FCT-GAN: Enhancing Global Correlation of Table Synthesis via Fourier Transform

FCT-GAN: Enhancing Global Correlation of Table Synthesis via Fourier Transform

Zilong Zhao*
Delft University of Technology
Delft, Netherlands
z.zhao-8@tudelft.nl

Robert Birke
University of Turin
Turin, Italy
robert.birke@unito.it

Lydia Y. Chen
Delft University of Technology
Delft, Netherlands
lydiaychen@ieee.org

ABSTRACT

An alternative method for sharing knowledge while complying with strict data access regulations, such as the European General Data Protection Regulation (GDPR), is the emergence of synthetic tabular data. Mainstream table synthesizers utilize methodologies derived from Generative Adversarial Networks (GAN). Although several state-of-the-art (SOTA) tabular GAN algorithms inherit Convolutional Neural Network (CNN)-based architectures, which have proven effective for images, they tend to overlook two critical properties of tabular data: (i) the global correlation across columns, and (ii) the semantic invariance to the column order. Permuting columns in a table does not alter the semantic meaning of the data, but features extracted by CNNs can change significantly due to their limited convolution filter kernel size. To address the above problems, we propose FCT-GAN – the first conditional tabular GAN to adopt Fourier networks into table synthesis. FCT-GAN enhances permutation invariant GAN training by strengthening the learning of global correlations via Fourier layers. Extensive evaluation on benchmarks and real-world datasets show that FCT-GAN can synthesize tabular data with better (up to 27.8%) machine learning utility (i.e. a proxy of global correlations) and higher (up to 26.5%) statistical similarity to real data. FCT-GAN also has the least variation on synthetic data quality among 7 SOTA baselines on 3 different training-data column orders.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

GAN, Fourier transform, Tabular data.

ACM Reference Format:
Zilong Zhao, Robert Birke, and Lydia Y. Chen. 2023. FCT-GAN: Enhancing Global Correlation of Table Synthesis via Fourier Transform. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Currently working at TU Munich. Email: zilong.zhao@tum.de

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

While data sharing is crucial for knowledge development, privacy concerns and strict regulations limit its full effectiveness. An emerging solution is to leverage synthetic data generated by machine learning models. Starting from images [5], generative adversarial networks (GAN) have powered data synthesis of various types, e.g., text [22] and audio [4]. Due to its ample application scenarios in areas such as medicine [3] and finance [1], synthetic tabular data is of primary interest. Compared to image data, tabular data differs by two key properties. First, unlike pixel positions, the column order has no impact on the semantics of a table row and column correlations are independent from the distance between columns. Second, tabular data is composed of different column types such as continuous, categorical or mixed variables instead of only a continuous one, i.e., pixel. The former impacts the synthesis quality of GANs designed for images on tabular data. The latter calls for extra feature engineering for non-continuous variables.

Convolutional Neural Networks (CNN) brought a breakthrough in image oriented machine learning due to their ability to extract local spatial features well [10]. Similarly, prior studies [23, 24] show that tabular GANs, which adopt CNNs as generator and discriminator, achieve better synthesis quality than using fully-connected networks (FCN). Adopting CNNs however may overlook relations between columns which happen to be too distant due to the size of the convolution filter. Even though permuting training data columns does not change its semantic meaning, the local feature presentation extracted by convolution layers is distorted. Consequently, the column order impacts the model performance and a given original column order may not maximise the CNN effectiveness. Feature encoding exacerbates this issue. While one-hot encoding is shown to better recover the categorical variable distribution for tabular GAN [21], it inevitably increases the data dimensions (i.e., number of columns) making it increasingly challenging for CNN-based tabular GANs to effectively capture global relations. Hence the column order can be seen as yet another hyper parameter to tune. To limit tuning efforts and training costs, it becomes imperative to devise an algorithm which ensures stable and high quality synthesis independent of the training column order.

To address the above problems, we propose a Fourier conditional tabular GAN - FCT-GAN leveraging a transformer-style architecture using CNN, to capture local relations as tokens, and Fourier Network Blocks (FNB), to model global dependencies. Our results show that FCT-GAN outperforms state-of-the-art (SOTA) up to 27.8% in machine learning utility and 26.5% in statistical similarity on 7 datasets. Thanks to Fourier blocks ability to capture global relations, our results also show that among 3 different column orders, FCT-GAN has the least variation in synthesis quality among

all comparisons. The main contributions of this study can be summarized as follows: (1) We introduce the Fourier transform into tabular GAN training and use it to design the generator and discriminator architectures. (2) We adopt a CNN-based input tokenizer feeding into FNB layers for discriminator. This novel architecture can capture both local (i.e., by tokenizer) and global (i.e., by Fourier layer) correlations of tabular data. (3) We leverage FNB layers in the generator to better incorporate global relations during upscaling leading to superior quality synthesised data. (4) We extensively evaluate FCT-GAN on 7 datasets against 7 SOTA synthesizers including models such as GAN, Variational AutoEncoder (VAE) and Diffusion model.

2 RELATED WORK

Deep Generative Models. Current state-of-the-art introduces several deep generative models for tabular data synthesis. TableGAN [16] implements an auxiliary classification model along with discriminator training to enhance column dependency in the synthetic data. TVAE [21], CT-GAN [21] and CTAB-GAN [23] improve data synthesis by introducing several preprocessing steps for categorical, continuous or mixed data types which encode data columns into suitable form for GAN training. The conditional vector designed by CT-GAN and later improved by CTAB-GAN also helps the GAN training to reduce mode-collapse on minority categories. CTAB-GAN+ [24], DTGAN [**?**], and IT-GAN [11] generate tabular data without risking privacy of original data by either adopting differential privacy or controlling the negative log-density of real records during the GANs training. [25] is the first paper to discuss the permutation invariant in table synthesis. It uses the same framework as IT-GAN, which uses an autoencoder to transform input table into latent vector. TabDDPM [9] is based on denoising diffusion probabilistic models (DDPM) [7], it uses two different diffusion models to synthesize categorical and continuous columns.

Fourier Networks. The Fourier transform has played an important role in image processing for decades [20]. Incorporating Fourier transform into the neural network architecture design has been studied in many vision works [2, 15, 19]. Recent work also leverages the Fourier transform to design deep neural networks to solve partial differential equations (PDE) [13] and natural language processing tasks [12]. Our Fourier network block architecture design is inspired by the Global Filter Network (GFNet) [18]. We take the design of the input tokenization and global filter layer from GFNet and use it in our design of the generator and discriminator.

3 FCT-GAN

We start with the use of CNN and FNB then their role in FCT-GAN.

3.1 CNN-based Tokenization and Fourier Network Blocks

Combining CNN tokenization process and FNB is originally explored for image classification [18] and used to compose the architecture illustrated in Fig. 1. The CNN tokenizes the input into local relation embeddings and the FNB transforms these local embeddings into the frequency domain to extract global relations.

As shown in Fig. 1, each input data row is encoded as a square-like image with pixels representing the column values. We define a

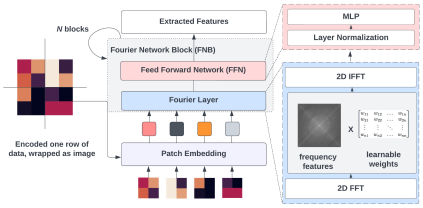

Figure 1: Input Tokenization and Fourier Network Block.

CNN layer with a $k \times k$ kernel and stride k the same as the kernel size. With this setting the CNN effectively acts as a tokenizer which divides an image of size $N \times N$ into $\frac{N}{k} \times \frac{N}{k}$ tokens embedding the local relations of a subpart of the original image.

Fourier Network Blocks consist of two parts: (1) Fourier layer and (2) Feed Forward Network. The Fourier layer consists of three operations: (i) 2D discrete Fourier transform, (ii) element-wise multiplication between frequency-domain features and learnable weights and (iii) 2D inverse discrete Fourier transform. The frequency-domain features represent relations between the inputs and the learnable weights allow to selectively filter them. We use real fast Fourier transform (rFFT) for operation since our input is a real tensor with no imaginary component [18].

3.2 Design of FCT-GAN

We first introduce the the architectures of the generator and discriminator. Then, we discuss the training procedure and loss functions.

3.2.1 Generator. The objective of the generator is to capture the joint probability distribution of all columns to synthesize high fidelity data. In FCT-GAN, we leverage the FNB to achieve the above goal. We opt for a design which iteratively upscales the resolution at different stages using FNB to incorporate the relations. Hence our generator gradually increases the input sizes and reduces the embedding dimension at each stage. Fig. 2 depicts on the left a generator design where the target resolution is 32×32 as an example. The latent noise and conditional vector are fed into an embedding layer (i.e., a Multi-Layer Perceptron having input dimension the same as the latent noise plus the conditional vector and the output dimension $N = H_0 \times W_0 \times C_0$) to convert them to a $H_0 \times W_0 \times C_0$ (by default we use H_0=W_0=4, C_0=256) vector. The vector is then reshaped into a $H_0 \times W_0$ resolution feature map with each point being a C_0-dimensional embedding. The result is fed into the first FNB. After each FNB, we insert an upsampling module. There are several choices to achieve resolution-upscaling, such as bicubic interpolation [8] or transpose convolutional operation. To mitigate memory usage and computation, we use the PIXELSHUFFLE function. This upscales the resolution of feature maps by a factor of 2 while reducing embedding dimension to a quarter. We repeat the FNB and UpScale stages until we reach the target resolution. The final linear flatten layer is used to project the embedding into 1-dimension.

3.2.2 Discriminator. Fig. 2 shows on the right our discriminator architecture. The objective of the discriminator is to distinguish real and fake data. We leverage transformer style architecture with repeated layers of FNBs from Fig. 1 acting as attention layers. Since FCT-GAN adopts the training structure of Wasserstein GAN plus gradient penalty (WGAN+GP) [6], our discriminator just outputs a

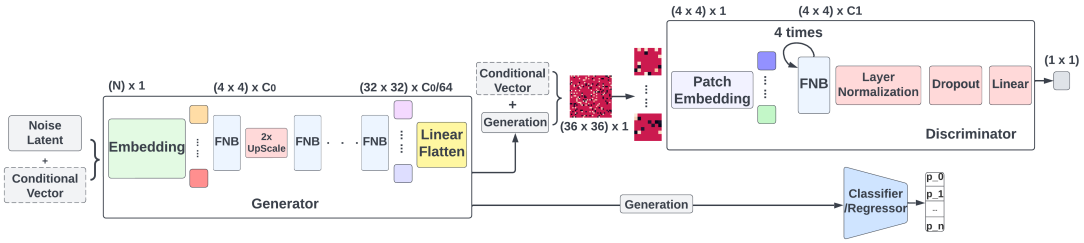

Figure 2: Structure of FCT-GAN. Target generation dimension and discriminator input dimension are setting to 32×32 and 36×36 as an example.

scalar score. The generated data from the generator and the conditional vector are concatenated, wrapped as an image (padding missing values with zeros), and fed to the patch embedding layer of the discriminator. Within patch embedding, there is a CNN filter with a $k \times k$ kernel ($k = 9$ by default), C_1 ($C_1 = 256$ by default) output channels, and stride same as the kernel size. Different from the generator, the inputs and outputs of the FNBs have constant dimensions, which shares the same setting as in [18] to construct image classifier. After 4 successive FNBs, the extracted features are flattened and downscaled to one single value.

3.2.3 GAN training loss. To improve the stability of GAN training, FCT-GAN adopts WGAN+GP [6] loss. Moreover, to elevate the synthesizing performance, we add three extra training losses for the generator: (1) information loss, (2) generator loss and (3) classifier/regressor loss. The information loss matches the first-order (i.e., mean) and second-order (i.e., standard deviation) statistics of synthesized and real records. This leads to synthetic records with the same statistical characteristics as real records. The generator loss measures the difference between the given condition and the output class of the generator. This loss helps the generator learn to produce the exact same class as the given conditions. FCT-GAN incorporates an auxiliary classifier/regressor as suggested in [16, 23, 24]. For each synthesized data item the classifier/regressor outputs a predicted value using the synthesized features. The classifier/regressor loss quantifies the discrepancy between the synthesized and predicted value, helps to increase the semantic integrity of synthetic records.

4 EVALUATION

In this section, we evaluate the effectiveness of synthetic data generated by FCT-GAN in maintaining the global correlation and statistical similarity to the original data. We also investigate FCT-GAN ability to counter the influence of column permutation.

4.1 Experiment Setup

Datasets. All algorithms are tested on 8 machine learning datasets. **Intrusion**, **Adult** and **Covertype** are from the UCI machine learning repository[1]. **Credit** and **Loan** are from Kaggle[2]. These five tabular datasets are defined to have a categorical variable as the target for conducting classification tasks. **Insurance** and **King** from Kaggle are included as regression datasets.

Due to computing resource limitations, 50K rows of data are sampled randomly in a stratified manner with respect to the target variable for Covertype, Credit and Intrusion datasets. The Adult, Loan, Insurance and King datasets are taken in their entirety. We assume that the data type of each variable is known before training. For stability on column permutation analysis, three column orders are considered: (i) *Original*: maintains the order as in the data downloaded from dataset source. (ii) *Order by data type*: puts all the continuous columns at the beginning and all categorical columns at the end. (iii) *Order by data correlation*: first calculates the pair-wise correlations between all columns. Then it sorts columns based on the absolute correlation value with highly correlated pairs in front and less correlated pairs later. Duplicate columns are skipped.

Baselines FCT-GAN is compared with 7 other SOTA tabular data synthesizers: TabDDPM, IT-GAN, CTAB-GAN, CTAB-GAN+, TableGAN, CT-GAN and TVAE. To show the impact of newly designed generator and discriminator, we test two variants of FCT-GAN: FCT-GAN$_G$ and FCT-GAN$_D$. FCT-GAN$_G$/FCT-GAN$_D$ only keeps the generator/discriminator of FCT-GAN using as counterpart the CNN-based discriminator/generator from CTAB-GAN+.

Environment A machine with 32 GB memory, a GeForce RTX 2080 Ti GPU and a 10-core Intel i9 CPU under Ubuntu 20.04.

4.2 Evaluation Metrics

4.2.1 Machine learning (ML) utility. Classification and regression datasets are quantified using different metrics, but they share the same evaluation process. We first train each algorithm on the training data and use the trained model to generate synthetic data of the same size as the training data. Then we use the training data and synthetic data to train same set of ML algorithms. For classification dataset, we choose decision tree classifier, linear support-vector-machine (SVM), random forest classifier, multinomial logistic regression and multilayer perceptron (MLP). For regression dataset, we choose linear regression, ridge regression, lasso regression and Bayesian ridge regression model. Finally, we use the same test set to separately test the two sets of ML models trained on the original and synthetic data. We use accuracy, F1-score and AUC as evaluation metrics for classification, and mean absolute percentage error (MAPE), explained variance score (EVS) and R^2 score as the metrics for regression. In the end, we calculate the difference between the results of the two sets of ML models for each metric.

4.2.2 Statistical similarity (SS). Three metrics are used to quantify the statistical similarity between real and synthetic data.

Average Jensen-Shannon divergence (Avg-JSD). The JSD [14] provides a measure to quantify the difference between the probability mass distributions of individual categorical variables belonging to real and synthetic data. This metric is bounded between 0 and 1 and is symmetric allowing for an easy interpretation of results.

[1] http://archive.ics.uci.edu/ml/datasets
[2] https://www.kaggle.com/

Table 1: ML Utility and SS result. R./C. represent the results averaged on the regression/classification datasets. SS Difference averaged on all 7 datasets. Im. to 2nd Best shows improvement of FCT-GAN vs. 2nd best baseline.

Method	ML Utility Difference R.			ML Utility Difference C.			Statistical Similarity Difference		
	MAPE	EVS	R^2	Acc.(%)	F1-score	AUC	Avg-JSD	Avg-WD	Diff. Corr.
FCT-GAN	**0.037**	**0.022**	**0.043**	**4.92**	**0.065**	**0.039**	**0.010**	**0.034**	**1.57**
FCT-GAN$_D$	0.042	0.041	0.065	4.98	0.066	0.053	0.014	0.043	1.72
FCT-GAN$_G$	0.131	0.041	0.109	9.24	0.140	0.066	0.014	0.047	2.07
TabDDPM	0.066	0.030	0.047	7.91	0.113	0.143	0.018	0.114	3.41
IT-GAN	0.608	0.409	0.478	8.95	0.183	0.229	0.029	0.097	2.38
CTAB-GAN+	**0.037**	0.025	**0.043**	5.23	0.090	0.041	0.011	0.043	1.65
CTAB-GAN	0.059	0.594	0.707	8.90	0.107	0.094	0.021	0.056	1.70
CT-GAN	0.871	0.594	0.709	21.51	0.274	0.253	0.037	0.090	2.96
TVAE	0.243	0.078	0.215	11.11	0.100	0.230	0.025	0.122	2.41
Table-GAN	0.338	0.434	0.479	11.40	0.130	0.169	0.028	0.231	3.23
Im. to 2nd Best	0%	13.6%	0%	6.3%	27.8%	5.1%	10%	26.5%	5.1%

Avg-JSD averages the JSDs computed for each categorical column to obtain a compact comprehensible score.

Average Wasserstein distance (Avg-WD). The Wasserstein distance [17] is used to measure distance between real and synthetic continuous variable distributions. We use WD because JSD is numerically unstable for evaluating the quality of continuous variables, especially when there is no overlap between distributions.

Difference in pair-wise correlation (Diff. Corr.). To evaluate the preservation of column dependency in synthetic data, we first compute the pair-wise correlation matrix for the columns within real and synthetic datasets individually. *Pearson correlation coefficient* is used between any two continuous variables. Similarly, the *uncertainty coefficient* is used to measure the correlation between any two categorical features. And the *correlation ratio* between categorical and continuous variables is used. Note that the dython[3] library is used to compute. Finally, the difference between pair-wise correlation matrices for real and synthetic datasets is computed.

To quantify the synthesis quality variation caused by column permutations, we define a metric **Maximal Absolute Variation (MAV)**. For the values V^i_{DM} of a metric $\mathcal{M}$ (e.g., Avg-JSD) on dataset $\mathcal{D}$, computed on the i^{th} column order out of N, $\mathcal{L} = \{V^1_{DM}, V^2_{DM} \cdots V^N_{DM}\}$, MAV is computed as: MAV = $max(\mathcal{L}) - min(\mathcal{L})$.

4.3 Result Analysis

4.3.1 Quality of synthetic data. Tab. 1 shows the ML utility and statistical similarity results of 10 algorithms on 7 datasets. The table is grouped into three sections: averaged ML Utility results across 2 regression datasets, averaged ML utility for 5 classification datasets and averaged statistical similarity on all 7 datasets. Best results are highlighted in bold. One can see that FCT-GAN outperforms all the baselines on all the metrics. Excluding its own variants, FCT-GAN also significantly outperforms the second best algorithm, i.e., CTAB-GAN+, in most metrics. Note that in [9], it reports that TabDDPM outperforms CTAB-GAN+ in most of their datasets. From our experiments, except **Intrusion** dataset, results of TabDDPM and CTAB-GAN+ are relatively similar. But for **Intrusion**, which contains 20 classes in the target column, TabDDPM performs much worse than most of the SOTAs. The results show that FCT-GAN improves CTAB-GAN+ by 27.8% on F1-score, which confirms that the new architecture can enhance the ability to capture global column dependency. Among FCT-GAN and its variants, FCT-GAN$_D$ shows the performance relatively close to FCT-GAN. FCT-GAN$_G$ is still better than many SOTAs, but it degrades more from FCT-GAN compared to FCT-GAN$_D$. This shows the generator can not fully demonstrate its ability without a well-designed discriminator, but a stronger discriminator can level up the synthesizing ability of the generator by better identifying fake data and hence forcing the generator to improve the fidelity of the generated data.

[3]http://shakedzy.xyz/dython/modules/nominal/#compute_associations

Table 2: MAV of the difference of ML Utility and SS on three type of column orders between original and synthetic data.

Method	ML Utility Difference R.			ML Utility Difference C.			Statistical Similarity Difference		
	MAPE	EVS	R^2	Acc.(%)	F1-score	AUC	Avg-JSD	Avg-WD	Diff. Corr.
FCT-GAN	**0.04**	**0.02**	**0.03**	**1.12**	**0.01**	**0.01**	3e-3	2e-3	**0.11**
FCT-GAN$_D$	0.11	0.03	0.2	1.81	0.03	0.02	7e-3	5e-3	0.2
FCT-GAN$_G$	0.08	0.11	0.08	3.27	0.05	0.03	9e-3	**1e-3**	0.2
TabDDPM	0.12	0.03	0.05	2.14	0.03	**0.01**	7e-3	0.01	0.21
IT-GAN	0.17	0.05	0.11	1.21	0.03	0.02	8e-3	0.01	0.15
CTAB-GAN+	0.28	0.26	0.35	2.18	0.03	0.03	5e-3	4e-3	0.32
CTAB-GAN	0.05	0.56	0.67	6.88	0.17	0.12	6e-3	7e-3	0.3
CT-GAN	0.57	0.25	0.38	8.82	0.06	0.09	0.01	0.01	0.17
TVAE	0.15	0.03	0.05	3.64	0.05	0.05	**2e-3**	3e-3	0.15
Table-GAN	0.09	0.3	0.16	13.58	0.06	0.02	0.01	9e-3	0.28

4.3.2 Stability on column permutation. We show the MAV of each metric on all algorithms among three column orders in Tab. 2. FCT-GAN achieves the smallest MAV in seven out of nine metrics and is the second best on the other two metrics. Specifically, the seven metrics (i.e., ML utility metrics and Diff. Coff.) in which FCT-GAN performs the best are the indicators designed to measure the column correlations retained in the synthesized data, assessing its fidelity to the original data. This means that no matter which column order is used, FCT-GAN can always capture the global column dependencies better than any other baseline. IT-GAN also performs well in stability due to its usage of autoencoder for transforming input into a latent vector. TabDDPM performs well in several metrics due to its separate training of continuous and categorical columns. TVAE and CT-GAN all use FCN for both generator and discriminator, and TVAE shows better stability. That indicates the VAE framework is more stable than GAN under column permutation for table synthesis. Moreover one can note that changing training data column order generally influences more ML utility than statistical similarity. This means that for most of the tabular GAN algorithms, it is more difficult to capture the column dependencies rather than to model the distribution of every single variable.

5 CONCLUSION

In this paper, we present a novel tabular GAN algorithm – FCT-GAN. FCT-GAN leverages the transformer-style tokenizer and Fourier network blocks to design the generator and discriminator. For discriminator, the tokenizer uses a CNN-based filter to extract local spatial features from original input data and tokenizes them for Fourier network blocks. This design takes both local and global features into account. The generator imitates the design of CNN-based GANs, which piles up Fourier network blocks and upscales feature dimensions at each layer by 2× until reaching the target size. The proposed method surpasses state-of-the-art performance. It also shows brilliant stability to counter the impact of training data column permutations on the synthetic data quality.

Acknowledgements. This work was partly supported by the Spoke "FutureHPC & BigData" of the ICSC – Centro Nazionale di Ricerca in "High Performance Computing, Big Data and Quantum Computing", funded by European Union – NextGenerationEU.

REFERENCES

[1] Samuel A. Assefa, Danial Dervovic, Mahmoud Mahfouz, Robert E. Tillman, Prashant Reddy, and Manuela Veloso. 2020. Generating Synthetic Data in Finance: Opportunities, Challenges and Pitfalls. In *ICAIF* (New York, New York). New York, NY, Article 44, 8 pages. https://doi.org/10.1145/3383455.3422554

[2] Lu Chi, Borui Jiang, and Yadong Mu. 2020. Fast fourier convolution. *NeurIPS* 33 (2020), 4479–4488.

[3] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. 2017. Generating multi-label discrete patient records using generative adversarial networks. *arXiv preprint arXiv:1703.06490* (2017).

[4] Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208* (2018).

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *NeurIPS* 27 (2014).

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved Training of Wasserstein GANs. In *NeurIPS* (Long Beach, California, USA). 5769–5779.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.

[8] R. Keys. 1981. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29, 6 (1981), 1153–1160. https://doi.org/10.1109/TASSP.1981.1163711

[9] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2022. TabDDPM: Modelling Tabular Data with Diffusion Models. *arXiv preprint arXiv:2209.15421* (2022).

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (Lake Tahoe, Nevada) *(NIPS'12)*. Red Hook, NY, USA, 1097–1105.

[11] Jaehoon Lee, Jihyeon Hyeong, Jinsung Jeon, Noseong Park, and Jihoon Cho. 2021. Invertible Tabular GANs: Killing Two Birds with One Stone for Tabular Data Synthesis. *NeurIPS* 34 (2021), 4263–4273.

[12] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824* (2021).

[13] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895* (2020).

[14] J. Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory* 37, 1 (1991), 145–151. https://doi.org/10.1109/18.61115

[15] Michael Mathieu, Mikael Henaff, and Yann LeCun. 2013. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851* (2013).

[16] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data Synthesis Based on Generative Adversarial Networks. *Proc. VLDB Endow.* 11, 10 (2018), 1071–1083.

[17] Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. 2017. On Wasserstein Two-Sample Testing and Related Families of Nonparametric Tests. *Entropy* 19, 2 (2017). https://doi.org/10.3390/e19020047

[18] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. 2021. Global filter networks for image classification. *NeurIPS* 34 (2021), 980–993.

[19] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. *NeurIPS* 33 (2020), 7462–7473.

[20] G.K. Wallace. 1992. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics* 38, 1 (1992), xviii–xxxiv. https://doi.org/10.1109/30.125072

[21] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular data using Conditional GAN. In *NeurIPS*, Vol. 32. Curran Associates, Inc., 7335–7345. https://proceedings.neurips.cc/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf

[22] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

[23] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. 2021. CTAB-GAN: Effective Table Data Synthesizing. In *Proceedings of The 13th Asian Conference on Machine Learning*, Vol. 157. 97–112. https://proceedings.mlr.press/v157/zhao21a.html

[24] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. 2022. CTAB-GAN+: Enhancing Tabular Data Synthesis. *arXiv preprint arXiv:2204.00401* (2022).

[25] Yujin Zhu, Zilong Zhao, Robert Birke, and Lydia Y Chen. 2022. Permutation-Invariant Tabular Data Synthesis. *arXiv preprint arXiv:2211.09286* (2022).