NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Applied Mathematics and Informatics"

Research Project Report on the Topic: Tabular Data Synthesis Using Generative Models

Submitted by the Student:

group #БПМИ232, 2nd year of study Abramov Georgiy Evgenievich

Approved by the Project Supervisor:

Grinberg Petr Markovich
Visiting Teacher
Faculty of Computer Science, HSE University

Moscow 2025

Contents

Annotation

Training of modern machine and deep learning models requires large datasets. However, such datasets can be unbalanced and contain private or missing information. This can cause trained computer models to be biased, leak personal information, or be of subpar quality. One of the methods for solving this problem is to synthesize new data with generative models. In this work, we explore existing approaches to this problem for tabular data and create our own solution based on deep learning algorithms. We use Generative Adversarial Networks as such models and show that they can be a viable solution for this problem.

Аннотация

Для обучения современных систем машинного и глубокого обучения требуются большие наборы данных. Однако такие данные могут быть несбалансированными, содержать приватную или пропущенную информацию. Из-за этого полученные компьютерные модели могут быть необъективными, передавать чужую информацию без разрешения или иметь плохое качество. Одним из методов решения данной проблемы является синтезирование новых данных с помощью генеративных моделей. В данной работе исследуются существующие подходы к задаче для табличных данных и создается свое решение на основе алгоритмов глубокого обучения. В качестве таких моделей используются Генеративно-Состязательные Сети и приводятся результаты, которые показывают, что они могут служить достойным решением данной проблемы.

Keywords

Data synthesis, Data augmentation, Deep learning, Generative adversarial networks

1 Introduction

A large and high-quality dataset is one of, if not the most important part when it comes to training a machine learning model. Such datasets allow models to generalize the data and extract more useful patterns from it, overall making them better. Data synthesis can be used to enlarge an existing dataset, augment it, oversample the minority class in order to make the dataset more balanced, or generate an entirely new dataset in order to avoid private data leaks.

Some datasets contain personal information, and it is important to anonymize them. While identifiers (name, personal address, age, etc.) can be removed to secure privacy, it can negatively impact the quality of the dataset. Moreover, using background knowledge, even with the removal of the personal identifiers, they can be recovered using other attributes from the dataset [13]. This problem can be solved by using the original data to train a generative model that will create a synthetic dataset similar to the original one, which can be used later without privacy concerns.

A lot of datasets are imbalanced. For example, datasets for transaction fraud contain way more entries of one class than the other [5]. This can lead to classification models being suboptimal at differentiating between the classes, as they train almost exclusively on one and can not learn enough about the other, potentially leading to biased decisions. Creating more entries of the minority class via generative models can aid in this problem [7].

In this work, Generative Adversarial Networks (GANs) [8] and their modifications are used as generative models. GANs serve as a lightweight alternative to Large Language Models(LLMs) [3] in the task of tabular data generation. Although displaying better results [3], LLMs require a lot of time, effort, and money to train. At the same time, GANs can achieve fairly good quality while being more efficient (in terms of compute and budget) and relatively fast in training and inference.

GAN architecture faces several challenges when it comes to tabular data. With table datasets containing several continuous and discrete columns, GANs need to model data entries, where each feature can represent complex or unbalanced distributions. For this reason, several models specific to this problem were introduced since the emergence of GANs, which we discuss in this paper.

We aim to test and combine best practices by comparing the quality of classification models (e.g., gradient boosting [16]) trained on original datasets versus the ones synthesized by the generative model. Also, we test the impact of oversampling the minority class on imbalanced datasets via the comparison between the quality of the aforementioned classifiers trained on original data versus the augmented data. We find that GAN-based models generate data, which is similar to real data for the purposes of using it to train machine learning algorithms. We also find that synthesizing new entries of the minority class via GANs can improve classification model performance on unbalanced datasets. The implementation of all models from this paper can be found on GitHub[1].

2 Related work

2.1 Generative adversarial networks (GANs)

Generative adversarial network (GAN) [8] consists of two neural networks: *generator* and *discriminator*. Generator tries to map latent space to the distribution of the objects we try to generate. Therefore, when we want to synthesize a new dataset entry, we can just feed it with random noise. Discriminator tries to differentiate between real data and the data that was synthesized by the generator. Both networks are trained at the same time, so as the discriminator becomes better at finding out the fake data, the generator becomes better at generating the objects that resemble the real data.

Generator and discriminator can be convolutional neural networks (CNNs); this type of GAN is called Deep Convolutional GAN (DCGAN) [17] and it is used to generate more complex objects (mostly images). Another variation of GANs is Conditional GAN [14]. We can add the label vector to an input of discriminator and generator nets in order to train GAN to generate a specific class of object. It can be useful when we want to oversample a minority class.

2.2 GAN application to data synthesis

When it comes to GANs in data synthesis, [18] shows that GAN is a viable method for synthesizing data, generating datasets similar to real ones. It also tackles the problem of balancing the dataset by oversampling the minority class. It is done by training a GAN solely on the minority class and generating data entries from it until the dataset is balanced. The architecture that is used in this paper is basic GAN architecture with multilayer perceptrons (MLPs).

Table-GAN (TGAN) [15] is the modification of GAN designed specifically for table dataset generation. Dataset features are reshaped into square matrices (padded with zeros) and generated using DCGAN architecture. Also, Table-GAN introduces another neural network in addition to the generator and discriminator – *classifier*. It predicts the label of the object by learning from the real data simultaneously with the generator and discriminator. Table-GAN also uses 2 new losses

[1] https://github.com/abramovgeorge/hse-dl-coursework-gan

for the generator – *information loss* and *classification loss*. The former measures the statistical differences between the 'embeddings' (extracted features from the discriminator before the sigmoid function) of the real data and generated data as the sum of ℓ_2 norms of the mean and standard deviation of vectors. The latter measures the difference between the label of generated data and the label that the classifier (trained on the real data) assigns to the same generated data.

Conditional tabular GAN (CTGAN) [19] is another GAN-based model for the synthesis of tabular datasets. Its first main feature is *mode-specific normalization*. Every continuous column C_i is estimated by the sum of scaled Gaussian distributions using VGM (variational Gaussian mixture) [2]. Next, for each value c_{ij} in C_i we choose one Gaussian distribution based on their probability densities in c_{ij} and normalize the value according to this distribution. As a result, we get a one-hot vector describing which distribution we sampled, as well as the normalized value. The second feature is the use of conditional GAN architecture. Every discrete feature D_i is encoded as a one-hot vector. Conditional vector, describing a certain state of discrete feature $D_i = t$, is defined as a concatenation of m_j – mask vectors for each discrete feature j. m_j is filled with $|D_j|$ zeros if $i \neq j$, and if $i = j$, m_j is a one-hot vector for $d_i = t$. In order to guarantee that the generator will create a data entry with $D_i = t$, another loss is introduced – *generator loss*, which is equal to the cross-entropy between the mask vector m_i and the vector of the generated entry responsible for the one-hot encoding of D_i. Another feature is *training-by-sampling* – change in data sampling during training. Instead of choosing objects uniformly, each training step a discrete column D_i is chosen uniformly. After that, we choose the value $t \in D_i$ at random with probability equal to the logarithm of its frequency in D_i. Finally, we choose the object with $D_i = t$ uniformly at random. This way the model will see minority values in discrete features more often.

CTABGAN [20] combines the methods of Table-GAN and CTGAN, as well as handling mixed variables, long-tail distributions, and extending training-by-sampling to continuous columns.

3 Experimental setup

3.1 Datasets

We pick two commonly used datasets: `adult` [1] and `credit`[2]. The former has a lot of discrete columns, so we can observe the impact of the conditional generator. The latter is extremely unbalanced – it has only 492 fraud entries, while having 284807 total entries. Table 3.1 contains the description of the datasets.

[2]https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud (accessed on March 31, 2025)

Table 3.1: Datasets description.

name	train, #entries	test, #entries	% of minority class	#continuous columns	#discrete columns
adult	32561	16281	23.9282%	6	9
credit	227845	56962	0.1727%	30	1

3.2 Models

In this paper we test four models. The first one is vanilla GAN, similar to [18]. One change we introduce is the conditional architecture. We introduce condition (i.e., a one-hot encoded vector of the target class), which indicates from which distribution we want to sample. Thus, we change the way oversampling is done: in the paper, to oversample the minority class, GAN is trained exclusively on the minority class labeled data. Instead, we train a CGAN on the whole dataset. This way, when we want to oversample a minority class, we can just generate more data entries specifically of this class.

The second model is Table-GAN [15]. We implement the similar change to Table-GAN, adding a condition vector to the features/noise in discriminator/generator respectively, before reshaping the resulting vector into the square matrix. Another change that we introduce is the way in which we approximate the information loss. Instead of the moving weighted average, we just compute its approximation, similar to other losses. We also introduce the w_{info} hyperparameter – the weight with which we scale the information loss. We observe that without it (i.e., with $w_{\text{info}} = 1$), information loss dominates all other losses and slows down their convergence. We set $w_{\text{info}} = 0.1$.

The third model is CTGAN [19]. In the paper, WGAN loss with gradient penalty [9] is used. We change it to classic GAN loss, as it is used in all other models. It is also worth noting that instead of a simple MLP, the paper uses residual linear layers in the generator and the PacGAN [12] architecture in the discriminator.

The forth model combines the techniques of Table-GAN and CTGAN, similarly to CTAB-GAN [6]. We use the CNNs instead of MLP and add the classifier network along with all losses from the CTGAN and Table-GAN papers. As with Table-GAN, we change the information loss computation.

Across all models, MLPs have three linear neural layers with the hidden layer size equal to 256. CNNs have the architecture suggested in [17], where each convolution layer halves the matrix side and doubles the number of channels. For `adult` CNN has 3 layers, while for `credit` – 2. The number of initial channels for CNN is set to 64.

All models are trained for 300 epochs, with batch size equal to 500 and epoch length equal to 65. All neural networks use the Adam [11] optimizer with constant learning rate equal to $2 \cdot 10^{-4}$.

3.3 Evaluation

In order to evaluate the models, we measure the difference between the quality of classifiers trained on the synthesized and original data. More specifically, we split the dataset on the `train` and `test` datasets. We train the synthesizers on `train` dataset, then generate the `synthetic` dataset with the same number of entries for each target class. After that we train classification models (gradient boosting [16], random forest [10], logistic regression [4]) on the `train` and `synthetic` datasets and compare their performance on `test` dataset. It is done by comparing F1 and ROC-AUC metrics.

Also, we evaluate the oversampling of the minority class in imbalanced datasets. Similarly, we divide the dataset into `train` and `test` and train the synthesizers on `train`. After that, we add to the `train` dataset new generated entries of the minority class and compare the performance of the aforementioned classification models trained on `train` and on the new supplemented dataset.

3.4 Ablation

We also conduct several ablation studies:

- One of the main features of Table-GAN is using CNNs on the features reshaped into 2D matrices. We test its utility by replacing the 2D convolutions in CTABGAN by linear layers.
- The main feature of CTGAN model is training-by-sampling (TBS). We remove it from CTABGAN by sampling discrete features and objects uniformly from the dataset instead of log frequency. We still keep the conditional generator loss.
- CTGAN discriminator uses the PacGAN idea, while Table-GAN discriminator does not. We can add it to the CTABGAN discriminator and compare it with the CTABGAN without this modification.

4 Results

First we present results and a comparison of different synthesizers, and then the oversampling results.

4.1 Comparison of the models

We compare the metrics achieved by several classification models, averaged across 5 random seeds[3]. We use gradient boosting[4], random forest, and logistic regression[5]. All classifiers are used with default parameters. Results are summed up in Table 4.1.

Table 4.1: Comparison of the model performances on different datasets with several classifiers: gradient boosting, linear regression, random forest. Best results for each classifier achieved on synthesized data are bolded.

	GAN		TableGAN		CTGAN		CTABGAN		Original data	
dataset name	F1 ↑	AUC ↑	F1 ↑	AUC ↑	F1 ↑	AUC ↑	F1 ↑	AUC ↑	F1 ↑	AUC ↑
Gradient boosting										
`adult`	0.288	0.565	0.467	0.649	0.446	0.642	**0.609**	**0.781**	0.702	0.792
`credit`	0.003	0.43	0.003	0.5	0.562	0.746	**0.61**	**0.873**	0.84	0.884
Logistic regression										
`adult`	0.315	**0.585**	**0.347**	0.467	0.178	0.547	0.34	0.581	0.368	0.608
`credit`	0	0.142	0.003	0.506	0.503	0.758	**0.624**	**0.867**	0.626	0.793
Random forest										
`adult`	0.296	0.569	0.472	0.654	0.409	0.625	**0.61**	**0.782**	0.663	0.769
`credit`	0.003	0.43	0.003	0.499	**0.63**	0.765	0.549	**0.884**	0.836	0.879

As we can see, our synthesizers can create tabular data that can be used for training machine learning models, achieving similar results with models trained on original data. We observe that CTGAN and CTABGAN work well even on extremely unbalanced datasets, such as `credit`, while Table-GAN and vanilla GAN fail to correctly learn from such distributions. We also note that the introduced features from the Table-GAN and CTGAN papers do not interfere with each other and can be combined to achieve better quality.

4.2 Ablation results

We present ablation study results for CTABGAN model in Table 4.2.

From these results we can see that training-by-sampling is crucial for imbalanced datasets, such as `credit`, as without it models do not see enough minority class entries and fail to learn the distribution properly, resulting in 0 on F1 metric.

We observe that CTABGAN with PacGAN architecture displays slightly worse results than the one without modifications, which in turn displays slightly worse results than the CTABGAN with fully connected layers. While the difference in metrics is not huge, when we look at the distributions, we notice that CTABGAN with MLP modification is displaying mode collapse.

[3]We omit standard deviation since it has the order of 10^{-3} for every evaluation.
[4]We use Catboost 1.2.5 library.
[5]We use scikit-learn 1.5.1 library.

Table 4.2: Ablation study results for CTABGAN.

	CTABGAN		w./o. TBS		w. pac = 10		MLP	
dataset name	F1 ↑	AUC ↑	F1 ↑	AUC ↑	F1 ↑	AUC ↑	F1 ↑	AUC ↑
Gradient boosting								
`adult`	0.609	0.781	0.353	0.604	0.582	0.713	0.637	0.781
`credit`	0.61	0.873	0.009	0.502	0.595	0.878	0.649	0.873
Logistic regression								
`adult`	0.34	0.581	0.329	0.557	0.394	0.617	0.382	0.602
`credit`	0.624	0.867	0.429	0.844	0.607	0.873	0.401	0.861
Random forest								
`adult`	0.61	0.782	0.206	0.557	0.572	0.707	0.627	0.768
`credit`	0.549	0.884	0.349	0.764	0.504	0.89	0.328	0.899

CTABGAN without changes experiences it less, while CTABGAN with pac = 10 almost does not. For example, it can be seen in the `adult` dataset in Figure 4.1. The difference in metrics can be explained by the fact that CTABGAN with convolutions has more parameters than CTABGAN with fully connected layers, so the former, especially with pac = 10, requires longer to train. It is also worth noting that CNNs are more scalable: with mode-specific normalization, table datasets with a lot of complex continuous features or discrete features with a lot of different values can produce highly dimensional inputs that CNNs can handle with fewer parameters.

4.3 Oversampling results

We successively add more and more CTABGAN synthesized entries of the minority class to the train dataset and measure the aforementioned two metrics, as well as common measures for binary classification tasks: accuracy, recall, and precision. We omit random forest since its results are similar to gradient boosting. Table 4.3 contains the results with respect to the number of added generated entries of the minor class to the original train dataset, consisting of 227845 total entries, of which 405 belong to the minor class.

Table 4.3: GAN-based oversampling results for `credit`.

	Gradient boosting					Logistic regression				
#added	F1 ↑	AUC ↑	Acc ↑	Rec ↑	Prec ↑	F1 ↑	AUC ↑	Acc ↑	Rec ↑	Prec ↑
0	0.84	0.884	1.0	0.768	0.928	0.626	0.793	0.999	0.586	0.671
400	0.839	0.889	1.0	0.778	0.911	0.659	0.83	0.999	0.661	0.662
1k	0.833	0.888	1.0	0.776	0.9	0.659	0.841	0.999	0.682	0.643
2k	0.832	0.89	1.0	0.78	0.892	0.666	0.849	0.999	0.698	0.642
5k	0.833	0.892	1.0	0.783	0.889	0.631	0.854	0.999	0.708	0.586
10k	0.832	0.892	1.0	0.785	0.887	0.631	0.857	0.999	0.715	0.579
20k	0.831	0.892	1.0	0.784	0.883	0.623	0.859	0.999	0.719	0.564
50k	0.829	0.892	1.0	0.784	0.88	0.584	0.861	0.998	0.724	0.518

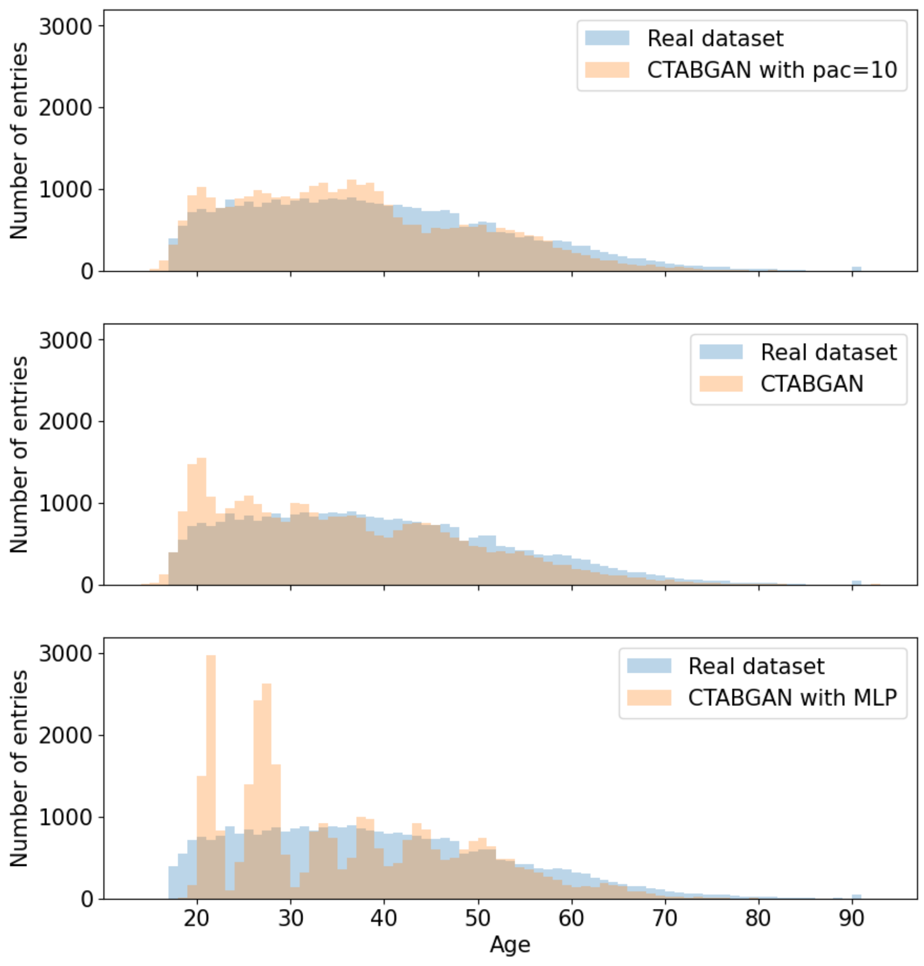

Figure 4.1: "Age" feature in the `adult` dataset generated by different configurations of CTABGAN compared to the real data.

For gradient boosting, the model performs well on the original imbalanced dataset, so the changes for it are minimal, while for logistic regression we observe the increase in F1 and ROC-AUC metrics. It is also worth noting that recall increases with the number of added generated data for all models. This means that oversampling can be useful in problems where the recall is crucial, for example, medical tasks, where we want to minimize the number of false negatives, i.e., undiagnosed diseases.

We compare this method to two basic techniques used for balancing the unbalanced dataset: introducing weights to the classes, giving a larger weight to a minor class, and removing entries of the major class. Results are displayed in Table 4.4 and Table 4.5 respectively. Table 4.4 presents the results with respect to w – the weight of the minor class, while the weight of the major class is always equal to 1. Table 4.5 displays the results with respect to the number of major class entries left in the dataset; "original" represents the original dataset.

We can see that, as expected, these methods increase the recall value of the models. For gradient boosting, quality remains largely the same, while for logistic regression, it drops propor-

Table 4.4: Balancing with weighted classes results for `credit`.

	Gradient boosting					Logistic regression				
w	F1 ↑	AUC ↑	Acc ↑	Rec ↑	Prec ↑	F1 ↑	AUC ↑	Acc ↑	Rec ↑	Prec ↑
1	0.84	0.884	1.0	0.768	0.928	0.626	0.793	0.999	0.586	0.671
2	0.842	0.884	1.0	0.769	0.931	0.623	0.813	0.999	0.626	0.626
5	0.844	0.886	1.0	0.772	0.931	0.587	0.819	0.999	0.64	0.557
10	0.846	0.887	1.0	0.775	0.931	0.574	0.829	0.998	0.658	0.525
20	0.846	0.888	1.0	0.776	0.93	0.539	0.841	0.998	0.683	0.474
50	0.845	0.888	1.0	0.777	0.927	0.494	0.85	0.997	0.703	0.421
100	0.844	0.89	1.0	0.779	0.921	0.453	0.859	0.996	0.721	0.378
500	0.838	0.891	1.0	0.782	0.904	0.405	0.868	0.992	0.743	0.335

Table 4.5: Balancing by deletion results for `credit`.

	Gradient boosting					Logistic regression				
#major	F1 ↑	AUC ↑	Acc ↑	Rec ↑	Prec ↑	F1 ↑	AUC ↑	Acc ↑	Rec ↑	Prec ↑
original	0.84	0.884	1.0	0.768	0.928	0.626	0.793	0.999	0.586	0.671
100k	0.83	0.886	1.0	0.772	0.898	0.613	0.813	0.999	0.626	0.609
50k	0.828	0.888	1.0	0.776	0.887	0.599	0.831	0.999	0.663	0.561
20k	0.814	0.892	0.999	0.783	0.851	0.557	0.837	0.998	0.675	0.49
10k	0.803	0.894	0.999	0.788	0.824	0.52	0.846	0.998	0.694	0.447
5k	0.787	0.895	0.999	0.79	0.793	0.483	0.854	0.997	0.711	0.403
2k	0.74	0.896	0.999	0.793	0.725	0.439	0.862	0.996	0.727	0.36
1k	0.68	0.898	0.998	0.797	0.654	0.399	0.868	0.994	0.741	0.323

tional to the severity of the method. This is not the case for the GAN oversampling, where the F1 metric and accuracy do not sharply decrease with the number of introduced synthetic entries.

5 Conclusion

We present an overview of best practices of using GANs for tabular data synthesis. We conclude that GAN-based models are capable of generating data that can be used to train machine learning algorithms comparable with algorithms trained on original data and can do so even for extremely unbalanced datasets while being relatively lightweight and cost-efficient. With the addition of conditional generator, it is possible to generate new entries of the chosen class. We use it to test the oversampling technique and observe that it performs better than the classic balancing methods and can improve the quality of classification models on unbalanced datasets. In conclusion, we argue that GANs can be successfully applied to the task of tabular data synthesis and be a viable tool for data anonymization and augmentation without requiring large amounts of time and compute.

References

[1] Barry Becker and Ronny Kohavi. *Adult*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20. 1996.

[2] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[3] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. “Language models are realistic tabular data generators”. In: *arXiv preprint arXiv:2210.06280* (2022).

[4] David R Cox. “The regression analysis of binary sequences”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 20.2 (1958), pp. 215–232.

[5] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. “Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.8 (2018), pp. 3784–3797. DOI: 10.1109/TNNLS.2017.2736643.

[6] Gabriel Eilertsen, Apostolia Tsirikoglou, Claes Lundström, and Jonas Unger. *Ensembles of GANs for synthetic training data generation*. 2021. arXiv: 2104.11797 [cs.CV]. URL: https://arxiv.org/abs/2104.11797.

[7] Val Andrei Fajardo, David Findlay, Charu Jaiswal, Xinshang Yin, Roshanak Houmanfar, Honglei Xie, Jiaxi Liang, Xichen She, and D.B. Emerson. “On oversampling imbalanced data with deep conditional generative models”. In: *Expert Systems with Applications* 169 (2021), p. 114463. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.114463. URL: https://www.sciencedirect.com/science/article/pii/S0957417420311155.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. “Improved training of wasserstein gans”. In: *Advances in neural information processing systems* 30 (2017).

[10] Tin Kam Ho. “Random decision forests”. In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282.

[11] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[12] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. *PacGAN: The power of two samples in generative adversarial networks*. 2018. arXiv: 1712.04086 [cs.LG]. URL: https://arxiv.org/abs/1712.04086.

[13] Boris Lubarsky. "Re-identification of "anonymized data"". In: *Georgetown Law Technology Review. Available online: https://www. georgetownlawtechreview. org/re-identification-of-anonymized-data/GLTR-04-2017 (accessed on 10 September 2021)* (2010).

[14] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG]. URL: https://arxiv.org/abs/1411.1784.

[15] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. "Data synthesis based on generative adversarial networks". In: *Proceedings of the VLDB Endowment* 11.10 (June 2018), pp. 1071–1083. ISSN: 2150-8097. DOI: 10.14778/3231751.3231757. URL: http://dx.doi.org/10.14778/3231751.3231757.

[16] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. "CatBoost: unbiased boosting with categorical features". In: *Advances in neural information processing systems* 31 (2018).

[17] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG]. URL: https://arxiv.org/abs/1511.06434.

[18] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. *Data Augmentation Using GANs*. 2019. arXiv: 1904.09135 [cs.LG]. URL: https://arxiv.org/abs/1904.09135.

[19] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. "Modeling tabular data using conditional gan". In: *Advances in neural information processing systems* 32 (2019).

[20] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. "Ctab-gan: Effective table data synthesizing". In: *Asian Conference on Machine Learning*. PMLR. 2021, pp. 97–112.