876**	0.813	0.714	0.707	0.768	0.730	0.760	0.725	0.746	0.787	**0.715**	0.630	0.454	0.369	0.543	0.470	0.597	0.550	0.587	0.694
90	0.870	0.793	0.659	0.631	0.738	0.780	**0.871**	0.811	0.795	0.803	**0.703**	0.676	0.483	0.458	0.496	0.581	0.585	0.611	0.599	0.683
120	**0.902**	0.810	0.696	0.710	0.765	0.820	0.835	0.850	0.814	0.817	0.723	**0.775**	0.428	0.435	0.527	0.670	0.710	0.736	0.620	0.707
150	**0.884**	0.847	0.747	0.729	0.825	0.850	0.848	0.860	0.835	0.826	0.713	0.679	0.516	0.463	0.671	0.702	**0.726**	0.720	0.639	0.695
180	0.865	**0.891**	0.788	0.720	0.803	0.840	0.847	0.858	0.844	0.854	0.730	0.752	0.579	0.448	0.614	0.650	0.731	**0.760**	0.642	0.718
210	0.904	0.849	0.809	0.765	0.832	0.840	0.853	**0.909**	0.851	0.843	**0.759**	0.719	0.657	0.473	0.667	0.669	0.713	0.750	0.650	0.734
240	0.875	0.829	0.795	0.771	0.862	0.782	0.856	**0.877**	0.837	0.870	**0.800**	0.738	0.591	0.540	0.726	0.578	0.752	0.760	0.673	0.720
270	0.915	0.874	0.832	0.736	0.783	0.850	0.815	**0.917**	0.841	0.876	0.796	0.697	0.678	0.447	0.567	0.679	0.603	**0.802**	0.658	0.715
300	**0.924**	0.892	0.806	0.790	0.842	0.807	0.891	0.870	0.856	0.861	0.789	0.751	0.604	0.586	0.685	0.642	0.771	**0.810**	0.684	0.725
330	**0.911**	0.866	0.824	0.773	0.820	0.890	0.848	0.911	0.862	0.873	**0.805**	0.728	0.651	0.547	0.640	0.764	0.708	0.803	0.677	0.728
Total Avg.	**0.893**	0.846	0.767	0.733	0.804	0.819	0.842	0.859	0.828	0.841	**0.753**	0.715	0.564	0.477	0.614	0.640	0.690	0.730	0.643	0.712
win/tie/lose		9/1/0	10/0/0	10/0/0	10/0/0	10/0/0	9/1/0	8/1/1	10/0/0	9/1/0		9/1/0	10/0/0	10/0/0	10/0/0	10/0/0	9/1/0	9/1/0	10/0/0	9/1/0

TABLE 5. Overview of comparison results for effect of sample size: comparing the performance of SDNN with benchmarked methods overall ten sub-datasets of MW1 with p-value and Hedges' g. The boldface represents the significant better results of the SDNN with p-value <0.05 and Hedges' g >0.5, F1 stands for the F-measure.

SDNN		SDNN$^-$	DNN	LSTM	DBN	NB	TNB	DTB	LR	Bag
F1	p-value	**0.035**	**0.014**	**0.007**	**0.016**	**0.021**	**0.033**	**0.047**	**0.023**	**0.036**
	Hedges' g	**0.704**	**1.604**	**1.786**	**1.510**	**1.104**	**0.688**	**0.522**	**1.109**	**0.743**
MCC	p-value	**0.043**	**0.017**	**0.013**	**0.026**	**0.035**	**0.046**	0.051	**0.037**	**0.048**
	Hedges' g	**0.528**	**1.297**	**1.483**	**1.312**	**1.140**	**0.716**	0.484	**1.143**	**0.745**

Moreover, overview performances of SDNN method with including 10 sub-datasets of MW1 are compared with benchmarked methods on the basis of p-value<0.05 and Hedges' g [50], the summary of comparison results is shown in Table 5.

In regard to F1, SDNN is significant better than the benchmarked methods (p-values are not more than 0.05). And the Hedges' g for the last five methods are not less than 1.0 (Hedges' g = 1.604, 1.786, 1.510, 1.104 and 1.109, respectively), which is explicated as a great improvement. As to the SDNN$^-$, TNB, DTB and Bag methods, the Hedges' g = 0.704, 0.688, 0.522, 0.743, respectively, which is viewed as a medium-size effect ($0.50 < Hedges' g < 1.0$).

In terms of MCC, SDNN keeps on maintaining a great advantage in comparison with DNN, LSTM, DBN, NB and LR methods by a great improvement (Hedges' g = 1.297, 1.483, 1.312, 1.140 and 1.143, respectively), and is comparable to SDNN$^-$, TNB and Bag, as see the Hedges' g = 0.528, 0.716 and 0.745, respectively, which is also regarded as a medium effect. While, the DTB's p-value = 0.051 and Hedges' g = 0.484.

2) RQ2:HOW MUCH TIME DOES IT TAKE FOR USING SDNN?

a: MOTIVATION

We have examined the effectiveness of the proposed SDNN. Time efficiency of an approach is also a crucial indicator to evaluate whether the approach is good enough or not.

b: APPROACH

To investigate SDNN's the time efficiency, for each dataset, we run SDNN 20 times and measured the average model training time and testing time on each baselined dataset. Model training time relates to the time taken from data preprocessing to obtaining a strong classifier. The testing time is the time taken from preprocessing testing data until the class labels are predicted. We compare the training and testing time of SDNN with the benchmarked methods used in RQ1.

c: RESULTS

Tables 6 presents the model training and testing time (in seconds) respectively on each baselined dataset. From Table 6, we note that SDNN method spends the second largest average training time, i.e., 13.176 s, and the main reason why SDNN takes so much training time is that amount of time is taken in training phase to select the best parameters.

Meanwhile, we can see that the average testing time across all baselined datasets of our SDNN method and the benchmarked datasets, i.e., SDNN$^-$, DNN, LSTM, DBN, NB, TNB, DTB, LR, and Bag, are 0.029 s, 0.026 s, 0.027 s, 0.042 s, 0.023 s, 0.016 s, 0.017 s, 0.015 s, 0.018 s, and 0.020 s, respectively. Although SDNN also spends the second amount of average testing time, we believe it is still acceptable. Furthermore, we can use many technologies, e.g., parallel computing, to speed up the training of SDNN in the future.

3) RQ3:DOSE THE UNBALANCED RATE AFFECT THE PERFORMANCE OF SDNN?

a: MOTIVATION

To evaluate the stabilization of our SDNN, this paper compares the stable performance of SDP methods developing with different class-unbalanced datasets.

b: APPROACH

First, we divided each dataset into some new databases with increasing class-unbalanced rate; and then we calculated the average(μ) and standard deviation(σ) of AUC for

TABLE 6. Time (in seconds).

Dataset	Training time										Testing time									
	SDNN	SDNN$^-$	DNN	LSTM	DBN	NB	TNB	DTB	LR	Bag	SDNN	SDNN$^-$	DNN	LSTM	DBN	NB	TNB	DTB	LR	Bag
AR1	6.515	6.441	6.520	8.776	6.712	1.123	1.847	1.442	1.638	1.863	0.023	0.021	0.020	0.026	0.012	0.013	0.015	0.012	0.014	0.016
AR4	6.284	6.143	6.291	8.375	6.132	1.017	1.036	1.029	1.031	1.048	0.018	0.015	0.016	0.017	0.011	0.007	0.006	0.009	0.013	0.015
AR6	6.008	6.005	6.006	7.017	6.011	1.009	1.005	1.003	1.011	1.016	0.013	0.007	0.009	0.016	0.006	0.008	0.007	0.005	0.006	0.011
CM1	8.705	8.898	8.466	10.788	8.452	2.039	2.426	2.523	2.321	2.669	0.032	0.023	0.026	0.042	0.023	0.016	0.015	0.014	0.016	0.018
MW1	7.463	7.108	7.997	9.673	7.346	2.032	2.021	2.023	2.019	2.052	0.029	0.025	0.026	0.038	0.021	0.014	0.012	0.013	0.011	0.015
KC1	33.461	30.866	29.878	44.764	21.348	3.074	3.071	3.075	3.066	3.086	0.039	0.031	0.034	0.068	0.033	0.019	0.018	0.021	0.019	0.024
KC2	9.371	9.218	9.178	12.768	9.036	2.136	2.728	2.825	2.622	2.774	0.034	0.031	0.032	0.049	0.025	0.018	0.017	0.016	0.018	0.022
PC1	16.877	15.431	15.978	25.246	15.036	2.687	2.877	2.774	2.668	2.898	0.035	0.033	0.034	0.052	0.032	0.021	0.023	0.019	0.024	0.026
PC3	19.756	19.014	18.957	27.764	18.783	2.747	2.737	2.643	2.852	2.881	0.037	0.036	0.035	0.058	0.034	0.023	0.027	0.021	0.026	0.027
PC4	17.324	17.036	16.879	26.359	16.343	2.741	2.733	2.639	2.787	2.976	0.033	0.034	0.033	0.054	0.033	0.022	0.025	0.024	0.028	0.029
Total Avg.	13.176	12.616	12.615	18.153	11.520	2.061	2.248	2.198	2.202	2.326	0.029	0.026	0.027	0.042	0.023	0.016	0.017	0.015	0.018	0.020

TABLE 7. Mean and standard deviation of each method, and the best performance is noted in boldface.

Model	AR4		KC2		AR6		KC1		PC3		PC4		CM1		MW1		AR1		PC1	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
SDNN	**0.941**	**0.003**	0.916	**0.011**	**0.921**	**0.008**	0.892	**0.007**	**0.909**	0.011	**0.917**	**0.013**	**0.896**	0.013	**0.895**	**0.011**	0.865	**0.012**	**0.875**	0.020
SDNN$^-$	0.917	0.008	0.899	0.007	0.889	0.011	0.814	0.020	0.846	0.018	0.861	0.028	0.812	0.037	0.848	0.013	0.850	0.023	0.777	0.039
DNN	0.823	0.024	0.829	0.038	0.829	0.013	0.802	0.023	0.772	0.052	0.802	0.027	0.730	0.064	0.726	0.036	0.677	0.060	0.647	0.049
LSTM	0.813	0.028	0.817	0.043	0.802	0.033	0.744	0.037	0.717	0.051	0.710	0.035	0.695	0.043	0.651	0.036	0.622	0.053	0.607	0.056
DBN	0.827	0.027	0.794	0.039	0.817	0.031	0.839	0.039	0.722	0.054	0.779	0.038	0.711	0.055	0.709	0.035	0.664	0.063	0.639	0.057
NB	0.859	0.015	0.838	0.014	0.850	0.019	0.880	0.043	0.812	0.034	0.821	0.029	0.777	0.038	0.740	0.037	0.681	0.063	0.669	0.060
TNB	0.917	0.010	0.893	0.014	0.886	0.015	0.827	0.037	0.835	**0.010**	0.852	0.020	0.749	0.037	0.794	0.020	0.719	0.034	0.702	0.041
DTB	0.938	0.008	**0.918**	0.014	0.908	0.009	**0.897**	0.015	0.905	0.014	0.900	0.015	0.827	**0.012**	0.891	0.014	**0.884**	0.019	0.845	**0.012**
LR	0.863	0.018	0.851	0.019	0.868	0.021	0.878	0.046	0.823	0.037	0.834	0.032	0.789	0.034	0.791	0.035	0.672	0.054	0.694	0.051
Bag	0.916	0.017	0.891	0.015	0.894	0.013	0.876	0.014	0.901	0.017	0.889	0.018	0.842	0.013	0.886	0.017	0.879	0.022	0.838	0.016

TABLE 8. The CV of performance of each SDP method.

Method	AR4	KC2	AR6	KC1	PC3	PC4	CM1	MW1	AR1	PC1
SDNN	0.319	1.201	0.920	0.776	1.212	1.421	1.453	1.231	1.392	2.294
SDNN$^-$	0.866	0.794	1.243	2.461	2.132	3.250	4.561	1.532	2.713	**5.021**
DNN	2.922	4.581	1.610	2.873	**6.741**	3.372	**8.770**	4.961	**8.861**	**7.562**
LSTM	3.441	**5.257**	4.162	4.972	**7.113**	4.932	**6.192**	**5.534**	**8.521**	**9.224**
DBN	3.262	4.912	3.786	4.650	**7.483**	4.881	**7.742**	4.942	**9.491**	**8.920**
NB	1.752	1.674	2.229	4.891	4.193	3.526	4.887	4.976	**9.251**	**8.970**
TNB	1.093	1.572	1.687	4.465	1.201	2.348	4.941	2.523	4.731	**5.843**
DTB	0.851	1.519	0.987	1.672	1.546	1.667	1.448	1.572	2.153	1.422
LR	1.786	1.812	2.274	4.863	4.211	3.547	4.896	4.905	**9.327**	**8.992**
Bag	0.937	1.824	1.994	2.949	2.109	3.849	2.942	3.783	4.042	**5.608**

each method. But, the average performance of AUC is different on a same dataset, we cannot directly use the standard deviation to measure the dispersion of performance of each prediction model. So we use the ratio of between standard deviation and mean, namely coefficient of variation (CV) [52] to evaluate the stabilization of capability of each SDP model. The computation of CV is shown in formula(5).

$$CV = \frac{\sigma}{\mu} * 100\% \tag{5}$$

where σ is the standard deviations of AUC, and μ is the averages of AUC. The higher the CV is, the more unstable the performance of the method is; in other words, the greater impact of class imbalance on the performance of a SDP method.

c: RESULTS

With different class-unbalanced rates, the performance changes of each method are shown in Figure 5. On most datasets, it can be seen that the performances of the three methods(LSTM, DBN and DNN) decrease significantly with increasing class-unbalanced rate, indicating that these three methods are more sensitive to the problem of class-unbalanced distribution. While, we can also conclude that the performances of two methods(SDNN and DTB) remain relatively stable. If only from the performance level, the performance of the SDNN is significantly higher than other methods.

Table 7 presents the average (μ) and standard deviation (σ) of each method with different class-unbalanced rates, it can be seen that the average of method is different. To eliminate the influence of different averages on the comparison of performance stability, the coefficient of variation (CV) is used to evaluate the stabilization of capability of each method, and the results are shown in Table 7. The boldface is used to mark data whose coefficient of variation(CV) is greater than 5%.

Table 8 shows that the CV values of LSTM, DBN and DNN are relatively high on most datasets, indicating that the performance of these three methods are highly susceptible to the class-unbalanced distribution, while the performances of the SDNN and DTB remain relatively stable, which is consistent with the results described in Figure 5.

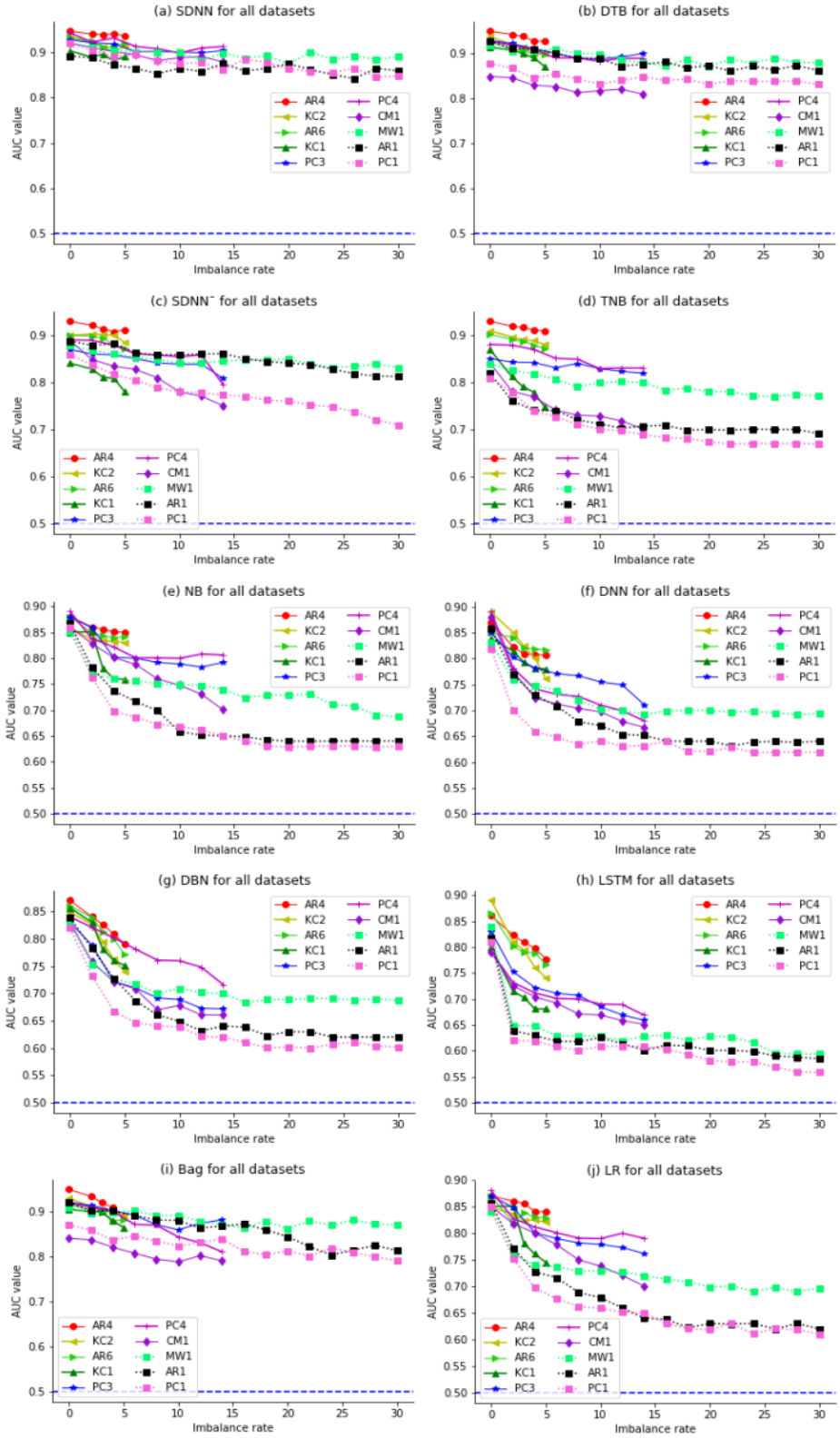


FIGURE 5. Comparison of the performance changes of each method with different unbalanced rates.

In addition, the performance of the same method on the specific dataset may be different from that its performance on other dataset. Such as SDNN^{-}, TNB and Bag methods show only slightly instability on the PC1 dataset (CV = 5.021%, 5.843% and 5.608%, respectively), the NB and LR methods exhibit instability on both AR1 and PC1 datasets. This reason

TABLE 9. Comparative results for effect of sample size: averages of PD and PF on ten sub-datasets of CM1. First line shows the logograms of ten methods and Total Avg. is the overview average of every method on ten sub-datasets. The best performances are signed in boldface. #NI denotes the number of instances.

#NI	SDNN		SDNN$^-$		DNN		LSTM		DBN		NB		TNB		DTB		LR		Bag	
	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF	PD	PF
80	0.825	**0.125**	0.800	0.281	0.791	0.449	0.835	0.463	0.801	0.286	0.807	0.420	**0.842**	0.462	0.821	0.327	0.797	0.435	0.818	0.346
120	**0.828**	**0.114**	0.794	0.191	0.816	0.249	0.821	0.451	0.810	0.301	0.811	0.290	0.792	0.438	0.818	0.268	0.804	0.309	0.826	0.271
160	0.820	**0.211**	0.807	0.238	0.781	0.380	0.710	0.362	**0.844**	0.218	0.827	0.380	0.823	0.359	0.825	0.315	0.813	0.375	0.831	0.303
200	0.832	0.235	0.819	**0.173**	0.831	0.250	0.654	0.311	0.826	0.263	0.834	0.410	**0.839**	0.376	0.808	0.215	0.844	0.394	0.830	0.243
240	0.850	**0.137**	0.822	0.193	0.812	0.331	0.627	0.290	0.831	0.206	0.863	0.407	**0.878**	0.394	0.841	0.273	0.851	0.421	0.851	0.264
280	**0.869**	0.230	0.808	**0.188**	0.840	0.242	0.787	0.338	0.811	0.302	0.835	0.384	0.863	0.308	0.824	0.234	0.827	0.391	0.834	0.261
320	**0.888**	0.163	0.815	0.215	0.870	0.231	0.688	0.384	0.835	0.294	0.853	0.218	0.849	0.336	0.861	**0.152**	0.864	0.243	0.843	0.192
360	0.873	**0.121**	0.823	0.193	0.840	0.228	0.711	0.355	0.861	0.228	0.846	0.382	0.851	0.401	**0.875**	0.175	0.854	0.378	0.861	0.189
400	**0.901**	**0.159**	0.811	0.183	0.822	0.250	0.760	0.303	0.841	0.304	0.839	0.373	0.861	0.388	0.854	0.276	0.827	0.391	0.842	0.243
440	**0.898**	**0.134**	0.816	0.175	0.870	0.291	0.743	0.355	0.851	0.402	0.850	0.381	0.844	0.359	0.880	0.180	0.844	0.376	0.859	0.192
Total Avg.	**0.858**	**0.163**	0.812	0.203	0.828	0.290	0.734	0.361	0.831	0.280	0.837	0.365	0.844	0.382	0.841	0.242	0.833	0.371	0.840	0.250

TABLE 10. Comparative results for effect of sample size: averages of F-measure and MCC on ten sub-datasets of CM1. First line indicates the logograms of ten methods and Total Avg. is the overview average of every method on ten sub-datasets. Last line is a sum-up of p-value for SDNN compared with other methods. The best performances are signed in boldface. #NI denotes the number of instances.

#NI	F-measure										MCC									
	SDNN	SDNN$^-$	DNN	LSTM	DBN	NB	TNB	DTB	LR	Bag	SDNN	SDNN$^-$	DNN	LSTM	DBN	NB	TNB	DTB	LR	Bag
80	**0.845**	0.840	0.732	0.647	0.750	0.741	0.787	0.795	0.743	0.778	**0.710**	0.704	0.574	0.565	0.507	0.590	0.632	0.702	0.578	0.676
120	0.859	**0.861**	0.761	0.674	0.800	0.812	0.812	0.806	0.798	0.790	0.702	**0.720**	0.592	0.589	0.612	0.668	0.650	0.690	0.654	0.688
160	**0.845**	0.839	0.734	0.717	0.788	0.811	0.821	0.819	0.805	0.815	0.705	0.689	0.614	0.593	0.563	0.624	0.645	**0.718**	0.611	0.674
200	0.864	**0.871**	0.771	0.695	0.795	0.817	0.802	0.834	0.816	0.824	**0.724**	0.710	0.653	0.610	0.588	0.672	0.686	0.706	0.662	0.712
240	**0.887**	0.850	0.795	0.732	0.811	0.809	0.836	0.852	0.824	0.817	0.717	0.690	0.588	0.620	0.630	0.647	0.701	**0.726**	0.653	0.702
280	**0.874**	0.835	0.807	0.731	0.810	0.821	0.854	0.845	0.819	0.832	**0.745**	0.717	0.686	0.619	0.639	0.646	0.725	0.712	0.638	0.691
320	**0.907**	0.865	0.810	0.732	0.832	0.842	0.861	0.878	0.834	0.846	0.738	0.701	0.697	0.624	0.693	0.712	**0.748**	0.733	0.697	0.678
360	**0.919**	0.855	0.821	0.731	0.825	0.869	0.871	0.864	0.829	0.853	0.729	0.707	0.707	0.602	0.730	**0.737**	0.736	0.727	0.720	0.719
400	**0.921**	0.871	0.831	0.751	0.836	0.851	0.868	0.893	0.841	0.867	**0.752**	0.720	0.680	0.636	0.715	0.746	0.714	0.737	0.732	0.689
440	**0.912**	0.866	0.820	0.749	0.854	0.845	0.872	0.904	0.847	0.892	0.746	0.705	0.685	0.629	0.750	**0.764**	0.754	0.741	0.724	0.717
Total Avg.	**0.883**	0.855	0.788	0.716	0.810	0.822	0.838	0.849	0.816	0.831	**0.727**	0.706	0.648	0.609	0.643	0.681	0.699	0.719	0.667	0.695
win/tie/lose		9/1/0	10/0/0	10/0/0	10/0/0	10/0/0	9/1/0	8/1/1	10/0/0	9/1/0		8/1/1	10/0/0	10/0/0	10/0/0	10/0/0	9/1/0	8/1/1	10/0/0	9/1/0

TABLE 11. Overview of comparison results for effect of sample size: comparing the performance of SDNN with benchmarked methods overall ten sub-datasets of CM1 with p-value and Hedges' g. The boldface represents the significant better results of the SDNN method with p-value <0.05 and Hedges' g >0.5, F1 stands for the F-measure.

SDNN		SDNN$^-$	DNN	LSTM	DBN	NB	TNB	DTB	LR	Bag
F1	p-value	0.057	**0.007**	**0.003**	**0.011**	**0.020**	**0.028**	**0.048**	**0.031**	**0.043**
	Hedges' g	0.433	**1.892**	**2.390**	**1.814**	**1.712**	**1.092**	**0.510**	**1.823**	**0.547**
MCC	p-value	**0.049**	**0.020**	**0.011**	**0.030**	**0.038**	**0.045**	0.072	**0.036**	**0.047**
	Hedges' g	**0.511**	**1.311**	**1.202**	**1.201**	**1.002**	**0.587**	0.325	**1.097**	**0.507**

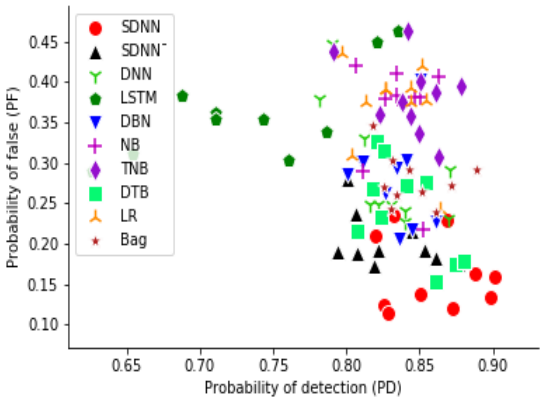


FIGURE 6. Scatter plots of ten methods on ten sub-datasets of CM1.

is that the non-defect instances is much more than that of defect entities, which seriously affects the stabilization of a SDP method.

V. DISCUSSION

A. EFFECTIVE ANALYSIS OF SDNN MODEL

The proposed SDNN approach explores the advantage of Siamese networks for learning a few samples. Compared with the benchmarked SDP methods, our proposed SDNN is more excellent for main three reasons that are summarized below.

(1) Base learner. This paper proposes the SDNN method to implement the SDP by using two identical fully-connected networks, and the twin networks have been shown to be more powerful when it comes to learning a few samples. Furthermore, the experimental results have been proved that this Siamese architecture is more effective than the single-branch network for SDP.

(2) This study proposes a metering function that composed of the Euclidean and cosine-proximity functions. Where the Euclidean function is used to learn the distance between samples, and the cosine-proximity function is utilized to

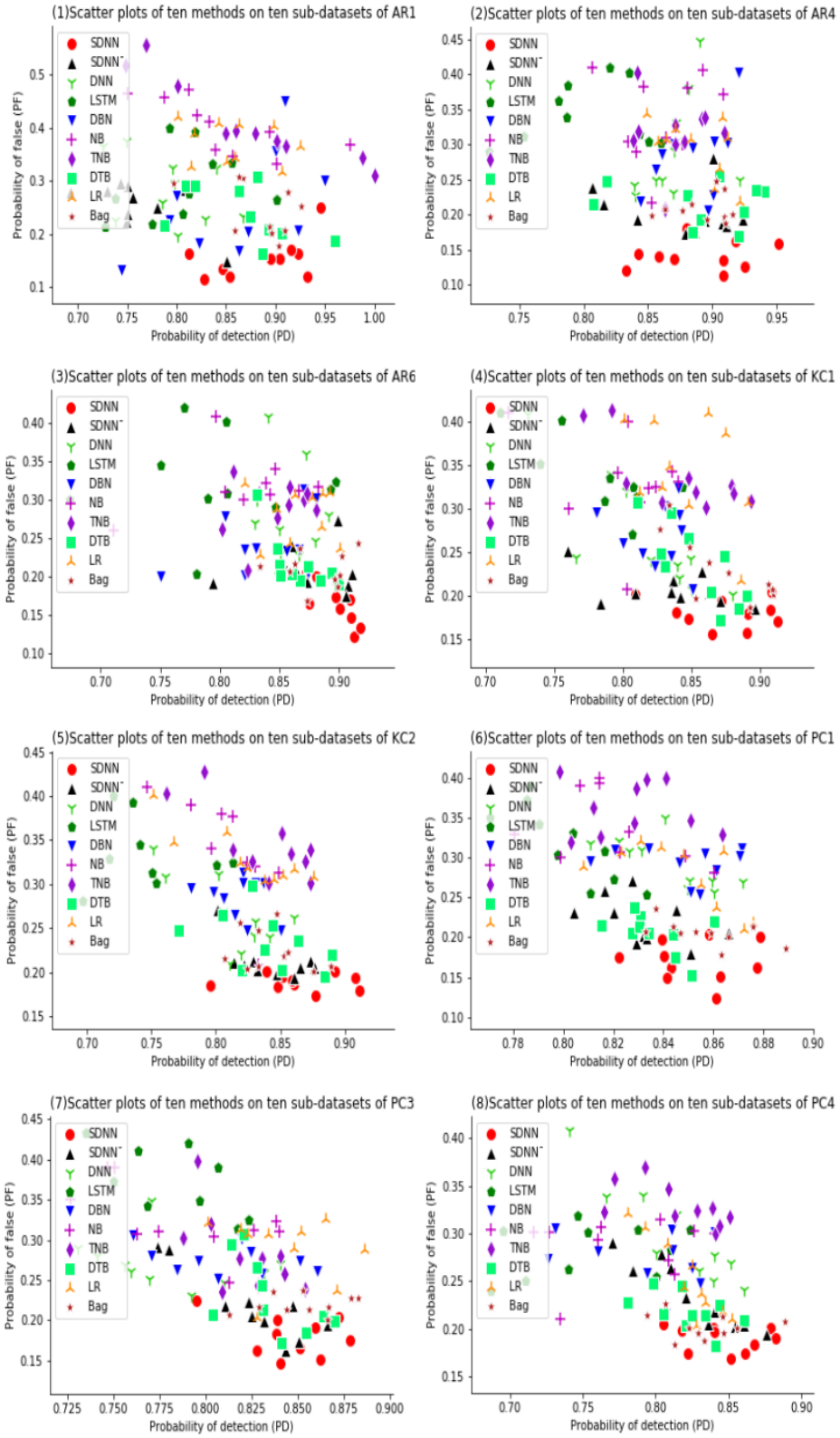


FIGURE 7. Scatter plots of ten methods on remaining datasets.

calculate the distance of intra-pairing samples. Moreover, the SDNN has a better result than the SDNN$^-$ due to the cosine-proximity in metering function.

(3) Many performance indexes. Such as PD, PF, F-measure, MCC and AUC, which have been used to assess the performance of our SDNN. Moreover, the p-value

TABLE 12. Overview of comparison results for effect of sample size: comparing the performance of SDNN with benchmarked methods remaining datasets with p-value and Hedges' g. The boldface represents the significant better results of the SDNN with p-value <0.05 and Hedges' g >0.5, F1 stands for the F-measure.

SDNN		SDNN$^-$	DNN	LSTM	DBN	NB	TNB	DTB	LR	Bag
AR1										
F1	p-value	**0.043**	**0.011**	**0.014**	**0.024**	**0.017**	**0.028**	**0.046**	**0.034**	**0.042**
	Hedges' g	**0.572**	**2.185**	**2.097**	**1.098**	**1.834**	**1.062**	**0.591**	**1.003**	**0.614**
MCC	p-value	**0.038**	**0.019**	**0.012**	**0.028**	**0.031**	**0.044**	0.068	**0.037**	**0.046**
	Hedges' g	**0.984**	**2.007**	**2.372**	**1.316**	**1.017**	**0.597**	0.406	**1.024**	**0.553**
AR4										
F1	p-value	0.053	**0.016**	**0.015**	**0.023**	**0.029**	**0.031**	**0.043**	**0.039**	**0.045**
	Hedges' g	0.491	**1.704**	**1.990**	**1.107**	**1.016**	**1.016**	**0.608**	**0.917**	**0.578**
MCC	p-value	0.054	**0.017**	**0.014**	**0.024**	**0.033**	**0.043**	0.076	**0.039**	**0.481**
	Hedges' g	0.479	**1.974**	**2.192**	**1.386**	**1.007**	**0.671**	0.403	**0.984**	**0.572**
AR6										
F1	p-value	0.062	**0.009**	**0.005**	**0.017**	**0.022**	**0.026**	0.056	**0.029**	**0.048**
	Hedges' g	0.395	**2.492**	**2.893**	**2.073**	**1.112**	**1.006**	0.432	**1.004**	**0.507**
MCC	p-value	0.057	**0.023**	**0.015**	**0.032**	**0.036**	**0.041**	0.061	**0.031**	0.051
	Hedges' g	0.434	**1.034**	**1.887**	**1.008**	**0.977**	**0.787**	0.436	**1.006**	0.486
KC1										
F1	p-value	0.053	**0.013**	**0.015**	**0.021**	**0.024**	**0.029**	**0.046**	**0.036**	**0.046**
	Hedges' g	0.493	**2.107**	**2.002**	**1.183**	**1.072**	**0.996**	**0.531**	**0.887**	**0.504**
MCC	p-value	**0.036**	**0.013**	**0.021**	**0.037**	**0.031**	**0.041**	0.058	**0.034**	**0.043**
	Hedges' g	**1.078**	**2.145**	**1.196**	**0.998**	**1.026**	**0.612**	0.398	**0.992**	**0.608**
KC2										
F1	p-value	0.052	**0.018**	**0.014**	**0.019**	**0.023**	**0.025**	**0.039**	**0.043**	**0.038**
	Hedges' g	0.481	**1.802**	**2.091**	**2.001**	**1.102**	**1.033**	**0.914**	**0.610**	**0.985**
MCC	p-value	0.051	**0.014**	**0.016**	**0.020**	**0.034**	**0.042**	0.057	**0.032**	0.053
	Hedges' g	0.498	**2.089**	**1.944**	**2.178**	**0.994**	**0.695**	0.483	**1.002**	0.492
PC1										
F1	p-value	**0.045**	**0.021**	**0.016**	**0.018**	**0.022**	**0.024**	0.056	**0.034**	**0.044**
	Hedges' g	**0.585**	**1.178**	**2.108**	**2.004**	**1.108**	**1.041**	0.422	**0.988**	**0.602**
MCC	p-value	**0.036**	**0.019**	**0.016**	**0.024**	**0.035**	**0.039**	0.054	**0.031**	**0.046**
	Hedges' g	**0.869**	**1.998**	**2.006**	**1.067**	**0.866**	**0.911**	0.485	**1.055**	**0.521**
PC3										
F1	p-value	**0.042**	**0.015**	**0.016**	**0.023**	**0.021**	**0.036**	0.055	**0.037**	**0.046**
	Hedges' g	**0.694**	**1.997**	**2.005**	**1.104**	**1.191**	**0.867**	0.476	**0.776**	**0.512**
MCC	p-value	0.054	**0.019**	**0.017**	**0.026**	**0.028**	**0.035**	0.068	**0.041**	**0.045**
	Hedges' g	0.482	**1.989**	**2.086**	**1.001**	**0.994**	**0.898**	0.304	**0.707**	**0.587**
PC4										
F1	p-value	0.056	**0.012**	**0.017**	**0.025**	**0.028**	**0.032**	0.059	**0.039**	**0.046**
	Hedges' g	0.429	**2.132**	**2.064**	**1.014**	**1.008**	**0.998**	0.403	**0.904**	**0.528**
MCC	p-value	0.056	**0.018**	**0.014**	**0.028**	**0.031**	**0.043**	0.054	**0.043**	0.052
	Hedges' g	0.423	**1.793**	**2.063**	**1.056**	**1.012**	**0.596**	0.476	**0.601**	0.493

<0.05 is used for the win-tie-loss analysis, the Hedges' g is utilized to evaluate the effect size, and the coefficient of variation (CV) is used for assessing the stabilization capability of the SDNN. All the results indicate that SDNN is an effective SDP approach for a software project with limited data.

B. VALIDITY THREATS

As with each empirical experiment, our results typically include a discussion of three different types of threats to validity, which will be analyzed below.

The threat to internal validity involves the influence of some uncertain factors on the experimental results, such as the initialization of model parameters and the quality of baseline datasets [53]. Although we have ensured the rationality of the experiments, there may still be some mistakes.

External validity is related to the summary of our research results. This paper validated the SDNN method on open datasets from NASA repository, and gained some meaningful and valuable finds. However, we still cannot suppose that these finds can be expanded to practical applications owing to insufficient empirical research. Empirical study will be conducted on more defect datasets in future work to alleviate the threats to external validity.

In relation to construct validity, the performance indexes considered for our study, this paper cannot generalize the results to other types of defect metrics just by using several performance metrics. Moreover, there is no consensus on metrics for assessing the predictive performance of unbalanced data. Considering the limited data, the limited resources and the latest study on SDP, five metrics including PD, PF, F-measure, MCC and AUC are employed in this study. This may cause a few potential threats to the construct validity.

VI. CONCLUSION

Building an effective SDP model with insufficient software defect data may be a difficult thing. This paper proposes the SDNN to address this issue and verifies the validity of the proposed method. A set of experiments are executed on 10 public software defect datasets in terms of PD, PF, F-measure, MCC and AUC. Although the results of the different metrics are slightly different, statistical analysis based on these metrics tend to support the same conclusions:

(1) SDNN shows the best overall performance among all comparison baselined methods;

(2) SDNN performs obviously better than benchmarked methods with limited samples;

(3) SDNN method also achieves more stable performance than the benchmarked methods under different class imbalance distribution.

Although these conclusions show that our SDNN could be an efficient solution for SDP, we still need to do more empirical researches on more defect datasets and try to extend this method to multi-categories defect prediction in future work.

ACKNOWLEDGMENT

The authors would like to thank all reviewers for their suggestions.

APPENDIX A

Table 9 presents the results of PD and PF on ten sub-datasets of CM1. Figure 6 presents these results in a scatter plots.

Table 10 presents the results of two equilibrium indicators with F-measure and MCC on ten sub-datasets of CM1. And the results of p-value and Hedges' g presented in Table 11.

Figure 7 shows the results of PD and PF on remaining datasets with the scatter plots.

Table 12 presents the overview of comparison results for effect of sample size with p-value and Hedges' g on remaining datasets.

REFERENCES

[1] Q. Yu, S. Jiang, and Y. Zhang, "A feature matching and transfer approach for cross-company defect prediction," *J. Syst. Softw.*, vol. 132, pp. 366–378, Oct. 2017.

[2] J. Hernández-González, D. Rodriguez, I. Inza, R. Harrison, and J. A. Lozano, "Learning to classify software defects from crowds: A novel approach," *Appl. Soft Comput.*, vol. 62, pp. 1–29, Jan. 2017.

[3] H. Tong, B. Liu, and S. Wang, "Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning," *Inf. Softw. Technol.*, vol. 96, pp. 94–111, Apr. 2018.

[4] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Negative samples reduction in cross-company software defects prediction," *Inf. Softw. Technol.*, vol. 62, no. 1, pp. 67–77, 2015.

[5] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," M.S. thesis, Graduate Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2015.

[6] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales. (2017). "Learning to compare: Relation network for few-shot learning." [Online]. Available: https://arxiv.org/abs/1711.06025

[7] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, " Signature verification using a 'siamese' time delay neural network," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1993, pp. 737–744.

[8] P. Neculoiu, M. Versteegh, and M. Rotaru, "Learning text similarity with siamese recurrent networks," in *Proc. Represent. Workshop ACL*, 2016, pp. 148–157.

[9] Q. Zhang, M. Pei, M. Chen, and Y. Jia, "Vehicle verification based on deep siamese network with similarity metric," in *Proc. Pacific Rim Conf. Multimedia*, 2017, pp. 773–782.

[10] T. McCabe, "A complexity measure," *IEEE Trans. Softw. Eng.*, vol. SE-2, no. 4, pp. 308–320, Dec. 1976.

[11] A. Abuasad and I. M. Alsmadi, "The correlation between source code analysis change recommendations and software metrics," in *Proc. Int. Conf. Inf. Commun. Syst.*, 2012, pp. 1–5.

[12] H. Aman, S. Amasaki, T. Sasaki, and M. Kawahara, "Lines of comments as a noteworthy metric for analyzing fault-proneness in methods," *IEICE Trans. Inf. Syst.*, vol. E98.D, no. 12, pp. 2218–2228, 2015.

[13] K. Yamashita *et al.*, "Thresholds for size and complexity metrics: A case study from the perspective of defect density," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur.*, Aug. 2016, pp. 191–201.

[14] J. Li, P. He, J. Zhu, and M. R. Lyu, "Software defect prediction via convolutional neural network," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur.*, Jul. 2017, pp. 318–328.

[15] Y. LeCun, B. Yoshua, and H. Geoffrey, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[16] A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, "Software bug prediction using machine learning approach," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 2, pp. 78–83, 2018.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[18] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 4353–4361.

[19] W. Fu and T. Menzies, "Easy over hard: A case study on deep learning," in *Proc. Joint Meeting Found. Softw. Eng.*, 2017, pp. 49–60.

[20] S. Majumder, N. Balaji, K. Brey, W. Fu, and T. Menzies. (2018). "500+ times faster than deep learning (a case study exploring faster methods for text mining StackOverflow)." [Online]. Available: https://arxiv.org/abs/1802.05319

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[22] L. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–10.

[23] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, Apr. 2006.

[24] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," in *Proc. Neural Inf. Process. Syst.*, 2016, pp. 1–9.

[25] H. Edwards and A. Storkey, "Towards a neural statistician," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–13.

[26] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 1–11.

[27] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—A comprehensive evaluation of the good, the bad and the ugly," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.

[28] Q. Wang, J. Gao, and Y. Yuan, "Embedding structured contour and location prior in siamesed fully convolutional networks for road detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 230–241, Jan. 2018.

[29] I. Melekhov, J. Kannala, and E. Rahtu, "Siamese network features for image matching," in *Proc. Int. Conf. Pattern Recognit.*, 2017, pp. 378–383.

[30] C.-H. Shih, B.-C. Yan, S.-H. Liu, and B. Chen, "Investigating Siamese LSTM networks for text categorization," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Summit Conf.*, 2018, pp. 641–646.

[31] D. Gray, D. Bowes, N. Davey, B. Christianson, and Y. Sun, "Reflections on the NASA MDP data sets," *IET Softw.*, vol. 6, no. 6, pp. 549–558, 2012.

[32] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," *IEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1208–1215, Sep. 2013.

[33] S. Khan and T. Yairi, "A review on the application of deep learning in system health management," *Mech. Syst. Signal Process.*, vol. 107, no. 1, pp. 241–265, 2018.

[34] C.-L. Zhang, J.-H. Luo, X.-S. Wei, and J. Wu, "In defense of fully connected layers in visual representation transfer," in *Proc. Adv. Multimedia Inf. Process. (PCM)*, 2017, vol. 36, no. 1007, pp. 807–817.

[35] H. Jindal, S. S. Kasana, and S. Saxena, "Underwater pipelines panoramic image transmission and refinement using acoustic sensors," *Int. J. Wavelets Multiresolution Inf. Process.*, vol. 16, no. 3, pp. 13–50, 2017.

[36] R. Shatnawi, "The application of ROC analysis in threshold identification, data imbalance and metrics selection for software fault prediction," *Innov. Syst. Softw. Eng.*, vol. 13, nos. 2–3, pp. 201–217, 2017.

[37] S. Liu, J. Zhang, Y. Xiang, and W. Zhou, "Fuzzy-based information decomposition for incomplete and imbalanced data learning," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 6, pp. 1476–1490, Dec. 2017.

[38] C.-S. Jung, M.-Y. Kim, and H.-G. Kang, "Normalized minimum-redundancy and maximum-relevancy based feature selection for speaker verification systems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 4549–4552.

[39] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 20–29, 2004.

[40] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2786–2792.

[41] Y. Lecun and F. J. Huang, "Loss functions for discriminative training of energy-based models," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, 2005, pp. 1–8.

[42] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2005, pp. 539–546.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Comput. Sci.*, pp. 1–15, Dec. 2014.

[44] M. Shepperd, D. Bowes, and T. Hall, "Researcher bias: The use of machine learning in software defect prediction," *IEEE Trans. Softw. Eng.*, vol. 40, no. 6, pp. 603–616, Jun. 2014.

[45] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Comput.*, vol. 3, pp. 1–17, Jan. 2018.

[46] H. K. Dam *et al.* (Feb. 2018). "A deep tree-based model for software defect prediction." [Online]. Available: https://arxiv.org/abs/1802.00921

[47] G. Lu, Z. Lie, and L. Hang, "Deep belief network software defect prediction model," *Comput. Sci.*, vol. 44, no. 4, pp. 229–233, 2017.

[48] R. S. Wahono and N. Suryana, "Combining particle swarm optimization based feature selection and bagging technique for software defect prediction," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 7, no. 7, pp. 153–166, 2013.

[49] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.
[50] V. B. Kampenes, T. Dybå, J. E. Hannay, and D. I. K. Sjøberg, "A systematic review of effect size in software engineering experiments," *Inf. Softw. Technol.*, vol. 49, nos. 11–12, pp. 1073–1086, 2007.
[51] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007.
[52] J. Forkman, "Estimator and tests for common coefficients of variation in normal distributions," *Commun. Statist.-Theory Methods*, vol. 38, no. 2, pp. 233–251, 2009.
[53] C. Tantithamthavorn, S. Mcintosh, A. E. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," in *Proc. 38th Int. Conf. Softw. Eng. (ICSE)*, 2016, pp. 321–332.

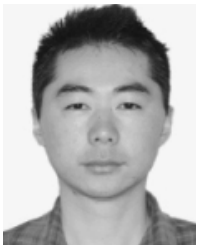
LINCHANG ZHAO received the B.S. degree from the College of Computer Science, Northeast Petroleum University, Daqing, China, in 2013, and the master's degree from the School of Mathematics and Statistics, Qiannan Normal College for Nationalities, Duyun, China, in 2017. He is currently pursuing the Ph.D. degree with the College of Computer Science, Chongqing University.

His research interests include pattern recognition, computer vision, machine learning, and deep learning. He is a Student Member of the IEEE.

ZHAOWEI SHANG received the Ph.D. degree in electronics information from Xi'an Jiao University, in 2005, and the Postdoctoral station in computer science from Chongqing University, China, in 2010. He is currently a Professor with the Department of Computer Science, Chongqing University, and a Visiting Research Fellow with the Faculty of Science and Technology, University of Macau.

He has published extensively in the IEEE TRANSACTIONS ON IMAGE PROCESSING, *Pattern Recognition*, and *Neurocomputing*. His research interests include pattern recognition, image processing, and machine learning. He is a member of the IEEE.

LING ZHAO received the B.S. degree from the College of Materials Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2005, and the master's degree from the School of Management, Fudan University, Shanghai, China, in 2008. He is currently the Manager of the United Imaging (Guizhou) Healthcare Co., Ltd. His research interests include data mining, Bigdata analyzing, and machine learning. He is a member of the IEEE.



ANYONG QIN received the B.S. degree in information and computational science from Chongqing University, Chongqing, China, in 2012, where he is currently pursuing the Ph.D. degree in computer science.

His research interests include pattern recognition, image processing, hyperspectral images, and machine learning. He is a Student Member of the IEEE.

YUAN YAN TANG (F'04) received the B.Sc. degree in electrical and computer engineering from Chongqing University, Chongqing, China, in 1966, the M.Eng. degree in electrical engineering from the Beijing Institute of Posts and Telecommunications, Beijing, China, in 1981, and the Ph.D. degree in computer science from Concordia University, Montreal, QC, Canada, in 1990.

He is currently a Chair Professor with the Faculty of Science and Technology, University of Macau, Macau, China, and a Professor/Adjunct Professor/Honorary Professor with several institutes, including Chongqing University, Concordia University, and Hong Kong Baptist University, Hong Kong. He has published over 400 academic papers and has authored/co-authored more than 25 monographs/books/book chapters. His current research interests include wavelets, pattern recognition, and image processing.

Dr. Tang is a Fellow of IAPR. He is the Founder and the Chair of the Pattern Recognition Committee in the IEEE International Conference on Systems, Man, and Cybernetics. He has served as the general chair, the program chair, and a committee member for several international conferences. He is the Founder and the General Chair of the series International Conferences on Wavelets Analysis and Pattern Recognition. He is the Founder and the Chair of the Macau Branch of International Associate of Pattern Recognition. He is the Founder and Editor-in-Chief of the *International Journal on Wavelets, Multiresolution, and Information Processing* and an associate editor of several international journals.