Received: October 23, 2021. Revised: December 17, 2021.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Improved SMOTE and Optimized Siamese Neural Networks for Class Imbalanced Heterogeneous Cross Project Defect Prediction

Nataraj Kalaivani[1]* **Raman Beena[2]**

[1]*Department of Computer Science, Kongunadu Arts and Science College, Coimbatore – 641029,Tamilnadu, India*
[2]*Department of Information Technology, Sri Ramakrishna College of Arts and Science, Coimbatore – 641006, Tamilnadu, India*
* Corresponding author's Email: kalaivhani@gmail.com

Abstract: Heterogeneous Cross-Project Defect Prediction (HCPDP) predicts the defects by incorporating the project datasets with different distribution and metrics. However, the class imbalance problem and shallow feature learning of classifiers reduce the overall performance. This paper presents an HCPDP using Improved Synthetic Minority Oversampling Technique (ISMOTE) and hybrid deep learning classifier of Golden Eagle Optimized Siamese Neural Networks (GEO-SNN). Initially, Peters filter removes irrelevant instances in source-target dataset based on heterogeneous metrics. ISMOTE, enhanced through fuzzy mutual information based KNN attribute weights assignment, is applied solve the class imbalance problem. Features are scaled using Z-Score standardization and metric matching by Spearman's Rank Correlation (SRC). Finally, GEO-SNN, developed by SNN parameter optimization using GEO, performs deep semantic feature learning for defect prediction. Experiments using benchmark datasets showed that the proposed ISMOTE and GEO-SNN based HCPDP technique has provided 99% defect prediction accuracy and reduced processing time by 20%than the state-of-the-art methods.

Keywords: Improved synthetic minority oversampling technique, Software defect prediction, Heterogeneous cross-project defect prediction, Golden eagle optimization, Siamese neural networks, Peters filter.

1. Introduction

Software Defect Prediction (SDP) has become an integral part of the testing phase in the Software Development Life Cycle (SDLC) [1]. The huge size and high complexity in production issues limit the performance of the entire SDLC and create complexities in identifying software defects in the early stages of development. SDP methods can help in precisely identifying the modules which have higher tendencies to become defects. SDP can solve the energy constraint problem of the developers and limit SDLC periods so that the quality of software can be improved. SDP requires high knowledge of the historical data of the projects but has proved to provide highly accurate predictions. Data mining and machine learning algorithms [2] have been mostly utilized for SDP with common metrics such as complexity metrics (lines of code, Halstead metrics and McCabe metrics) [3], object-oriented metrics [4] and process metrics. Early prediction of the defects can be performed by SDP as within-project defect prediction (WPDP) [5]. However, the WPDP does not support new different projects or projects with limited or insufficient historical data. Unlike WPDP, Cross-Project Defect Prediction (CPDP) is an SDP that utilizes the existing historical data of the other projects to provide the prediction results of the given project [6]. Although some CPDP methods use heterogeneous source and target projects, they utilise only the common metrics among them. Therefore, the current CPDP methods are not adequate for heterogeneous projects with different metrics.

Heterogeneous Cross-Project Defect Prediction (HCPDP) or simply Heterogeneous Defect Prediction (HDP) [7] is the heterogeneous type of SDP based on the principle of CPDP with the additional characteristic of independent nature on whether the

source and target projects have common metrics or not. Different metrics based multiple source projects is still challenging for HCPDP. Additionally, the class imbalance problem and the shallow feature learning of the machine learning classifiers are also greater challenges [8]. To overcome these issues, the proposed HCPDP method utilizes an improved sampling technique and a hybrid deep learning classifier for deep feature learning.

The main notion is to construct a projective matrix between the heterogeneous source and target projects for converting the source project into the target project space for utilizing the deep learning classifier. Initially, the source and target project datasets are filtered using Peters filter together to select the relevant instances based on heterogeneous metrics of the source project after analysing the unlabeled modules in the target project. This reduces the negative impact of the irrelevant instances. Then the newly formed source project will be used for training and the feature extractions will be performed in the first stage. The noise removal is then performed and the class imbalance problem is overcome by the Improved Synthetic Minority Oversampling Technique (ISMOTE), which is developed by enhancing the underlying KNN classifier by defining the attribute weights by fuzzy mutual information. Z-Score standardization is used as the feature scaling method to minimize the irrelevant features. Then the feature weight training is performed based on the position vectors and the distance metrics to highlight the important features. The metric matching phase is utilized through Spearman's Rank Correlation (SRC) technique to estimate this association between feature pairs to represent the highly correlated feature pairs based on the heterogeneous metric collection. Finally, the deep semantic feature learning and defect prediction are achieved using the hybrid deep learning classifier of Golden Eagle Optimized Siamese Neural Networks (GEO-SNN) in which the parameters of SNN are optimized using Golden Eagle Optimizer. The proposed GEO-SNN classifier effectively learns the deep semantic features with high accuracy and low complexity to ensure better prediction of the defects. Experiments are conducted over benchmark project data to evaluate the performance of the proposed method and compared it with existing HCPDP methods.

The remainder of the article is organized as follows: Related works in Section 2, Methodology in Section 3 followed by the experimental results in Section 4 and conclusion and possible future directions in Section 5.

2. Related works

Recent years have seen an increasing number of studies being conducted for HCPDP. Still, attaining the perfect results in the HCPDP method is challenging due to the use of WPDP with different metrics. Kalaivani and Beena [9] presented an HCPDP method using Boosted Relief Feature Subset Selection (BRFSS) and Firefly Particle Swarm Multivariate Linear Regression (FFLYPSMVLR). This method used the BRFSS to handle different projects with heterogeneous feature sets through the mapping process and FFLYPSMVLR to provide the final optimal prediction. This method achieved high accuracy, precision and recall. Yet, the class imbalance problem still prevails. Jing et al. [10] introduced unified metric representation (UMR) and canonical correlation analysis (CCA)-based transfer learning for HDP. UMR was constructed for source company data and the target-company specific metrics based on which the CCA made the data distributions of source and target projects. However, this method does not resolve the class imbalance problem before applying transfer learning. Xu et al. [11] developed heterogeneous domain adaptation with dictionary learning for HDP. This method employed the domain adaptation method to insert the data from the two projects and then measured the difference between them using dictionary learning to predict the defects. Though it improved the F-measure, Balance, and AUC, this method suffers from class imbalance problems and the randomness in selecting the entities.

Li et al. [12] suggested a new cost-sensitive transfer kernel canonical correlation analysis (CTKCCA) for HDP with highly balanced data through different misclassification costs for defective and defect-free classes. This method reduced the class imbalance problem and increased the accuracy of defect prediction. Yet, this method has a higher storage space requirement for handling multi-source projects. Li et al. [13] developed an HDP model using two-stage ensemble learning of ensemble multi-kernel domain adaptation (EMDA) and ensemble data sampling (EDS). This ensemble model utilized Ensemble Multiple Kernel Correlation Alignment (EMKCA) predictor to estimate the defects through multiple kernel learning and domain adaptation. This model achieved high AUC, F-measure and balance. However, this model has high complexity in terms of processing time. Tong et al. [14] proposed a novel kernel spectral embedding transfer ensemble (KSETE) approach for HDP. This method solved the class-imbalance problem of the source data and identified the latent common feature space by

combining kernel spectral embedding, transfer learning, and ensemble learning. This method improved the AUC, G-measure and MCC values but has limitations in terms of processing time.

Gong et al. [15] illustrated Conditional domain adversarial adaptation (CDAA) motivated by generative adversarial networks (GANs) for HDP. CDAA used a generator for transfer learning of source project to target space, one discriminator to transfer the source project and one classifier to correctly label and predict the defects. However, this model also does not consider the class imbalance problem. Yin et al. [16] developed an HCPDP method using multiple source projects based on transfer learning. In this method, multiple heterogeneous source projects were used for defect prediction based on the projective matrix for transfer learning. This method achieved high prediction accuracy and fewer false positives. Yet, this method has limitations in handling the class imbalance problem.

Wang et al. [17] proposed a few-shot learning-based balanced distribution adaptation (FSLBDA) approach for imbalanced HDP. This method removed the redundant metrics using extreme gradient boosting and then the data variation is reduced using balanced distribution adaptation. Finally, adaptive boosting based few-shot learning is used for prediction. Although this method achieved high AUC, G-mean and F-measure than the existing methods, this method has the limitation of high complexity due to complex architecture. Wu et al. [18] presented multi-source HCPDP using multi-source transfer learning and autoencoder (MSTL-AE). This method considered the negative effect of transfer learning and developed modified autoencoder based HCPDP to extract the intermediate features from the original datasets. The multi-source transfer learning algorithm reduced the negative impact and improved the prediction accuracy. However, this method also does not consider the class imbalance and the noise problem. Wang et al. [19] developed HDP using Federated Transfer Learning based on the knowledge distillation (FTLKD) approach. This approach used trained convolutional neural networks (CNN) to utilize knowledge distillation for detecting the different metrics of projects and improved the accuracy, AUC and G-mean.

The vital points inferred from the literature are that the class imbalance problem is not effectively considered in many studies. The noise problem and the feature scaling problems are also briefly considered. The major point from the recent studies is that the deep learning methods can improve the feature learning process and increase the prediction accuracy without increasing the processing time or other complexities. Based on these observations, this proposed method developed an efficient HCPDP method using ISMOTE and GEO-SNN classifier.

3. Methodology

The proposed HCPDP method aims at resolving the error-prone limitations in noisy, unbalanced and different scaled features of the source and target projects' datasets. Initially, the source and target projects are filtered and merged based on the relevant instances using the Peters filter. Then the features are extracted and the noise removal, class balancing, feature scaling and feature weight training tasks are performed. The metric matching phase is utilized through the SRC technique to determine the highly correlated feature pairs to feed the GEO-SNN classifier for accurate defect prediction. The proposed HCPDP model is illustrated in Fig. 1.

3.1 Instance selection for the source project

Peters filter is used to perform instance selection for the source project. Peters filter lets the instances in the training dataset (TDS) find their nearest Test instances [20]. These instances are selected for the final filtered training dataset. The Peters filter uses the k-nearest neighbor algorithm to select similar labeled modules in the source project from training instances, which are more similar to the unlabeled modules in the target project. Algorithm 1 shows the Peters filter based instance selection.

Algorithm 1: Peters filter for instance selection
Input: Source dataset T, target dataset U, TDS
Output: New Source Dataset T'
While T' not unique
    For each module um in U do
        For each module tm in T do
            Distance (tm, um)
        End for
        Distance (TDS, Test)
        Nearest modules of TDS selected
        $T' = T' \cup selected\ modules$
    End for
        $T' = unique\ (T')$
    Return T'
End while

3.2 Feature extraction and pre-processing

First, the multiple quantifiable change features from the source data are extracted and they are divided into training and testing instances. Then the features are pre-processed using three steps:

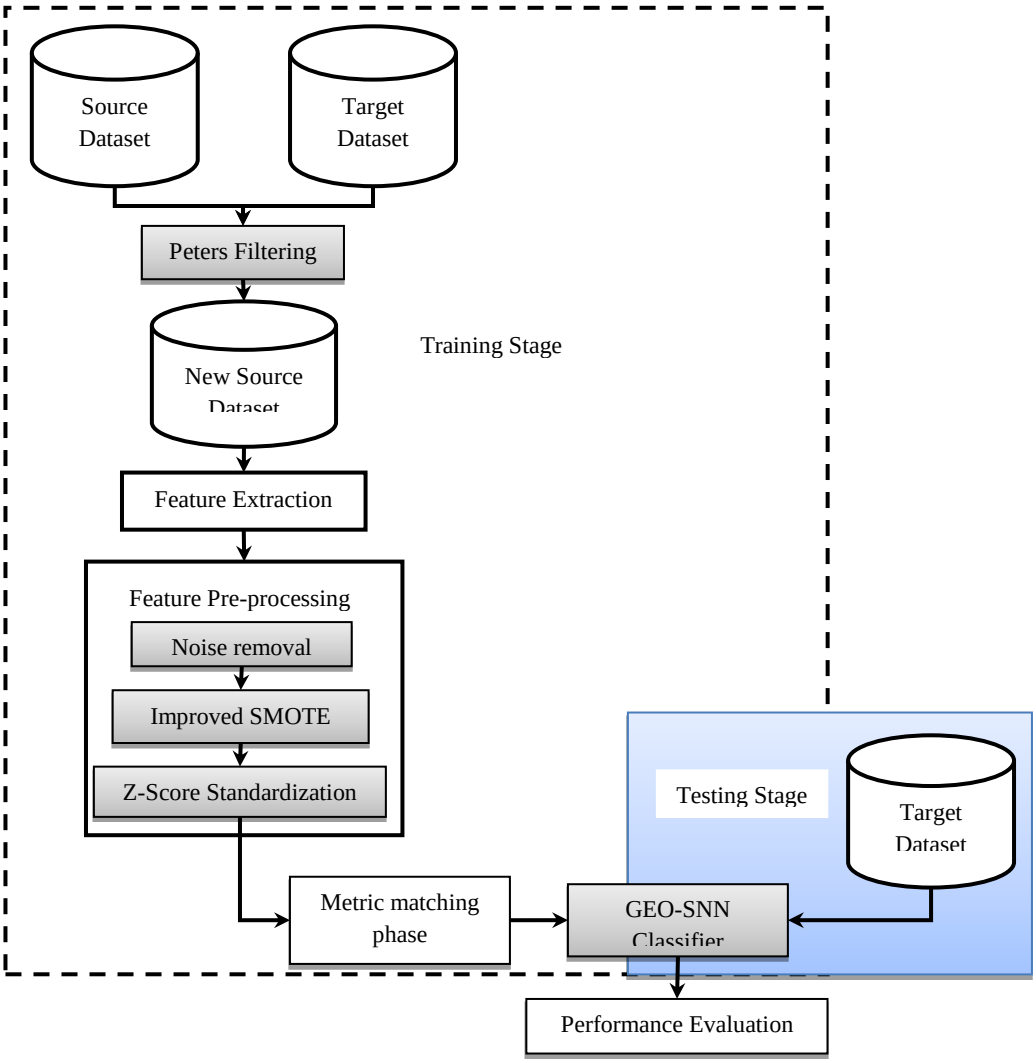

Figure. 1 Overall process of proposed HCPDP methodology

denoising, class imbalance processing and data standardization. For denoising, dynamic thresholding is used in which a dynamic threshold value is set and the data instances exceeding the threshold are filtered as noise. Then the class imbalance processing is performed using ISMOTE and data standardization by Z-score standardization.

3.2.1. Class imbalance processing

Class imbalance is a very common problem in most defect prediction methods. Analysing many software project modules, the distribution of the defects is in the Pareto principle, i.e. 80% non-defects and 20% defects. This means the number of defects is relatively very less than the non-defects which might result in poor prediction results. To overcome the problem, the ISMOTE is used.

ISMOTE is an improved sampling technique where the underlying KNN imputes used for class enhancement are modified defining the KNN attribute weights by fuzzy mutual information. ISMOTE performs the tasks as in SMOTE [21] by computing the information gain of every attribute and the weighted mutual information. Then the value of k clusters in KNN is defined and the training set is divided into k clusters. The Euclidean distance is computed between the training and testing samples. Finally, majority voting is applied to determine the class probability and the class labels. The fuzzy mutual information is computed for two attributes X and Y as

$$FMI(X;Y) = FH(X) + FH(Y) - FH(X,Y) \quad (1)$$

Here $FMI(X;Y)$ denotes the fuzzy mutual information of the attributes X and Y; $FH(X)$ and $FH(Y)$ denotes the fuzzy entropy of X and Y, respectively and $FH(X,Y)$ denotes the fuzzy entropy of X, Y [22].

$$FH(X) = -PX \log PX; FH(Y) = -PY \log PY \quad (2)$$

Where, X and Y are attributes and PX and PY are the fuzzy equivalent of the joint probability of training samples of a class with attributes X and Y, given as

$$PX = \frac{\sum_k \mu(x_{ik})}{NP}; PY = \frac{\sum_k \mu(y_{jk})}{NP} \quad (3)$$

Here, NP denotes the number of probabilities, $\mu(x_{ik})$ and $\mu(y_{jk})$ denotes fuzzy membership functions of k-th vector in i-th class of X and Y attributes.

3.2.2. Z-score standardization

The distribution of the extracted defect change feature values is not in the same order of magnitude. If the original feature values are used for defect prediction, the function of the higher values will be emphasized, and the function of the lower values is relatively destabilized. Therefore, to ensure the reliability of the result, z-score standardization is used [23]. For each value f_i of f features, the normalized value z_i can be computed as

$$z_i = \frac{f_i - mean\,(f)}{var\,(f)} \quad (4)$$

Here $mean\,(f)$ denotes the mean and $var\,(f)$ denotes the variance of f features.

3.2.3. Feature weight training

Each dimensional feature has a different effect on defect prediction and hence the vital features must be identified and different weights must be assigned. The weight values are assigned from 0 to 1 where 0 means that the features have no importance and the closer the weight value is to 1, the feature is more important. w_i is a control gate weight vector automatically trained by an adaptive trainable function, and is computed as a Sigmoid nonlinear activation function to obtain the final weighted vector to distinguish the vital features. The distinguishing rate r' of feature-weights is given as

$$r' = sigmoid\,(w_i) * r \quad (5)$$

Here w_i is the weights of features and r denote the initial weight difference rate.

3.3 Metric matching phase

The metric matching phase is performed to provide a common ground between the heterogeneous metrics of the source and target projects. To compute the correlation, Spearman's Rank Correlation is used. SRC is computed for the two metrics a and b as

$$\rho = 1 - \frac{6\sum md_i^2}{n(n^2-1)} \quad (6)$$

Here ρ denote the SRC measure, md_i denotes the difference between the ranks of each metric $a$and b, and n denote the number of instances.

3.4 GEO-SNN

The proposed GEO-SNN is a hybrid model using the SNN [24] and GEO algorithms [25]. The SNN is similar to the CNN and it includes an embedding layer (EL), convolutional layer (CL) and a pooling layer (PL), and two fully connected layers (FCL). It is made of two single neural network architectures combined with the GEO algorithm. The loss objectives of the SNN are alternatively computed using the predicted and true classes. The focal loss function is predominantly utilized for this purpose. The SNN is a typical few-shot learning model that included two single neural networks whose inputs are two instances. The output is computed by comparing the output layers of the first FCL layers of the single networks. It is estimated by applying the FCL on the difference of the two samples to estimate their similarity and using a cross-entropy loss on the similarity. Fig. 2 shows the architecture of the proposed GEO-SNN.

The proposed hybrid SNN is training the single architectures separately and adapting them as the top and bottom classes, respectively. The hybrid SNN has three technical attributes namely multi-task architecture, class-specific similarity and sampling process. Initially, the single network is trained and then used to initialize the SNN. After initialization, the SNN is trained separately as it might forget the knowledge obtained by the single networks. If the number of training instances is large, the instance pairs will be squared and become huge in numbers. SNN will not be feasible to train such huge instances and hence the training set is reduced to a sample subset. This might infuse the over-fitting problem in SNN.

To overcome this problem, the multi-task architecture is based on the Single SNN. The loss function is given as

$$\mathcal{L} = \lambda_S \mathcal{L}_S + \lambda_{m1} \mathcal{L}_{m1} + \lambda_{m2} \mathcal{L}_{m2} \quad (7)$$

Here $\mathcal{L}_S$ denotes the loss of SNN. $\mathcal{L}_m$denotes the loss of two single networks and λ denotes the network constraint. When there is no loss of

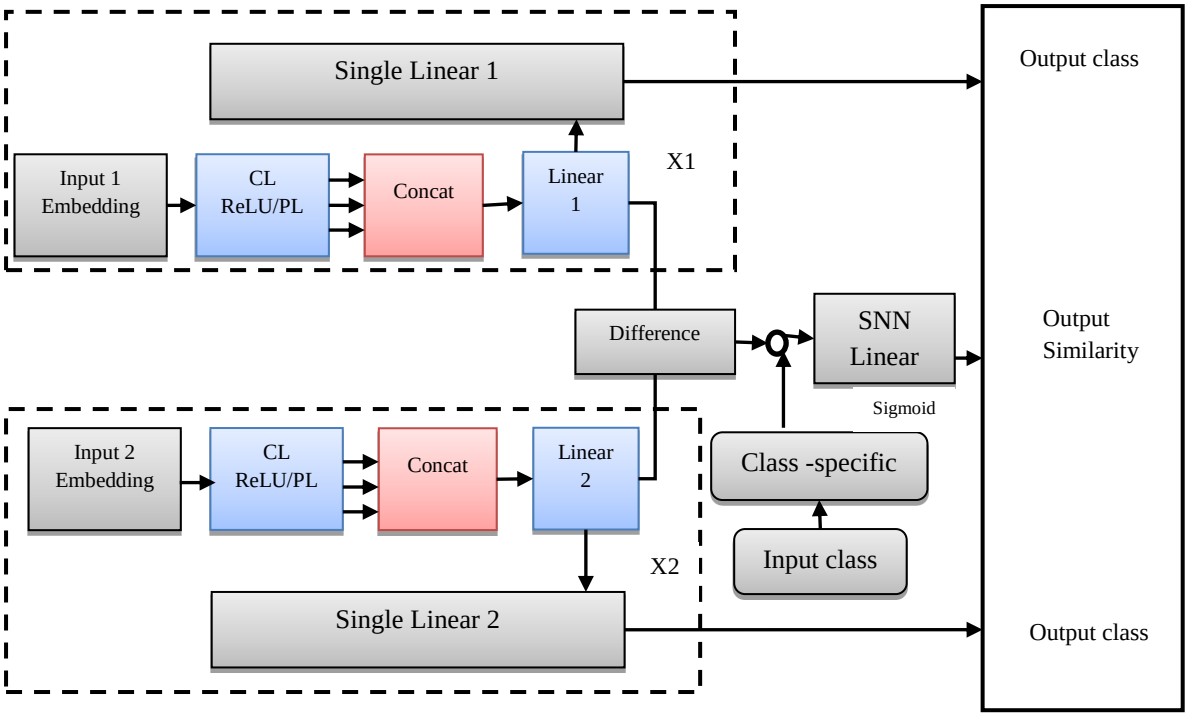

Figure. 2 Architecture of GEO-SNN

generality, $\lambda_S = \lambda_{m1} = \lambda_{m2} = 1$. $\mathcal{L}_{m1}$ and $\mathcal{L}_{m2}$ are equal for the similar pair of instances while $\mathcal{L}_{m1}$ and $\mathcal{L}_{m2}$ are unequal for dissimilar instances. The single network is considered as the constraint to avoid forgetting and over-fitting the SNN.

SNN has only single similarity output which is limited for estimating the similarity between a large number of individual classes and multiple classes. When an instance has multiple classes and each class has a separate representation vector, it becomes difficult for the SNN to learn all instances to obtain the single similarity output.in this case, the class-specific similarity is used in the SNN architecture to obtain the informative features in the similarity outputs. For two input instances d_i, d_j with a class q, the class-specific similarity is estimated as

$$h = |d_i - d_j| \circ h_q \tag{8}$$

Here $h_q = \sigma(\frac{w.p_q+b}{\sqrt{Q}})$. σ is the ReLU activation function and $\circ$ is the element-wise multiplication, w and b are the weight and bias, p_q is the class constant and Q is the set of all class labels.

The class-specific sampling process is also applied to the training data features by the randomly selected similar pairs $((d_i, q)(d_j, q))$ and dissimilar pairs $((d_i, q_i)(d_j, q_j))$ with the ratio of 1:1. The specific sampling process will generate one triplet (d_i, d_j, q) for training and generates, triplet instances with the ratio of similar and dissimilar pairs are thus 1:2.

GEO is applied at this stage to optimally tune the parameters. The EL, CL, PL, FCL and activation functions are optimally tuned using the GEO. GEO is based on the hunting behaviour of the golden eagles in tuning speed at different hunting stages [25]. It includes two tasks of golden eagles for searching the prey and attacking the prey. The golden eagles will cruise around the search space and determine which prey to be attacked since there are many organisms in the search space. GEO is based on the spiral motion of the golden eagles. The population is initialized as $f \in \{1,2,\dots,Popsize\}$. In each iteration, the golden eagle must select and attack only one prey. It is formulated as the attack vector.

$$\vec{A}_i = \vec{X}_f^* - \vec{X}_i \tag{9}$$

Here $\vec{A}_i$ denotes the attack vector of the eagle, $\vec{X}_f^*$ denote the best location of the prey found by the eagle f and $\vec{X}_i$ denote the current position of the eagle.

The cruise vector $\vec{C}_i$ is calculated based on the attack vector and it is used for the exploration process. The element of $\vec{C}_i$ is given as

$$c_k = \frac{d - \sum_{j,j\neq k} a_j}{a_k} \tag{10}$$

Here c_k denotes the k-th element of the destination point C of the eagle, a_j denote the j-th

element of the attack vector $\vec{A}_i$, d symbolizes the dimension of the search space, a_k denotes the k-th element of the attack vector $\vec{A}_i$, and k denotes the index of the fixed variable.

Based on the attack and cruise vectors, the step vector is formulated for each eagle at iteration t.

$$\Delta x_i = \vec{r}_1 p_a \frac{\vec{A}_i}{\|\vec{A}_i\|} + \vec{r}_2 p_c \frac{\vec{C}_i}{\|\vec{C}_i\|} \quad (11)$$

Here $\vec{r}_1$ and $\vec{r}_2$ are the random vectors in the interval [0,1] and p_a and p_c denotes the attack coefficient and cruise coefficient in t iterations, respectively. $\|\vec{A}_i\|$ and $\|\vec{C}_i\|$ denotes the Euclidean norm of the attack and cruise vectors, respectively.

The position of the golden eagles in iteration $t+1$ are computed as adding the step vector in iteration t to the positions in iteration t. This will be the position update equation

$$x^{t+1} = x^t + \Delta x_i^t \quad (12)$$

If the fitness of the new position of the golden eagle i is better than the position in its memory, the memory of this eagle is updated with the new position. Or else, the memory remains the same and the eagle will move to the next location. Based on this algorithm, the weights and the bias of SNN are tuned and thus the optimal values are set for the hyper-parameters. The fitness function is replaced by that of SNN as $f(y) = f(\delta, \eta)$. This fitness function is related to the EL, CL, Pl, FCL and activation function of the SNN for the two networks. While many configurations for the GEO-SNN are found by the tuning process, GEO selects the configuration with a minimum error rate. The process of GEO for SNN is presented as follows.

Algorithm 2: GEO for tuning SNN parameters

Population Initialization of golden eagles (P)
Set Iteration = 0
Initialize the GEO parameters
Map the SNN parameters to GEO solution search
Calculate the fitness and select the best search agent
Initialize the population memory
For each golden eagle$i = 1\ to\ n$ do
    Randomly select $\vec{X}_i$as the initial solution
    Calculate the attack vector
    If the attack vector length ≠ 0
        Calculate the cruise vector
        Calculate the step vector
        Update the position of search agents
        Update the GEO parameters
        Adjust the out-of-boundary eagle
        Compute fitness
        Update the best solution
        Increment Iteration by 1
        Until maximum iteration
    End if
End for
Return the best solution $\vec{X}_f^*$ from population memory
End

The top configuration obtained for SNN using GEO is shown in Table 1. The error rate obtained for this configuration is 12.2. For optimal structure, the GEO-SNN required 3 layers of FCL and one layer each for EL and CL. This configuration enables the learning of deep spatial features by SNN for multi-source projects. The input dimension is (1, 256, 24) where 1 denotes the channel of input data, 256 denote the number of data, and 24 denote the time steps to process them.

The networks are formed using this configuration. The single network is trained separately and the SNN is initialized using the single network. In the next stage, the trained features are used to classify the test data. Based on the similarity, the instance pairs are formed and the majority voting based approach is used for classifying the instances. Once the classifier results of the single SNN are obtained, they are merged based on a threshold which is determined by the number of instances in a class.

4. Results and discussion

The performance of the proposed HCPDP method using ISMOTE and GEO-SNN is deployed and evaluated using MATLAB software. The experimental results are obtained over six benchmark datasets of five project groups [26] described in Table 2 for evaluation.

The performance metrics namely accuracy, precision, recall, f-measure, specificity and

Table 1. Optimal configuration obtained for GEO-SNN

Layer	Name	Parameter	Dimension
0	Input	-	(1, 256, 24)
1	EL Activation (ReLU)	(16, 3, 3) -	(16, 256, 24) -
2	Convolution Pooling Activation (ReLU)	(64, 3, 3) (2, 2) -	(64, 64, 6) (64, 32, 3) -
3	Flatten	-	(1536,)
4	FCL	278	(278,)
5	FCL	278	(278,)

Table 2. Benchmark evaluation project datasets

Group	Dataset	No of instances	No of metrics	Prediction attribute
AEEEM	EQ	324	61	class
MORPH	Velocity-1.4	196	20	class
SOFTLAB	Ar1	121	29	function
NASA	CM1	344	37	function
	PC2	1585	36	function
ReLink	Apache	194	26	file

Table 3. Performance over NASA group CM1 and SOFTLAB group Ar1

Methods	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)	Specificity (%)	Processing time (s)
BRFSS [9]	95.68	93.2	94.7	96.67	92.56	127.56
CTKCCA [12]	92.16	95.11	93.67	91.39	89.43	154.28
KSETE [14]	95.67	91.56	91.67	95.67	90.9	110.65
CDAA [15]	95.5	90.17	92.38	96.44	92.14	116.37
FSLBDA [17]	90.89	88.97	89.34	91.56	84.63	126.55
MSTL-AE [18]	92.77	90.32	93.69	92.86	90.17	120.2
FTLKD-CNN [19]	94.5	92.18	91.66	95.22	88.67	115.21
GEO-SNN	**98.76**	**96.87**	**97.86**	**98.75**	**94.56**	**103.6**

Table 4. Performance over NASA group PC2 and RELINK group Apache

Methods	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)	Specificity (%)	Processing time (s)
BRFSS [9]	95.07	92.83	94.29	93.67	90.17	196.75
CTKCCA [12]	90.35	91.5	92.76	94.81	91.34	220.15
KSETE [14]	94.68	92.67	92.65	91.10	93.56	203.56
CDAA [15]	94.78	91.44	91.88	90.75	95.77	211.78
FSLBDA [17]	88.67	85.25	87.5	91.67	87.9	231.9
MSTL-AE [18]	92.15	92.44	93.19	94.36	91.05	226.5
FTLKD-CNN [19]	93.97	91.48	93.98	94.58	92.16	213.29
GEO-SNN	**98.99**	**96.62**	**99.75**	**98.13**	**95.82**	**186.75**

Table 5. Performance over AEEM group EQ and MORPH group Velocity-1.4

Methods	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)	Specificity (%)	Processing time (s)
BRFSS [9]	96.35	94.16	95.08	94.67	90.45	61.23
CTKCCA [12]	90.16	92.5	92.67	90.66	91.45	56.89
KSETE [14]	88.98	85.42	88.76	92.45	90.76	78.67
CDAA [15]	87.67	87.34	87.71	95.69	88,14	64.55
FSLBDA [17]	89.98	86.76	89.15	89.1	88.33	76.11
MSTL-AE [18]	92.45	90.23	91.4	91.73	92.89	70.9
FTLKD-CNN [19]	96.56	93.65	95.61	95.55	92.56	64.23
GEO-SNN	**99.23**	**98.47**	**98.67**	**99.11**	**97.45**	**49.5**

processing time are evaluated. The state-of-the-art methods of BRFSS [9], CTKCCA [12], KSETE [14], CDAA [15], FSLBDA [17], MSTL-AE [18] and FTLKD-CNN [19] are also implemented in the same simulation setting to compare their performance with the proposed method denoted as GEO-SNN. For comparison, the NASA Group dataset CM1 (source) and SOFTLAB group dataset Ar1 (target) are grouped as heterogeneous projects. Similarly, ASA group dataset PC2 (source) and RELINK group dataset Apache (target), and the AEEM group dataset EQ (source) and MORPH group dataset Velocity-1.4 (target) are also grouped as heterogeneous projects.

Table 3 shows the performance comparison of NASA Group dataset CM1 and SOFTLAB group dataset Ar1.Table 4 shows the performance comparison of NASA group dataset PC2 and RELINK group dataset Apache. Table 5 shows the performance comparison of AEEM group dataset EQ and MORPH group dataset Velocity-1.4.

The results in Table 3 show that the proposed GEO-SNN method has conveniently outperformed

the other methods with high values of accuracy, precision, recall, f-measure, and specificity and less processing time. The GEO-SNN has approximately 3%, 6%, 3%, 3%, 8%, 6% and 4% high accuracy than BRFSS [9], CTKCCA [12], KSETE [14], CDAA [15], FSLBDA [17], MSTL-AE [18] and FTLKD-CNN [19], respectively. The use of effective class imbalance processing and deep feature learning with parameter tuned SNN has been the major reason for this improvement.

The results in Table 4 show that the proposed GEO-SNN method has conveniently outperformed the other methods. GEO-SNN has approximately 3%, 8%, 4%, 4%, 10%, 6% and 5% high accuracy than BRFSS [9], CTKCCA [12], KSETE [14], CDAA [15], FSLBDA [17], MSTL-AE [18] and FTLKD-CNN [19], respectively. The proposed GEO-SNN has provided better defect prediction with heterogeneous metrics very effectively.

The results in Table 5 show that the proposed GEO-SNN method has conveniently outperformed the other methods for the AEEM group source dataset EQ and MORPH group target dataset Velocity-1.4. GEO-SNN has approximately 3%, 9%, 11%, 12%, 10%, 7% and 3% high accuracy than BRFSS [9], CTKCCA [12], KSETE [14], CDAA [15], FSLBDA [17], MSTL-AE [18] and FTLKD-CNN [19], respectively. This performance improvement can be attributed to the parameter tuned deep learning model of SNN using the advanced GEO optimization.

5. Conclusion

This paper presented an efficient HCPDP method using ISMOTE and GEO-SNN classifier. This proposed method resolved the class imbalance problem by using the ISMOTE with modified KNN using fuzzy mutual information. Z-score standardization and metric matching using the SRC measure improves the optimal feature selection. Finally, the GEO-SNN classifier used these features to classify the test data into the defect and non-defect modules. Experimental results showed that GEO-SNN has approximately 3%, 4-9%, 3-11%, 3-12%, 8-10%, 6-7% and 3-5% high accuracy than BRFSS [9], CTKCCA [12], KSETE [14], CDAA [15], FSLBDA [17], MSTL-AE [18] and FTLKD-CNN [19], respectively. In future, more company projects datasets will be used as multi-source datasets to detect the defects. Additionally, the possibility of integrating more heterogeneous metrics will be investigated.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

This work is a contribution of the authors: "Conceptualization, N. Kalaivani and R. Beena; methodology, N. Kalaivani; software, N. Kalaivani; validation, N. Kalaivani and R. Beena; formal analysis, N. Kalaivani; writing—original draft preparation, N. Kalaivani; writing—review and editing, N. Kalaivani and R. Beena.

References

[1] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings", *IEEE Transactions on Software Engineering*, Vol. 34, No. 4, pp. 485-496, 2008.

[2] V. U. B. Challagulla, F. B. Bastani, I. L. Yen, and R. A. Paul, "Empirical assessment of machine learning based software defect prediction techniques", *International Journal on Artificial Intelligence Tools*, Vol. 17, No. 02, pp. 389-400, 2008.

[3] R. S. Wahono and N. Suryana, "Combining particle swarm optimization based feature selection and bagging technique for software defect prediction", *International Journal of Software Engineering and Its Applications*, Vol. 7, No. 5, pp. 153-166, 2012.

[4] M. Jureczko and D. Spinellis, "Using object-oriented design metrics to predict software defects", *Models and Methods of System Dependability. OficynaWydawniczaPolitechnikiWrocławskiej*, pp. 69-81, 2010.

[5] S. D. Palma, D. D. Nucci, F. Palomba, and D. A. Tamburri, "Within-project defect prediction of infrastructure-as-code using product and process metrics", *IEEE Transactions on Software Engineering, Early Access*, pp. 1-12, 2021.

[6] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction", *Automated Software Engineering*, Vol. 19, No. 2, pp. 167-199, 2012.

[7] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous defect prediction", *IEEE Transactions on Software Engineering*, Vol. 44, No. 9, pp. 874-896, 2017.

[8] X. Chen, Y. Mu, K. Liu, Z. Cui, and C. Ni, "Revisiting heterogeneous defect prediction methods: How far are we?", *Information and Software Technology*, Vol. 130, p. 106441, 2021.

[9] N. Kalaivani and R. Beena, "Boosted Relief Feature Subset Selection and Heterogeneous Cross Project Defect Prediction using Firefly

Particle Swarm Optimization", *International Journal of Recent Technology and Engineering*, Vol. 8, No. 5, pp. 2605-2613, 2020.

[10] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning", In: *Proc. of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pp. 496-507, 2015.

[11] Z. Xu, P. Yuan, T. Zhang, Y. Tang, S. Li, and Z. Xia, "HDA: Cross-project defect prediction via heterogeneous domain adaptation with dictionary learning", *IEEE Access*, Vol. 6, No. 1, pp. 57597-57613, 2018.

[12] Z. Li, X. Y. Jing, F. Wu, X. Zhu, B. Xu, and S. Ying, "Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction", *Automated Software Engineering*, Vol. 25, No. 2, pp. 201-245, 2018.

[13] Z. Li, X. Y. Jing, X. Zhu, H. Zhang, B. Xu, and S. Ying, "Heterogeneous defect prediction with two-stage ensemble learning", *Automated Software Engineering*, Vol. 26, No. 3, pp. 599-651, 2019.

[14] H. Tong, B. Liu, and S. Wang, "Kernel spectral embedding transfer ensemble for heterogeneous defect prediction", *IEEE Transactions on Software Engineering*, Vol. 47, No. 9, pp. 1886-1906, 2019.

[15] L. Gong, S. Jiang, and L. Jiang, "Conditional domain adversarial adaptation for heterogeneous defect prediction", *IEEE Access*, Vol. 8, No. 1, pp. 150738-150749, 2020.

[16] X. Yin, L. Liu, H. Liu, and Q. Wu, "Heterogeneous cross-project defect prediction with multiple source projects based on transfer learning", *Mathematical Biosciences and Engineering*, Vol. 17, No. 2, pp. 1020-1040, 2020.

[17] A. Wang, Y. Zhang, H. Wu, K. Jiang, and M. Wang, "Few-shot learning based balanced distribution adaptation for heterogeneous defect prediction", *IEEE Access*, Vol. 8, No. 1, pp. 32989-33001, 2020.

[18] J. Wu, Y. Wu, N. Niu, and M. Zhou, "MHCPDP: multi-source heterogeneous cross-project defect prediction via multi-source transfer learning and autoencoder", *Software Quality Journal*, Vol. 29, No. 2, pp. 1-26, 2021.

[19] A. Wang, Y. Zhang, and Y. Yan, "Heterogeneous Defect Prediction Based on Federated Transfer Learning via Knowledge Distillation", *IEEE Access*, Vol. 9, No. 1, pp. 29530-29540, 2021.

[20] F. Peters, T. Menzies, and A. Marcus, "Better cross company defect prediction", In: *Proc. of 2013 10th Working Conference on Mining Software Repositories, IEEE*, pp. 409-418, 2013.

[21] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary", *Journal of Artificial Intelligence Research*, Vol. 61, No. 1, pp. 863-905, 2018.

[22] N. Hoque, H. A. Ahmed, D. K. Bhattacharyya, and J. K. Kalita, "A fuzzy mutual information-based feature selection method for classification", *Fuzzy Information and Engineering*, Vol. 8, No. 3, pp. 355-384, 2016.

[23] C. Cheadle, M. P. Vawter, W. J. Freed, and K. G. Becker, "Analysis of microarray data using Z score transformation", *The Journal of Molecular Diagnostics*, Vol. 5, No. 2, pp. 73-81, 2003.

[24] D. Chicco, "Siamese neural networks: An overview", *Artificial Neural Networks*, In: *Proc. of Cartwright H. (eds) Artificial Neural Networks. Methods in Molecular Biology*, Vol. 2190, No. 1, pp. 73-94, 2021.

[25] A. M. Balani, M. D. Nayeri, A. Azar, and M. T. Yazdi, "Golden eagle optimizer: A nature-inspired metaheuristic algorithm", *Computers and Industrial Engineering*, Vol. 152, No. 1, pp. 107050-107061, 2021.

[26] S. Herbold, A.Trautsch, and J. Grabowski, "A comparative study to benchmark cross-project defect prediction approaches", *IEEE Transactions on Software Engineering*, Vol. 44, No. 9, pp. 811-833, 2017.