Computers, Materials & Continua CMC, vol.65, no.2, pp.1467-1486, 2020

Software Defect Prediction Based on Non-Linear Manifold Learning and Hybrid Deep Learning Techniques

Kun Zhu[1], Nana Zhang[1], Qing Zhang[2], Shi Ying[1,*] and Xu Wang[3]

Abstract: Software defect prediction plays a very important role in software quality assurance, which aims to inspect as many potentially defect-prone software modules as possible. However, the performance of the prediction model is susceptible to high dimensionality of the dataset that contains irrelevant and redundant features. In addition, software metrics for software defect prediction are almost entirely traditional features compared to the deep semantic feature representation from deep learning techniques. To address these two issues, we propose the following two solutions in this paper: (1) We leverage a novel non-linear manifold learning method - SOINN Landmark Isomap (SL-Isomap) to extract the representative features by selecting automatically the reasonable number and position of landmarks, which can reveal the complex intrinsic structure hidden behind the defect data. (2) We propose a novel defect prediction model named DLDD based on hybrid deep learning techniques, which leverages denoising autoencoder to learn true input features that are not contaminated by noise, and utilizes deep neural network to learn the abstract deep semantic features. We combine the squared error loss function of denoising autoencoder with the cross entropy loss function of deep neural network to achieve the best prediction performance by adjusting a hyperparameter. We compare the SL-Isomap with seven state-of-the-art feature extraction methods and compare the DLDD model with six baseline models across 20 open source software projects. The experimental results verify that the superiority of SL-Isomap and DLDD on four evaluation indicators.

Keywords: Software defect prediction, non-linear manifold learning, denoising autoencoder, deep neural network, loss function, deep learning.

1 Introduction

Software defect prediction plays an important role in software quality assurance, and it can help to allocate limited testing resources reasonably, rank the testing priority of different software modules, and improve software quality by inspecting as many

[1] School of Computer Science, Wuhan University, Wuhan, 430072, China.

[2] School of Information Science and Engineering, Qufu Normal University, Rizhao, 276826, China.

[3] Department of Computer Science, Vrije University Amsterdam, Amsterdam, 1081HV, The Netherlands.

[*] Corresponding Author: Shi Ying. Email: yingshl@whu.edu.cn.

Received: 07 May 2020; Accepted: 08 June 2020.

potentially defective software modules (such as components, files, classes) as possible before releasing the new software product [Hall, Beecham, Bowes et al. (2012)]. Nevertheless, a serious challenge that threatens the modeling process of defect prediction is the high dimensionality of defect datasets, i.e., datasets that contain excessive irrelevant and redundant features. To solve this issue, a few feature extraction methods have been proposed to alleviate irrelevant and redundant features by constructing new, combined features form the original features, which have not been thoroughly investigated in software defect prediction [Kondo, Bezemer, Kamei et al. (2019)]. In this paper, we leverage a non-linear manifold learning method - SOINN Landmark Isomap (SL-Isomap) [Gan, Shen, Zhao et al. (2014)] to extract the representative features form the original defect features by selecting automatically the reasonable number and position of landmarks, which can reveal the complex intrinsic structure hidden behind the defect data. SL-Isomap adopts the SOINN (Self-Organizing Incremental Neural Network) algorithm [Shen, Tomotaka and Osamu (2007)] to automatically select the reasonable number of landmarks, thus characterizing topological structure of defect data in the high dimensional input space. In addition, SL-Isomap also utilizes the L-Isomap (Landmark Isomap) algorithm to search low dimensional manifolds from high dimensional defect data based on selected landmarks.

At present, deep learning techniques are research hotspot in the field of artificial intelligence, and have been successfully used in many domains, such as image classification [Zhang, Wang, Lu et al. (2019)]. In this paper, in order to bridge the research gap, while taking into account the superior prediction performance of deep learning techniques, we leverage hybrid deep learning techniques-denoising autoencoder (DAE) [Vincent, Larochelle Bengio et al. (2008)] and deep neural network (DNN) to construct a novel defect prediction model named DLDD by further processing the defect features extracted by SL-Isomap. Denoising autoencoder can remove noise through training to learn true input features that are not contaminated by noise, and reconstruct a clean "repaired" input form the "corrupted" input, thus learning the reconstructed distribution by changing the reconstruction error term. The learned features not only have more robust feature representation, but also stronger generalization capability. Then we integrate these defect features processed by denoising autoencoder into the abstract deep semantic features by deep neural network. The deep neural network trained by these deep semantic features has stronger discriminative capacity for different classes [Wang, Jiang, Luo et al. (2019); Zhou, Tan, Yu et al. (2019)]. For the loss function of the entire DLDD model, we combine the squared error loss function of denoising autoencoder with the cross entropy loss function of deep neural network to reinforce the learned defect feature representation by controlling a hyperparameter θ, thereby achieving the best defect prediction effect.

The main contributions of this paper can be summarized as follows:

(1) We utilize a novel non-linear manifold learning method - SOINN Landmark Isomap (SL-Isomap) to extract the representative features from the original defect features by selecting automatically the reasonable number and position of landmarks, which can reveal the complex intrinsic structure hidden behind the defect data.

(2) Encouraged by the superior performance of deep learning techniques, we propose a

novel defect prediction model called DLDD based on hybrid deep learning techniques, which leverages denoising autoencoder to learn the reconstructed distribution and more robust feature representation by changing the reconstruction error term, and utilizes deep neural network to learn the abstract deep semantic features.

(3) For the loss function of the entire DLDD model, we combine the squared error loss function of denoising autoencoder with the cross entropy loss function of deep neural network to achieve the best performance of defect prediction by adjusting a hyperparameter.

(4) To verify the performance of SL-Isomap and DLDD, we conduct extensive experiments for feature extraction and defect prediction across 20 software defect projects from large open source datasets. We compare the SL-Isomap with seven state-of-the-art feature extraction methods, and compare the DLDD model with six baseline models contain five classic defect predictors and deep neural network. The experimental results demonstrate that the effectiveness of SL-Isomap and DLDD on four evaluation indicators.

2 Related work

Software defect prediction is a research hotspot in software engineering domain. The majority of previous studies use different machine learning methods to construct defect prediction models. Li et al. [Li, Jing, Zhu et al. (2018)] leverage a new Two-Stage Ensemble Learning (TSEL) method to conduct software defect prediction model, and the method includes two stages: ensemble multi-kernel domain adaptation stage and ensemble data sampling stage. Wang et al. [Wang, Zhang, Jing et al. (2016)]. propose a semiboost defect prediction model called NSSB based on non-negative sparse graphs, which can utilize the adaboost algorithm to boost the model performance. The experimental results demonstrate that the NSSB model can effectively address the issues of label instances inadequacy and class imbalance. Chen et al. [Chen and Ma (2015)] use six regression models to conduct extensive empirical studies, and the experimental results show that decision tree regression can achieve the best prediction performance. Lov et al. [Lov, Saikrishna, Ashish et al. (2018)] construct the defect prediction model based on Least Squares Support Vector Machine (LSSVM) associated with linear, polynomial and radial basis function kernel functions.

Different from previous studies, we leverage hybrid deep learning techniques - denoising autoencoder and deep neural network to construct a novel software defect prediction model in this paper.

3 Feature extraction based on SL-Isomap

We utilize a non-linear manifold learning technique-SOINN Landmark Isomap (SL-Isomap) [Gan, Shen, Zhao et al. (2014)] to extract the representative features form the original defect features, which can reveal the complex intrinsic structure hidden behind the defect data. SL-Isomap is a variant of Isomap [Li, Zhang, Zhang et al. (2017)], which leverages the SOINN (Self-Organizing Incremental Neural Network) algorithm to automatically select the reasonable number and position of landmarks, so as to depict topological structure of defect data in the high dimensional input space and lessen short-circuit errors.

In addition, L-Isomap (Landmark Isomap) algorithm is adopted to search low dimensional manifolds from high dimensional defect data based on selected landmarks.

The implementation process for SL-Isomap is as follows. The data points on each software project are defined as $X = \{(x_i, y_i)|i = 1,2, \dots, N\}$.

1) Select the reasonable number and position of SOINN landmarks

We utilize the SOINN algorithm to select the reasonable number and position of landmarks automatically. We first initialize the following variables: The output nodes: $O = \{x_1, x_2\}$, the number of local cumulative signals: $S_{x_1} = S_{x_2} = 1$, the thresholds: $B_{x_1} = B_{x_2} = D_{E(1,2)}$, the connection value: $C = \emptyset$, the connection age: $a_{(1,2)} = 0$. We can find the winner w_1 and second winner (second-nearest) w_2 by searching the output nodes O from the input data points $x_i (i \in [3, N])$ one by one, as shown in Eqs. (1) and (2):

$$w_1 = \underset{n_c \in O}{\operatorname{argmin}} ||x_i - n_c||. \tag{1}$$

$$w_2 = \underset{n_c \in O \setminus w_1}{\operatorname{argmin}} ||x_i - n_c||. \tag{2}$$

The x_i is a new data node and $O = O \cup x_i$ when $||x_i - n_{w_1}|| > B_{w_1}$ or $||x_i - n_{w_2}|| > B_{w_2}$, and go back to find winner again. If there is no the connection between w_1 and w_2, we need to recreate the connection and reset the connection age $a_{(w_1,w_2)}$ to 0, and assign 1 to the age of all edges in w_1 and increase the number of local cumulative signals S_{w_1} by 1, and then adjust the winner w_1 to input data x_i by a certain fraction ε and delete invalid edges and connections.

The SOINN can automatically determine the number of landmarks *n*. After updating the threshold and removing noise nodes, we can obtain the node set $O = \{n_1, n_2, \dots, n_n\}$ and Nearest neighbors of *O* in *D*, namely the landmark set $L = \{x_{l_1}, x_{l_2}, \dots, x_{l_n}\}$, which can be expressed as follows:

$$x_{l_i =} \underset{x_c \epsilon D}{\operatorname{argmin}} ||n_i - n_c||. \tag{3}$$

2) Apply MDS on SOINN landmarks

We utilize MDS (MultiDimensional Scaling) to construct matrix H_n based on the selected *n* landmarks, as shown in Eq. (4):

$$H_n = -M_n I_n M_n / 2, \tag{4}$$

$$(M_n)_{ij} = \varphi_{ij} - 1/n, \tag{5}$$

where I_n denotes the matrix of squared *G*, *G* represents landmarks-only distance matrix.

Next the *l*-dimensional coordinates of *n* landmarks are represented as the columns of matrix *U*:

$$U = \begin{bmatrix} \sqrt{\theta_1}\vec{\mu}_1^T \\ \sqrt{\theta_2}\vec{\mu}_2^T \\ \vdots \\ \sqrt{\theta_l}\vec{\mu}_l^T \end{bmatrix}, \tag{6}$$

where θ_i denotes the *i*th biggest eigenvalues of H_n, $\vec{\mu}_i$ represents the corresponding

eigenvector.

3) LMDS based on SOINN landmarks

We calculate embedding coordinates for the remaining data nodes according to the distances from the SOINN landmarks. First, we need to conduct *n* times Dijkstra algorithm to compute single-source shortest path matrix *G'(n*N)*, which denotes the approximate geodesic distance between landmarks and remaining data nodes. Second, we leverage LMDS (Landmark MDS) to generate the low dimensional embedding, the embedding of *x* can be expressed as follows:

$$\vec{x} = \frac{1}{2} U'(\bar{I}_n - I_x), \tag{7}$$

$$U' = \begin{bmatrix} \vec{\mu}_1^T / \sqrt{\theta_1} \\ \vec{\mu}_2^T / \sqrt{\theta_2} \\ \vdots \\ \vec{\mu}_l^T / \sqrt{\theta_l} \end{bmatrix}, \tag{8}$$

where I_x represents the column vector of squared distances between data node *x* and *n* landmarks (one column vector in squared *G'*), $\bar{I}_n$ denotes the average value of the column for I_n.

Finally, we utilize PCA to reorient the axes to reflect the entire distribution of $\{x_1, x_2, \ldots, x_n\}$, thus extracting the representative features X_f.

4 The proposed DLDD model

4.1 Robustness feature representation based on denoising autoencoder

Denoising autoencoder can remove noise through training to learn true input features that are not contaminated by noise, thereby reconstructing a clean "repaired" input form the "corrupted" input. We utilize denoising autoencoder to further process these defect features X_f extracted by SL-Isomap, aiming to generate more robust feature representation, which has stronger generalization capability.

Denoising autoencoder regards the corrupted data as the input and the predicted undamaged data as the output, and can learn useful information by changing the reconstruction error term. The training process of denoising autoencoder is shown in Fig. 1. Denoising autoencoder is trained to reconstruct clean data points *x* from damaged version $\tilde{x}$, which can be achieved by minimizing the loss $L = -\log P_d(x|h = f(\tilde{x}))$, where $\tilde{x}$ is the damaged version of the each defect instance *x* by the damage process $C(\tilde{x}|x)$. Denoising autoencoder can learn the reconstructed distribution $P_r(X|\tilde{X})$ from the data pairs $(x, \tilde{x})$ according to the follow training process:

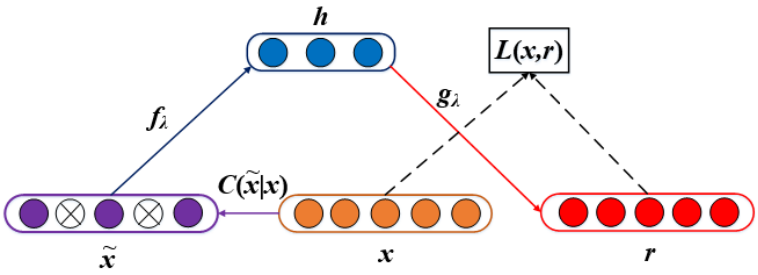


Figure 1: The training process schematic diagram of denoising autoencoder

First, given a *d*-dimensional input vector $x \in R^d$, we introduce a damage process $C(\tilde{x}|x)$ by adding Gaussian noise, the conditional distribution denotes the probability that the given defect instance X generates the corrupted instance $\tilde{X}$. Gaussian noise is a type of noise whose probability density function obeys the Gaussian distribution (i.e., normal distribution). The probability density function can be expressed as shown in Eq. (9):

$$s(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{9}$$

where σ represents the standard deviation and μ represents the expectation.

Then, we leverage the training instances $(x, \tilde{x})$ to estimate the reconstructed distribution $p_r(x|\tilde{x}) = p_d(x|h)$, which contains two stages: encoder and decoder. For the encoder stage, the *d*-dimensional Gaussian noise input data $\tilde{x}$ is mapped to the *k*-dimensional hidden layer h, as shown in Eq. (10); for the decoder stage, the hidden layer h is reconstructed to the *d*-dimensional output r, as shown in Eq. (11) .

$$h = f(W\tilde{x} + b_1), \tag{10}$$

$$r = g(W'h + b_2), \tag{11}$$

where *f(.)* and *g(.)* denote the activation function of the encoder and decoder, respectively, $W \in R^{d\times k}$ and $W' \in R^{k\times d}$ present the weight matrix of the encoder and decoder, respectively, $b_1 \in R^k$ and $b_2 \in R^d$ denote the bias of hidden layer and output layer, respectively, and the parameter of denoising autoencoder can be defined as follows: $\lambda = (W, W', b_1, b_2)$.

The parameter λ is trained to minimize the reconstruction error, as shown in Eq. (12):

$$\lambda' = \underset{\lambda}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} L(X^{(i)}, r^{(i)}) = \underset{\lambda}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} L(X^{(i)}, g_\lambda(f_\lambda(\tilde{X}^{(i)}))), \tag{12}$$

where *L(.)* denotes the squared error loss function, *N* represents the total number of training instances.

We adopt the squared error (the average reconstruction error) as the loss function of denoising autoencoder. The smaller the value, the better the performance of denoising autoencoder. The loss function L_{DAE} of denoising autoencoder is as shown in Eq. (13):

$$L_{DAE} = L(\tilde{X}, r) = ||\frac{1}{N}\sum_{i=1}^{N}(\left(g_\lambda\left(f_\lambda\left(\tilde{x}^{(i)}\right)\right)\right)^2 - \left(x^{(i)}\right)^2)||^2, \tag{13}$$

where || || is the norm of the squared error.

4.2 Feature integration based on deep neural network

We integrate these defect features processed by denoising autoencoder into the abstract deep semantic features by deep neural network (DNN). The deep neural network trained by these deep semantic features has stronger discriminative capacity for different classes (defective or non-defective). We utilize the trained deep neural network to predict whether the defect of unknown label is defective or non-defective.

According to the location of different layers, the network layers of deep neural network can be divided into three categories: input layer, hidden layer and output layer. Generally speaking, the first layer is the input layer, the last layer is the output layer, and the middle layers are all hidden layers. The neurons among various network layers are fully connected, whereas the neurons within the same layer have no direct connections. Moreover, the number of the neurons in the input and output layers are determined in accordance with specific applications, while the number of hidden layers and the number of neurons for each hidden layer are determined empirically.

The network structure of deep neural network in this paper is shown in Fig. 2. The output of the first hidden layer can be expressed as shown in Eq. (14):

$$H_k = g(\sum_{m=1}^{M} w_{mk} x_m + b_k), \tag{14}$$

where x_m presents the mth input vector, w_{mk} presents the input weight vector connecting the mth input node and the kth hidden node, b_k denotes the bias of the kth hidden node, $g(.)$ denotes the nonlinear activation function.

The output of the output layer is as follows:

$$O_j = g(\sum_{s=1}^{S} w_{sj} r_s), \tag{15}$$

where w_{sj} presents the output weight connecting the jth output node and the sth hidden node, and r_s presents the output value of the sth hidden node. O_j denotes the probability that a specific module belongs to the jth class.

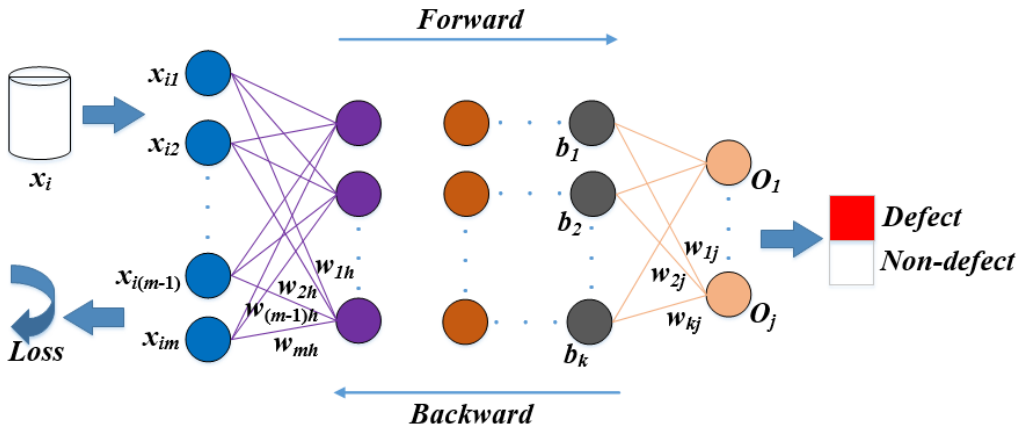


Figure 2: The network structure of deep neural network

The training process of deep neural network is mainly divided into the forward transmission of the information and the backpropagation of the loss. In the training

process of deep neural network, the loss is used for updating the network parameters (weights and biases) by gradient descent, aiming to maximize the probability of the correct class label and minimize the probability of the incorrect class label, in other words, to minimize the classification loss on the given training set. In this paper, the deep neural network adopts cross entropy loss function to train the network parameters. From the perspective of classification, it is the probability that the input instances are predicted to belong to a certain class. The smaller the cross entropy, the more accurate the prediction result. The equation for cross entropy loss function is as shown in Eq. (16):

$$L_{DNN} = -\sum_{j=1}^{C} T_j(\chi_t) \log O_j(\chi_t), \tag{16}$$

where $T_j(\chi_t)$ presents the actual probability of the *j*th input vector of the *t*th module χ_t, $O_j(\chi_t)$ presents the output probability of the *j*th input vector of the *t*th module χ_t by deep neural network, and *C* presents the number of defect classes.

4.3 Hybrid loss function for the DLDD model

For the loss function of the entire DLDD model, we combine the squared error loss function of denoising autoencoder with the cross entropy loss function of deep neural network to further reinforce the learned defect feature representation. The equation of the hybrid loss function for the DLDD model is as shown in Eq. (17):

$$L_{DLDD} = \theta L_{DAE} + L_{DNN}$$
$$= \theta || \frac{1}{N} \sum_{i=1}^{N} (\left(g_\lambda \left(f_\lambda \left(\tilde{x}^{(i)} \right) \right) \right)^2 - (x^{(i)})^2) ||^2 - \sum_{j=1}^{C} T_j(\chi_t) \log O_j(\chi_t) . \tag{17}$$

For the hyperparameter θ, $\theta \in [0, 1]$, we can adjust θ according to the experimental result. In this paper, we discuss the performance of the DLDD model when the hyperparameter θ=0.25, 0.5, 0.75 and 1, respectively. This experiment part will introduce the experimental results for different hyperparameter θ in detail.

We utilize the proposed DLDD model to learn the deep semantic features with stronger discriminative capacity for the training set. After using the defect instances with known labels to train the proposed DLDD model, the weights and biases of deep neural network will no longer change. For the defect instances with unknown labels in the test set, we feed them to the DLDD model for prediction with the same mapping rule, the class label with the highest probability manifests that the defect instance belongs to this class (defective or not-defective).

5 Experimental setup

In this section, we introduce the experimental setup, including benchmark datasets, evaluation indicators and baseline models.

5.1 Benchmark datasets

We conduct extensive experiments on 20 software projects, including 14 projects from the PROMISE data repository and 6 projects from the NASA data repository, which are publicly available and well-known datasets in software defect prediction study [Lov, Saikrishna, Ashish et al. (2018)]. Tab. 1 summarizes the basic information of 14 projects

(the first fourteen rows) from the PROMISE data repository and 6 projects (the latter six rows) from the NASA data repository respectively.

For all software projects, we adopt the SMOTE (Synthetic Minority Oversampling Technique) algorithm for class imbalance processing and the z-score method for data normalization in this paper. Moreover, we conduct 10 times 10-fold cross-validation to evaluate the performance of the models in this paper. In this paper, we adopt four commonly used evaluation indicators-F1, MCC (Matthews correlation coefficient), pf and G-measure [Zhu, Zhang, Ying et al. (2020)] to evaluate the model performance.

Table 1: The statistics of 20 projects from the PROMISE and NASA data repository

Projects	# of features	# of instances	# of defective instances	# of non-defective instances	Defective Ratio (%)	Imbalance ratio
ant-1.6	20	351	92	259	26.21	2.82
ant-1.7	20	745	166	579	22.28	3.49
camel-1.0	20	339	13	326	3.83	25.08
camel-1.2	20	608	216	392	35.53	1.81
camel-1.4	20	872	145	727	16.63	5.01
ivy-2.0	20	352	40	312	11.36	7.80
jedit-4.1	20	312	79	233	25.32	2.95
jedit-4.2	20	367	48	319	13.08	6.65
jedit-4.3	20	492	11	481	2.24	43.73
poi-2.0	20	314	37	277	11.78	7.49
prop-6	20	660	66	594	10.00	9.00
synapse-1.2	20	256	86	170	33.59	1.98
xalan-2.4	20	723	110	613	15.21	5.57
xerces-1.2	20	440	71	369	16.14	5.20
KC2	21	522	107	415	20.50	3.88
CM1	37	327	42	285	12.84	6.79
MC1	38	1988	46	1942	2.31	42.22
MW1	37	253	27	226	10.67	8.37
PC1	37	705	61	644	8.65	10.56
PC2	36	745	16	729	2.15	45.56

5.2 Baseline models

To validate the feature extraction capability of SL-Isomap and the prediction performance of DLDD, we conduct extensive experiments for feature extraction and software defect prediction. For our DLDD model, we conduct experiments respectively when the hyperparameter θ=0.25, 0.5, 0.75 and 1.

For feature extraction, we compare the SL-Isomap model with seven state-of-the-art feature extraction methods, including Factor Analysis (FA) [Ali, Ahmed, Ferzund et al.

(2017)], Principal Component Analysis (PCA) [Kondo, Bezemer, Kamei et al. (2019)]], Stochastic Proximity Embedding (SPE) [Eberhardt, Stote and Dejaegere (2018)], Stochastic Neighbor Embedding (SNE) [Bunte, Haase, Biehl et al. (2012)], Neighborhood Preserving Embedding (NPE) [Zhao, Zou and Gao (2013)], Generalized Discriminant Analysis-Gaussion (GDA-G) [Uddin and Hassan (2015)] and Isometric Mapping (Isomap) [Li, Zhang, Zhang et al. (2017)]. These feature extraction methods all use the DLDD as the defect predictor.

For software defect prediction, we compare the SL-Isomap model with five classic defect predictor, include Support Vector Machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), Decision Tree (DT) and Logistic Regression (LR). These defect predictors all use the features extracted by SL-Isomap.

In addition, we also compare the DLDD with Deep Neural Network (DNN) that does not combine denoising autoencoder (DAE), and the DNN also use the features extracted by the SL-Isomap.

6 Experimental results

We detail the experimental results by the following three research questions (RQ) in the section.

RQ1: How about the feature extraction capability of the non-linear manifold learning method SL-Isomap compared with seven state-of-the-art feature extraction methods in software defect prediction?

To verify the effectiveness of the representative features extracted by the non-linear manifold learning method SL-Isomap, we compare the SL-Isomap with seven state-of-the-art feature extraction methods with the same defect predictor-DLDD (θ=0.75), including FA, PCA, SPE, SNE, NPE, GDA-G and Isomap. In RQ2, the experiment results demonstrate that the DLDD model can achieve the best defect prediction performance when θ=0.75, so we choose the DLDD with θ=0.75 in RQ1.

Tabs. 2-4 show the F1, MCC and G-measure of SL-Isomap and seven state-of-the-art feature extraction methods across all 20 projects. Note that the highest value of each row is marked in bold. From Tabs. 2-4, we can observe that our method SL-Isomap achieves the best average performance in terms of F1, MCC and G-measure. More specifically, the average F1 (0.7957) by SL-Isomap gains improvement between 4.40% (for Isomap) and 18.46% (for NPE) with an average improvement of 8.96%, the average MCC (0.5714) by SL-Isomap yields improvement between 12.48% (for Isomap) and 89.46% (for NPE) with an average improvement of 34.83% and the average G-measure (0.7820) by SL-Isomap achieves improvement between 4.78% (for PCA) and 23.15% (for NPE) with an average improvement of 11.35%.

Fig. 3 shows the box-plots of four indicators for our method SL-Isomap and seven feature extraction methods across all 20 projects. From Figs. 3(a)-3(d), we can observe that the median values gained by SL-Isomap are higher than those gained by seven feature extraction methods from the point of F1, MCC and G-measure respectively, and the median value gained by SL-Isomap is lower than those gained by seven feature extraction methods from the point of pf, which can fully demonstrate the superiority of

our method SL-Isomap. In addition, for F1, MCC and G-measure, the median values by SL-Isomap are higher than the maximum values by SPE and NPE, respectively.

Compared with other feature extraction methods, our method SL-Isomap can achieve the best experimental results. This is because SL-Isomap can utilize the SOINN algorithm to automatically select the reasonable number of landmarks, thereby characterizing topological structure of defect data in the high dimensional input space. Moreover, SL-Isomap also leverages the L-Isomap algorithm to search low dimensional manifolds from high dimensional defect data based on selected landmarks.

Conclusion 1: Our method SL-Isomap performs better than seven state-of-the-art feature extraction methods in terms of F1, MCC and G-measure. The SL-Isomap can achieve the average 8.96%, 34.83% and 11.35% performance improvements compared with seven feature extraction methods across all 20 projects in terms of F1, MCC and G-measure. In terms of pf, the median value gained by SL-Isomap is lower than those gained by other seven methods.

Table 2: The F1 for our method SL-Isomap compared with seven feature extraction methods

Datasets	FA	PCA	SPE	SNE	NPE	GDA-G	Isomap	SL-Isomap
ant-1.6	0.7241	0.7308	0.6792	0.6909	0.6316	0.7097	0.7119	**0.7586**
ant-1.7	0.7273	0.7225	0.7018	0.7363	0.6637	0.7358	0.7437	**0.7834**
camel-1.0	0.7727	**0.8430**	0.7344	0.8331	0.7059	0.7786	0.8281	0.8105
camel-1.2	0.5443	0.5271	0.5191	0.5691	0.5042	0.5170	0.5410	**0.6014**
camel-1.4	0.6967	0.6638	0.6201	0.6824	0.5914	0.6741	0.7177	**0.7414**
ivy-2.0	0.7619	0.7576	0.7097	0.7606	0.6857	0.7302	0.7937	**0.8116**
jedit-4.1	0.7097	0.7250	0.6970	0.7213	0.6575	0.7143	0.7273	**0.7742**
jedit-4.2	0.7838	**0.8485**	0.7564	0.8125	0.7170	0.7799	0.7925	0.8344
jedit-4.3	0.8426	0.8364	0.7327	0.8436	0.7085	0.8360	0.8725	**0.8804**
poi-2.0	0.7692	0.7949	0.7105	**0.8148**	0.6933	0.7391	0.7789	0.7949
prop-6	0.7835	0.7273	0.6730	0.7396	0.6452	0.7447	0.7717	**0.8259**
synapse-1.2	0.7241	**0.7458**	0.7018	0.6984	0.6441	0.7077	0.7119	0.7368
xalan-2.4	0.7079	0.7018	0.6946	0.7019	0.6387	0.7111	0.7487	**0.7677**
xerces-1.2	0.7133	0.7448	0.6757	**0.7778**	0.6587	0.6822	0.7328	0.7682
KC2	0.7552	0.7652	0.7059	0.7581	0.6446	0.7692	0.7156	**0.7910**
CM1	0.7692	0.7714	0.7671	**0.8462**	0.7273	0.7937	0.8205	0.8421
MC1	0.8229	0.8407	0.6454	0.8161	0.7586	0.8152	0.8554	**0.8936**
MW1	0.7234	0.7458	0.7368	0.7636	0.7407	0.7547	0.7500	0.7931
PC1	0.7629	0.7735	0.7160	0.8084	0.7038	0.7631	0.8043	**0.8551**
PC2	0.8104	0.8244	0.6128	**0.8497**	0.7132	0.8178	0.8253	0.8492
Avg	0.7453	0.7545	0.6895	0.7612	0.6717	0.7387	0.7622	**0.7957**

RQ2: How about the prediction performance of the proposed DLDD model compared with five classic defect predictors with the same feature extraction

Table 3: The MCC for our method SL-Isomap compared with seven feature extraction methods

Datasets	FA	PCA	SPE	SNE	NPE	GDA-G	Isomap	SL-Isomap
ant-1.6	0.4971	0.5530	0.4559	0.4584	0.3361	0.4497	0.4687	**0.5602**
ant-1.7	0.4598	0.4867	0.4627	0.4643	0.1979	0.4142	0.4895	**0.5030**
camel-1.0	0.5342	0.7049	0.4691	0.6318	0.3810	0.5490	**0.6567**	0.6055
camel-1.2	-0.1086	0.0520	0.0226	**0.1712**	0.0736	-0.0933	0.1233	0.1342
camel-1.4	0.4207	0.3979	0.3186	0.3689	0.0591	0.3221	0.4527	**0.5317**
ivy-2.0	0.5345	0.4995	0.4426	0.4715	0.3116	0.4719	**0.5972**	0.5948
jedit-4.1	0.4370	0.3886	0.3798	0.4679	0.2180	0.3915	0.3977	**0.5621**
jedit-4.2	0.5012	0.5766	0.3779	0.5000	0.2531	0.4191	0.4523	**0.6047**
jedit-4.3	0.6468	0.6286	0.4361	0.6551	0.3182	0.6873	0.7281	**0.7387**
poi-2.0	0.4107	**0.4764**	0.2928	0.4383	0.2674	0.1240	0.2437	**0.4764**
prop-6	0.5667	0.3186	0.2759	0.4850	0.1897	0.5103	0.5789	**0.6350**
synapse-1.2	0.4971	0.5287	0.4656	0.4092	0.4184	0.4147	0.4658	**0.5290**
xalan-2.4	0.4633	0.4656	0.2140	0.3592	0.2813	0.3439	0.4898	**0.5219**
xerces-1.2	0.3526	0.4123	0.2955	**0.4936**	0.0438	0.3900	0.4789	0.4347
KC2	0.4949	0.5762	0.4505	0.5323	0.3262	0.5759	0.5163	**0.5805**
CM1	0.4152	0.5015	0.4584	**0.6130**	0.5222	0.6422	0.5471	**0.6130**
MC1	0.6457	0.6817	0.0180	0.6346	0.5648	0.6583	0.7116	**0.7928**
MW1	0.5845	0.5379	0.5290	0.5860	0.5516	0.5818	0.5551	**0.6281**
PC1	0.4590	0.4899	0.2626	0.5694	0.3307	0.5513	0.5681	**0.6792**
PC2	0.6077	0.6325	0.2900	0.6780	0.3882	0.6943	0.6391	**0.7031**
Avg	0.4710	0.4955	0.3459	0.4994	0.3016	0.4549	0.5080	**0.5714**

method SL-Isomap?

In this paper, we combine the squared error loss function of denoising autoencoder with the cross entropy loss function of deep neural network to further reinforce the learned defect feature representation by controlling the hyperparameter θ, $\theta \in [0,1]$. To investigate the influence of θ on the performance of the DLDD model, we select θ=0.25, 0.5, 0.75, 1, and discuss the performance of the DLDD model when the hyperparameter θ=0.25, 0.5, 0.75, 1, respectively. In addition, this question is also designed to evaluate the effectiveness of the DLDD model compared with five classic defect predictors with the same feature extraction method SL-Isomap, including SVM, NB, KNN, DT and LR.

Table 4: The G-measure for our method SL-Isomap compared with seven feature extraction methods

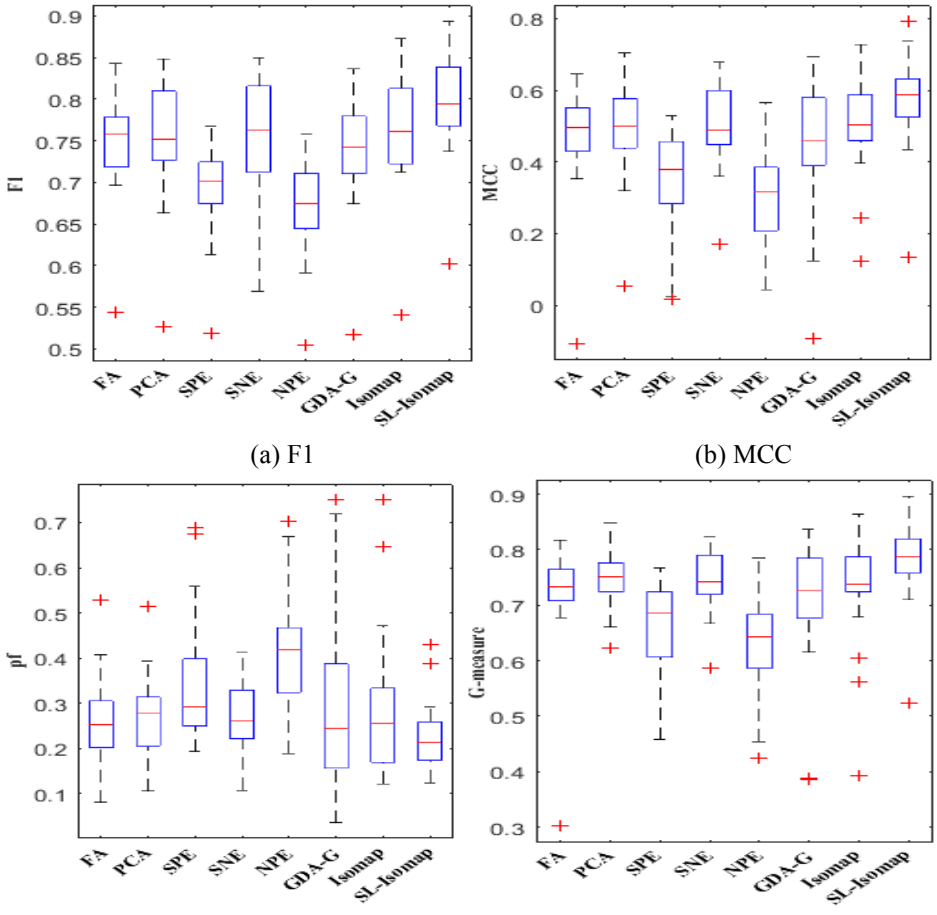
Datasets	FA	PCA	SPE	SNE	NPE	GDA-G	Isomap	SL-Isomap
ant-1.6	0.7500	0.7590	0.7151	0.7248	0.6677	0.7213	0.7358	**0.7817**
ant-1.7	0.7223	0.7231	0.7317	0.7283	0.5911	0.7093	0.7383	**0.7530**
camel-1.0	0.7639	**0.8489**	0.7344	0.8031	0.6809	0.7724	0.8281	0.8213
camel-1.2	0.3018	**0.6228**	0.5053	0.5858	0.5342	0.3878	0.5615	0.5230
camel-1.4	0.7097	0.6874	0.6473	0.6837	0.4541	0.6496	0.7264	**0.7578**
ivy-2.0	0.7648	0.7497	0.7164	0.7214	0.6457	0.7336	**0.7961**	0.7916
jedit-4.1	0.7184	0.7254	0.6851	0.7328	0.5811	0.6719	0.6048	**0.7810**
jedit-4.2	0.7569	0.7700	0.6886	0.7448	0.6154	0.7045	0.7221	**0.8098**
jedit-4.3	0.8116	0.7952	0.7182	0.8224	0.6317	0.8373	0.8640	**0.8663**
poi-2.0	0.7059	**0.7402**	0.6490	0.7168	0.6373	0.3864	0.3936	**0.7402**
prop-6	0.7818	0.6604	0.6314	0.7396	0.5785	0.7481	0.7764	**0.8180**
synapse-1.2	0.7441	**0.7625**	0.7248	0.7038	0.6878	0.7018	0.7309	0.7568
xalan-2.4	0.7219	0.7248	0.4581	0.6668	0.6406	0.6168	0.7445	**0.7591**
xerces-1.2	0.6777	0.7068	0.5769	**0.7487**	0.4237	0.6808	0.7295	0.7096
KC2	0.7006	0.7823	0.7241	0.7663	0.6630	**0.7850**	0.7378	0.7755
CM1	0.6847	0.7530	0.7293	0.7867	0.7617	0.7941	0.7529	**0.7980**
MC1	0.8174	0.8136	0.5810	0.8148	0.7847	0.8259	0.8446	**0.8963**
MW1	0.7473	0.7699	0.7671	0.7940	0.7742	0.7863	0.6786	**0.8164**
PC1	0.7132	0.7339	0.4761	0.7739	0.6534	0.7669	0.7791	**0.8331**
PC2	0.7968	0.7972	0.6354	0.8227	0.6936	0.8242	0.8125	**0.8515**
Avg	0.7195	0.7463	0.6548	0.7441	0.6350	0.7052	0.7279	**0.7820**

Tabs. 5-7 show the F1, MCC and G-measure of the DLDD (θ=0.25, 0.5, 0.75, 1) model compared with those of five classic predictors across all 20 projects, respectively. Note that the highest value of each row is marked in bold. From Tabs. 5-7, compared with DLDD (θ=0.25, 0.5, 1), we can find that the DLDD model is the best performer in terms of F1, MCC and G-measure when the hyperparameter θ=0.75. Moreover, compared with SVM, NB, KNN, DT and LR, we can observe that the proposed DLDD (θ=0.75) model also achieves the best average performance in terms of F1, MCC and G-measure. More specifically, the average F1 (0.7957) by DLDD achieves improvement between 7.14% (for DT) and 19.19% (for NB) with an average improvement of 11.31%, the average MCC (0.5714) by DLDD yields improvement between 19.97% (for LR) and 97.65% (for NB) with an average improvement of 46.55% and the average G-measure (0.7820) by DLDD achieves improvement between 8.04% (for LR) and 27.15% (for NB) with an average improvement of 15.08%.

Fig. 4 shows the box-plots of four indicators for the proposed DLDD (θ=0.25, 0.5, 0.75, 1) model and five classic defect predictors across all 20 projects. From Figs. 4(a)-4(d), we can observe that the median values gained by DLDD (θ=0.25, 0.5, 0.75, 1) are higher than those gained by five classic defect predictors from the point of F1, MCC, and G-measure, respectively. We also find that the median values by DLDD (θ=0.75) are higher than the

maximum values by NB respectively in terms of F1, MCC, G-measure, and the median value by DLDD (θ=0.75) is lower than the minimum values by SVM, KNN and DT in terms of pf. In addition, the median values gained by DLDD (θ=0.75) are higher than those gained by DLDD (θ=0.25, 0.5, 1) respectively in terms of MCC and G-measure, and the median value gained by DLDD (θ=0.75) is lower than those gained by DLDD (θ=0.25, 0.5, 1) respectively in terms of pf (the smaller the pf, the better the performance).

Compared with five classic defect predictors, our DLDD model can achieve the best prediction performance. This is because the DLDD model adopts denoising autoencoder to learn the reconstructed distribution and more robust feature representation by changing the reconstruction error term, and utilizes deep neural network to learn the abstract deep semantic features. The deep semantic features have stronger discriminative capacity for different classes. In addition, the model performance can be affected by the degree of noise added, the DLDD model can achieve the best experimental performance when θ=0.75.



(a) F1 (b) MCC

(c) pf (d) G-measure

Figure 3: The box-plots for our method SL-Isomap compared with seven feature extraction methods in terms of four indicators

Conclusion 2: The proposed DLDD model can achieve the best prediction performance in terms of F1, MCC, pf and G-measure when the hyperparameter θ=0.75. The DLDD (θ=0.75) can achieve the average 11.31%, 46.55% and 15.08% performance improvements compared with five defect predictors across all 20 projects in terms of F1, MCC and G-measure.

Table 5: The F1 for our model DLDD compared with five classic defect predictors

Datasets	SVM	NB	KNN	DT	LR	SLDD (0.25)	SLDD (0.5)	SLDD (0.75)	SLDD (1)
ant-1.6	0.6901	0.6452	0.7038	0.6870	0.7467	0.7241	0.7458	**0.7586**	0.7541
ant-1.7	0.6866	0.6585	0.6890	0.7445	0.7069	0.7562	0.7761	0.7834	**0.7905**
camel-1.0	0.7568	0.7368	0.7832	0.8048	0.7933	0.8387	**0.8444**	0.8105	0.8000
camel-1.2	0.5507	0.5263	0.6033	0.5873	0.5685	0.5793	**0.6179**	0.6014	0.5857
camel-1.4	0.6841	0.6469	0.6473	0.6900	0.6653	0.7194	0.7265	**0.7414**	0.7137
ivy-2.0	0.6912	0.6441	0.7398	0.7237	0.7380	**0.8308**	0.8235	0.8116	0.8060
jedit-4.1	0.7055	0.6914	0.7670	0.7312	**0.7973**	0.7813	0.7536	0.7742	0.7353
jedit-4.2	0.6667	0.6560	0.7621	0.7348	0.7454	0.7947	0.8153	0.8344	**0.8435**
jedit-4.3	0.7543	0.7013	0.8601	0.7814	0.8432	0.8421	0.8700	**0.8804**	0.8071
poi-2.0	0.7399	0.6609	0.7170	0.7670	**0.8072**	0.7895	0.8052	0.7949	0.7901
prop-6	0.7585	0.6914	**0.8357**	0.7622	0.7574	0.8058	0.8061	0.8259	0.8200
synapse-1.2	0.6923	0.6753	0.6475	0.6803	0.6667	0.6866	0.6913	**0.7368**	0.7143
xalan-2.4	0.6992	0.6486	0.6853	0.7071	**0.7762**	0.7385	0.7594	0.7677	0.7407
xerces-1.2	0.6277	0.6038	0.6649	0.6787	0.6316	0.7463	0.7518	0.7682	**0.7703**
KC2	0.7004	0.6753	0.7032	0.7697	0.7360	0.7778	0.7797	**0.7910**	0.7534
CM1	0.6950	0.6692	0.7063	0.8087	0.7692	0.8205	0.8312	**0.8421**	0.8267
MC1	0.7206	0.6857	0.7582	0.8221	0.7817	0.8059	0.8682	**0.8936**	0.8420
MW1	0.7162	0.7253	0.7417	0.7544	0.7379	0.7755	0.7797	0.7931	**0.8235**
PC1	0.7117	0.6769	0.7667	0.8028	0.7632	0.7673	0.8293	**0.8551**	0.8295
PC2	0.7436	0.7326	0.7835	0.8160	0.8069	0.8048	0.8163	**0.8492**	0.8473
Avg	0.6996	0.6676	0.7283	0.7427	0.7419	0.7693	0.7846	**0.7957**	0.7797

RQ3: Does the proposed DLDD (θ = 0.75) model outperform the single deep neural network that does not combine denoising autoencoder?

Denoising autoencoder can remove noise through training to learn more robust feature representation that are not contaminated by noise, and reconstruct a clean "repaired" input form the "corrupted" input. To explore the influence of the denoising autoencoder on the prediction performance of the DLDD (θ=0.75) model, we compare the DLDD (θ=0.75) model (with denoising autoencoder) with deep neural network (without denoising autoencoder) in this experiment.

Table 6: The MCC for our model DLDD compared with five classic defect predictors

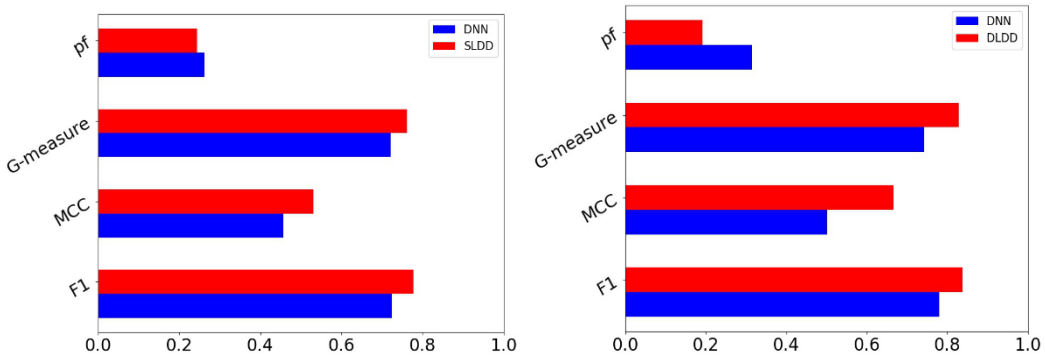
Datasets	SVM	NB	KNN	DT	LR	SLDD (0.25)	SLDD (0.5)	SLDD (0.75)	SLDD (1)
ant-1.6	0.2704	0.3594	0.3307	0.3091	0.5123	0.4971	0.5318	**0.5602**	0.5397
ant-1.7	0.3426	0.1280	0.4122	0.4677	0.4475	0.5065	**0.5488**	0.5030	0.5437
camel-1.0	0.4366	0.3881	0.5135	0.6242	0.5556	**0.6876**	0.6792	0.6055	0.6093
camel-1.2	0.0567	0.0847	0.2112	0.1943	0.2018	0.0715	**0.2652**	0.1342	0.1132
camel-1.4	0.5643	0.2659	0.2967	0.3764	0.3599	0.4084	0.4996	**0.5317**	0.4314
ivy-2.0	0.3412	0.3262	0.4648	0.4316	0.4746	**0.6566**	0.6251	0.5948	0.5932
jedit-4.1	0.3639	0.4620	0.4892	0.4104	0.5336	**0.5640**	0.4834	0.5621	0.4481
jedit-4.2	0.2599	0.3416	0.5170	0.4729	0.4591	0.5073	0.5233	0.6047	**0.6502**
jedit-4.3	0.4425	0.2909	0.7113	0.5207	0.6771	0.6547	0.7074	0.7387	**0.8222**
poi-2.0	0.4429	0.3050	0.2531	0.4892	**0.5839**	0.4880	0.5147	0.4764	0.4288
prop-6	0.5970	0.2347	**0.6390**	0.4658	0.4824	0.5813	0.6068	0.6350	0.6251
synapse-1.2	0.3511	0.2234	0.2630	0.3876	0.2599	0.3586	0.4679	**0.5290**	0.4981
xalan-2.4	0.3748	0.1863	0.3107	0.3969	0.4976	0.4690	**0.5320**	0.5219	0.4898
xerces-1.2	0.2395	0.1465	0.2824	0.3101	0.3633	**0.4854**	0.4518	0.4347	0.4555
KC2	0.3477	0.2234	0.4162	0.4971	0.5515	**0.6306**	0.5916	0.5805	0.4265
CM1	0.3232	0.3219	0.3893	0.5837	0.5057	0.5471	0.5800	**0.6130**	0.5818
MC1	0.4039	0.3116	0.4034	0.6223	0.5132	0.6062	0.7387	**0.7928**	0.6839
MW1	0.3396	0.4115	0.3775	0.4576	0.4027	0.6474	0.6011	0.6281	**0.7107**
PC1	0.5132	0.3439	0.5063	0.5607	0.5814	0.5839	0.6172	**0.6792**	0.6612
PC2	0.4658	0.4272	0.5070	0.6018	0.5626	0.6171	**0.7497**	0.7031	0.6899
Avg	0.3738	0.2891	0.4147	0.4590	0.4763	0.5284	0.5658	**0.5714**	0.5501

Fig. 5 shows the average F1, MCC, pf and G-measure of the DLDD (θ=0.75) model (with denoising autoencoder) compared with deep neural network (without denoising autoencoder) on the PROMISE and NASA, respectively. From Fig. 5, we can observe that the DLDD (θ=0.75) model perform better than deep neural network in terms of F1, MCC, pf and G-measure. More specifically, the average F1 (0.7778), MCC (0.5309), pf (0.2433) and G-measure (0.7621) by DLDD (θ=0.75) yield improvements of 7.37%, 16.25%, 7.31% and 5.66% compared with deep neural network without denoising autoencoder on PROMISE respectively, and the average F1 (0.8374), MCC (0.6661), pf (0.1919) and G-measure (0.8285) by DLDD (θ=0.75) yield improvements of 7.36%, 32.85%, 38.81% and 11.67% compared with deep neural network without denoising autoencoder on NASA respectively.

Compared with single deep neural network, our DLDD model not only adopt deep neural network, but also remove noise to learn more robust feature representation by combining denoising autoencoder. Therefore, the DLDD model can achieve better prediction performance than single deep neural network.

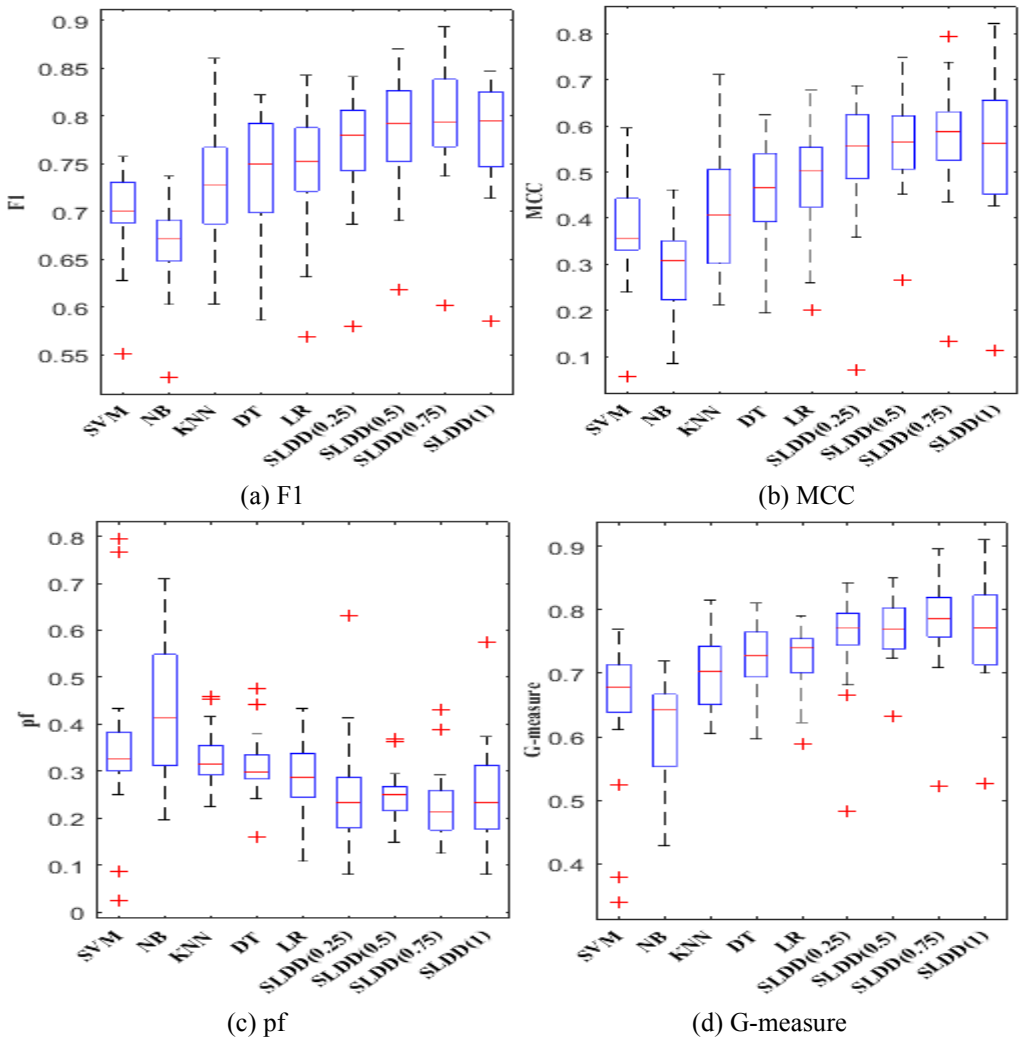
Table 7: The G-measure for our model DLDD compared with five classic defect predictors

Datasets	SVM	NB	KNN	DT	LR	SLDD (0.25)	SLDD (0.5)	SLDD (0.75)	SLDD (1)
ant-1.6	0.3393	0.6743	0.6534	0.6251	0.7564	0.7500	0.7674	**0.7817**	0.7686
ant-1.7	0.6715	0.4286	0.7010	0.7313	0.7171	0.7492	0.7701	0.7530	**0.7741**
camel-1.0	0.7241	0.6418	0.7469	0.8104	0.7269	**0.8431**	0.7315	0.8213	0.7044
camel-1.2	0.5236	0.5389	0.6054	0.5963	0.5896	0.4821	**0.6328**	0.5230	0.5252
camel-1.4	0.6887	0.6332	0.6483	0.6881	0.6774	0.6814	0.7435	**0.7578**	0.7150
ivy-2.0	0.6700	0.6615	0.7306	0.7156	0.7369	**0.8283**	0.8096	0.7916	0.7956
jedit-4.1	0.6802	0.7195	0.7409	0.7012	0.7432	**0.7813**	0.7237	0.7810	0.7114
jedit-4.2	0.6228	0.6614	0.7584	0.7357	0.7294	0.7603	0.7629	0.8098	**0.8310**
jedit-4.3	0.7076	0.5820	0.8145	0.7567	0.7722	0.8242	0.8281	0.8663	**0.9107**
poi-2.0	0.7207	0.6450	0.6154	0.7409	**0.7906**	0.7500	0.7623	0.7402	0.7018
prop-6	0.7695	0.5004	0.8154	0.7229	0.7408	0.7895	0.8028	**0.8180**	0.8130
synapse-1.2	0.3778	0.5474	0.6270	0.6903	0.6228	0.6657	0.7343	**0.7568**	0.7368
xalan-2.4	0.6877	0.5572	0.6569	0.6973	0.7389	0.7340	**0.7647**	0.7591	0.7445
xerces-1.2	0.6112	0.5667	0.6258	0.6387	0.6508	**0.7388**	0.7283	0.7096	0.7252
KC2	0.6709	0.5474	0.7072	0.7455	0.7482	0.7920	**0.7941**	0.7755	0.7130
CM1	0.6548	0.6599	0.6903	0.7896	0.7522	0.7529	0.7760	**0.7980**	0.7863
MC1	0.7015	0.6457	0.7058	0.7945	0.7488	0.7895	0.8512	**0.8963**	0.8181
MW1	0.6748	0.7049	0.6937	0.7235	0.6861	0.7970	0.8015	0.8164	**0.8425**
PC1	0.7488	0.6720	0.7445	0.7727	0.7799	0.7906	0.7979	**0.8331**	0.8285
PC2	0.7309	0.7128	0.7375	0.7923	0.7668	0.8084	0.8488	**0.8515**	0.8427
Avg	0.6488	0.6150	0.7009	0.7234	0.7238	0.7554	0.7716	**0.7820**	0.7644



(a) PROMISE (b) NASA

Figure 5: The average performance comparison of SLDD and DNN on PROMISE and NASA datasets



(a) F1 (b) MCC

(c) pf (d) G-measure

Figure 4: The box-plots for our proposed SLDD (θ=0.25, 0.5, 0.75, 1) compared with five classic predictors in terms of four indicators

> **Conclusion 3:** The DLDD (θ = 0.75) model outperforms the single deep neural network that does not combine denoising autoencoder, and the experimental results prove that the denoising autoencoder can boost the prediction performance of the DLDD (θ=0.75) model.

7 Conclusion

Software defect prediction can effectively guide the direction of software testing by allocating reasonably limited testing resources to highly risky modules before releasing the new software product. In this work, we construct an effective software defect prediction model based on a novel non-linear manifold learning feature extraction method and hybrid deep learning techniques. First, we leverage an advanced non-linear

manifold learning method - SL-Isomap to extract the representative features from the original defect features. Second, we propose a novel defect prediction model called DLDD based on hybrid deep learning techniques, which leverages denoising autoencoder to learn the reconstructed distribution and more robust feature representation by changing the reconstruction error term, and utilizes deep neural network to learn the abstract deep semantic features. In addition, we also combine loss functions of two deep learning techniques to achieve the best performance of defect prediction by adjusting a hyperparameter. We conduct extensive experiments for feature extraction and defect prediction across 20 software defect projects from large open source datasets, and the experimental results demonstrate that the effectiveness of SL-Isomap and DLDD.

In future work, in order to verify generalization capability and practicability of SL-Isomap and DLDD, we will evaluate them in more open source and commercial projects. Moreover, we also plan to extend SL-Isomap and DLDD to cross-project defect prediction.

Funding Statement: This work is supported in part by the National Science Foundation of China (Grant Nos. 61672392, 61373038), and in part by the National Key Research and Development Program of China (Grant No. 2016YFC1202204).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Ali, M. U.; Ahmed, S.; Ferzund, J.; Mehmood, A.; Rehman, A. (2017): Using PCA and factor analysis for dimensionality reduction of bio-informatics data. arXiv, arXiv-1707.

Bunte, K.; Haase, S.; Biehl, M.; Villmann, T. (2012): Stochastic neighbor embedding (SNE) for dimension reduction and visualization using arbitrary divergences. *Neurocomputing*, vol. 90, pp. 23-45.

Chen, M. M.; Ma, Y. T. (2015): An empirical study on predicting defect numbers. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, pp. 397-402.

Eberhardt, J.; Stote, R. H.; Dejaegere, A. (2018): Unrolr: structural analysis of protein conformations using stochastic proximity embedding. *Journal of Computational Chemistry*, vol. 39, no. 30, pp. 2551-2557.

Gan, Q.; Shen, F. R.; Zhao, J. X. (2014): An extended isomap for manifold topology learning with SOINN landmarks. *Proceedings of the 22th International Conference on Pattern Recognition*, pp. 1579-1584.

Hall, T.; Beecham, S.; Bowes, D.; Gray, D.; Counsell, S. (2012): A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276-1304.

Kirihata, M.; Ma, Q. (2019): A trend-shift model for global factor analysis of investment products. *IEICE Transactions on Information and Systems*, vol. 102, no. 11, pp. 2205-2213.

Kondo, M.; Bezemer, C. P.; Kamei, Y.; Ahmed, E. H.; Osamu, M. (2019): The impact of feature reduction techniques on defect prediction models. *Empirical Software Engineering*, vol. 24, no. 4, pp. 1925-1963.

Li, W.; Zhang, L. P.; Zhang, L. F.; Du, B. (2017): GPU parallel implementation of isometric mapping for hyperspectral classification. *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 9, pp. 1532-1536.

Li, Z. Q.; Jing, X. Y.; Zhu, X. K.; Zhang, H. Y.; Xu, B. W. et al. (2019): Heterogeneous defect prediction with two-stage ensemble learning. *Automated Software Engineering*, vol. 26, no. 3, pp. 599-651.

Lov, K.; Saikrishna, S.; Ashish, S.; Santanu, K. R. (2018): Effective fault prediction model developed using least square support vector machine (LSSVM). *The Journal of Systems and Software*, vol. 137, no. 3, pp. 686-712.

Shen, F. R.; Tomotaka, O.; Osamu, H. (2007): An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, vol. 20, no. 8, pp. 893-903.

Uddin, M. Z.; Hassan, M. M. (2015): A depth video-based facial expression recognition system using radon transform, generalized discriminant analysis, and hidden Markov model. *Multimedia Tools and Applications*, vol. 74, no. 11, pp. 3675-3690.

Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P. A. (2008): Extracting and composing robust features with denoising autoencoders. *The 25th International Conference on Machine*, pp. 1096-1103.

Wang, T. J.; Zhang, Z. W.; Jing, X. Y.; Liu, Y. L. (2016): Non-negative sparse-based semiboost for software defect prediction. *Software Testing, Verification and Reliability*, vol. 26, no. 7, pp. 498-515.

Wang, W.; Jiang, Y. B.; Luo, Y. H.; Li, J.; Wang, X. (2019): An advanced deep residual dense network (DRDN) approach for image super-resolution. *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, pp. 1592-1601.

Zhang, J. M.; Wang, W.; Lu, C. Q.; Wang, J.; Sangaiah, A. K. (2019): Lightweight deep network for traffic sign classification. *Annals of Telecommunications*, pp. 1-11.

Zhao, L.; Zou, D.; Gao, G. (2013): Subsampling based neighborhood preserving embedding for image classification. In *Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 358-360.

Zhou, L. L.; Tan, F.; Yu, F.; Liu, W. (2019): Cluster synchronization of two-layer nonlinearly coupled multiplex networks with multi-links and time-delays. *Neurocomputing*, vol. 359, pp. 264-275.

Zhu, K.; Zhang, N.; Ying, S.; Wang, X. (2020): Within-project and cross-project software defect prediction based on improved transfer Naive Bayes algorithm. *Computers, Materials & Continua*, vol. 63, no. 2, pp. 891-910.