Computer Science and Information Systems 22(3):673–691 https://doi.org/10.2298/CSIS231113014K

A GAN-Based Hybrid Approach for Addressing Class Imbalance in Machine Learning

Dae-Kyoo Kim[1] and Yeasun K. Chung[2]

[1] Computer Science and Engineering, Oakland University
Rochester, Michigan 48309, USA
kim2@oakland.edu
[2] Spears School of Business, Oklahoma State University
Stillwater, Oklahoma 74078, USA
y.chung@okstate.edu

Abstract. Class imbalance is a common problem in machine learning where the majority class has a significantly higher number of instances than the minority class, which leads to bias towards the majority class. The problem can be effectively addressed by using Generative Adversarial Network (GAN) to generate realistic synthetic samples. In this work, we present a GAN-based approach that makes use of hybrid models that combine oversampling techniques with undersampling and ensemble techniques to reduce overfitting. The proposed approach was evaluated on two datasets with different level of class imbalance using six widely used classifiers and compared with two popular class balancing techniques – SMOTEENN and SMOTETomek. The results show that the proposed approach outperforms them in highly imbalanced datasets.

Keywords: class imbalance, classification, GAN, hybrid model, machine learning.

1. Introduction

Class imbalance is a prevalent issue that can arise in many machine learning problems where one class has a significantly higher number of instances than the other class. For instance, in fraud detection (e.g., [11,35]), the majority of observations may be negative cases (e.g., no fraud), while only a few cases are positive, but critical (e.g., fraudulent transactions). Class imbalance may lead machine learning to be biased towards the majority class at the expense of the minority class [14,15,20], resulting in lower performance metrics (e.g., precision, recall, F1 score) [5].

To address class imbalance, oversampling techniques such as Synthetic Minority Oversampling Technique (SMOTE) [7,36] have been used [4]. More recently, there has been much work (e.g., [1,3,9,10,18,19,22,24,31,32]) using Generative Adversarial Network (GAN) [13] for oversampling which can generate more diverse and realistic synthetic samples, leading to competitive and, in some cases, superior performance to SMOTE. Oversampling can address class imbalance by creating artificial samples for the minority class, but it may lead to an overfitting problem. To mitigate the problem, hybrid models such as SMOTEENN and SMOTETomek have been proposed, combining an oversampling technique with an undersampling techniques like Edited Nearest Neighbours (ENN) [33] and Tomek-Links [29], which help remove noisy samples in the majority

class. However, the use of GAN as an oversampling technique with undersampling techniques has not been studied.

In this paper, we present a GAN-based hybrid approach to address class imbalance. The approach introduces a set of GAN-based hybrid models – GANBoost, GANENN, GANRUS, GANRUSBoost, and GANTomek, which combine a GAN with undersampling or ensemble techniques such as AdaBoost [12], Edited Nearest Neighbors (ENN) [7], Random Under Sampling (RUS) [27], RUSBoost (RUS+AdaBoost), and TomekLinks [29]. These models enable generating realistic samples by GANs, while removing ambiguous examples in the majority class using undersampling and ensemble techniques

We evaluated the proposed approach using two datasets – Hotel Booking Cancellation (HBC) and Financial Fraud Detection (FFD) on six different classifiers – Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), XGBoost (XGB), k-Nearest Neighbors (KNN), and Light Gradient Boosting Machine (LGBM). The datasets were purposely chosen from different domains with notably varying sizes and degree of class imbalance for the sake of diversity. The classifiers were chosen for their popular use for binary classification as concerned in the datasets. The performance of the GAN-based hybrid models are compared with SMOTEENN and SMOTETomek which are widely used hybrid models for class imbalance [4]. The results showed that GAN-based hybrid models consistently outperform both SMOTEENN and SMOTETomek in the highly imbalanced FFD dataset, which suggests the effectiveness of the proposed approach for significant class imbalance. The contributions of the work are as follows.

- Introducing a GAN-based hybrid approach to address class imbalance.
- Evaluating the approach on datasets that vary in domain, size, and degree of class imbalance.
- Comparing the performance of the approach with widely used hybrid models, and demonstrating the effectiveness of the approach on highly imbalanced datasets.

The paper is organized as follows. Section 2 gives an overview of related work using GAN to address class imbalance. Section 3 describes commonly used hybrid techniques for dealing with class imbalance. Section 4 presents the proposed GAN-based hybrid approach. Section 5 discusses the benchmark datasets used in the evaluation of the approach. Section 6 describes the results of the evaluation and compares them with existing hybrid techniques. Finally, Section 7 concludes the paper with a discussion of future research.

2. Related Work

Odena et al. [24] introduced the Auxiliary Classifier Generative Adversarial Network (AC-GAN) model to improve the training and quality of GANs for image synthesis. This model incorporates label conditioning to enable the generation of high-resolution images with greater global coherence across all classes of the ImageNet dataset. The AC-GAN model includes an auxiliary classifier within the discriminator to output class labels for training data, making the model class-conditional but also capable of reconstructing class labels. The model was evaluated using discriminability and diversity metrics, showing that 128x128 samples were more than twice as discriminable as 32x32 samples and 84.7% of classes matched or exceeded the diversity of real ImageNet data.

Mariani et al. [22] introduced Balancing Generative Adversarial Network (BAGAN) to address class imbalance in image classification by generating image samples via initializing the autoencoder of the discriminator and generator of a GAN. Class conditioning was used in the latent space to steer the generation process of image samples in a certain direction. They reported that the initialization of the autoencoder helped learn class conditioning and reduce convergence issues that arise in conventional GANs.

Antoniou et al. [3] presented Data Augmentation Generative Adversarial Network (DAGAN) to learn a model of a larger invariance space using a conditional GAN. The learned DAGAN was used to improve few-shot target domains by augmenting the data in Matching networks [30] with relevant comparator points generated from the DAGAN. The approach was evaluated on three datasets, Omniglot, EMNIST, and VGG-Face for classification tasks via vanilla classifiers (e.g., DenseNet) and Matching networks.

Wang et al. [31] introduced a traffic data augmentation method called PacketCGAN, which extends their earlier work [32] using Conditional GAN. The method was evaluated using three deep learning (DL) models on four types of encrypted traffic datasets and compared with ROS and SMOTE, and vanilla GAN. They reported that the PacketCGAN outperformed the compared methods.

Wang et al. [32] proposed FlowGAN, a GAN-based method to generate synthesized samples for encrypted traffic classification. The synthesized data was combined with real data to create a new training dataset. They evaluated the method against a Multi-Layer Perceptron (MLP) on three datasets and reported that FlowGAN outperformed the MLP on both the imbalanced dataset and oversampled dataset.

Fiorea et al. [10] used a GAN to address class imbalance in credit card fraud detection. The GAN was used to generate synthetic examples from the original minority class which were merged with original data to obtain an augmented training set. They reported that the classifier trained on the augmented set outperformed the same classifier trained on the original data.

Ali-Gombe and Elyan [1] proposed Multiple Fake Class Generative Adversarial Network (MFC-GAN) to address class imbalance in multi-classification tasks. MFC-GAN uses a multi-fake class GAN to preserve the structure of the minority class and generate samples for each class by learning the correct data distribution. The approach conditions sample generation on real class labels only and modifies the classification objective to reduce noise appearing in generated samples. They reported that MFC-GAN results in improved performance.

Jiang et al. [16] presented an anomaly detection method using GANs, specifically designed for imbalanced time series data. The method involves an encoder-decoder-encoder architecture within the generator and is trained on normal samples to understand the distribution of normal data. This approach addresses class imbalance by focusing on learning the distribution of the normal class and identifying deviations as anomalies during testing. The method was tested on benchmark rolling bearing datasets, which are widely used in the field of mechanical engineering to evaluate the performance of diagnostic algorithms and models. They reported that their method achieved 100% accuracy in distinguishing between normal and abnormal samples.

Lei et al. [19] proposed Imbalanced Generative Adversarial Fusion Network (IGAFN) to address class imbalance in credit scoring task. The network consists of a fusion module and a balance module, The fusion model is used for feature exploration, leveraging a feed-

forward neural network and bidirectional long short-term memory (Bi-LSTM) network to learn user profile and behavior data. The balance module is used for data generation, applying an imbalanced generative adversarial network (IGAN) to approximate the real data distribution and generate new samples for the minority class. The network uses a minimax algorithm to optimize the generative and discriminative networks.

Similar to Mariani et al.'s work [22], Kim et al. [17] proposed a GAN-based model to address class imbalance in defect detection within industrial inspections. The model incorporates an autoencoder as the generator, alongside two distinct discriminators for normal and anomalous inputs. They introduced Patch Loss and Anomaly Adversarial Loss functions to optimize the training process. The model was evaluated on MNIST, Fashion MNIST, CIFAR-10/100, and a real-world industrial dataset concerning smartphone case defects, achieving an average accuracy of 99.03% on the latter.

Qasim et al. [25] presented Red-GAN, a GAN-based approach equipped with class conditioning and a segmentor to mitigate class imbalance in medical imaging. This GAN is conditioned both at the pixel-level and global-level, allowing for controlled synthesis of image-label pairs tailored to specific classes. A segmentor is incorporated to ensure that the synthesized images are relevant for segmentation tasks. The method was experimented on two medical datasets – BraTS and ISIC, showing increases in DICE scores of up to 5% and 2%, respectively.

Lee and Park [18] used a vanilla GAN to address imbalanced data in intrusion detection and compared the performance of the GAN with SMOTE and single RF with no handling of data imbalance. They reported that their model outperformed both SMOTE and single RF.

Similar to Lei et al.'s work [19], Engelmann and Lessmann [9] proposed a GAN-based approach for oversampling data in credit scoring. The method was evaluated on seven credit scoring datasets and compared against benchmark oversampling methods (e.g., SMOTE, Random Oversampling) as well as without any oversampling methods, using five different classification algorithms (e.g., RF, DT, KNN). Their method outperformed four variants of SMOTE and Random Oversampling on most datasets. However, predictions made without any oversampling method generally performed better than those using SMOTE variants, and maintaining the original class distribution also delivered competitive results.

Yang and Zhou [34] introduced Imbalanced Data Augmentation Generative Adversarial Network (IDA-GAN) to tackle class imbalance in datasets. IDA-GAN incorporates a variational autoencoder to learn class distributions in the latent space, allowing for the generation of diverse samples of minority classes, thus mitigating the mode collapse issue [26] in GANs. The approach was experimented on five benchmark datasets (MNIST, Fashion-MNIST, SVHN, CIFAR-10, and GTSRB). They compared their work with AC-GAN [24] and BAGAN [22], reporting outperformance in precision, recall, and F1-score.

Bhagwani et al. [6] used GANs to address class imbalance in datasets by generating synthetic samples of the minority class. The method was evaluated on a credit card fraud detection dataset using a Support Vector Machine (SVM) for the classification of the generated samples. They reported that their approach demonstrates a classification accuracy of 99.89%, compared to SMOTE's 58.29%.

Deng et al. [8] presented Imputation Balanced Generative Adversarial Network (IB-GAN) to address the classification of multivariate time series data with strong class im-

balance. IB-GAN combines data augmentation and classification in a unified process using an imputation-balancing approach. This method employs imputation and resampling techniques to generate synthetic samples from randomly masked vectors, improving the classification of minority classes. The approach was tested on the UCR data and a 90K product dataset. They reported that their work outperforms similar existing work in F1-score.

Sharma et al. [28] proposed SMOTified-GAN which combines SMOTE and GANs to address class imbalance in datasets. It uses SMOTE for the initial oversampling of the minority class, which is then refined through GANs to produce more samples. The method was experimented on various datasets with performance improvements of up to 9% on F1-score measurements compared to other algorithms.

In summary, most of the discussed works reported the positive effect of GAN on addressing class imbalance. However, no work studied the effect of GAN with hybrid techniques of oversampling and undersampling.

3. Hybrid Class Balancing Techniques

In this section, we discuss two commonly used hybrid techniques for addressing class imbalance, namely SMOTEENN and SMOTETomek.

SMOTEENN is a data sampling method that combines SMOTE [7] and the ENN algorithm [33] to address class imbalance in datasets. SMOTEENN applies the SMOTE method to create synthetic samples for the minority class, which are generated by interpolating between the nearest neighbors of each minority class observation. This increases the number of minority class samples and helps balance the class distribution. However, SMOTE can also generate noisy samples that can negatively impact the model's performance. To address this issue, the ENN algorithm is applied, which removes samples that are misclassified by their k-nearest neighbors. For each sample, the k-NN algorithm is used to locate its nearest neighbors. Subsequently, the class of each neighbor is compared to the class of the observation. If there is a difference in class, indicating potential anomalies, both the observation and its corresponding k-nearest neighbors are eliminated from the dataset. Let D denote the dataset, and m be an individual observation within D. The class of observation m is then denoted as C_m, while the majority class of its k-nearest neighbors, denoted as k, is represented by C_k. If C_m is not equal to C_k, it signifies a discrepancy in class labels. In such cases, the observation m and its k-nearest neighbors are removed from the dataset using the set operation $D \setminus m \cup k$. This process ensures that only relevant samples are included in the dataset, which improves the overall quality of the data and reduces the impact of noise. Figure 1 illustrates the under-sampling by ENN when $k = 4$ and an example of before and after application of SMOTEENN. The example uses a synthetic classification dataset with 300 samples, 4 features, and 2 classes in a weight of 0.8 for the majority class and a weight of 0.2 for the minority class. The combination of SMOTE and ENN leads to a more balanced dataset with reduced noise, making it a useful method for improving the performance of machine learning models on imbalanced datasets [4].

SMOTETomek is another hybrid technique that combines SMOTE with Tomek Links [29]. After generating synthetic samples by SMOTE, Tomek Links identifies pairs of samples belonging to opposite classes where one sample is from the majority class and the

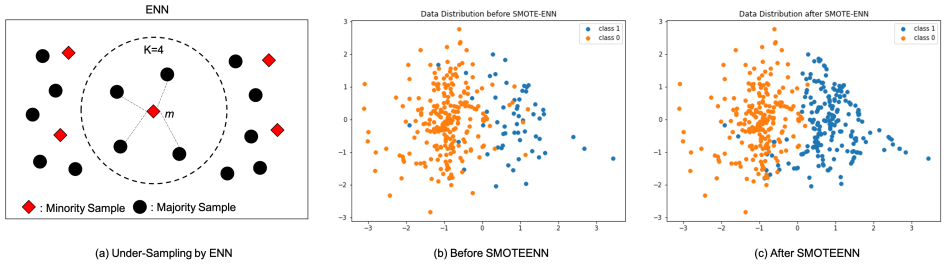

Fig. 1. Data Balancing by SMOTEENN

other is from the minority class. These paired samples are considered the closest neighbors to each other. To determine if a pair of samples, denoted as (x_i, x_j), forms a Tomek link, the Euclidean distance between them, represented as $d(x_i, x_j)$, is calculated. For a given minority class sample x_i and a majority class sample x_j, a Tomek link exists if there is no other sample x_k in the k-nearest neighbors such that $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j, x_k) < d(x_j, x_i)$. That is, a Tomek link is present when there are no neighboring samples of x_i and x_j that are closer to each other than x_i and x_j. The decision boundary between classes is improved by removing the majority class instance from a Tomek link. The process of applying SMOTE and Tomek links is repeated until a balanced dataset is achieved. Figure 1 illustrates the undersampling by Tomek when $k = 3$ and an example of before and after application of SMOTETomek.

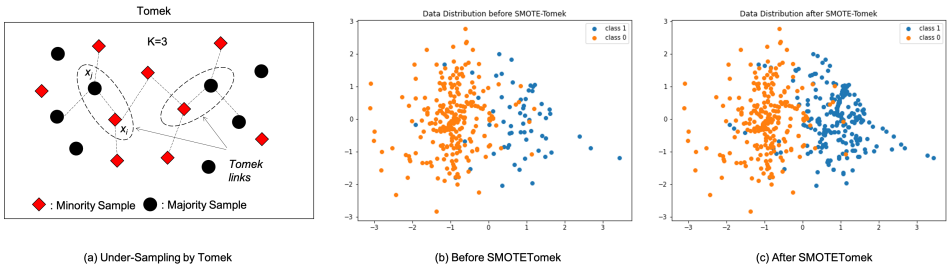

Fig. 2. Data Balancing by SMOTETomek

4. GAN-Based Hybrid Models

In this work, we present a GAN-based hybrid approach that makes use of GANs as an oversampling technique together with undersampling techniques (e.g., ENN, Tomek Links, RUS) or ensemble techniques (e.g., AdaBoost, RUSBoost) to address class imbalance problems. The approach enables creating more diverse and realistic synthetic samples to better capture the underlying data distribution using a GAN, while removing noisy,

irrelevant, or ambiguous examples in the majority class using an undersampling/ensemble technique, which helps reduce overfitting.

A GAN consists of the generator G, the discriminator D, and the classifier C [23]. G takes random noise z as input and generates synthetic samples $G(z)$, learning to map the noise distribution to the distribution of the minority class. The function $G : \mathcal{Z} \rightarrow \mathcal{T} \rightarrow \mathbb{R}^{n_i}$ can be defined as follows:

$$G(z) = g_i(t(z|i))$$

where $t : \mathcal{Z} \rightarrow \mathcal{T}$ maps the latent space z to the intermediate space $\mathcal{T}$, and $g_i : \mathcal{T} \rightarrow \mathbb{R}^{n_i}$ maps the intermediate space $\mathcal{T}$ to a vector of weights for the existing points n_i in the minority class X_i using the softmax activation function. The objective of the generator is to generate synthetic samples that resemble the real samples from the minority class by fooling the discriminator, aiming to maximize the probability of the discriminator misclassifying the synthetic samples as real:

$$\max_{\forall s \in G(z)} D(s)$$

The discriminator D distinguishes between real and synthetic samples, learning to classify whether a given sample is from the minority class or generated by the generator. The function $D : \mathbb{R}^{n_i} \rightarrow [0, 1]$ takes a real sample x from the minority class and outputs a probability $D(x)$. The objective of the discriminator is to correctly identify the real samples and distinguish them from the synthetic ones. That is, the discriminator aims at maximizing the probability of correctly classifying real samples, while minimizing the probability of misclassifying synthetic samples:

$$\max_{\forall x \in X_i} D(x) \wedge \min_{\forall s \in G(z)} D(s)$$

The classifier C evaluates the performance of the generated synthetic samples. It is initially trained on the original imbalanced dataset and then is used to assess the quality of the synthetic samples produced by the generator through a two-player game between the generator and the discriminator. The generator tries to produce synthetic samples that can fool the discriminator, while the discriminator aims to correctly classify between real and synthetic samples. This adversarial training process continues iteratively through backpropagation and gradient descent until the generator is capable of generating realistic synthetic samples that are indistinguishable from the real minority class samples according to the discriminator.

Undersampling techniques reduce the number of instances in the majority class to balance the distribution of the target variable. In addition to ENN and Tomek Links discussed in Section 3, we also use Random Under Sampling (RUS) [27] for undersampling. RUS reduces the size of the majority class by randomly selecting a subset of examples from the majority class, resulting in a more balanced dataset. The removed examples are discarded or used for validation purposes. This technique is simple and computationally efficient, making it a popular choice for addressing class imbalance. When combined with GANs, RUS can help reduce overfitting in training.

Ensemble techniques combine multiple classifiers to improve classification performance. For ensemble techniques, we utilize Adaptive Boosting (AdaBoost) [12] and Random Under-Sampling Boosting (RUSBoost). AdaBoost is a boosting algorithm that combines multiple weak classifiers into a strong ensemble. In each iteration, AdaBoost assigns

more weight to the misclassified examples, which allows the subsequent weak classifiers to focus on the misclassified examples and improve their performance. By combining the predictions of all the weak classifiers, AdaBoost generates a strong classifier that can accurately classify both the majority and minority classes. RUSBoost is a hybrid method that combines AdaBoost with random undersampling. RUSBoost first randomly undersamples the majority class to balance the class distribution and then applies the AdaBoost algorithm to the balanced dataset. By randomly removing examples from the majority class, RUSBoost reduces the computational cost of the AdaBoost algorithm while still providing a balanced dataset for training.

Figure 3 presents the proposed GAN-based hybrid approach. It first computes the number of minority samples by summing up all the samples labelled "1" (representing the minority class) and the number of majority samples by subtracting the minority count from the total number of samples. Then, it determines the significance of class imbalance by comparing the ratio of the minority count to the majority count with the threshold value. If the class imbalance is significant, it computes the number of synthetic minority samples to generate based on the difference between the majority and minority classes and the user-defined percentage of synthetic samples to generate. The GAN is then trained on the minority class samples, and synthetic minority samples are generated using the trained GAN. These synthetic minority samples are combined with the original dataset. The combined dataset undergoes balancing using Tomek, ENN, RUS, RUSBoost, or AdaBoost techniques to remove noisy and ambiguous samples introduced by the GAN. Finally, the algorithm returns the resampled dataset with a reduced class imbalance.

An instance of the GAN model is created using the $build\_gan()$ function based on the dimensionality of the input data and the dimensionality of the noise vector input. The function first builds a generator and a discriminator for the GAN instance using the $build\_generator()$ and $build\_discriminator()$ function, respectively and then combine them sequentially to create a GAN instance which is compiled with binary cross-entropy loss and $Adam$ optimizer.

The $build\_generator()$ function takes in a noise vector of length as input and outputs a synthetic sample with the same shape as the input data of dimension. It creates a sequential model with three layers. The first layer is a dense layer with 128 neurons and the input dimension (i.e., the noise vector). The second layer is the $LeakyReLU$ activation function with a slope of 0.2, which helps prevent the generator from collapsing and improves the quality of the generated samples. The third layer is another dense layer with the hyperbolic tangent ($tanh$) activation function, which scales the output values to be between -1 and 1, similar to the range of the real data.

The $build\_discriminator()$ function takes in the input dimension of the discriminator network and creates a sequential model with four layers. The first layer is a dense layer with 128 nodes and the second layer is an activation layer with $LeakyReLU$ introducing nonlinearity to the output of the first layer. A dropout layer is added after the activation function to prevent overfitting. It randomly drops out 50% of the nodes in the layer during training. Lastly, a dense output layer with a single node and the $sigmoid$ activation function is added to produce a scalar output indicating the probability that the input data is from the minority class. The model is then compiled with binary cross-entropy loss and $Adam$ optimizer.

Algorithm 1: GAN-based hybrid model

Input: X: array-like or sparse matrix, shape (n_samples, n_features); y: array-like, shape (n_samples,); $synth\_ratio$: float, the percentage of synthetic samples to generate; $threshold$: float, the threshold value for significant class imbalance; $epochs$: int, the number of epochs to train the GAN; $batch\_size$: int, the number of samples per batch to use when training the GAN; $latent\_dim$: int, the size of the noise vector input to the generator;

Output: X_bal, y_bal: balanced dataset with synthetic samples

```
G = (majority_count - minority_count) * synth_ratio          ▷ Computes # of synthetic minority samples to generate
gan = build_gan(X_min, latent_dim)                           ▷ Build a GAN on the minority class in X
for epoch in 1 to epochs do
    train_gan(X_min, generator, discriminator, gan, latent_dim, batch_size)   ▷ Train the GAN
end
X_syn = gan.generate_samples(G)                              ▷ Generate synthetic minority samples
y_syn = [1] * G                                              ▷ Assign a label of 1 to the synthetic minority samples
X_bal = combine_datasets(X, X_syn)
y_bal = combine_datasets(y, y_syn)
perform Tomek|ENN|RUS|RUSBoost|AdaBoost on combined samples
return X_bal, y_bal
def build_gan(input_dim, latent_dim):
    generator = build_generator(input_dim, latent_dim)
    discriminator = build_discriminator(input_dim)
    gan = create_model()
    gan.add(generator)
    gan.add(discriminator)
    gan.compile(loss='binary_crossentropy', optimizer='adam')
    return GAN(generator, discriminator, gan)
def build_generator(input_dim, latent_dim):
    model = create_model()
    model.add(Dense(128, input_dim=latent_dim))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(input_dim, activation='tanh'))
    return model
def build_discriminator(input_dim):
    model = create_model()
    model.add(Dense(128, input_dim=input_dim))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam')
    return model
def train_gan(X, generator, discriminator, gan, latent_dim, batch_size):
    noise = generate_noise(batch_size, latent_dim)
    gen_samples = generator(noise)
    idx = get_random_indices(X.shape[0], batch_size)
    real_samples = X[idx]
    X_combined = combine_array(real_samples, gen_samples)
    y_combined = combine_label(ones(batch_size), zeros(batch_size))
    d_loss = discriminator.train_on_batch(X_combined, y_combined)
    noise = generate_noise(batch_size, latent_dim)
    y_mislabeled = ones(batch_size)
    g_loss = gan.train_on_batch(noise, y_mislabeled)
def generate_samples(num_samples):
    noise = get_random_indices(num_samples, latent_dim)
    synthetic_samples = generator(noise)
    return synthetic_samples
minority_count = count_minority(y)                           ▷ Compute # of minority samples
majority_count = count_majority(y)                           ▷ Compute # of majority samples
if majority_count / minority_count < threshold then
    X_bal = X, y
    return X_bal
end
```

Fig. 3. GAN-based hybrid model

The $train\_gan()$ function trains the GAN model, taking in the input data, the number of epochs, and batch size. It generates fake samples using the generator and noise and selects real samples from the input data. Then, it combines the generated and real samples with their corresponding labels (0 for fake and 1 for real) and trains the discriminator on

the combined dataset. After that, it generates noise and mislabels the generated samples as real to train the generator.

The $generate\_samples()$ function generates $num\_samples$ number of new samples using the generator and passes it along with noise to the generator to generate new samples.

Figure 4 illustrates data balancing by the proposed GAN-based hybrid models on a synthetic classification dataset with 300 samples, 4 features, and 2 classes of a weight (0.8, 0.2). Figure 4(a) shows the data distribution before applying the approach and Figure 4(b)-(f)) show the application of the approach in order of GANRUS, GANENN, GANRUSBoost, GANBoost, and GANTomek.

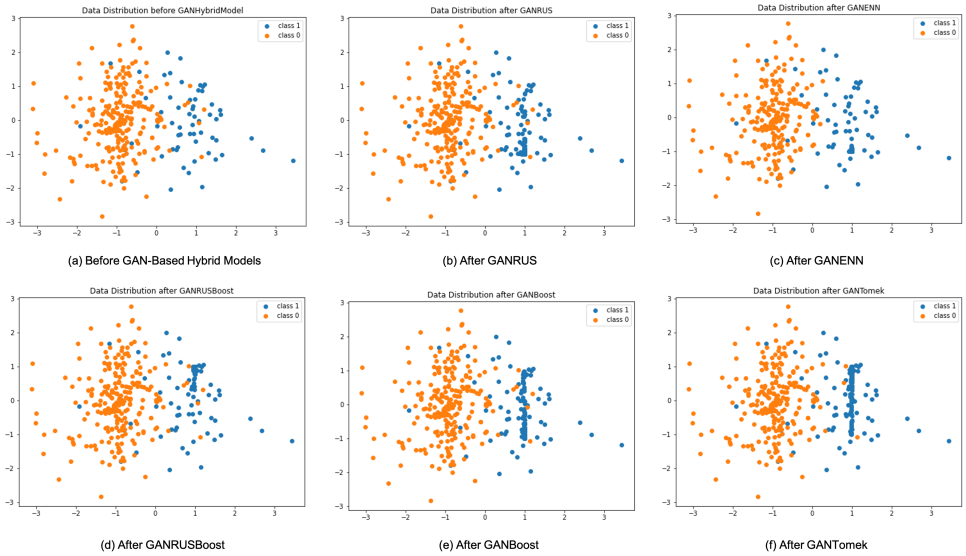

Fig. 4. Data Balancing by GAN-Based Hybrid Models

5. Datasets

We evaluate the proposed approach using two datasets with different levels of class imbalance. One is hotel booking cancellation [2] obtained from two hotels situated in Portugal, a resort hotel (RH) and a city hotel (CH) during the period between July, 2015 and August, 2017. The dataset has a size of 119,390, consisting of 40,060 RH bookings and 79,330 CH bookings. Out of these, 11,120 bookings (28%) of RH and 33,079 (42%) bookings of CH were canceled, resulting in a total of 44,224 (37%) cancellations. The dataset features are presented in Table 5. In order to ensure accurate learning, we preprocessed the dataset for null values and non-numerical values. The $country$, $agent$, and $company$ attribute contained 488, 16,340, and 112,593 null values, respectively where the null values for

$country$ were replaced with *Unknown* and those for $agent$ and $company$ were replaced with 0. The *Undefined* value for $meal$ was replaced with *SC* (self-catering). The samples with zero guests were excluded, and categorical feature values were replaced with numerical values to facilitate efficient learning.

Variable	Description
hotel	Hotel - Resort Hotel, City Hotel
is_canceled	Value indicating if the booking was canceled (1) or not (0)
lead_time	Number of days that elapsed between the entering date of the booking and the arrival date
arrival_date_year	Year of arrival date
arrival_date_month	Month of arrival date with 12 categories: "January" to "December"
arrival_date_week_number	Week number of the arrival date
arrival_date_day_of_month	Day of the month of the arrival date
stays_in_weekend_nights	Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
stays_in_week_nights	Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel
adults	Number of adults
children	Number of children
babies	Number of babies
meal	Type of meal booked. Categories are presented in standard hospitality meal packages - Undefined/SC, BB, HB, FB
country	Country of origin
market_segment	Market segment designation - TA (Travel Agents), TO (Tour Operators)
distribution_channel	Booking distribution channel - TA (Travel Agents), TO (Tour Operators)
is_repeated_guest	Value indicating if the booking name was from a repeated guest (1) or not (0)
previous_cancellations	Number of previous bookings that were cancelled by the customer prior to the current booking
previous_bookings_not_canceled	Number of previous bookings not cancelled by the customer prior to the current booking
reserved_room_type	Code of room type reserved.
assigned_room_type	Code for the type of room assigned to the booking.
booking_changes	Number of changes/amendments made to the booking from the moment the booking was entered until the moment of check-in or cancellation.
deposit_type	Indication on if the customer made a deposit to guarantee the booking.
agent	ID of the travel agency that made the booking.
company	ID of the company/entity that made the booking or responsible for paying the booking.
days_in_waiting_list	Number of days the booking was in the waiting list before it was confirmed to the customer.
customer_type	Type of booking, assuming one of four categories
adr	Average Daily Rate
required_car_parking_spaces	Number of car parking spaces required by the customer
total_of_special_requests	Number of special requests made by the customer (e.g. twin bed or high floor).
reservation_status	Reservation last status, assuming one of three categories - Canceled, Check-Out, No-Show
reservation_status_date	Date at which the last status was set.

Fig. 5. Data Features of Hotel Booking Cancellation

Another dataset is financial fraud detection obtained by PaySim [21], a data simulator that emulates real transactions with malicious transactions incorporated. This dataset comprises 6.9 million transactions with 0.13% identified as fraudulent. The features of the dataset are presented in Figure 6. In the dataset, the type of the $nameOrig$ and $nameDest$ features is of object data type, while the type of the remaining features is numerical. To ensure accurate learning, the dataset underwent validation and normalization processes where null values were eliminated, and numerical variables were transformed to values ranging between -1 and 1. The $isFlaggedFraud$ feature was omitted, as it is a subset of $isFraud$.

Feature	Description
step	It maps a unit of time in the real world. 1 step is 1 hour of time. Total steps 744 (30 days simulation).
type	Type of the transaction. There are four types -- CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.
amount	Amount of the transaction in local currency.
nameOrig	Customer who started the transaction.
oldbalanceOrg	Initial balance before the transaction.
newbalanceOrig	New balance after the transaction.
nameDest	Customer who is the recipient of the transaction.
oldbalanceDest	Initial balance recipient before the transaction. Note that there is not information for customers that start with M (Merchants).
newbalanceDest	New balance recipient after the transaction. Note that there is not information for customers that start with M (Merchants).
isFraud	Transaction made by a fraudulent agent. In this dataset, the fraudulent behavior aims to take control of a customer account and transfer the funds to another account and then cashing out of the system.
isFlaggedFraud	Transaction flagged as illegal attempt (more than 200,000 in a single transaction) by the business model monitoring massive transfers from one account to another.

Fig. 6. Data Features of Financial Fraud Detection

The financial fraud detection dataset has a minority class (fraudulent) proportion of 0.13%, which is approximately 769 times more imbalanced than the hotel booking cancellation dataset with a minority class (defective) proportion of 37%. Using the two different datasets in the level of class imbalance allows for evaluating the effectiveness of GAN-based hybrid models depending on the level of class imbalance.

6. Evaluation

We evaluated the proposed GAN-based hybrid models on six widely used machine learning classifiers – Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), XGBoost (XGB), K-Nearest Neighbors (KNN), and Light Gradient Boosting Machine (LGBM) and compared them with SMOTEENN and SMOTETomek which are commonly used hybrid models for handling class imbalance. The performance of the models was measured in terms of accuracy, precision, recall, F1 score, ROC AUC, and PR AUC.

Figure 7 shows the performance of GAN-based hybrid models on the hotel booking cancellation dataset which has 10,885 samples with 63% non-cancelled and 37% cancelled, which is lllized in Figure 8. The results show that SMOTEENN outperformed all other models across multiple classifiers in terms of accuracy. Specifically, SMOTEENN achieved the highest accuracy scores for RF (0.9643), XGB (0.9368), and KNN (0.9458). On the other hand, GANENN consistently delivered the best accuracy among the hybrid GAN models. When it comes to precision, both GANENN and SMOTEENN demonstrate superior performance across most classifiers. They take the lead in precision scores, highlighting their effectiveness in minimizing false positives. In recall, SMOTEENN stands

out with the highest scores for all classifiers, except for the LR model where GANENN outperforms it. This indicates that SMOTEENN is particularly adept at capturing instances of the minority class. Similar to recall, SMOTEENN dominates the F1 metric for all classifiers, except for LR, where GANENN achieves the highest score. This demonstrates the effectiveness of SMOTEENN in achieving a balance between precision and recall. In terms of ROC AUC, SMOTEENN maintains strong performance across classifiers, followed by GANENN and GANRUSBoost. This suggests that SMOTEENN provides an effective balance between true positive rate and false positive rate. SMOTEENN also outperforms other models in PR AUC, with GANENN and GANRUSBoost closely following. This indicates that SMOTEENN is effective in capturing positive instances, while minimizing false positives. Overall, the results demonstrate that SMOTEENN consistently outperforms GAN-based hybrid models across most classifiers and metrics, while GAN-based hybrid models prove their competitive performance and GANENN outperforms other GAN-based models.

Figure 9 presents the results on the financial fraud detection dataset which contains 6.9 million samples with only 0.13% identified as fraudulent. The results are visualized in Figure 10. The results reveal that overall, GAN-based models outperformed SMOTEENN and SMOTETomek. Specifically, GANBoost and GANENN performed better on DT, RF, XGB, and LGBM in terms of accuracy, and other GAN-based models performed competitively. For precision, GANBoost and GANENN also outperformed SMOTEENN and SMOTETomek on all classifiers. In recall, SMOTEENN and SMOTETomek continue to obtain higher scores across classifiers with a few instances where GAN-based models achieve similar or marginally better performance, such as GANENN in the LR classifier. F1 scores are highly competitive among all models. GAN-based models occasionally outperform SMOTEENN and SMOTETomek across the classifiers with GANENN and GANRUSBoost showing slightly better results in the LR classifier. When considering ROC AUC and PR AUC scores, GAN-based models are either on par with or surpass SMOTEENN and SMOTETomek in most cases across the classifiers. In particular, GANBoost, GANENN, GANRUS, and GANRUSBoost demonstrate prominent performance in the RF and XGB classifiers for both ROC AUC and PR AUC scores. Overall, the proposed GAN-based hybrid models demonstrate strong performance compared to SMOTEENN and SMOTETomek across the considered classifiers.

The results from both the datasets show that GAN-based hybrid models demonstrate competitive or superior performance when compared to well-established techniques such as SMOTEENN and SMOTETomek. In the hotel booking cancellation dataset where the minority class constitutes 37%, GAN-based hybrid models exhibit competitive performance across various metrics and classifiers, although they were slightly outperformed by SMOTEENN. In particular, GANENN shows promise among the proposed GAN-based models. On the other hand, in the financial fraud detection dataset with a highly imbalanced minority class of only 0.13%, GAN-based models consistently outperformed SMOTEENN and SMOTETomek in most metrics and classifiers, demonstrating their effectiveness in addressing a high degree of class imbalance in large-scale datasets. These results witness the potential of the proposed models, especially GANENN, as an efficient and competitive technique for handling a more significant class imbalance in different domains and across various classifiers.

Model	Metric	GANBoost	GANENN	GANRUS	GANRUSBoost	GANTomek	SMOTEENN	SMOTETomek
DT	accuracy	0.8435	0.8926	0.8436	0.8436	0.8523	**0.9415**	0.8585
	precision	0.8203	0.9074	0.8200	0.8200	0.8382	**0.9440**	0.8548
	recall	0.8279	0.9032	0.8287	0.8287	0.8418	**0.9501**	0.8636
	f1	0.8241	0.9053	0.8243	0.8243	0.8400	**0.9470**	0.8592
	roc_auc	0.8447	0.8906	0.8449	0.8449	0.8549	**0.9406**	0.8622
	pr_auc	0.7620	0.8756	0.7621	0.7621	0.7857	**0.9243**	0.8131
RF	accuracy	0.8825	0.9264	0.8825	0.8824	0.8896	**0.9643**	0.8970
	precision	0.9012	0.9472	0.9014	0.9013	0.9094	**0.9687**	0.9100
	recall	0.8251	0.9220	0.8248	0.8247	0.8444	**0.9663**	0.8811
	f1	0.8614	0.9344	0.8614	0.8613	0.8757	**0.9675**	0.8953
	roc_auc	0.9499	0.9776	0.9500	0.9500	0.9558	**0.9944**	0.9628
	pr_auc	0.9494	0.9844	0.9494	0.9494	0.9575	**0.9953**	0.9662
LR	accuracy	0.7865	0.7758	0.7909	0.7775	0.7719	**0.7979**	0.7490
	precision	0.8249	0.8073	**0.8449**	0.8112	0.8053	0.8232	0.7663
	recall	0.6575	0.7957	0.6466	0.6483	0.6658	**0.8055**	0.7166
	f1	0.7317	0.8015	0.7325	0.7207	0.7289	**0.8142**	0.7406
	roc_auc	0.8402	0.8589	0.8446	0.8288	0.8345	**0.8756**	0.8206
	pr_auc	0.8584	0.9114	0.8621	0.8481	0.8592	**0.9170**	0.8556
XGB	accuracy	0.8638	0.8989	0.8636	0.8630	0.8672	**0.9368**	0.8670
	precision	0.8943	0.9212	0.8957	0.8939	0.8985	**0.9433**	0.8906
	recall	0.7852	0.8990	0.7831	0.7836	0.8024	**0.9416**	0.8368
	f1	0.8362	0.9100	0.8356	0.8351	0.8477	**0.9425**	0.8628
	roc_auc	0.9355	0.9628	0.9353	0.9354	0.9399	**0.9842**	0.9420
	pr_auc	0.9358	0.9749	0.9358	0.9358	0.9432	**0.9878**	0.9493
KNN	accuracy	0.8016	0.8900	0.8016	0.8014	0.8096	**0.9458**	0.8114
	precision	0.7955	0.9193	0.7960	0.7955	0.8084	**0.9340**	0.7867
	recall	0.7429	0.8842	0.7423	0.7425	0.7689	**0.9700**	0.8543
	f1	0.7683	0.9014	0.7682	0.7680	0.7882	**0.9517**	0.8191
	roc_auc	0.8752	0.9506	0.8751	0.8752	0.8868	**0.9860**	0.8967
	pr_auc	0.8520	0.9545	0.8520	0.8520	0.8671	**0.9822**	0.8711
LGBM	accuracy	0.8585	0.8899	0.8587	0.8586	0.8611	**0.9218**	0.8545
	precision	0.9036	0.9182	0.9035	0.9037	0.9067	**0.9346**	0.8905
	recall	0.7616	0.8852	0.7623	0.7618	0.7786	**0.9224**	0.8083
	f1	0.8265	0.9014	0.8269	0.8267	0.8378	**0.9284**	0.8474
	roc_auc	0.9311	0.9570	0.9312	0.9313	0.9349	**0.9772**	0.9325
	pr_auc	0.9319	0.9710	0.9320	0.9320	0.9389	**0.9825**	0.9410

Fig. 7. Experiment Results on Hotel Booking Cancellation

7. Conclusion

We have presented a GAN-based hybrid approach to address class imbalance by combining GANs with undersampling/ensemble techniques such as Boost, ENN, RUS, RUSBoost, and Tomek Links. The use of a GAN enables the creation of diverse and realistic synthetic samples, while undersampling/ensemble techniques help remove noisy or ambiguous examples in the majority class. The evaluation on two datasets – hotel booking cancellation and financial fraud detection shows that the proposed effective when the level of class imbalance is high.

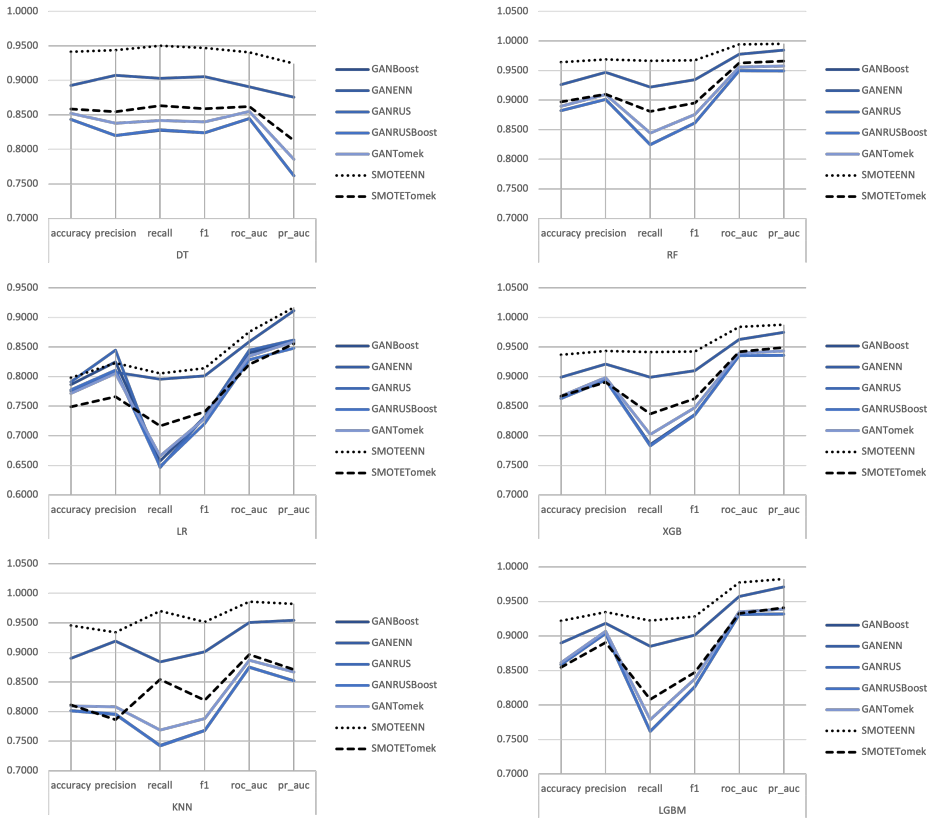

Fig. 8. Visualization of Experiment Results on Hotel Booking Cancellation

References

1. Ali-Gombe, A., Elyan, E.: MFC-GAN: Class-imbalanced dataset classification using Multiple Fake Class Generative Adversarial Network. Neurocomputing 361, 212–221 (2019)
2. Antonio, N., Almeida, A., Nunes, L.: Hotel booking demand datasets. Data in Brief 22, 41–49 (2019)
3. Antoniou, A., Storkey, A., Edwards, H.: Data Augmentation Generative Adversarial Networks. https://arxiv.org/abs/1711.04340 (2018)
4. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter 6(1), 20–29 (2004)
5. Bekkar, M., Djemaa, H.K., , Alitouche, T.A.: Evaluation measures for models assessment over imbalanced data sets. Journal of Information Engineering and Applications 3 (2013)
6. Bhagwani, H., Agarwal, S., Kodipalli, A., Martis, R.J.: Targeting Class Imbalance Problem Using GAN. In: Proceeding of the 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT). pp. 318–322 (2021)

Model	Metric	GANBoost	GANENN	GANRUS	GANRUSBoost	GANTomek	SMOTEENN	SMOTETomek
DT	accuracy	**0.9997**	**0.9997**	0.9996	**0.9997**	0.9996	0.9994	0.9994
	precision	**0.9996**	0.9995	0.9995	0.9995	0.9995	0.9992	0.9990
	recall	0.9994	0.9995	0.9994	0.9994	0.9994	**0.9997**	**0.9997**
	f1	**0.9995**	**0.9995**	**0.9995**	**0.9995**	0.9994	0.9994	0.9994
	roc_auc	**0.9996**	**0.9996**	**0.9996**	**0.9996**	**0.9996**	0.9994	0.9994
	pr_auc	**0.9992**	**0.9992**	0.9991	**0.9992**	0.9990	0.9990	0.9989
RF	accuracy	**0.9998**	**0.9998**	**0.9998**	**0.9998**	0.9997	0.9997	0.9995
	precision	**1.0000**	**1.0000**	0.9999	**1.0000**	**1.0000**	0.9995	0.9991
	recall	0.9993	0.9994	0.9994	0.9993	0.9993	**0.9999**	**1.0000**
	f1	**0.9997**	**0.9997**	**0.9997**	**0.9997**	0.9996	**0.9997**	0.9995
	roc_auc	0.9999	**1.0000**	**1.0000**	**1.0000**	0.9999	**1.0000**	**1.0000**
	pr_auc	0.9999	**1.0000**	0.9999	0.9999	0.9999	**1.0000**	**1.0000**
LR	accuracy	0.9977	0.9976	**0.9985**	0.9984	0.8310	0.9407	0.9193
	precision	0.9949	0.9941	0.9971	**0.9974**	0.7015	0.9435	0.9176
	recall	0.9984	**0.9986**	0.9984	0.9979	0.5000	0.9379	0.9214
	f1	0.9966	0.9964	**0.9978**	0.9976	0.4986	0.9407	0.9195
	roc_auc	0.9975	0.9977	0.9985	0.9984	**0.9990**	0.9849	0.9732
	pr_auc	0.9799	0.9798	0.9878	0.9855	**0.9903**	0.9864	0.9765
XGB	accuracy	**0.9998**	**0.9998**	**0.9998**	**0.9998**	**0.9998**	0.9994	0.9993
	precision	**1.0000**	0.9999	0.9999	0.9999	0.9999	0.9990	0.9987
	recall	0.9995	0.9995	0.9995	0.9995	0.9994	0.9998	**1.0000**
	f1	**0.9997**	**0.9997**	**0.9997**	**0.9997**	**0.9997**	0.9994	0.9993
	roc_auc	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	pr_auc	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
KNN	accuracy	**0.9989**	0.9988	0.9988	0.9988	**0.9989**	0.9984	0.9982
	precision	**0.9997**	0.9995	0.9995	0.9995	0.9996	0.9991	0.9989
	recall	0.9982	0.9981	0.9981	0.9981	0.9982	0.9987	**0.9992**
	f1	0.9989	0.9988	0.9988	0.9988	0.9989	0.9989	**0.9991**
	roc_auc	**0.9999**	**0.9999**	**0.9999**	**0.9999**	**0.9999**	0.9998	0.9998
	pr_auc	**0.9999**	**0.9999**	**0.9999**	**0.9999**	**0.9999**	0.9998	0.9998
LGBM	accuracy	**0.9993**	0.9992	0.9992	0.9992	**0.9993**	0.9988	0.9986
	precision	**0.9999**	0.9998	0.9998	0.9998	**0.9999**	0.9995	0.9993
	recall	0.9987	0.9986	0.9986	0.9986	0.9987	0.9992	**0.9997**
	f1	0.9993	0.9992	0.9992	0.9992	0.9993	0.9993	**0.9995**
	roc_auc	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9999	0.9999
	pr_auc	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9999	0.9999

Fig. 9. Experiment Results on Financial Fraud Detection

7. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Oversampling Technique. Journal Of Artificial Intelligence Research 16, 321–357 (2002)
8. Deng, G., Han, C., Dreossi, T., Lee, C., Matteson, D.S.: IB-GAN: A Unified Approach for Multivariate Time Series Classification under Class Imbalance. In: Proceedings of the SIAM International Conference on Data Mining (SDM). pp. 217–225 (2022)
9. Engelmann, J., Lessmann, S.: Conditional Wasserstein GAN-based oversampling of tabular data for Imbalanced Learning. Expert Systems With Applications 174 (2021)
10. Fiorea, U., Santisb, A.D., Perlaa, F., Zanettia, P., Palmieribi, F.: Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. Information sciences 479, 448–455 (2019)
11. Fotouhi, S., Asadi, S., Kattan, M.W.: A comprehensive data level analysis for cancer diagnosis on imbalanced data. Journal of Biomedical Informatics 90, 103089 (2019)
12. Freund, Y., Shapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Processings of the 2nd European Conference on Computational Learning Theory. pp. 23–37 (1995)

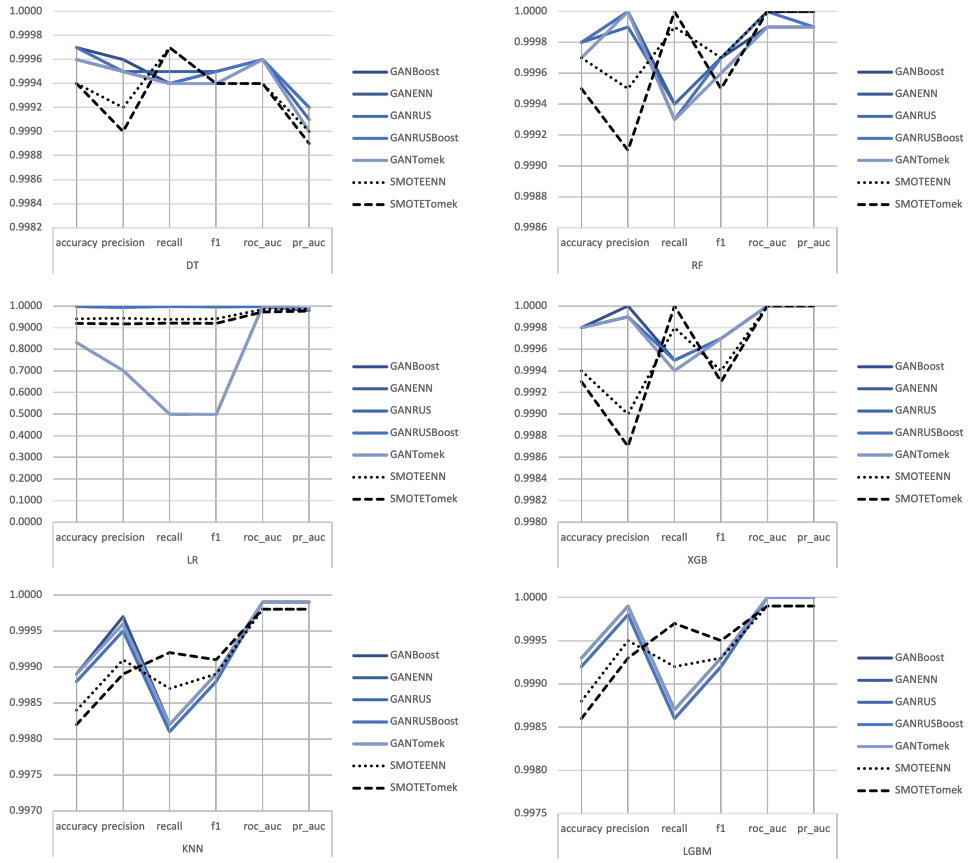

Fig. 10. Visualization of Experiment Results on Financial Fraud Detection

13. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 27. Curran Associates, Inc. (2014)
14. Guo, X., Yin, Y., Dong, C., Yang, G., Guangtong, Z.: On the Class Imbalance Problem. In: Processings of the 4th International Conference on Natural Computation. pp. 192–201 (2008)
15. Japkowicz, N., Stephen, S.: The class imbalance problem: a systematic study. Intelligent Data Analysis 6, 429–449 (2002)
16. Jiang, W., Hong, Y., Zhou, B., He, X., Cheng, C.: A GAN-Based Anomaly Detection Approach for Imbalanced Industrial Time Series. IEEE Access 7, 143608–143619 (2019)
17. Kim, J., Jeong, K., Choi, H., Seo, K.: GAN-Based Anomaly Detection in Imbalance Problems. In: Proceedings of Computer Vision–ECCV Workshops: Part VI 16. pp. 128–145 (2020)
18. Lee, J., Park, K.: GAN-based imbalanced data intrusion detection system. Personal and Ubiquitous Computing 25, 121–128 (2021)
19. Lei, K., Xie1, Y., Zhong, S., Dai1, J., Yang, M., Shen, Y.: Generative adversarial fusion network for class imbalance credit scoring. Neural Computing and Applications 32, 8451–8462 (2020)

20. Liu, X., Zhou, Z., Wu, J.: Exploratory Undersampling for Class-Imbalance Learning. IEEE Transactions on Systems, Man, and Cybernetics 39, 539–550 (2009)
21. Lopez-Rojas, E.A., Elmir, A., Axelsson, S.: PaySim: A financial mobile money simulator for fraud detection. In: Proceedings of The 28th European Modeling and Simulation Symposium (2016)
22. Mariani, G., Scheidegger, F., Istrate, R., Bekas, C., Malossi, C.: BAGAN: Data Augmentation with Balancing GAN. https://arxiv.org/abs/1803.09655 (2018)
23. Mullick, S.S., Datta, S., Das, S.: Generative Adversarial Minority Oversampling (2020)
24. Odena, A., Olah, C., Shlens, J.: Conditional Image Synthesis with Auxiliary Classifier GANs. In: International conference on machine learning. pp. 2642–2651 (2017)
25. Qasim, A.B., Ezhov, I., Shit, S., Schoppe, O., Paetzold, J.C., Sekuboyina, A., Kofler, F., Lipkova, J., Li, H., Menze, B.: Red-GAN: Attacking Class Imbalance via Conditioned Generation. Yet Another Medical Imaging Perspective. In: Proceedings of Medical imaging with deep learning. pp. 655–668 (2020)
26. Roth, K., Lucchi, A., Nowozin, S., Hofmann, T.: Stabilizing Training of Generative Adversarial Networks through Regularization. Advances in neural information processing systems 30 (2017)
27. Seiffert, C., Khoshgoftaar, T.M., Hulse, J.V., Napolitano, A.: Rusboost: A hybrid approach to alleviating class imbalance. Systems Man and Cybernetics Part A: Systems and Humans IEEE Transactions 40(1), 185–197 (2010)
28. Sharma, A., Singh, P.K., Chandra, R.: SMOTified-GAN for Class Imbalanced Pattern Classification Problems. IEEE Access 10, 30655–30665 (2022)
29. Tomek, I.: Two Modifications of CNN. IEEE Transactions on Systems, Man, and Cybernetics SMC-6, 769–772 (1976)
30. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 29, pp. 3630–3638. Curran Associates, Inc. (2016)
31. Wang, P., Li, S., Ye, F., Wang, Z., Zhang, M.: PacketCGAN: Exploratory Study of Class Imbalance for Encrypted Traffic Classification Using CGAN. In: Processings of IEEE International Conference on Communications. pp. 1–7 (2020)
32. Wang, Z., Wang, P., Zhou, X., Li, S., Zhang, M.: FLOWGAN:Unbalanced network encrypted traffic identification method based on GAN. In: Processings of IEEE International Conference on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking (2019)
33. Wilson, D.L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man, and Cybernetics 2(3), 408–421 (1972)
34. Yang, H., Zhou, Y.: IDA-GAN: A Novel Imbalanced Data Augmentation GAN. In: Proceedings of the 25th International Conference on Pattern Recognition (ICPR). pp. 8299–8305 (2021)
35. Zhenchuan, L., Mian, H., Guanjun, L., Changjun, J.: A hybrid method with dynamic weighted entropy for handling the problem of class imbalance with overlap in credit card fraud detection. Expert Systems with Applications 175, 114750 (2021)
36. Zheng, Z., Cai, Y., Li, Y.: Oversampling method for imbalanced classification. Computing and Informatics 34, 1017–1037 (2015)

Dae-Kyoo Kim is a professor in the Department of Computer Science and Engineering at Oakland University. He received a Ph.D. in computer science from Colorado State University in 2004. He worked as a technical specialist at the NASA Ames Research Center in 2002. His research interests include software engineering, software security,

data modeling in smart grids, and business process modeling. The author can be contacted at kim2@oakland.edu.

Yeasun K. Chung is Associate professor at Spears School of Business in Oklahoma State University. Prior to her academic career she worked in financial engineering and derivative department for a financial investment organization. Her research interests include risk management, strategic planning, business process management, and business intelligence. The author can be contacted at y.chung@okstate.edu

Received: November 13, 2023; Accepted: September 15 2024.