WiP: Generative Adversarial Network for Oversampling Data in Credit Card Fraud Detection

Akhilesh Kumar Gangwar[1,2] and Vadlamani Ravi[1](✉)

[1] Center of Excellence in Analytics, Institute for Development and Research in Banking Technology, Castle Hills Road #1, Masab Tank, Hyderabad 500057, India
gangwar.akhilesh1993@gmail.com, rav_padma@yahoo.com
[2] School of Computer and Information Sciences, University of Hyderabad, Hyderabad 500046, India

Abstract. In this digital world, numerous credit card-based transactions take place all over the world. Concomitantly, gaps in process flows and technology result in many fraudulent transactions. Owing to the spurt in the number of reported fraudulent transactions, customers and credit card service providers incur significant financial and reputation losses respectively. Therefore, building a powerful fraud detection system is paramount. It is noteworthy that fraud detection datasets, by nature, are highly unbalanced. Consequently, almost all of the supervised classifiers, when built on the unbalanced datasets, yield high false negative rates. But, the extant oversampling methods while reducing the false negatives, increase the false positives. In this paper, we propose a novel data oversampling method using Generative Adversarial Network (GAN). We use GAN and its variant to generate synthetic data of fraudulent transactions. To evaluate the effectiveness of the proposed method, we employ machine learning classifiers on the data balanced by GAN. Our proposed GAN-based oversampling method simultaneously achieved high precision, F1-score and dramatic reduction in the count of false positives compared to the state-of-the-art synthetic data generation based oversampling methods such as Synthetic Minority Oversampling Technique (SMOTE), Adaptive Synthetic Sampling (ADASYN) and random oversampling. Moreover, an ablation study involving the oversampling based on the ensemble of SMOTE and GAN/WGAN generated datasets indicated that it is outperformed by the proposed methods in terms of F1 score and false positive count.

Keywords: Fraud detection · Supervised classification · Deep learning · Generative Adversarial Network · Oversampling · SMOTE

1 Introduction

With the spectacular digitalization witnessed in the last decade, online payment methods become one of the significant offerings in the financial services industry. Online shopping, ticket booking, bill payments using credit card become the norm nowadays.

D. Garg et al. (Eds.): ICISS 2019, LNCS 11952, pp. 123–134, 2019.
https://doi.org/10.1007/978-3-030-36945-3_7

Fraudulent transactions involving credit cards for online payments are increasing owing to the shortcomings in the processes, technology and social engineering. Detecting these online card-based frauds and preventing them is a thriving research area. Implementing a powerful and versatile fraud detection system is paramount for any organization as they incur heavy monetary and reputational loss. State-of-the-art fraud detection systems take recourse to machine learning techniques which attempt to learn the profiles of fraudulent transactions from a dataset that contains normal as well as fraudulent transactions.

We now present the challenges in data-driven fraud detection. Data-driven fraud detection collapses into the classification task of supervised machine learning. Due to a disproportionately small number of fraudulent transactions vis-à-vis the normal ones, traditional classifiers incorrectly predict the fraudulent transaction as normal ones. For instance, fraudulent transactions account for 0.1% to 5% of the total data leading to the problem of data imbalance. Almost all machine learning based binary classifiers perform badly on the unbalanced data. However, fuzzy rule based classifiers, one-class classification using SVM, 3-layered or 5-layered auto encoders can work well in case of unbalanced data without having to balancing the data. Generating synthetic data of fraudulent transactions having the same distribution as that of the original data is a big challenge. There are many ways to oversample minority class data in data mining literature. But the problem with these methods is that they replicate the data, without bothering to learn the distribution of minority class. Consequently, false positive rates becomes very large. Cost of manually verifying false positives is much high compared to the cost incurred due to fraud.

We now briefly survey the related work. Sisodia et al. [1] evaluated the performance of class balancing techniques for credit card fraud detection. They used different SMOTE versions and performed undersampling and oversampling. They validated their methods using different classifiers. Randhawa et al. [2] employed a host of machine learning classifiers and designed an ensemble too using majority voting for credit card fraud classification. Vega-Márquez et al. [3] used the conditional GAN for generating synthetic data for both classes. They found that synthetic data improved the F1-score with the Xgboost classifier. They did not compare it with other synthetic data generation techniques and did not report the false positive count. Dos Santos Tanaka et al. [4] used GAN for generating minority class data in medical domain. They found that it is useful in the context of privacy preservation of sensitive data. Mottini et al. [5] proposed Cramer GAN for PNR data generation that is a combination of categorical and numerical data. They used softmax function in place of sigmoid for categorical data generation. Fiore et al. [6] has used vanilla GAN to increase the effectiveness of credit card fraud detection. Douzas et al. [7] used conditional GAN for generating the samples of the minority class containing fraudulent transactions. We propose GAN for generating the continuous data. The architecture of the generator in our proposed method is different from that of Cramer GAN.

The motivation for the present research is as follows: In credit card fraud detection, the major challenge is to obtain high sensitivity and low false positive rate simultaneously from statistical and machine learning classifiers. A supervised classifier can work properly if the data set is balanced. One way of balancing the data is to oversample the minority class. While simple oversampling replicates the minority class observations, oversampling based on synthetic data generation does not generate data that follows the same probability distribution as that of the minority class because it is based on the nearest neighbor method. Consequently, classifiers yield high false positive rate, which in turn increases the cost, as it needs manual verification of the false positives. The motivation is to oversample minority class by using GAN. This strategy could reduce the number of false positives thereby making the fraud detection system cost effective.

The main contributions of the present paper are as follows:

- We developed a generator and a discriminator for GAN. After training, the generator of GAN can generate synthetic data from the minority class.
- We demonstrated the effectiveness of the proposed oversampling method using machine learning classifiers.
- We compared our method with the extant oversampling techniques. We could reduce the number of false positives dramatically.
- We proposed Wasserstein GAN (WGAN) as well for structured data generation. WGAN was used in literature for image data. But, here, we used it for generating the structured data of fraudulent transactions of credit card. We compared the performance of the WGAN with that of vanilla GAN.
- We performed an ablation study too, where we designed an ensemble combining the data generated using GAN and SMOTE. We compared all methods using F1-score, precision and Recall.

The rest of the paper is organized as follows: Sect. 2 presents the detail of background knowledge to understand model; Sect. 3 presents in detail our proposed model; Sect. 4 presents the dataset description and evaluation metrics; Sect. 5 presents a discussion of the results and finally Sect. 6 concludes the paper and presents future directions.

2 Background

2.1 Generative Adversarial Networks (GANs)

Of late, GAN has gained traction across the machine learning community with excellent results replicating real-world rich content such as images, languages, and music. It is inspired by game theory. It consists of two models, namely, a generator, and a discriminator, both of which compete while supporting each other and making progress together. But training GAN is not an easy task, because since Goodfellow et al. [8] proposed this concept, it has been a problem of unstable training and easy collapse. The architecture of GAN depicted in Fig. 1. The two models of GAN are as follows:

Discriminator model is responsible for predicting the probability that a sample is from a real data set. It trains on real data and the generator's generated fake data and tries to discriminate both type of data as accurately as possible, thus pointing out the inadequacy of newly generated samples. In a sense, it is like a critic which tries to discriminate generator's generated data from the real data.

Generator G. This model is responsible for synthesizing the input noise signal z into a new sample (z contains the potential real data distribution) and then feeding it to the discriminator for judgment. Its goal is to capture the true data distribution based on the output of the discriminator, making the samples it generates as realistic as possible.

These two models compete with each other during the training process: the generator G tries to "spoof" the discriminator D, and the discriminator is constantly improving itself to avoid being cheated. It is this interesting zero-sum game that has given immense strength to GAN.

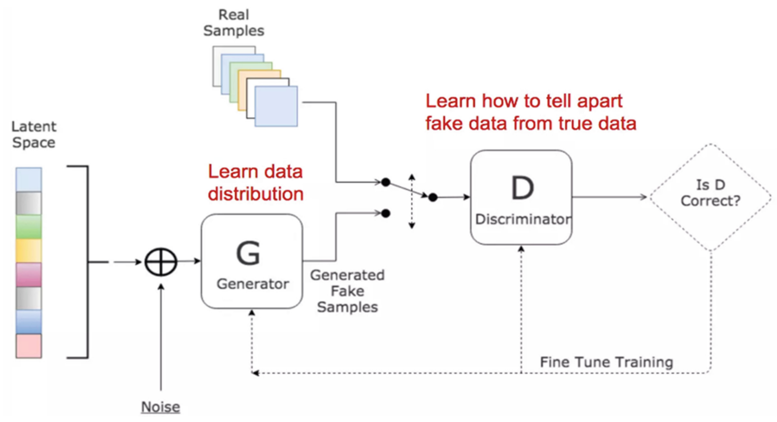

Fig. 1. Architecture of GAN with generator and discriminator

GAN [8] is a generative model that is built by combining two feed-forward neural networks. Generator component of GAN is used to generate synthetic data that looks like the original data in terms of probability distribution and quality. In the Fig. 1, z is a random input to the generator G. Discriminator D receives input as a combination of real data x and the data G(z) generated by the generator. θ_d and θ_g are parameters of discriminator and generator networks respectively. Discriminator and generator both have their respective objective functions as depicted in Fig. 2. In GAN, the generator must be a differentiable, feed-forward network. In GAN, the motive is to achieve a well trained generator so that it can generate the synthetic data that looks like the original data. Hence, optimizing the generator's weights is critical. The presence of the discriminator changed the way the generator is optimized. Discriminator of GAN works as a critic that helps in optimizing generator's weights. The goal of the discriminator is to distinguish the generated sample from the real sample as accurately as possible. That means the generator's objective is to generate synthetic samples that look so similar to the real ones that it can fool the discriminator. Therefore, the optimization of GAN is a min-max game. We optimize the weights of the generator and the discriminator alternately by freezing the weights of one of them.

$$\min_{G}\max_{D} V(D,G) = E_{x\sim P_{data(x)}}[\log(D(x)] + E_{z\sim P_z(z)}[\log(1 - D(G(z)))]$$

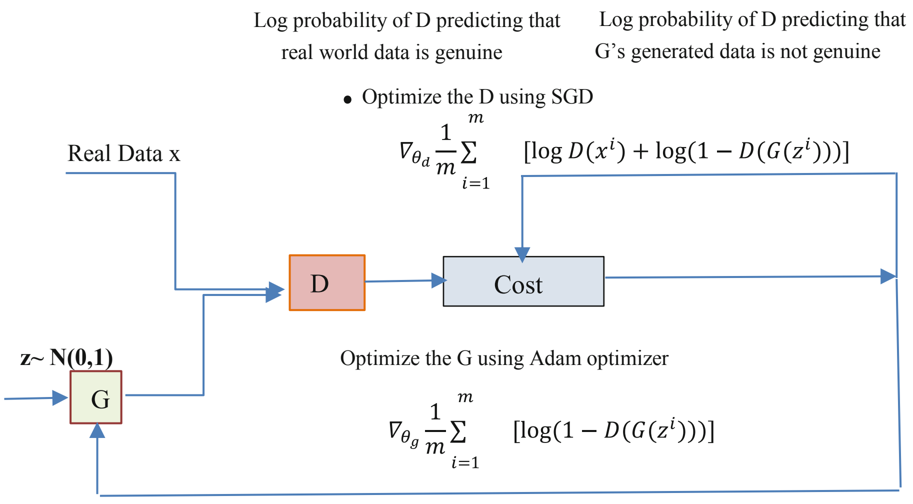
Log probability of D predicting that real world data is genuine
Log probability of D predicting that G's generated data is not genuine
• Optimize the D using SGD
$\nabla_{\theta_d} \frac{1}{m}\sum_{i=1}^{m} [\log D(x^i) + \log(1 - D(G(z^i)))]$
Real Data x
D
Cost
z~ N(0,1)
G
Optimize the G using Adam optimizer
$\nabla_{\theta_g} \frac{1}{m}\sum_{i=1}^{m} [\log(1 - D(G(z^i)))]$

Fig. 2. Architecture of GAN with generator and discriminator objective functions.

Wasserstein GAN [9] (WGAN) employs Wasserstein distance as the loss function. Wasserstein Distance is a measure of the distance between two probability distributions. It is also called Earth Mover's (EM) distance. Wasserstein distance is better than Kullback-Leibler divergence and Jensen-Shannon divergence to measure the similarity between distributions [10].

2.2 Oversampling Methods

Random Oversampling: It simply replicates the data points of the minority class to reduce imbalance. So, no new data is generated. Due to replication, the classifier does not learn a variety of patterns of the minority class.

Synthetic Minority Over-Sampling Technique (SMOTE): Proposed in [12] SMOTE, instead of replicating the minority class, it generates synthetic data based on the position of original samples. It selects a random data point from the minority class and finds its k nearest neighbors. It then creates a synthetic point near these neighbors.

Adaptive Synthetic (ADASYN) Sampling: Proposed in [13], ADASYN is an updated version of SMOTE. This method adds a small bias to artificial points so that they are not linearly correlated to their parent data points. In this paper, to implement SMOTE and ADYSYN, we used the imbalanced-learn python library [11].

3 Proposed Model

The schematic of our proposed model is depicted in Fig. 3. We used GAN for generating the synthetic records of the fraudulent class and machine learning classifiers to validate the quality of artificially generated data and classification purpose.

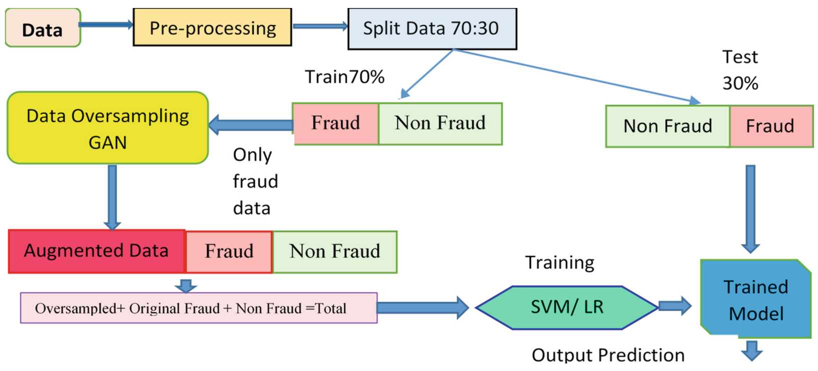

Fig. 3. Schematic diagram of the proposed method

First, we consider the whole data and perform some pre-processing like dropping the column, column standardization. We split the data using the hold-out method in the ratio of 70%:30%. Then, we oversample the training data by concentrating on the minority class of fraudulent transactions, while keeping the test data intact because the latter represents the reality.

3.1 GAN Architecture

Both generator and discriminator of GAN should be differentiable and feedforward networks [14]. As our data is a structured one, CNN and LSTM are not suitable. CNN is used for image data and LSTM is used in case of sequential data. Therefore, in our case, we used multilayer perceptron for both generator and discriminator. We used WGAN (consists of MLP as a generator) for structured data generation and to the best of our knowledge it is not attempted earlier. We used the following architectures for generator and discriminator of the vanilla GAN and WGAN. Only the objective function is different in them.

Architecture of the Generator:
Input Layer. The input layer receives randomly generated latent input of fixed length. Input data is transformed to synthetically generated data by the generator.

Hidden Layers.
- First hidden layer has 500 neurons with leaky ReLU activation [15].
- Second hidden layer has 500 neurons with leaky ReLU activation.
- Third hidden layer has 500 neurons followed by leaky ReLU activation.

Output Layer. This layer outputs generated data that has the same dimension as that of the original data. It uses the hyperbolic tangent (tanh) activation function to get non-linearity. We update the weights of this network so that generated data achieves the same distribution of as that of the original data. That update in weights is accomplished with the help of discriminator in GAN.

Architecture of the Discriminator:
Input Layer. The input layer of the discriminator receives the input vector with the same length as that of the original data.

Hidden Layers.
- First hidden layer has 500 neurons with leaky ReLU activation.
- Second hidden layer has 300 neurons with leaky ReLU activation.

Output Layer. This layer with sigmoid activation function, yields a binary output.

3.2 Classifiers

In the current work, we did not focus on the power of the classifier. We chose two traditional machine learning classifiers. We employed logistic regression (LR) with the elastic net regularization and support vector machine (SVM) with the elastic net penalty. We used Linear kernel and *SGDClassifier* [16] of *Sklearn*[1] library for SVM and LR by choosing *hinge loss* and *log loss* respectively. We trained the GAN with the minority class only. When training of GAN is completed, we generated synthetic samples from the trained generator. We then augmented the generated data with the samples of the fraudulent as well as non-fraudulent classes and used it to train the SVM and LR. The trained classifiers are then tested on the test data.

3.3 Experimental Setting

We used Adam [17] optimizer (with learning rate = 0.001) for generator and stochastic gradient descent optimizer with (learning rate = 0.001) for the discriminator. We generated 15000 data samples for each oversampling method. We trained GAN for 10,000 epochs. Input dimension of generator and discriminator is 100 and 30 respectively, while the output dimension of generator and discriminator is 30 and 1 respectively. We used Tensorflow [18] framework for deep learning and Python language for the classifiers. The combination of all the hyperparameters is obtained by a grid search.

4 Data Set Description and Evaluation Metrics

We validated the effectiveness of our proposed models on the Kaggle credit card data set [19]. This data set is highly unbalanced having 284,807 normal transactions and 492 fraudulent transactions. Thus, the positive class represents only 0.172% of the whole data. The data has 31 columns out of which *Time* and *Amount* are two named

[1] https://scikit-learn.org/stable/.

features and the names of *V1* to *V28* features have not been reported due to confidentiality reasons. All are continuous features. Feature *Time* represents time in seconds between transactions and feature *Amount* shows the amount value of the transaction. Last feature *Class* represents the label fraudulent or normal transaction. We divided the dataset in train and test parts in 70%:30% ratio as presented in Table 1.

Table 1. Train and test data

Data	Fraudulent samples	Normal samples	% of the whole data
Train set	338	199364	0.17% fraud
Test set	155	85443	0.18% fraud

4.1 Evaluation Metrics

The data set is highly unbalanced. Here our main intention is to reduce the count of false positives and not to measure the accuracy. We used precision, F1-score as additional performance measures. We also computed false positive, true positive, specificity and recall.

Precision = TP/Predicted Positive = TP/(TP + FP)
Recall = TP/Actual Positive = TP/(TP + FN)
Specificity = TN/Actual Negative = TN/(TN + FP)
F1-Score = 2 * (Precision * Recall)/(Precision + Recall)

Where True Positive (TP): Predicted positive and actual is also positive.
True Negative (TN): Predicted negative and actual is also negative.
False Positive (FP): Predicted positive but actual is negative.
False Negative (FN): Predicted negative but actual is positive.

5 Result and Discussion

Our main intention, in this research work, is to build a good oversampling method or synthetic data generator. We investigated the effectiveness of the proposed GAN and WGAN based oversampling methods. We compared the performance of the proposed oversampling methods with that of the existing oversampling methods using SVM and LR classifiers. The results of the classifiers on test data are presented in Tables 2 and 3 respectively. It can be seen from Table 2 that AUC scores are not good measures to check the effectiveness of the oversampling method. In fraud detection case, we would like to have high true positive count and less false positive count simultaneously. According to the Tables 2 and 3, we can say that false positive count is very high on the unbalanced data without using any oversampling method. But, compared to all oversampling methods, our proposed method performed the best in terms of controlling false positives.

Table 2. Performances comparison with SVM

Method	TP	FP	Specificity	Precision	Recall	F1-score	AUC
Plain SVM	145	2951	0.97	0.05	0.94	0.09	0.95
Random oversampling + SVM	139	1046	0.99	0.12	0.90	0.21	0.94
SMOTE + SVM	145	1595	0.98	0.08	0.94	0.15	0.96
ADASYN + SVM	145	4874	0.94	0.03	0.94	0.06	0.94
GAN + SVM	**135**	**170**	**0.99**	**0.58**	**0.85**	**0.69**	**0.93**
WGAN + SVM	**132**	**95**	**0.99**	**0.58**	**0.85**	**0.69**	**0.92**
SMOTE + GAN + SVM	**142**	**512**	**0.99**	**0.22**	**0.92**	**0.35**	**0.96**
SMOTE + WGAN + SVM	**141**	**422**	**0.91**	**0.25**	**0.91**	**0.39**	**0.95**

It is common knowledge that the fraudulent transactions are considered as positive class. Supervised machine learning classifiers misclassify non-fraudulent (normal) samples as fraudulent ones thereby increasing the false positive count. Even the true positive count is high due to soft boundary of a classifier separating the fraudulent and non-fraud datasets. Thus, even if a model yields high true positive count, it also misclassifies the normal transactions, that are near the soft boundary, as fraudulent ones. This increases the false positive count. Hence, such models are not preferred because every sample predicted as fraudulent will have to be manually verified and during the verification if they turn out to be false positives, it is a wastage of time and money. Therefore, reducing false positive count along with increasing the true positive count is the main concern nowadays.

GAN and WGAN based oversampling methods controlled the false positive count spectacularly without affecting the true positive count significantly. Precision and F1-score are very high compared to other oversampling methods. Our GAN with LR/SVM is better than plain LR/SVM because the latter, unsurprisingly, yielded high false positive count as well as high precision.

Our proposed method yielded excellent results (see Tables 2 and 3) compared to SMOTE, random oversampling and ADASYN because these oversampling methods generate the synthetic data without learning the distribution of the original class. Random oversampling does not generate any synthetic data. It replicates the existing minority class data to balance the classes. Therefore, it does not bring any variety to the data thereby afflicting the performance of the classifiers. While SMOTE and ADASYN generate new synthetic data points based on neighbors, they too do not learn the distribution of the data. Therefore, the data generated by them does not have variety.

Our proposed GAN generated data that has the same distribution as that of the minority class because the objective function of GAN is designed so as to learn the distribution There are many types of objective functions such as KL-divergence, JS-divergence used in GAN. These objective functions have a limitation in terms of learning the data distribution as follows: if the two distributions are in lower dimensional manifolds without overlaps, then KL- divergence would not perform better. We used WGAN to overcome this problem. WGAN employs Wasserstein distance that can provide meaningful and smooth representation of the distance if distributions are not

Table 3. Performance comparison with logistic regression

Method	TP	FP	Specificity	Precision	Recall	F1-score	AUC
Plain LR	145	3394	0.96	0.04	0.94	0.08	0.95
Random oversampling + LR	144	1476	0.98	0.09	0.93	0.16	0.95
SMOTE + LR	144	1454	0.98	0.09	0.93	0.16	0.95
ADASYN + LR	148	7215	0.92	0.02	0.95	0.04	0.93
GAN + LR	**132**	**120**	**0.999**	**0.52**	**0.85**	**0.65**	**0.93**
WGAN + LR	**132**	**78**	**0.999**	**0.63**	**0.85**	**0.72**	**0.92**
SMOTE + GAN + LR	**141**	**588**	**0.94**	**0.19**	**0.91**	**0.32**	**0.95**
SMOTE + WGAN + LR	**143**	**561**	**0.99**	**0.22**	**0.92**	**0.33**	**0.96**

*LR = Logistic Regression

overlapped. Reducing the distance between the distribution of artificially generated data and true data makes GAN perform better for data oversampling. WGAN outperformed GAN because WGAN has a better objective function. Training time of model is system and data dependent but after training we can use trained model for deployment and response will depend on the server. But we noticed that it is very fast (10–15 s for generating 15000 data points).

An ablation study was also conducted by taking the union of the datasets generated by both SMOTE and GAN/WGAN along with the original dataset as the new dataset and invoking classifiers on them. It is a sort of ensemble of the datasets generated by SMOTE and GAN/WGAN. Tables 2 and 3 clearly indicate that even this ensemble strategy failed to yield higher F1 score and lower false positive count compared to the GAN/WGAN based oversampling methods. Therefore, we infer that the proposed GAN/WGAN based oversampling method demonstrates its strength and power conclusively and convincingly.

6 Conclusion and Future Directions

In this paper, we proposed a GAN based minority oversampling method for credit card fraud detection. We compared GAN based oversampling method with the existing oversampling algorithms like random oversampling, SMOTE, ADASYN. We observed that the GAN based oversampling method is more effective compared to other oversampling methods in terms of reducing the false positive count. We have successfully generated a variety of data samples from the minority class after learning the distribution of the minority class using GAN and tested the quality using standard machine learning algorithms. We observed that the false positive count reduced drastically without affecting the true positive count significantly. We also proposed WGAN for data oversampling. WGAN has better objective function compared to the vanilla GAN for comparing the distribution of the generated data and with that of the original data. We found that WGAN is more effective compared to vanilla GAN. Further, an ablation study conducted involving ensembling the SMOTE and GAN/WGAN generated datasets also failed to outperform the proposed GAN/WGAN based oversampling methods in terms of F1 score and false positive count.

In the future, we would like to try different generator architecture in GAN. We will use the Variational Autoencoder (*VAE*) [20] in the generator part. We will extend the capacity of learning of GAN by using appropriate loss function. We want to investigate the effectiveness of the proposed model in another type of structured data that contain categorical, integer and numerical features. We want to use GAN in data privacy case also.

References

1. Sisodia, D.S., Reddy, N.K.: Performance evaluation of class balancing techniques for credit card fraud detection. In: 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), pp. 2747–2752 (2017)
2. Randhawa, K., Loo, C.K., Seera, M., Lim, C.P., Nandi, A.K.: Credit card fraud detection using adaboost and majority voting. IEEE Access **6**, 14277–14284 (2018)
3. Vega-Márquez, B., Rubio-Escudero, C., Riquelme, José C., Nepomuceno-Chamorro, I.: Creation of synthetic data with conditional generative adversarial networks. In: Martínez Álvarez, F., Troncoso Lora, A., Sáez Muñoz, J.A., Quintián, H., Corchado, E. (eds.) SOCO 2019. AISC, vol. 950, pp. 231–240. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-20055-8_22
4. Dos Santos Tanaka, F.H.K., Aranha, C.: Data augmentation using GANs. CoRRabs/1904.09135 (2019). http://arxiv.org/abs/1904.09135
5. Mottini, A., Lheritier, A., Acuna-Agost, R.: Airline passenger name record generation using generative adversarial networks. arXiv preprint https://arxiv.org/abs/1807.06657 (2018)
6. Fiore, U., De Santis, A., Perla, F., Zanetti, P., Palmieri, F.: Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. Inf. Sci. **479**, 448–455 (2019)
7. Douzas, G., Bacao, F.: Effective data generation for imbalanced learning using conditional generative adversarial networks. Expert Syst. Appl. **91**, 464–471 (2018)
8. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, Montreal, Canada, pp. 2672–2680, December 2014
9. Arjovsky, M., Chintala, S., Bottou. L.: Wasserstein GAN. arXiv preprint https://arxiv.org/abs/1701.07875 (2017)
10. Weng, L.: From GAN to WGAN. arXiv preprint https://arxiv.org/abs/1904.08994 (2019)
11. Guillaume, L., Nogueira, F., Aridas, C.K.: Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J. Mach. Learn. Res. **18**(1), 559–563 (2017)
12. Chawla, N.V., Bowyer, K.W., Hall, L.O.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
13. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322–1328 (2008)
14. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org
15. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: ICML 2013, pp. 3–8 (2013)
16. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

17. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. arXiv preprint https://arxiv.org/abs/1412.6980 (2014, 2017)
18. https://www.tensorflow.org/
19. Pozzolo, A.D., Caelen, O., Johnson, R.A., Bontempi, G.: Calibrating probability with undersampling for unbalanced classification. In: Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015, pp. 155–166 (2015)
20. Welling, M.: Auto-encoding variational Bayes. arXiv: https://arxiv.org/abs/1312.6114 (2014). ICLR 2013, vol. abs/1312.6114