sciendo

Int. J. Appl. Math. Comput. Sci., 2024, Vol. 34, No. 2, 291–307
DOI: 10.61822/amcs-2024-0021

amcs

A GAUSSIAN–BASED WGAN–GP OVERSAMPLING APPROACH FOR SOLVING THE CLASS IMBALANCE PROBLEM

QIAN ZHOU [a], BO SUN [a,*]

[a]Department of Computer Science and Technology
Shandong Agricultural University
61 Daizong Street, 271018, Tai'an, Shandong, China
e-mail: zq@sdau.edu.cn, sunbo87@126.com

In practical applications of machine learning, the class distribution of the collected training set is usually imbalanced, i.e., there is a large difference among the sizes of different classes. The class imbalance problem often hinders the achievable generalization performance of most classifier learning algorithms to a large extent. To ameliorate the learning performance, some effective approaches have been proposed in the literature, where the recently presented GAN-based oversampling methods are very representative. However, their generated minority class examples have the risk of high similarity and duplication degree. To further ameliorate the quality of the generated minority class examples, i.e., to make the generated examples effectively expand the minority class region, a novel oversampling approach named the GWGAN-GP is proposed, which is based on the Gaussian distribution label within the framework of a Wasserstein generative adversarial network with gradient penalty (WGAN-GP). Our GWGAN-GP approach incorporates the Gaussian distribution as an input label, thereby making the generated examples more diverse and dispersive. The examples are then combined with the original dataset to form a balanced dataset, which is subsequently utilized to evaluate the classification performance of three selected classification algorithms. Experimental results on 16 imbalanced datasets demonstrate that the GWGAN-GP not only generates examples that better conform to the distribution of the original dataset, but also achieves superior classification performance. Specifically, when combined with the KNN classifier, the GWGAN-GP significantly outperforms other oversampling approaches considered in the study.

Keywords: machine learning, class imbalance, generative adversarial networks, oversampling, data duplication.

1. Introduction

The class imbalance problem is a common challenge in the fields of machine learning and data mining, which refers to the unequal distribution of data among different classes. This problem has a significant impact on the efficiency and performance of classification algorithms (Sun *et al.*, 2009). Therefore, it is crucial to address the class imbalance problem when applying classification algorithms. There are two main approaches to tackle class imbalance: algorithmic improvements and data-level solutions. The algorithmic improvement approach focuses on modifying classification algorithms to mitigate their bias towards the majority class examples (Ohsaki *et al.*, 2017; Chaabane *et al.*, 2020; Zheng and Zhao, 2020b). Data-level solutions aim to address the class imbalance problem prior to the classification process. Oversampling algorithms (Chawla *et al.*, 2002), undersampling algorithms (Liu *et al.*, 2008), or hybrid algorithms (Park and Park, 2021; Janicka *et al.*, 2019) are employed to balance the dataset. Undersampling algorithms are based on removing examples from the majority class dataset, while oversampling algorithms involve generating new examples for the minority class dataset. By manipulating the dataset, both approaches aim to achieve a balance of classes before classifying, ultimately enhancing the efficiency of classification algorithms.

With the advent of generative adversarial networks (GANs) as a deep learning algorithm (Goodfellow *et al.*, 2014), a novel approach has been introduced for generating examples in oversampling approaches. GANs employ an adversarial training process, enabling the iterative generation of highly realistic new examples. This breakthrough has significantly contributed to the

*Corresponding author

© 2024 Q. Zhou and B. Sun.
This is an open access article distributed under
the Creative Commons Attribution-NonCommercial-NoDerivs license
(http://creativecommons.org/licenses/by-nc-nd/3.0/).

advancement of data generation techniques in the context of oversampling.

Oversampling approaches employ various strategies to generate new minority class examples. These strategies include random replication (Moreo *et al.*, 2016), interpolation approaches (Chawla *et al.*, 2002), and ensemble learning techniques (Chen *et al.*, 2022). The most influential one among them is the SMOTE algorithm (Chawla *et al.*, 2002; García *et al.*, 2016), and many subsequent oversampling approaches have been developed and expanded based on this algorithm (Fernández *et al.*, 2018). For example, Ren *et al.* (2023) proposed a novel oversampling approach (GB-SMOTE), which relocates new examples away from overlapping regions and applies appropriate corrections to the decision boundary. In addition, Kovács (2019) conducted a comprehensive comparison and evaluation of 85 variants of oversampling approaches, concluding that Polynom-fit-SMOTE (Gazzah and Amara, 2008), ProWSyn (Barua *et al.*, 2013), and SMOTE-IPF (Sáez *et al.*, 2015) typically achieve better performance than other oversampling approaches.

GANs are commonly employed for image data processing, including image generation (Miyato *et al.*, 2018) and image enhancement (Yang *et al.*, 2017). However, GANs also demonstrate promising performance in generating non-image data. For instance, Xie and Zhang (2018) introduced the DCGAN model and applied it to rotational machinery fault diagnosis. When aiming to generate examples specific to certain class, the conditional GAN (cGAN) (Mirza and Osindero, 2014) is often utilized. Douzas and Bacao (2018) employed the cGAN to generate examples that approximate the distribution of real examples, and experimental results confirmed the high quality of the generated examples. GANs have found extensive application in the generation of tabular data. For example, they are employed in the generation of healthcare (Nik *et al.*, 2023) and intrusion detection (Bourou *et al.*, 2021) data, serving a role in privacy preservation (Hernandez *et al.*, 2022). TableGAN (Park *et al.*, 2018) represents an early model designed for the generation of tabular data, encompassing both numerical and categorical columns. The CTGAN (Xu *et al.*, 2019) has the capability to generate tabular data with specific conditions. Building upon the core features of the CTGAN and TableGAN, Zhao *et al.* (2021) developed a novel conditional table GAN architecture (CTAB-GAN). It proves effective in modeling various data types with complex distributions. However, GANs suffer from limitations, such as generator gradient vanishing and mode collapse, resulting in repetitive and less diverse generated examples.

To mitigate these issues, improvements have been proposed for GANs. The WGAN (Arjovsky *et al.*, 2017) introduced the Wasserstein distance as a replacement for the traditional cross-entropy loss function to alleviate gradient vanishing. Zhang *et al.* (2021) utilized the WGAN to generate an imitation learning module, pre-training the reinforcement learning module to enhance learning efficiency. Zhang *et al.* (2023) proposed the G-GAN algorithm, which incorporates the Gaussian distribution as prior knowledge and employs the WGAN to generate examples. Cui *et al.* (2023) addressed the issue of insufficient rare attack examples by constructing multiple Gaussian distributions for minority class examples, establishing a mixed distribution model composed of several normal distributions. The WGAN-GP algorithm (Gulrajani *et al.*, 2017) replaced weight clipping with gradient penalty to address the mode collapse problem. Chen *et al.* (2023) proposed the pre-training WGAN-GP (PT-WGAN-GP) for aero-engine high speed bearing fault diagnosis for data imbalance. Zheng *et al.* (2020a) introduced the conditional WGAN-GP (CWGAN-GP) algorithm, which generates new examples by adding auxiliary conditional information, such as the class label.

These approaches are based on GANs and are applied for oversampling to address the class imbalance problem in various real-world domains. The generation of data using GAN-based approaches often leads to a problem of high data similarity, accumulation, and a lack of diversity that does not conform to the distribution of the real dataset. Relevant studies indicate that the intrinsic characteristics of imbalanced data also affect the efficiency of classification algorithms (López *et al.*, 2013), such as class overlap (Sun *et al.*, 2023), noisy data (He and Garcia, 2009), and small disjuncts (Japkowicz, 2003). Class overlap may lead to blurred decision boundaries and increase classification errors. Additionally, repetition of minority class data can also lead to a loss of minority class information, rendering it challenging for the model to accurately differentiate between various classes (Budach *et al.*, 2022; Zhao *et al.*, 2021).

To avoid these issues from occuring, Zhu *et al.* (2022) presented the GAN-based hybrid sampling (GBHS) algorithm, which utilizes the WGAN-GP to generate examples and then use undersampling approaches to tackle the duplication issue. However, to achieve direct generation of dispersed examples without the need for a secondary processing step, we present a Gaussian-based WGAN-GP approach (GWGAN-GP) in this paper. By incorporating the Gaussian distribution label, our method directly generates dispersed new examples, thereby reducing data duplication and enhancing data diversity.

The GWGAN-GP performs a novel oversampling at the data level. By utilizing the Gaussian distribution label as a condition and employing the WGAN-GP, synthetic examples of a minority class dataset are generated to address the class imbalance problem. In

contrast to other algorithms (Zhang *et al.*, 2023; Cui *et al.*, 2023) that require reshuffling input data to adhere to a Gaussian distribution, we maintain the original distribution of input data and disperse the generated new samples based on Gaussian distribution labels to minimize data duplication. In comparison with the previously mentioned GAN, WGAN, WGAN-GP and CWGAN-GP, the generated examples exhibit low similarity and increased diversity, and conform more to the distribution of a real dataset. Furthermore, the balanced datasets obtained through our oversampling approach demonstrate excellent performance and a significant practical value when used in various classifiers.

This paper is structured as follows. In Section 2, we provide an overview of the relevant algorithms, including the GAN, WGAN, WGAN-GP and CWGAN-GP. In Section 3, we present our novel approach, which integrates the Gaussian distribution label into the WGAN-GP framework. Section 4 outlines the experimental procedure. The results and a comprehensive analysis of the findings are presented in Section 5. Finally, Section 6 concludes the paper and contains a discussion of the future work.

2. Related works

2.1. GAN. The GAN is a deep learning model composed of two neural networks: the generator and the discriminator (Goodfellow *et al.*, 2014). The objective of the GAN is to generate realistic examples by training the generator and discriminator in an adversarial manner. The generator takes random noise as input and aims to generate synthetic examples that resemble real examples. Conversely, the discriminator attempts to distinguish between real and generated examples by outputting the probability of a sample being real. The generator uses this probability to update its parameters and improve the quality of generated examples, while the discriminator updates its parameters to better discriminate between real and generated examples. The training process of the GAN can be formulated as a minimax game, where the generator aims to minimize the discriminator's ability to distinguish between real and generated examples, while the discriminator aims to maximize its ability to correctly discriminate the examples. The loss function for this adversarial training is defined as follows:

$$\min_{\mathrm{G}}\max_{\mathrm{D}} V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))], \tag{1}$$

where $p_{\text{data}}(x)$ represents the distribution of real examples, x denotes an element from the distribution $p_{\text{data}}(x)$, $p_z(z)$ is a noise distribution, and z represents the input noise vector for the generator. $G(z)$ represents the generated examples, D is the discriminator, and $D(G(z))$ denotes the discriminator's probability of classifying the generated examples. From the equation above, it can be observed that the discriminator aims to maximize the loss function, while the generator aims to minimize $\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$.

During this adversarial process, if the discriminator performs exceptionally well from the beginning and can easily distinguish real examples from generated ones, the gradients may approach zero. This phenomenon limits the gradient propagation to the generator, leading to gradient vanishing where the generator's parameters cannot be effectively updated. Consequently, the generator becomes trapped in a specific mode, ignoring other potential modes, and continuously generates repetitive or similar examples, resulting in a lack of diversity. This issue is commonly referred to as mode collapse, where the generator fails to capture the full distribution and collapses to a limited set of modes.

2.2. Three GAN-based approaches (WGAN, WGAN-GP and CWGAN-GP). Compared with the GAN, the WGAN (Arjovsky *et al.*, 2017) utilizes the Wasserstein distance as the loss function for the discriminator:

$$\min_{\mathrm{G}}\max_{\mathrm{D}} V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))]. \tag{2}$$

The objective of the discriminator is to maximize the loss function, while that of the generator is to minimize $-\mathbb{E}_{z\sim p_z(z)}[D(G(z))]$.

This distance provides a more accurate measurement of the difference between real and generated examples, enabling more precise discriminator evaluation. As a result, the discriminator's gradient changes smoothly, alleviating the issue of gradient vanishing.

By introducing a gradient penalty mechanism, the WGAN-GP algorithm (Gulrajani *et al.*, 2017) applies a penalty on the gradients of the discriminator based on linear interpolations between real and generated examples. This penalty prevents the discriminator from excessively chasing a specific mode, thus mitigating the problem of mode collapse and enhancing algorithm stability. The improved loss function is expressed as follows:

$$\min_{\mathrm{G}}\max_{\mathrm{D}} V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{\tilde{x}\sim p_{\tilde{x}}}[D(\tilde{x})] - \lambda\mathbb{E}_{\hat{x}\sim p_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}}D(\hat{x})\|_2-1\right)^2\right], \tag{3}$$

where $p_{\tilde{x}}$ represents the linearly interpolated distribution between the real examples' distribution $p_{\text{data}}(x)$ and the generated examples' distribution $p_{\tilde{x}}$, with $\hat{x} = \varepsilon x + (1-\varepsilon)\tilde{x}$, where ε is a uniformly sampled random number

from $[0, 1]$ and $\tilde{x}$ stands for the generated examples from $G(z)$. The gradient penalty coefficient λ is set to 10. The objective of the discriminator is to maximize the loss function, while that of the generator is to minimize $-\mathbb{E}_{\tilde{x}\sim p_{\tilde{x}}}[D(\tilde{x})]$.

The WGAN-GP can be modified to incorporate the conditional information (CWGAN-GP), such as the class label (Zheng *et al.*, 2020a). By adding this conditional information as an additional input, the CWGAN-GP enables the generation of examples that satisfies a specific condition. The modified loss function can be expressed as follows:

$$\begin{aligned}&\min_{\mathrm{G}}\max_{\mathrm{D}} V(D,G)\\&= \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x \mid y)] - \mathbb{E}_{\tilde{x}\sim p_{\tilde{x}}}[D(\tilde{x} \mid y)]\\&\quad - \lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x} \mid y)\|_2 - 1\right)^2\right],\end{aligned} \tag{4}$$

where y represents the additional auxiliary information, which in this approach refers to the class label. By incorporating the class label into the loss function, the CWGAN-GP allows the generation of examples that conform to a specific class, enabling more controlled and targeted data synthesis.

For binary classification problems, the class label as input information may not hold significant meaning. In this paper, we propose employing a Gaussian distribution as the input label instead, aiming to mitigate issues such as data accumulation and overlap.

3. Proposed oversampling approach (GWGAN-GP)

By conducting feature correlation analysis (Guyon and Elisseeff, 2003) and incorporating a Gaussian distribution (Wasserman, 2004) as the input label, the GWGAN-GP enhances the diversity of the generated examples and ensures that it better captures the distribution characteristics of the real examples. Firstly, we perform feature correlation analysis to identify the features most relevant to the target variable, referred to as the key features. This analysis aims to identify the features that exhibit strong correlations with the class label. Subsequently, the mean and standard deviation of the key feature are calculated for the minority class dataset. These computed values serve as the parameters for the Gaussian distribution, with the mean indicating the central tendency and the standard deviation representing the spread. Alternatively, based on the different characteristics of the dataset, other suitable values can be chosen as the mean and standard deviation parameters for the Gaussian distribution. Ultimately, this Gaussian distribution is introduced as a conditional information y input in the WGAN-GP framework.

The procedure for identifying the column index of the key feature can be expressed using the following equation:

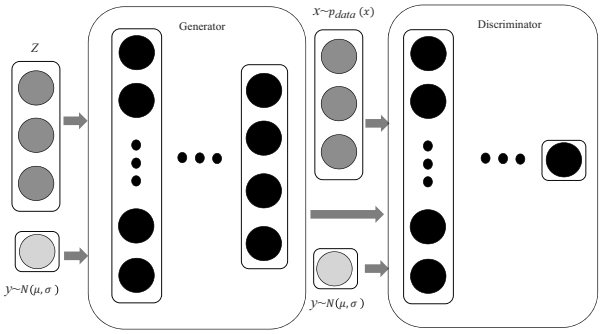


Fig. 1. Structure of our GWGAN-GP approach.

$$\begin{aligned}&C_j\\&= \arg\max_j \frac{\mu([(X-\mu(X,N))(Y-\mu(Y,N))],N)}{\sigma(X,N)\sigma(Y,N)}.\end{aligned} \tag{5}$$

where $X = X_j$, $Y = Y_{\text{class}}$. Specifically, X_j represents the vector composed of values from all examples in the *j*-th column for dataset *p*. The number of all examples is denoted as *N*. Y_{class} represents the class label column. The $\arg\max_j$ operation identifies the index j that maximizes the correlation value.

The mean $\mu\left(X_{C_j}, N_{\min}\right)$ and standard deviation $\sigma\left(X_{C_j}, N_{\min}\right)$ of the minority class dataset p_{data} about the column of C_j are computed, and these values are used as the mean and standard deviation for the Gaussian distribution. The size of p_{data} is $N_{\min}$.

For datasets with complex feature values, where a key feature may not have significant impact, it is possible to choose more suitable values for the mean and standard deviation of the Gaussian distribution. For example, $\mu = 10$ and $\sigma = 5$ can be used, although these values are not fixed and may vary to achieve better experimental results.

From the above equations and values, the conditional input y conforms to the distribution $N(y; \mu, \sigma)$. The structure of our GWGAN-GP approach is illustrated in Fig. 1.

In Fig. 1, the process begins by the generator producing fake examples based on the distribution of the conditional y and random noise z. Subsequently, the generated fake examples, the real minority class dataset, and the conditional y are fed into the discriminator. It calculates the loss function separately for each input and back-propagates the gradients to update its parameters. This iterative process continues with multiple iterations of generator and discriminator updates. Eventually, the trained model outputs new examples that conform to the distribution specified by the conditional y. The new loss

Algorithm 1. Procedures of our GWGAN-GP.

Require: Dataset p

Calculate C_j of the key feature column of p using Eqn. (5).
Extract the minority class dataset p_{data} from p.
Calculate the mean $\mu\left(X_{C_j}, N_{\min}\right)$ and standard deviation $\sigma\left(X_{C_j}, N_{\min}\right)$ of column C_j in p_{data}.
Generate the y distribution label by inputting μ and σ into $N(y; \mu, \sigma)$.
while $k \Leftarrow 1, 2, 3, \ldots, num_epochs$ **do**
  while $i \Leftarrow 1, 2, 3, \ldots, N_{\min}/batch_size$ **do**
    Sample a batch from noise data and y randomly.
    Sample a batch from p_{data} and y randomly.
    Generator()
    Compute_Gradient_Penalty()
    Discriminator()
    Calculate the loss functions for the generator and discriminator. {Referring to Eqn. (6)}
    Update the parameters of the generator and discriminator
  end while
end while
return the generated examples after oversampling

function can be expressed as follows:

$$\min_G \max_{\mathrm{D}} V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x), y\sim N(\mu,\sigma)}[D(x \mid y)] - \mathbb{E}_{\tilde{x}\sim p_{\tilde{x}}, y\sim N(\mu,\sigma)}[D(\tilde{x} \mid y)] - \lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}, y\sim N(\mu,\sigma)}\left[\left(\|\nabla_{\hat{x}} D(\hat{x} \mid y)\|_2 - 1\right)^2\right]. \tag{6}$$

The complete procedures of our GWGAN-GP, incorporating the structure and equations mentioned, can be described as follows.

Firstly, the calculations of the key feature columns, mean, and standard deviation, along with the generation of the y distribution label, ensure that the generated examples conform to the real distribution and incorporate the specified conditioning information. Secondly, the iterative process involves training the generator and discriminator by sampling batches from the noise data, the minority class dataset, and conditional y. Subsequently, the parameters are updated, and loss functions are optimized during each iteration. Finally, the generated examples are outputted after oversampling.

4. Experimental study

To demonstrate that the minority examples generated by the GWGAN-GP approach are more dispersed compared with those obtained with other GAN-based approaches, some experiments have been performed. Additionally, when combined with different classification algorithms, our approach can achieve better classification performance than other oversampling approaches. We aim to identify an optimal combination strategy of a classification algorithm with the GWGAN-GP and apply it to address real-world class imbalance problems. The following experimental procedure was conducted.

4.1. Imbalanced datasets used. The dataset utilized in this paper was sourced from imbalanced datasets available on KEEL (Derrac *et al.*, 2015), UCI (Dua and Graff, 2019), and Kaggle (2024). The sizes of these datasets varied, ranging from 213 to 26,851 instances, while the number of features encompassed a range of 3 to 96. Furthermore, the datasets exhibited varying degrees of class imbalance, with imbalance ratios ranging from 1.87 to 29.9. Table 1 presents the size, imbalance ratio, and number of features for each dataset.

Datasets with IDs 12-16 are sourced from real-world applications. The Adult dataset is extracted from the 1994 U.S. census data and includes detailed individual information such as the age, education level, and other features. The Bank dataset, focused on banking marketing, includes fundamental information like the user age, occupation, etc. The Telecom Churn dataset is a commonly used dataset for predicting customer churn. These datasets are widely employed in research related to income prediction and the field of business economics. Additionally, two healthcare-related datasets are the Lower Back Pain dataset and the Mammography dataset. The former is used to determine the presence of a lower back pain by inputting relevant features such as pelvic and sacral characteristics. The latter (Woods *et al.*, 1993) is used to predict the likelihood of cancer by inputting breast-related image data, age, and other information.

4.2. Generator and discriminator configuration. The generators and discriminators in GAN-based approaches (GAN, WGAN, WGAN-GP, CWGAN-GP, and GWGAN-GP) are implemented as three-layer neural networks. The hidden layers are set to 64–128 neurons. It is important to note that the parameters for the hidden layers should not be smaller than the size of features in the dataset. The batch size is set to 16, which should be smaller than the number of the minority class dataset. The number of epochs is 100, as setting a larger value would result in highly similar data generated by GAN-based approaches, significantly deviating from the real examples' distribution and affecting the classifier's judgment. The Adam optimizer is used in the algorithm. The detailed network topology and hyperparameter settings are shown in Table 2.

Table 1. Details of the datasets used.

ID	Dataset	Instances	Majority instances	Minority instances	Imbalance ratio	Features
1	Abalone9-18	731	689	42	16.40	8
2	Car-good	1728	1659	69	24.04	6
3	Ecoli2	336	284	52	5.46	7
4	Fare-F	306	267	39	6.85	11
5	Glass0	213	144	69	2.09	9
6	Haberman	289	210	79	2.66	3
7	Pima	768	500	268	1.87	8
8	Vehicle0	846	647	199	3.25	18
9	Winequality	1359	1306	53	24.64	11
10	Yeast1458vs7	688	658	30	21.93	8
11	Yeast2vs8	457	437	20	21.85	8
12	Adult	26851	20730	6121	3.39	96
13	Bank	4521	4000	521	7.68	16
14	Lower back pain	310	210	100	2.10	12
15	Mammography	7849	7595	254	29.90	6
16	Telecom Churn	7031	5163	1868	2.76	27

Table 2. Settings of GAN-based approaches.

Parameters	GAN	WGAN	WGAN-GP	CWGAN-GP	GWGAN-GP
Network topology	(input_dim,64) (64,128) (128,output_dim)	(input_dim,64) (64,128) (128,output_dim)	(input_dim,64) (64,128) (128,output_dim)	(input_dim+1,64) (64,128) (128,output_dim)	(input_dim+1,64) (64,128) (128,output_dim)
Activation function	Sigmoid()	Tanh()	Tanh()	ReLU() Tanh()	ReLU() Tanh()
Loss function	BCELoss()	Wasserstein Loss	Wasserstein Loss GradientPenalty Loss	Wasserstein Loss GradientPenalty Loss	Wasserstein Loss GradientPenalty Loss
Optimizer	Adam	Adam	Adam	Adam	Adam
Learning rate	0.0002	0.00005	0.00005	0.0002	0.0002
Number of epochs	100	100	100	100	100
Batch size	16	16	16	16	16
Lambda GP	–	–	10	10	10

4.3. Details of classification algorithms. We combined the examples generated by the GWGAN-GP algorithm with the original dataset, resulting in a balanced dataset that was used for evaluation by the classification algorithms.

The three commonly used classification algorithms include k-nearest neighbors (k-NN) (Cover and Hart, 1967), random forest (RF) (Breiman, 2001), and adaptive boosting (AB) (Freund and Schapire, 1997). The k-NN algorithm assigns a class label to a sample by considering the majority class labels among its k nearest neighbors ($k = 3$). The RF algorithm constructs an ensemble of decision trees (Breiman, 2017) and combines their predictions to make the final classification. We created a random forest classifier consisting of 10 trees, each with a maximum depth of 10. In each split, only one randomly selected feature is considered. The AB algorithm will employ 200 decision trees as the base classifiers, where each decision tree is constrained by a maximum depth limit of 1.

4.4. Performance evaluation metrics used. In order to demonstrate the effectiveness of the GWGAN-GP algorithm in addressing the challenges of duplication and high similarity, the Euclidean distances $D = \{d_i\}_{i=1}^{N_{\text{new}}-1}$ are computed between generated examples. Here, N_{new} denotes the size of the generated examples. Subsequently, the mean and variance of the calculated Euclidean distances are evaluated. A higher mean value indicates greater distances and lower similarity among the generated examples, while a larger variance implies a more dispersed data distribution.

Additionally, we employ principal component analysis (PCA) (Wold *et al.*, 1987) to visually demonstrate the changes in the quantity of the dataset before and after oversampling, as well as the distribution of the generated

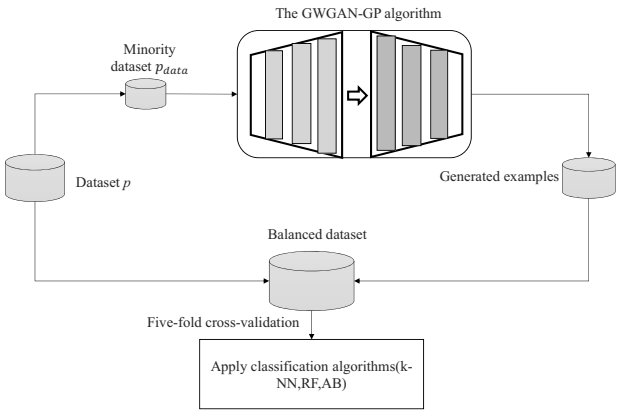


Fig. 2. Experimental workflow about datasets.

examples using the GAN-based approaches.

We performed five-fold cross-validation (Kohavi, 1995) to calculate various performance metrics for classification algorithms, including recall, F1 score, G-mean and the area under the curve (AUC) (He and Garcia, 2009; Powers, 2020). Recall measures the ability of the classifier to correctly identify positive instances, making it particularly suitable for applications that are concerned with the omission of positive instances, such as medical diagnosis. The F1 score is the harmonic mean of precision and recall, making it a good metric for classifiers that achieve a balance between positive and negative instances. G-mean is highly effective for classification problems with imbalanced datasets. It considers the performance of the classifier on both positive and negative classes, taking the geometric mean between them. Therefore, when considering the balance between different classes, G-mean is an appropriate choice. The AUC metric assesses the performance of the classification algorithm based on the receiver operating characteristic (ROC) curve, providing insights into the algorithm's ability to distinguish between positive and negative instances. The AUC is suitable for assessing the overall ranking ability of a classifier. Unlike the previous three metrics, it does not focus on a specific threshold but instead considers the overall performance across different thresholds. Moreover, it exhibits a certain tolerance to class imbalance.

By comparing the performance of the classification algorithms using the new balanced dataset, we were able to quantitatively evaluate the efficacy of the GWGAN-GP in generating diverse examples. The experimental workflow about datasets is illustrated in Fig. 2. The experimental results and performance metrics obtained will be presented and discussed in the subsequent sections.

5. Experimental results and analysis

5.1. Data distribution after oversampling. The distribution of newly generated examples varies among the ten small datasets for GAN-based approaches. The means and variances of the distances between the newly generated examples are illustrated in Fig. 3.

Based on the figure, it is evident that the means and variances of distances between the generated examples differ among GAN-based approaches across the ten small datasets. Specifically, the examples generated by the GAN exhibit relatively small means and variances of distances, indicating a high level of similarity and significant duplication. For comparison, the WGAN-GP and CWGAN-GP show slightly increased mean and variance values compared with the GAN. Notably, Datasets 8 exhibits the largest mean and variance of distances for the WGAN-GP. On other datasets, the GWGAN-GP generates examples with the highest means of distances, suggesting a lower similarity and duplication degree among the generated examples. Additionally, the larger variances of the distance about generated examples indicate a more dispersed data distribution. On the average, the GWGAN-GP generates examples with higher means and variances of distances compared with the other GAN-based approaches. This highlights the effectiveness of the GWGAN-GP in addressing problems related to data duplication and high similarity encountered in GAN-based oversampling approaches.

Figure 4 presents a more visually insightful depiction of the distribution of generated examples across three datasets. In order to present clear results, we chose three datasets with varying sizes, avoiding overly large datasets. Dataset 3, of moderate size, features a unique distribution where the minority class data predominantly occupies the central region, flanked by the majority class data on either side. This setup often results in a majority of safe examples in the minority class, which are generally more straightforward for classifiers to learn from (Napierala and Stefanowski, 2016). Dataset 5, smaller in size, presents a different scenario where there is notable overlap between the two classes. In this case, the minority class is characterized by a greater proportion of unsafe examples, complicating the classification task. Lastly, Dataset 7, larger in size, displays a higher rate of overlap between the two classes in one dimension, adding to the complexity of the classification challenge.

As depicted in Fig. 4, examples generated by the GAN tend to be relatively concentrated, displaying a high degree of duplication. Unlike this, the WGAN-GP and CWGAN-GP show a slight reduction in duplication for the generated examples compared with the GAN. Notably, examples generated by the GWGAN-GP exhibit greater dispersion with a lower duplication rate, aligning more

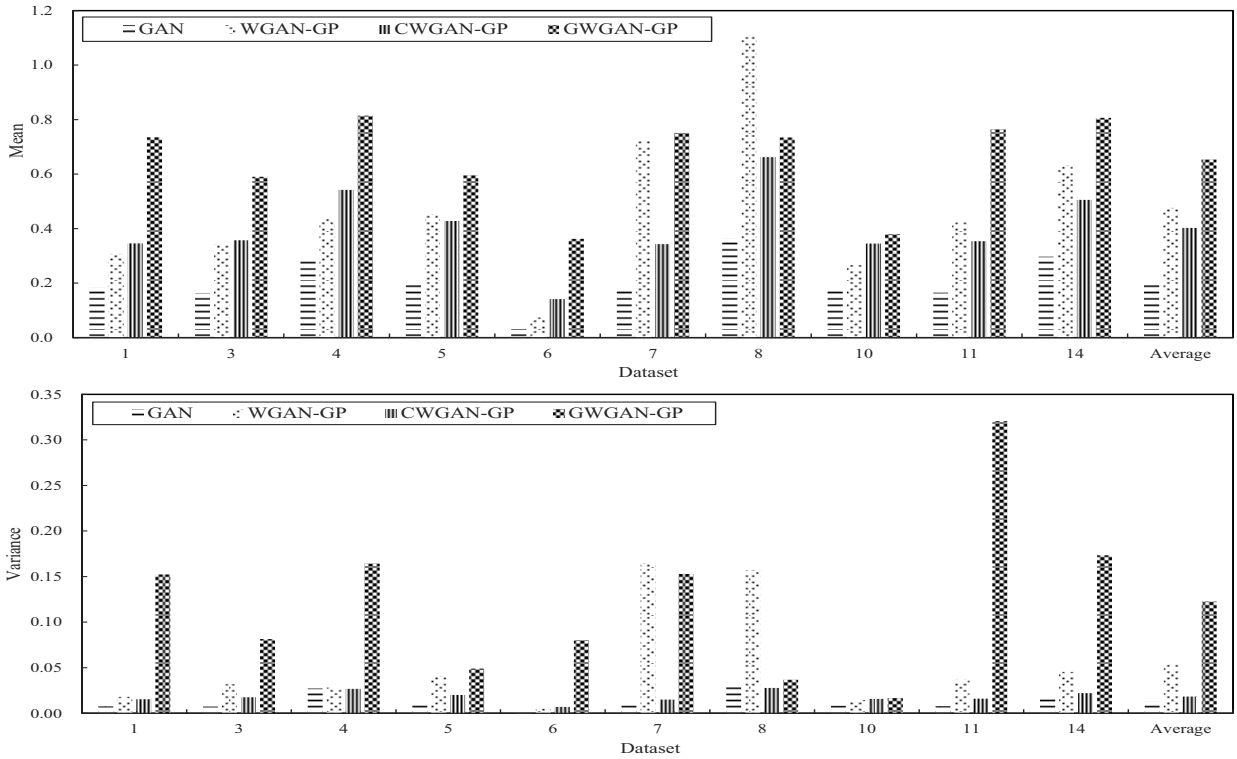


Fig. 3. Means and variances of the distances.

closely with the distribution of the real minority class dataset.

5.2. Obtained experimental results measured using recalls. To evaluate the performance of GAN-based oversampling approaches in comparison with oversampling algorithms such as SMOTE (Chawla *et al.*, 2002), Polynom-fit-SMOTE (PFS) (Gazzah and Amara, 2008), ProWSyn (Barua *et al.*, 2013) and SMOTE-IPF (Sáez *et al.*, 2015), the evaluation is conducted using a five-fold cross-validation approach on various datasets and classifiers. The mean recall values achieved by each approach are computed and compared across the different datasets and classifiers. The recall values are presented in Table 3.

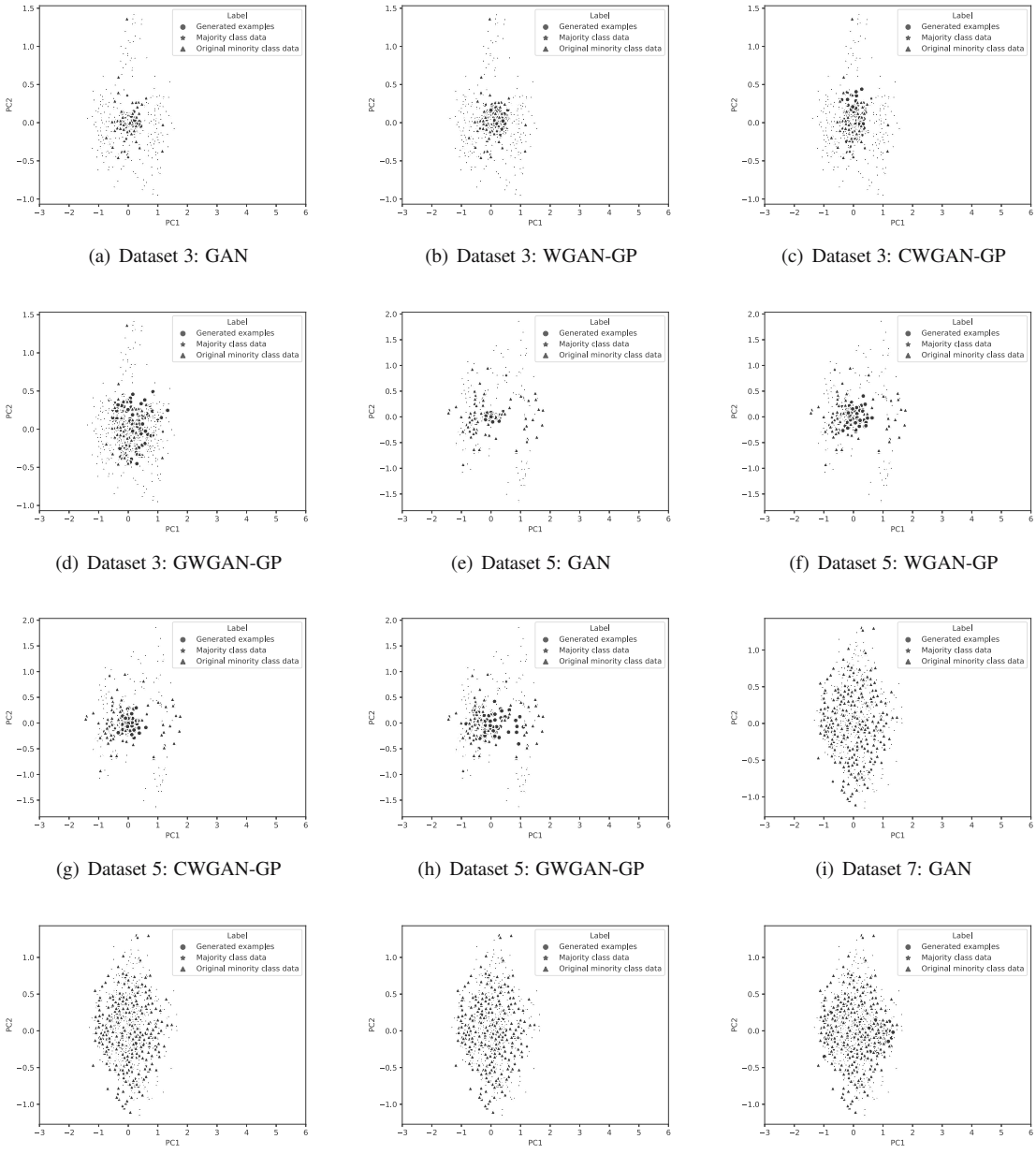
Based on the data presented in the table, it is evident that the GWGAN-GP outperforms other oversampling methods in recall across three classifiers. Specifically, in the five datasets (2, 9, 10, 11, 15) where the imbalance ratio exceeds 20, the combination of the GWGAN-GP and a classification algorithm also achieves superior recall values.

In the context of imbalanced datasets, the minority class examples are of particular importance as they often represent crucial, rare, or significant instances. Specifically, in oversampling algorithms, the generation of examples primarily focuses on the minority class (positive instances). Hence, it is desirable for the classifier to effectively capture a maximum number of minority class dataset. The recall metric accurately assesses the proportion of correctly identified minority class data among all real minority class dataset, thereby effectively evaluating the performance of oversampling approaches.

The higher frequency of the GWGAN-GP consistently yielding the best recall values across various datasets when combined with classifiers indicates its capability to enhance the classifier identifying the minority class dataset.

5.3. Obtained experimental results measured using F1 scores. The average F1 scores are presented in Table 4.

The table reveals that the GWGAN-GP algorithm attains superior optimal F1 scores, suggesting that the synergy between the GWGAN-GP algorithm and classifiers can strike an effective balance between precision and recall. Among the three classifiers, the fusion of the KNN classifier and the GWGAN-GP produces an average F1 score of 0.8917 across 16 datasets, surpassing other oversampling approaches by a margin of 0.0044 to 0.2156. Similarly, when the RF classifier is paired with the GWGAN-GP, the average F1 score reaches 0.8958, outperforming other oversampling algorithms by 0.0033 to 0.2490. Additionally, the



(a) Dataset 3: GAN (b) Dataset 3: WGAN-GP (c) Dataset 3: CWGAN-GP

(d) Dataset 3: GWGAN-GP (e) Dataset 5: GAN (f) Dataset 5: WGAN-GP

(g) Dataset 5: CWGAN-GP (h) Dataset 5: GWGAN-GP (i) Dataset 7: GAN

(j) Dataset 7: WGAN-GP (k) Dataset 7: CWGAN-GP (l) Dataset 7: GWGAN-GP

Fig. 4. Data distribution of the datasets (the pentagram symbol represents the majority class dataset, the triangle symbol represents the original minority class dataset, and the circle symbol represents the generated examples).

combination of the AB classifier and the GAN achieves an average F1 score of 0.8975, marginally surpassing the GWGAN-GP algorithm by 0.0008.

5.4. Obtained experimental results measured using G-means. The average G-mean metric results are presented in Table 5.

From Table 5, it can be observed that the GWGAN-GP algorithm performs well compared with other oversampling approaches across the three classifiers. It excels particularly with the KNN classifier, achieving the maximum G-mean values across 10 datasets. Considering the earlier F1 experimental results, the GWGAN-GP algorithm, in comparison with other oversampling approaches, tends to bring about a more balanced classification performance for the KNN classifier in both positive and negative classes.

5.5. Obtained experimental results measured using AUCs. Utilizing five-fold cross-validation, we

Table 3. Obtained recalls of each approach.

Dataset	Classifers	NONE	SMOTE	PFS	ProWSyn	SMOTE _IPF	GAN	WGAN	WGAN -GP	CWGAN -GP	GWGAN -GP
	KNN	0.5298	0.9245	0.8741	0.9013	0.9180	0.9611	0.9124	**0.9676**	0.9509	0.9555
1	RF	0.5943	0.9340	0.8913	0.9078	0.9289	0.9596	0.8935	**0.9633**	0.9509	0.9504
	AB	0.6324	0.9166	0.9498	0.9223	0.9289	0.9517	0.9139	0.9553	0.9422	**0.9598**
	KNN	0.6534	0.9436	0.9563	0.9250	0.9433	0.9653	0.9653	0.9653	0.9653	**0.9674**
2	RF	0.6474	0.9608	0.9255	0.9485	0.9551	0.9650	0.9665	**0.9677**	0.9662	0.9674
	AB	0.5187	0.8165	0.8230	0.8210	0.8177	**0.9792**	**0.9792**	0.9789	**0.9792**	0.9653
	KNN	0.9157	0.8924	0.9069	0.8959	0.9012	0.9180	0.8918	0.9180	0.9074	**0.9253**
3	RF	0.8716	0.9293	0.9235	0.9153	0.9311	0.9198	0.9286	0.9216	0.8929	**0.9329**
	AB	0.8894	0.9171	0.9122	0.9083	0.9312	0.9412	**0.9466**	0.9448	0.9465	0.9294
	KNN	0.5082	0.8352	0.8178	0.8165	0.8322	0.8939	0.8884	0.8883	0.8884	**0.9034**
4	RF	0.4991	0.8577	0.8818	0.8446	0.9020	0.8845	0.8921	0.8939	0.8940	**0.9034**
	AB	0.6380	0.8633	0.8743	0.8577	0.8741	**0.9034**	0.9015	0.8996	0.8996	0.8923
	KNN	0.8042	0.8333	0.8117	0.8229	0.8264	0.8125	0.7917	0.8056	0.8090	**0.8403**
5	RF	0.8007	0.8507	0.8268	0.8125	**0.8681**	0.8438	0.7917	0.8194	0.8160	0.8125
	AB	0.7938	0.8264	0.8264	0.8021	**0.8542**	0.8299	0.7951	0.8090	0.8088	0.8264
	KNN	0.5747	0.7357	0.7102	0.6667	0.7520	0.7377	0.6686	0.7639	0.7472	**0.7663**
6	RF	0.5391	0.7048	0.7245	0.7095	0.7550	0.7616	0.7354	0.7545	0.7593	**0.7641**
	AB	0.5648	0.7119	0.7618	0.7500	0.6986	**0.7877**	0.7520	0.7782	0.7830	0.7664
	KNN	0.6936	0.7860	**0.8238**	0.7611	0.7881	0.7800	0.7402	0.7771	0.7711	0.7892
7	RF	0.7003	0.7912	0.7699	0.7551	0.7921	0.8043	0.7484	0.7831	0.7922	**0.8055**
	AB	0.6955	0.7570	0.7930	0.7621	0.7701	**0.8002**	0.7544	0.7611	0.7882	0.7642
	KNN	0.9311	0.9590	0.9533	0.9552	**0.9606**	0.9583	0.9560	0.9583	0.9583	0.9591
8	RF	0.9138	0.9583	0.8663	0.9490	0.9482	0.9390	0.9228	0.9459	0.9506	**0.9714**
	AB	0.9788	0.9799	0.8667	0.9691	0.9776	0.9815	0.9490	0.9737	0.9776	**0.9846**
	KNN	0.5045	0.9043	0.7960	0.8672	0.9032	0.9733	0.9369	0.9687	0.9434	**0.9739**
9	RF	0.4981	0.9315	0.8279	0.8664	0.9173	**0.9775**	0.9112	0.9675	0.9491	0.9766
	AB	0.5470	0.8806	0.8433	0.8641	0.8679	0.9591	0.8836	0.9522	0.9330	**0.9739**
	KNN	0.5071	0.8320	0.7518	0.7689	0.8191	0.9263	0.8068	**0.9301**	0.9004	0.9217
10	RF	0.4969	0.8776	0.8112	0.7917	0.8768	0.9544	0.9453	0.9598	**0.9666**	0.9491
	AB	0.5257	0.8715	0.8444	0.8183	0.8723	0.9308	0.9301	0.9270	0.9286	**0.9491**
	KNN	0.7227	0.9109	0.9363	0.9187	0.9223	**0.9851**	0.9141	0.9770	0.9507	0.9441
11	RF	0.6989	0.9462	**0.9875**	0.9336	0.9623	0.9747	0.9759	0.9770	0.9667	0.9852
	AB	0.7466	0.9599	0.9852	0.9577	0.9588	0.9828	0.9782	0.9816	0.9828	**0.9863**
	KNN	0.6815	0.8303	0.7815	0.8101	0.8516	0.8394	0.8394	0.8378	0.8395	**0.8568**
12	RF	0.5019	0.7892	0.8267	0.8269	0.8049	0.8510	0.8523	**0.8524**	0.8522	0.8521
	AB	0.7221	0.8375	0.7933	0.8207	0.8546	0.8602	0.8592	0.8603	0.8596	**0.8635**
	KNN	0.5993	0.9140	0.8968	0.9057	0.9118	0.9195	0.9183	0.9189	0.9195	**0.9246**
13	RF	0.5357	0.8931	**0.9327**	0.9107	0.8982	0.9315	0.9318	0.9323	0.9318	0.9296
	AB	0.6534	0.9116	**0.9222**	0.9167	0.9134	0.9176	0.9183	0.9184	0.9188	0.9109
	KNN	0.7111	0.8000	0.7942	0.7833	0.8095	0.8071	0.7571	**0.8262**	0.8119	0.8117
14	RF	0.6752	0.8000	0.8309	0.7881	0.7976	0.8190	0.7881	0.8214	**0.8452**	0.8214
	AB	0.7261	0.8071	0.8006	0.7929	0.8000	0.8048	0.7905	0.8167	0.8238	**0.8405**
	KNN	0.7787	0.9532	0.9718	0.9555	0.9687	0.9864	0.9795	0.9856	0.9860	**0.9864**
15	RF	0.7239	0.9506	0.9732	0.9598	0.9651	0.9871	0.9849	**0.9875**	0.9873	0.9874
	AB	0.7299	0.9267	0.9715	0.9517	0.9461	0.9835	0.9820	**0.9861**	0.9856	0.9857
	KNN	0.6767	0.8059	0.7986	0.7932	0.8074	0.8040	0.8029	0.8040	0.8039	**0.8238**
16	RF	0.6911	0.8123	0.7784	0.8186	0.8125	0.8224	0.8173	0.8193	0.8038	**0.8393**
	AB	0.7143	0.8165	0.7788	0.7930	0.8198	0.8334	0.8330	**0.8349**	0.8328	0.8188

computed the average AUC (area under the curve) metrics for different oversampling algorithms combined with three classifiers across 16 datasets. The AUC results offer a comprehensive assessment of the classifier performance under various datasets and oversampling approaches. The detailed results are presented in Table 6.

On all 16 datasets and with three different classifiers, each oversampling approach exhibited its optimal AUC value. Overall, the GWGAN-GP performed best, closely followed by the GAN and WGAN-GP. This demonstrates that, overall, our approach can achieve good performance.

Table 4. Obtained F1 scores of each approach.

Dataset	Classifers	NONE	SMOTE	PFS	ProWSyn	SMOTE _IPF	GAN	WGAN	WGAN -GP	CWGAN -GP	GWGAN -GP
	KNN	0.5355	0.9242	0.8742	0.9009	0.9176	0.9595	0.9104	**0.9657**	0.9493	0.9519
1	RF	0.6185	0.9339	0.8910	0.9076	0.9288	0.9579	0.8912	**0.9612**	0.9493	0.9469
	AB	0.6571	0.9165	0.9497	0.9223	0.9288	0.9499	0.9120	0.9532	0.9405	**0.9565**
	KNN	0.5976	0.9433	0.9560	0.9242	0.9430	0.9647	0.9647	0.9647	0.9647	**0.9669**
2	RF	0.5675	0.9607	0.9248	0.9482	0.9549	0.9644	0.9659	**0.9671**	0.9656	0.9669
	AB	0.3813	0.7849	0.7914	0.7894	0.7861	**0.9786**	**0.9786**	0.9783	**0.9786**	0.9647
	KNN	0.9091	0.8908	0.9037	0.8947	0.8998	0.9147	0.8879	0.9152	0.9024	**0.9214**
3	RF	0.8692	0.9287	0.9222	0.9148	0.9306	0.9176	0.9263	0.9194	0.8851	**0.9316**
	AB	0.8818	0.9167	0.9111	0.9080	0.9311	0.9406	**0.9462**	0.9444	0.9461	0.9281
	KNN	0.4978	0.8303	0.8159	0.8104	0.8310	0.8734	0.8679	0.8676	0.8677	**0.8847**
4	RF	0.4924	0.8551	0.8779	0.8406	**0.9014**	0.8659	0.8715	0.8755	0.8755	0.8867
	AB	0.6257	0.8593	0.8702	0.8523	0.8711	**0.8852**	0.8811	0.8791	0.8791	0.8783
	KNN	0.7866	0.8281	0.8040	0.8160	0.8201	0.8064	0.7816	0.7991	0.8032	**0.8334**
5	RF	0.7997	0.8495	0.8253	0.8099	**0.8672**	0.8382	0.7802	0.8072	0.8088	0.8063
	AB	0.7913	0.8248	0.8231	0.7967	**0.8532**	0.8213	0.7863	0.7981	0.7977	0.8220
	KNN	0.5744	0.7313	0.7031	0.6660	0.7494	0.7143	0.6545	0.7369	0.7309	**0.7514**
6	RF	0.5271	0.7017	0.7067	0.7057	**0.7527**	0.7429	0.7119	0.7359	0.7408	0.7502
	AB	0.5525	0.7064	0.7366	0.7445	0.6941	**0.7680**	0.7339	0.7583	0.7647	0.7489
	KNN	0.6977	0.7856	**0.8242**	0.7607	0.7870	0.7758	0.7376	0.7729	0.7671	0.7862
7	RF	0.7074	0.7906	0.7614	0.7545	0.7917	0.8023	0.7473	0.7790	0.7896	**0.8043**
	AB	0.7012	0.7568	0.7892	0.7609	0.7698	**0.7983**	0.7524	0.7577	0.7857	0.7615
	KNN	0.9324	0.9590	0.9525	0.9551	**0.9605**	0.9582	0.9558	0.9582	0.9582	0.9589
8	RF	0.9194	0.9582	0.8543	0.9490	0.9481	0.9382	0.9221	0.9455	0.9503	**0.9714**
	AB	0.9741	0.9799	0.8487	0.9690	0.9776	0.9815	0.9490	0.9737	0.9776	**0.9846**
	KNN	0.5046	0.9032	0.7945	0.8651	0.9021	0.9730	0.9367	0.9684	0.9433	**0.9734**
9	RF	0.4891	0.9314	0.8232	0.8657	0.9172	**0.9772**	0.9109	0.9672	0.9489	0.9761
	AB	0.5521	0.8804	0.8412	0.8638	0.8677	0.9588	0.8832	0.9519	0.9327	**0.9734**
	KNN	0.4996	0.8247	0.7492	0.7547	0.8110	0.9254	0.8026	**0.9291**	0.8991	0.9213
10	RF	0.4873	0.8764	0.8085	0.7817	0.8746	0.9542	0.9449	0.9596	**0.9665**	0.9488
	AB	0.5290	0.8706	0.8428	0.8138	0.8712	0.9300	0.9292	0.9261	0.9277	**0.9489**
	KNN	0.7802	0.9100	0.9359	0.9184	0.9219	**0.9850**	0.9141	0.9768	0.9507	0.9435
11	RF	0.7282	0.9462	**0.9875**	0.9336	0.9622	0.9745	0.9757	0.9768	0.9665	0.9851
	AB	0.7974	0.9599	0.9852	0.9577	0.9588	0.9828	0.9782	0.9817	0.9828	**0.9863**
	KNN	0.6866	0.8293	0.7443	0.8017	**0.8522**	0.8282	0.8282	0.8282	0.8283	0.8502
12	RF	0.4397	0.7882	0.7751	**0.8251**	0.8051	0.8148	0.8161	0.8162	0.8160	0.8048
	AB	0.7388	0.8369	0.7397	0.7949	0.8544	0.8478	0.8463	0.8478	0.8468	**0.8559**
	KNN	0.6260	0.9135	0.8971	0.9055	0.9112	0.9118	0.9106	0.9112	0.9118	**0.9169**
13	RF	0.5389	0.8930	**0.9296**	0.9096	0.8982	0.9238	0.9240	0.9245	0.9240	0.9219
	AB	0.6831	0.9107	**0.9193**	0.9146	0.9124	0.9099	0.9105	0.9106	0.9110	0.9031
	KNN	0.6892	0.7779	0.7656	0.7597	0.7866	0.8052	0.7517	**0.8241**	0.8089	0.8053
14	RF	0.6696	0.7711	0.8221	0.7690	0.7718	0.8164	0.7862	0.8181	**0.8425**	0.8157
	AB	0.6946	0.7813	0.7736	0.7712	0.7717	0.8016	0.7850	0.8137	0.8211	**0.8371**
	KNN	0.8208	0.9532	0.9717	0.9555	0.9685	**0.9863**	0.9795	0.9856	0.9860	**0.9863**
15	RF	0.7863	0.9505	0.9732	0.9598	0.9651	0.9871	0.9849	**0.9875**	0.9873	0.9873
	AB	0.7713	0.9267	0.9715	0.9517	0.9462	0.9835	0.9820	**0.9861**	0.9856	0.9856
	KNN	0.6809	0.8038	0.7975	0.7917	0.8050	0.7945	0.7934	0.7945	0.7944	**0.8162**
16	RF	0.7074	0.8114	0.7213	0.8108	0.8116	0.8045	0.7962	0.7997	0.7668	**0.8282**
	AB	0.7267	0.8136	0.7216	0.7596	0.8170	0.8226	0.8219	**0.8241**	0.8218	0.8130

5.6. Algorithm comparison using statistical analysis. We employ the Wilcoxon signed-rank test method (James *et al.*, 2013) to calculate p-values, separately for various classifiers and metrics, comparing other oversampling approaches with the GWGAN-GP, as illustrated in Fig. 5.

From the figure, it can be observed that, in the results of AUCs, there is a significant difference when combined with the KNN classifier for the GWGAN-GP compared with other oversampling approaches. The difference is not pronounced when combined with the other two classifiers. For F1 scores and G-means, there is a significant difference compared with most oversampling approaches, regardless of the combined classifier. The differences between the GWGAN-GP

Table 5. Obtained G-means of each approach.

Dataset	Classifers	NONE	SMOTE	PFS	ProWSyn	SMOTE _IPF	GAN	WGAN	WGAN -GP	CWGAN -GP	GWGAN -GP
	KNN	0.5590	0.9282	0.8750	0.9043	0.9221	0.9651	0.9177	**0.9724**	0.9548	0.9594
1	RF	0.6560	0.9344	0.8959	0.9095	0.9298	0.9639	0.9012	**0.9683**	0.9545	0.9555
	AB	0.6738	0.9170	0.9526	0.9228	0.9293	0.9561	0.9190	0.9603	0.9470	**0.9654**
	KNN	0.6090	0.9473	0.9588	0.9304	0.9470	0.9683	0.9683	0.9683	0.9683	**0.9701**
2	RF	0.5907	0.9623	0.9311	0.9511	0.9572	0.9680	0.9695	**0.9706**	0.9692	0.9703
	AB	0.5062	0.8489	0.8550	0.8530	0.8500	**0.9818**	**0.9818**	0.9815	**0.9818**	0.9683
	KNN	0.9168	0.9017	0.9104	0.9019	0.9093	0.9282	0.8989	0.9279	0.9186	**0.9342**
3	RF	0.8830	0.9336	0.9271	0.9181	0.9351	0.9286	0.9380	0.9303	0.9096	**0.9400**
	AB	0.8926	0.9205	0.9160	0.9106	0.9336	0.9464	0.9507	0.9493	**0.9510**	0.9366
	KNN	0.5002	0.8516	0.8241	0.8320	0.8478	0.8791	0.8728	0.8740	0.8734	**0.8993**
4	RF	0.4946	0.8691	0.8953	0.8589	0.9087	0.9027	0.8772	0.9112	0.9108	**0.9204**
	AB	0.6458	0.8780	0.8892	0.8744	0.8854	**0.9196**	0.8860	0.8844	0.8844	0.9071
	KNN	0.8097	0.8466	0.8260	0.8379	0.8417	0.8228	0.8057	0.8172	0.8189	**0.8500**
5	RF	0.8280	0.8548	0.8300	0.8193	**0.8718**	0.8550	0.8012	0.8355	0.8287	0.8237
	AB	0.8094	0.8302	0.8326	0.8129	**0.8580**	0.8457	0.8002	0.8263	0.8238	0.8340
	KNN	0.5839	0.7436	0.7213	0.6674	0.7605	0.7460	0.6711	0.7718	0.7538	**0.7801**
6	RF	0.5351	0.7101	0.7474	0.7161	0.7628	0.7650	0.7355	0.7577	0.7627	**0.7701**
	AB	0.5599	0.7174	0.7731	0.7569	0.7017	**0.8018**	0.7610	0.7887	0.7953	0.7728
	KNN	0.7020	0.7873	**0.8286**	0.7620	0.7915	0.7893	0.7468	0.7859	0.7800	0.7957
7	RF	0.7128	0.7931	0.7817	0.7567	0.7938	**0.8094**	0.7511	0.7903	0.7977	**0.8094**
	AB	0.7059	0.7573	0.8005	0.7636	0.7709	**0.8060**	0.7578	0.7685	0.7947	0.7699
	KNN	0.9345	0.9605	0.9537	0.9567	**0.9616**	0.9598	0.9574	0.9598	0.9598	0.9609
8	RF	0.9237	0.9592	0.8879	0.9501	0.9499	0.9428	0.9255	0.9487	0.9529	**0.9719**
	AB	0.9752	0.9802	0.8932	0.9702	0.9780	0.9816	0.9494	0.9741	0.9778	**0.9849**
	KNN	0.5362	0.9121	0.7991	0.8768	0.9110	0.9750	0.9387	0.9704	0.9445	**0.9764**
9	RF	0.4892	0.9321	0.8379	0.8695	0.9185	**0.9791**	0.9131	0.9696	0.9503	0.9790
	AB	0.5559	0.8818	0.8469	0.8654	0.8689	0.9613	0.8861	0.9542	0.9358	**0.9764**
	KNN	0.4997	0.8521	0.7551	0.7897	0.8384	0.9316	0.8138	**0.9360**	0.9070	0.9249
10	RF	0.4874	0.8825	0.8194	0.8034	0.8841	0.9569	0.9487	0.9619	**0.9682**	0.9525
	AB	0.5303	0.8755	0.8517	0.8262	0.8767	0.9363	0.9358	0.9332	0.9337	**0.9518**
	KNN	0.8141	0.9165	0.9381	0.9218	0.9260	**0.9861**	0.9142	0.9792	0.9517	0.9459
11	RF	0.7728	0.9465	**0.9880**	0.9342	0.9627	0.9770	0.9778	0.9792	0.9688	0.9863
	AB	0.8138	0.9601	0.9856	0.9578	0.9590	0.9834	0.9788	0.9823	0.9834	**0.9871**
	KNN	0.6873	0.8345	0.8078	0.8238	0.8563	0.8521	0.8520	0.8493	0.8522	**0.8670**
12	RF	0.5599	0.7918	0.7811	**0.8343**	0.8081	0.8292	0.8305	0.8306	0.8304	0.8334
	AB	0.7430	0.8399	0.7481	0.8497	0.8579	0.8752	0.8747	**0.8755**	0.8749	0.8752
	KNN	0.6524	0.9193	0.8976	0.9082	0.9173	0.9321	0.9306	0.9312	0.9321	**0.9370**
13	RF	0.6260	0.8940	0.9406	0.9156	0.8989	0.9438	0.9440	**0.9445**	0.9440	0.9419
	AB	0.6957	0.9158	0.9306	0.9250	0.9179	0.9303	0.9309	0.9310	**0.9313**	0.9238
	KNN	0.7363	0.8170	0.8042	0.7948	0.8244	0.8099	0.7659	**0.8283**	0.8139	0.8151
14	RF	0.7197	0.8201	0.8392	0.7922	0.8115	0.8221	0.7898	0.8246	**0.8495**	0.8247
	AB	0.7576	0.8210	0.8082	0.8031	0.8100	0.8075	0.7961	0.8190	0.8263	**0.8435**
	KNN	0.8282	0.9532	0.9719	0.9556	0.9687	0.9867	0.9799	0.9860	0.9864	**0.9874**
15	RF	0.8082	0.9508	0.9735	0.9598	0.9651	0.9875	0.9853	0.9879	0.9877	**0.9882**
	AB	0.7820	0.9270	0.9717	0.9518	0.9462	0.9839	0.9823	0.9865	0.9860	**0.9866**
	KNN	0.6816	0.8129	0.8110	0.7989	0.8150	0.8179	0.8168	0.8178	0.8178	**0.8353**
16	RF	0.7153	0.8157	0.7326	0.8314	0.8159	0.8453	0.8413	0.8426	0.8354	**0.8546**
	AB	0.7299	0.8239	0.7330	0.8238	0.8273	0.8505	0.8496	**0.8518**	0.8495	0.8292

and other approaches like the GAN, WGAN-GP, and CWGAN-GP are not pronounced. This is attributed to the fact that GAN-based approaches tend to generate minority class examples that are more concentrated, often resulting in the creation of safe examples which are easier for classifiers to recognize (Napierala and Stefanowski, 2016). However, as demonstrated in Section 5.1, the data generated by the GWGAN-GP are more dispersed compared to these algorithms.

6. Conclusions and future work

In this study, we introduced an oversampling approach, the GWGAN-GP, to address the class imbalance problem.

Table 6. Obtained AUCs of each approach.

Dataset	Classifers	NONE	SMOTE	PFS	ProWSyn	SMOTE _IPF	GAN	WGAN	WGAN -GP	CWGAN -GP	GWGAN -GP
	KNN	0.6299	0.9596	0.9271	0.9455	0.9613	0.9683	0.9451	**0.9698**	0.9635	0.9690
1	RF	0.7190	**0.9795**	0.9662	0.9702	0.9794	0.9765	0.9529	0.9694	0.9698	0.9712
	AB	0.8162	0.9582	0.9782	0.9682	0.9655	**0.9868**	0.9398	0.9484	0.9583	0.9695
	KNN	0.9596	0.9807	0.9796	0.9592	0.9804	0.9792	0.9798	0.9797	0.9798	**0.981**8
2	RF	0.8875	0.9964	0.9791	**0.9983**	0.9975	0.9792	0.9792	0.9788	0.9791	0.9791
	AB	0.8879	0.9458	0.9247	0.9474	0.9477	0.9744	0.9807	**0.9902**	0.9804	0.9792
	KNN	0.9482	0.9363	0.9514	0.9361	0.9455	0.9611	0.9322	0.9611	0.9487	**0.9769**
3	RF	0.9623	0.9744	0.9639	0.9649	0.9787	0.9880	0.9870	0.9845	0.9839	**0.9911**
	AB	0.9506	0.9683	0.9568	0.9641	0.9738	0.9782	0.9837	0.9740	0.9714	**0.9955**
	KNN	0.6075	0.8743	0.8974	0.8672	0.8992	0.9247	0.9240	0.9199	0.9311	**0.9372**
4	RF	0.5971	0.9519	0.9350	0.9567	**0.9716**	0.9434	0.9420	0.9420	0.9336	0.9569
	AB	0.7923	0.9652	0.9279	0.9663	**0.9692**	0.9317	0.9309	0.9286	0.9365	0.9386
	KNN	0.8931	0.9009	0.9121	0.8996	0.8982	0.8980	0.8842	0.8910	0.8887	**0.9223**
5	RF	0.9086	0.9262	0.9054	0.8819	0.9294	**0.9311**	0.8827	0.9155	0.9095	0.8891
	AB	0.8829	0.8851	0.8868	0.8880	0.8927	**0.9228**	0.8715	0.9097	0.9132	0.8889
	KNN	0.6264	0.7716	0.7752	0.7248	0.7915	0.8068	0.6963	0.8050	0.7954	**0.8108**
6	RF	0.6283	0.7852	0.8303	0.7900	**0.8446**	0.7985	0.7789	0.8111	0.8089	0.8216
	AB	0.5640	0.7793	0.8236	0.7839	0.7406	0.8328	**0.8390**	0.8319	0.8368	0.7608
	KNN	0.7476	0.8446	**0.8778**	0.8247	0.8465	0.8390	0.7909	0.8378	0.8345	0.8540
7	RF	0.7989	0.8623	0.8826	0.8396	0.8737	**0.8861**	0.8397	0.8746	0.8779	0.8740
	AB	0.7923	0.8424	0.8688	0.8359	0.8464	**0.8823**	0.8353	0.8458	0.8692	0.8576
	KNN	0.9775	0.9865	0.9878	0.9842	0.9852	0.9867	0.9855	0.9866	0.9866	**0.9929**
8	RF	0.9870	0.9947	0.9797	0.9920	0.9935	0.9926	0.9824	0.9943	0.9930	**0.9969**
	AB	0.9971	0.9978	0.9763	0.9942	0.9988	0.9981	0.9859	0.9947	0.9954	**0.9995**
	KNN	0.5560	0.9450	0.8829	0.9232	0.9452	0.9786	0.9599	0.9769	0.9644	**0.9809**
9	RF	0.6620	**0.9828**	0.9153	0.9453	0.9773	0.9800	0.9603	0.9787	0.9736	0.9826
	AB	0.5917	0.9404	0.9003	0.9298	0.9407	**0.9821**	0.9409	0.9780	0.9639	0.9817
	KNN	0.5901	0.8931	0.8178	0.8353	0.8965	0.9563	0.8551	0.9590	0.9485	**0.9644**
10	RF	0.6245	0.9627	0.9036	0.8835	0.9667	0.9763	0.9678	**0.9790**	0.9735	0.9784
	AB	0.6939	0.9036	0.9133	0.8825	0.9232	0.9609	0.9666	0.9712	0.9666	**0.9762**
	KNN	0.7891	0.9694	**0.9874**	0.9605	0.9674	0.9851	0.9598	0.9770	0.9642	0.9705
11	RF	0.8861	0.9881	0.9908	0.9860	**0.9933**	0.9884	0.9889	0.9904	0.9781	0.9892
	AB	0.8584	0.9874	0.9886	0.9856	0.9886	0.9897	0.9888	0.9897	0.9883	**0.9898**
	KNN	0.7733	0.8754	0.8378	0.8884	0.9065	0.9072	0.9069	0.9047	0.9073	**0.9335**
12	RF	0.8414	0.8661	0.8334	0.9299	0.8832	0.9534	0.9514	0.9514	**0.9539**	0.9529
	AB	0.8782	0.9164	0.8244	0.9532	0.9313	**0.9624**	**0.9624**	**0.9624**	**0.9624**	0.9497
	KNN	0.7087	0.9556	0.9472	0.9546	**0.9560**	0.9333	0.9329	0.9330	0.9333	0.9349
13	RF	0.8262	0.9609	0.9449	**0.9718**	0.9616	0.9354	0.9342	0.9483	0.9364	0.9350
	AB	0.8800	**0.9763**	0.9373	0.9716	0.9755	0.9560	0.9410	0.9437	0.9383	0.9347
	KNN	0.7836	0.8456	0.8464	0.8358	0.8472	0.8642	0.8327	0.8687	0.8543	**0.8731**
14	RF	0.7926	0.8379	0.8598	0.8438	0.8664	0.8764	0.8455	0.8641	**0.8921**	0.8683
	AB	0.8578	0.8597	0.8434	0.8269	0.8547	0.8656	0.8498	0.8659	0.8788	**0.9000**
	KNN	0.8728	0.9701	0.9836	0.9778	0.9842	0.9876	0.9863	0.9871	0.9873	**0.9880**
15	RF	0.9253	0.9832	**0.9948**	0.9906	0.9940	0.9911	0.9945	0.9903	0.9880	0.9906
	AB	0.9126	0.9663	0.9922	0.9893	0.9864	0.9956	0.9950	**0.9962**	0.9835	0.9900
	KNN	0.7454	0.8663	0.8784	0.8622	0.8699	0.8796	0.8793	0.8797	0.8797	**0.9083**
16	RF	0.8280	0.8927	0.8332	0.9254	0.8930	0.9368	0.9357	0.9377	0.9398	**0.9409**
	AB	0.8424	0.9169	0.8214	0.9343	0.9177	**0.9416**	0.9414	0.9415	0.9412	0.9313

Utilizing the Gaussian distribution as label information within the WGAN-GP framework, we generated new examples for the minority class. The GWGAN-GP algorithm was then integrated with three classifiers (k-NN, RF, AB), and their performance was assessed across 16 datasets using metrics such as recall, F1 score, G-mean, and the AUC.

On the basis of classification experimental results and statistical tests, we have the following conclusions.

(i) Compared with other GAN-based approaches, the GWGAN-GP can separate the generated examples, reduce the repetition of single-class data, and avoid issues such as wasting resources, overlooking important features, and exhibiting overly optimistic

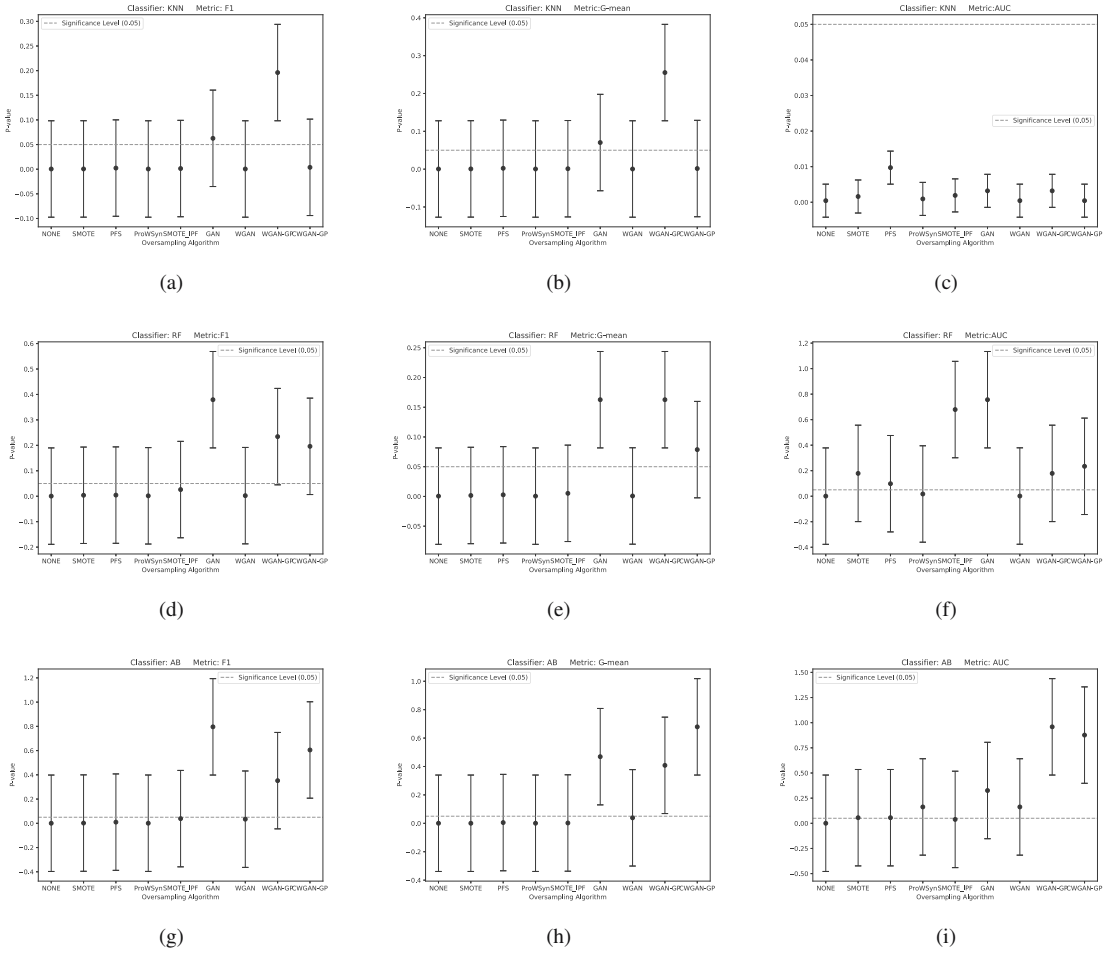


Fig. 5. Results of the Wilcoxon signed-rank test.

performance on the test set.

(ii) When combined with different classifiers, the GWGAN-GP performs well on performance metrics.

(iii) While the GWGAN-GP may not outperform other oversampling methods on all classifiers, we observed that the combination of the GWGAN-GP algorithm with the kNN classifier significantly outperforms the other oversampling approaches.

Despite the performance of our approach and its ability to reduce duplication, finding the most appropriate values for the mean and standard deviation of the Gaussian distribution remains a challenge. Setting too small parameters results in high duplication among the generated examples, while setting them too large significantly deviates from the overall distribution of the real dataset. Although this paper employed the key feature analysis technique, it may not be suitable for all datasets, and parameter tuning is still necessary to find appropriate values for the mean and standard deviation in some datasets. In future research, it would be worthwhile to explore improvements to the approach that enable it to automatically generate the most suitable mean and standard deviation based on the characteristics of the dataset itself.

In addition, our proposed algorithm did not show a significant improvement in classification performance compared with the other oversampling approaches. One reason for this is that the oversampling approaches we selected had already been proven to be effective (Kovács, 2019). We are considering further efforts to enhance the algorithm's classification performance, especially when applied to specific datasets, to explore the potential for greater improvement. The second reason is that GAN-based oversampling approaches share similar principles in data generation. The distinction lies in our approaches' ability to directly disperse generated data, making it closer to the distribution of real data. However, an excessive concentration of duplicated data might lead to better classifier performance (Napierala and Stefanowski, 2016). To mitigate

this issue, we employed five-fold cross-validation. This explains why our method did not outperform other GAN-based oversampling approaches overall. In summary, GAN-based oversampling approaches demonstrate efficient capabilities in generating new non-image examples, warranting further exploration and application in real-world class imbalance problems.

Acknowledgment

This work was supported by the Natural Science Foundation of Shandong Province (ZR2023MF098).

References

Arjovsky, M., Chintala, S. and Bottou, L. (2017). Wasserstein generative adversarial networks, *International Conference on Machine Learning, Sydney, Australia*, pp. 214–223.

Barua, S., Islam, M.M. and Murase, K. (2013). PROWSYN: Proximity weighted synthetic oversampling technique for imbalanced data set learning, *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia*, pp. 317–328.

Bourou, S., El Saer, A., Velivassaki, T.-H., Voulkidis, A. and Zahariadis, T. (2021). A review of tabular data synthesis using GANs on an IDS dataset, *Information* **12**(09): 375.

Breiman, L. (2001). Random forests, *Machine Learning* **45**(1): 5–32.

Breiman, L. (2017). *Classification and Regression Trees*, Routledge, London.

Budach, L., Feuerpfeil, M., Ihde, N., Nathansen, A., Noack, N., Patzlaff, H., Naumann, F. and Harmouch, H. (2022). The effects of data quality on machine learning performance, *arXiv:* 2207.14529.

Chaabane, I., Guermazi, R. and Hammami, M. (2020). Enhancing techniques for learning decision trees from imbalanced data, *Advances in Data Analysis and Classification* **14**(3): 1–69.

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002). SMOTE: Synthetic minority over-sampling technique, *Journal of Artificial Intelligence Research* **16**: 321–357.

Chen, J., Huang, H., Cohn, A.G., Zhang, D. and Zhou, M. (2022). Machine learning-based classification of rock discontinuity trace: SMOTE oversampling integrated with GBT ensemble learning, *International Journal of Mining Science and Technology* **32**(2): 309–322.

Chen, J., Yan, Z., Lin, C., Yao, B. and Ge, H. (2023). Aero-engine high speed bearing fault diagnosis for data imbalance: A sample enhanced diagnostic method based on pre-training WGAN-GP, *Measurement* **213**(7): 112709.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification, *IEEE Transactions on Information Theory* **13**(1): 21–27.

Cui, J., Zong, L., Xie, J. and Tang, M. (2023). A novel multi-module integrated intrusion detection system for high-dimensional imbalanced data, *Applied Intelligence* **53**(1): 272–288.

Derrac, J., Garcia, S., Sanchez, L. and Herrera, F. (2015). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, *Journal of Multiple-Valued Logic and Soft Computing* **17**(2–3): 255–287.

Douzas, G. and Bacao, F. (2018). Effective data generation for imbalanced learning using conditional generative adversarial networks, *Expert Systems with Applications* **91**(1): 464–471.

Dua, D. and Graff, C. (2019). *UCI Machine Learning Repository*, `http://archive.ics.uci.edu/ml`.

Fernández, A., Garcia, S., Herrera, F. and Chawla, N.V. (2018). Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary, *Journal of Artificial Intelligence Research* **61**: 863–905.

Freund, Y. and Schapire, R.E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences* **55**(1): 119–139.

García, S., Luengo, J. and Herrera, F. (2016). Tutorial on practical tips of the most influential data preprocessing algorithms in data mining, *Knowledge-Based Systems* **98**(7): 1–29.

Gazzah, S. and Amara, N.E.B. (2008). New oversampling approaches based on polynomial fitting for imbalanced data sets, *2008 8th IAPR International Workshop on Document Analysis Systems, Nara, Japan*, pp. 677–684.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative adversarial nets, *Advances in Neural Information Processing Systems* **27**: 2672–2680.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A.C. (2017). Improved training of Wasserstein GANs, *Advances in Neural Information Processing Systems* **30**: 5767–5777.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection, *Journal of Machine Learning Research* **3**(Mar): 1157–1182.

He, H. and Garcia, E.A. (2009). Learning from imbalanced data, *IEEE Transactions on Knowledge and Data Engineering* **21**(9): 1263–1284.

Hernandez, M., Epelde, G., Alberdi, A., Cilla, R. and Rankin, D. (2022). Synthetic data generation for tabular health records: A systematic review, *Neurocomputing* **493**(27): 28–45.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications to R*, 2nd Edn, Springer, New York.

Janicka, M., Lango, M. and Stefanowski, J. (2019). Using information on class interrelations to improve classification of multiclass imbalanced data: A new resampling

algorithm, *International Journal of Applied Mathematics and Computer Science* **29**(4): 769–781, DOI: 10.2478/amcs-2019-0057.

Japkowicz, N. (2003). Class imbalances: Are we focusing on the right issue, *Workshop on Learning from Imbalanced Data Sets II, Washington, USA*, p. 63.

Kaggle (2024), Datasets: *Lower Back Pain*, `https://www.kaggle.com/datasets/sammy123/lower-back-pain-symptoms-dataset`, and *Telecom Churn*, `https://www.kaggle.com/datasets/mnassrib/telecom-churn-datasets`.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection, *14th International Joint Conference on Artificial Intelligence (IJCAI), Montreal, Canada*, pp. 1137–1145.

Kovács, G. (2019). An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets, *Applied Soft Computing* **83**(9): 105662.

Liu, X.-Y., Wu, J. and Zhou, Z.-H. (2008). Exploratory undersampling for class-imbalance learning, *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics* **39**(2): 539–550.

López, V., Fernández, A., García, S., Palade, V. and Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, *Information Sciences* **250**(33): 113–141.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets, *arXiv:* 1411.1784.

Miyato, T., Kataoka, T., Koyama, M. and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks, *arXiv:* 1802.05957.

Moreo, A., Esuli, A. and Sebastiani, F. (2016). Distributional random oversampling for imbalanced text classification, *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, Pisa, Italy*, pp. 805–808.

Napierala, K. and Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers from imbalanced data, *Journal of Intelligent Information Systems* **46**: 563–597.

Nik, A.H.Z., Riegler, M.A., Halvorsen, P. and Storås, A.M. (2023). Generation of synthetic tabular healthcare data using generative adversarial networks, *International Conference on Multimedia Modeling, Bergen, Norway*, pp. 434–446.

Ohsaki, M., Wang, P., Matsuda, K., Katagiri, S., Watanabe, H. and Ralescu, A. (2017). Confusion-matrix-based kernel logistic regression for imbalanced data classification, *IEEE Transactions on Knowledge and Data Engineering* **29**(9): 1806–1819.

Park, N., Mohammadi, M., Gorde, K., Jajodia, S., Park, H. and Kim, Y. (2018). Data synthesis based on generative adversarial networks, *Proceedings of the VLDB Endowment* **11**(10): 1071–1083.

Park, S. and Park, H. (2021). Combined oversampling and undersampling method based on slow-start algorithm for imbalanced network traffic, *Computing* **103**(3): 401–424.

Powers, D.M. (2020). Evaluation: From precision, recall and f-measure to ROC, informedness, markedness and correlation, *arXiv:* 2010.16061.

Ren, J., Wang, Y., Cheung, Y.-m., Gao, X.-Z. and Guo, X. (2023). Grouping-based oversampling in kernel space for imbalanced data classification, *Pattern Recognition* **133**(1): 108992.

Sáez, J.A., Luengo, J., Stefanowski, J. and Herrera, F. (2015). SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, *Information Sciences* **291**(2): 184–203.

Sun, B., Zhou, Q., Wang, Z., Lan, P., Song, Y., Mu, S., Li, A., Chen, H. and Liu, P. (2023). Radial-based undersampling approach with adaptive undersampling ratio determination, *Neurocomputing* **553**(39): 126544.

Sun, Y., Wong, A.K. and Kamel, M.S. (2009). Classification of imbalanced data: A review, *International Journal of Pattern Recognition and Artificial Intelligence* **23**(04): 687–719.

Wasserman, L. (2004). *All of Statistics: A Concise Course in Statistical Inference*, Springer, New York.

Wold, S., Esbensen, K. and Geladi, P. (1987). Principal component analysis, *Chemometrics and Intelligent Laboratory Systems* **2**(1–3): 37–52.

Woods, K.S., Doss, C.C., Bowyer, K.W., Solka, J.L., Priebe, C.E. and Kegelmeyer Jr, W.P. (1993). Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography, *International Journal of Pattern Recognition and Artificial Intelligence* **7**(06): 1417–1436.

Xie, Y. and Zhang, T. (2018). Imbalanced learning for fault diagnosis problem of rotating machinery based on generative adversarial networks, *2018 37th Chinese Control Conference (CCC), Wuhan, China*, pp. 6017–6022.

Xu, L., Skoularidou, M., Cuesta-Infante, A. and Veeramachaneni, K. (2019). Modeling tabular data using conditional GAN, *Advances in Neural Information Processing Systems* **32**: 7335–7345.

Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O. and Li, H. (2017). High-resolution image inpainting using multi-scale neural patch synthesis, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA*, pp. 6721–6729.

Zhang, M., Wan, X., Gang, L., Lv, X., Wu, Z. and Liu, Z. (2021). An automated driving strategy generating method based on WGAIL–DDPG, *International Journal of Applied Mathematics and Computer Science* **31**(3): 461–470, DOI: 10.34768/amcs-2021-0031.

Zhang, Y., Liu, Y., Wang, Y. and Yang, J. (2023). An ensemble oversampling method for imbalanced classification with prior knowledge via generative adversarial network, *Chemometrics and Intelligent Laboratory Systems* **235**(4): 104775.

Zhao, Y., Li, H., Bissyandé, T.F., Klein, J. and Grundy, J. (2021). On the impact of sample duplication in machine-learning-based android malware detection, *ACM Transactions on Software Engineering and Methodology* **30**(3): 1–38.

Zhao, Z., Kunar, A., Birke, R. and Chen, L.Y. (2021). CTAB-GAN: Effective table data synthesizing, *Asian Conference on Machine Learning*, pp. 97–112, (virtual).

Zheng, M., Li, T., Zhu, R., Tang, Y., Tang, M., Lin, L. and Ma, Z. (2020a). Conditional wasserstein generative adversarial network-gradient penalty-based approach to alleviating imbalanced data classification, *Information Sciences* **512**(7): 1009–1023.

Zheng, W. and Zhao, H. (2020b). Cost-sensitive hierarchical classification for imbalance classes, *Applied Intelligence* **50**(8): 2328–2338.

Zhu, B., Pan, X., vanden Broucke, S. and Xiao, J. (2022). A GAN-based hybrid sampling method for imbalanced customer classification, *Information Sciences* **609**(28): 1397–1411.

Qian Zhou received her master's degree in computer science and technology from Shandong Normal University in 2014. She now works as an assistant professor in the Department of Computer Science and Technology at Shandong Agricultural University. Her research interests include machine learning and data mining, and she has published several international papers in this field.

Bo Sun received his PhD degree in computer science and technology from the Nanjing University of Aeronautics and Astronautics in 2016. He now works as an associate professor in the Department of Computer Science and Technology at Shandong Agricultural University. His research interests include machine learning and artificial neural networks, and he has published several high quality journal papers in this field.

Received: 31 July 2023
Revised: 12 December 2023
Re-revised: 16 February 2024
Accepted: 21 February 2024