Missing Data Imputation for Real Time-series Data in a Steel Industry using Generative Adversarial Networks

Kisan Sarda
Department of Engineering
University of Sannio
Benevento 82100, Italy
kirasarda@unisannio.it

Amol Yerudkar
Department of Engineering
University of Sannio
Benevento 82100, Italy
ayerudkar@unisannio.it

Carmen Del Vecchio
Department of Engineering
University of Sannio
Benevento 82100, Italy
c.delvecchio@unisannio.it

IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society | 978-1-6654-3554-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IECON48115.2021.9589716

***Abstract*—On the verge of technology, manufacturing industries revolutionize into smart industries, which create a large amount of multivariate time-series data. However, due to sensors' failure, extreme environment, etc., the collected data are incomplete and have missing values at several instances that result in an erroneous analysis of the data. The key to resolving this problem is data imputation, i.e., replacing the missing values with synthetic values. In this paper, we introduce a generative adversarial network (GAN) framework to generate the synthetic data pertaining to the data imputation. Over the last decade, GANs have presented excellent results to generate synthetic data for images. By following this stream of research, we consider multivariate time-series data from a steel manufacturing industry and propose a GAN-based data imputation technique. We perform several computer simulations to validate and compare the performance of the proposed GAN method with state-of-the-art data imputation techniques.**

***Index Terms*—Generative adversarial networks, manufacturing systems, multivariate time-series data, data imputation.**

I. Introduction

With the advent of Industry 4.0, manufacturing industries are being adapted with advanced technology instead of traditional ways for maintenance, thereby generating a large amount of time-series data. However, due to extreme environments, sensors' failure, miscommunications, etc., the data are incomplete and are not useful to obtain fruitful information using analytical methods. The inaccuracy will become meaningless for the better and efficient performance of the manufacturing processes. Hence, an incomplete dataset causes a significant problem in data analysis. The problem has a solution of data imputation, whereby all occurrences of missing values (nan values) within a variable are replaced by similar values such as mean, median. Since the time-series data are not linearly dependent on past and current values, the statistical value imputation, i.e., mean or median values imputation, may endure inaccurate results for prediction or forecasting. Based on occurrence behavior, the missing values are mainly classified into three groups: missing completely at random (MCAR), missing at random (MAR), missing not at random (MNAR). Industrial data generally has missing values completely at random in the manufacturing processes. The missing data occur for many reasons as sensors' failure, collection errors, manual interference, etc. [1]. Practitioners generally adopt conventional methods, such as either case deletion or replacement of the samples based on direct imputation methods [2], [3], to resolve the problem. In the case deletion method, missing data samples are discarded that might lose the valuable information of the dataset, whereas discarding also affects the results when the percentage of missing values is higher or missing values have occurred at a rare event. The statistical imputation methods rely on replacement by a parameter value, such as mean/zero imputation, median imputation, most frequent value imputation, etc., that do not consider data distribution and reduces variability of the data. However, the following methods based on machine learning algorithms, namely expectation-maximization (EM) [4], k-nearest neighbor (KNN) based imputation [5], matrix factorization-based imputations, rarely consider the data distribution. Multivariate imputation by chained equation (MICE) is a data imputation technique based on chained equations [6]. It considers the missing data behavior with MCAR and MAR. Furthermore, some imputation methods consider the data distribution while generating missing samples [7], [8] and the references therein.

Recently, generative adversarial networks (GANs), a new approach based on deep neural networks (DNNs) and adversarial learning [9] has become popular among practitioners as a synthetic data generator. It consists of two models, the generator and the discriminator. In competitive training of generator and discriminator, the generator learns the distribution of samples and generates synthetic samples similar to the real ones, and the discriminator authenticates its realness. GANs proved their applicability in various areas such as image synthesis [10], face completion [11], audio enhancement and synthesis [12], anomaly detection [13] and so on. Recent surveys about developments and applications of GANs can be found in [14], [15]. Recently, GANs have also been exploited to solve the problem of missing values via data imputations, see, for example [3], [16]–[18]. The performance of GAN depends on its hyper-parameters, mainly loss function, optimizer, learning rate, etc. A review on customized loss functions based on GAN models used for different purposes is given in [19].

In this paper, we consider multivariate real-time data from a steel plant [20] situated in the south part of Italy. The time-series data has information about signals such as vibration, temperature, and pressure. Due to extreme environmental conditions such as extreme stress and vibrations, the sensors malfunction; thereby, we obtain the missing data during the operating condition of the process or production phase. However, the data includes a fault situation in which the values of the features have risen to a great extent. Thus, the data has an imbalanced distribution over the complete duration. Therefore, to obtain comprehensive data, it is necessary to get the samples at random within the dataset and learn a model over complete distribution. Motivated by this, we present a GAN-based data imputation method to substitute the missing values with the synthetic data. We compare the outcome of the presented technique with the one obtained using various data imputation methods such as mean, median, most frequent, KNN-based, and MICE. From the various simulation tests, we conclude that the GAN-based imputation method outperforms the rest of the methods under consideration.

II. Methodology

This section describes the structure and mathematical idea of the GAN. Further, we discuss the different data imputation methods used for comparative study.

A multivariate time-series data X with total n samples $(t_0, \ldots, t_{n-1})$, and d features is given as $X = (x_{t_0}, \ldots, x_{t_i}, \ldots, x_{t_{n-1}})^\top \in \mathbb{R}^{n \times d}$, where $\mathbb{R}$ is a set of real numbers and x_{t_i} is a d-dimensional observation/sample at t_i. An incomplete data $X_{incomplete}$ consists of missing values, denoted by nan. In the interest of the missing values, we use the mask matrix M, introduced in [16], [17] to identify the missing values' location. The aim of the mask matrix is to replace synthetic data over the missing values. For a missing value occurrence at sample t_i, and for a feature/column j, value in the mask matrix $M_{t_i}^j = 1$ otherwise the value is 0.

A. Generative adversarial networks (GANs)

Over the last decade, research on GAN has achieved many encouraging results about synthetic data generation in images, acoustics, and time-series data.

1) GAN structure: A basic GAN model consists of two neural networks (NNs) models, namely generator ($\mathcal{G}$) and discriminator ($\mathcal{D}$). As per the application's requirement, modifications could be done in the structure and hyper-parameters of the GAN. A Hyper-parameter is a parameter on which the performance of the GAN model is dependent. Basically, $\mathcal{G}$ and $\mathcal{D}$ are like forger and identifier between fake and original data, respectively. $\mathcal{G}$ generates fake samples using random noise samples z as input and provides it to $\mathcal{D}$. $\mathcal{D}$ model also gets original samples x as input for training/learning the data distribution. $\mathcal{D}$'s task is to identify these samples correctly as real or fake. In between $\mathcal{G}$ and $\mathcal{D}$, whosoever fails in their tasks; is trained back to improve itself. This training is known as adversarial training. Finally, $\mathcal{G}$ trains itself such that it generates indistinguishable samples close to original/real samples. In other words, the training tries to achieve Nash equilibrium [21]. As shown in Fig. 1, $\mathcal{D}$ model is trained with samples to learn the original data distribution and flag these samples as real while the generated samples from $\mathcal{G}$ as fake. In the next case, $\mathcal{G}$ is trained while $\mathcal{D}$ remains idle. When $\mathcal{D}$ is trained with generated samples, on the correct prediction of the samples from $\mathcal{D}$, it is required to train $\mathcal{G}$ model to generate better samples that can fool $\mathcal{D}$. The $\mathcal{D}$ in Fig. 1 works as a binary classifier that labels any sample into either real or fake.

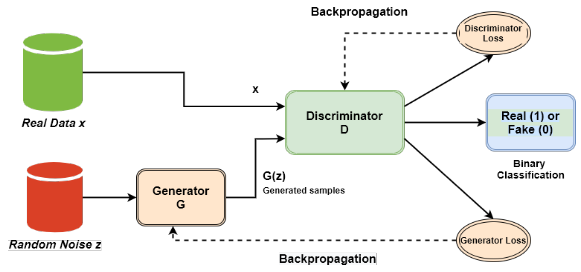


Fig. 1: GAN model

2) Mathematical model: The GAN framework is based on a mathematical formulation wherein the aim is to achieve min-max optimization. Consider G, and D are functions of model $\mathcal{G}$, and $\mathcal{D}$, respectively. $\mathcal{G}$, and $\mathcal{D}$ have input random samples z, and original data x, respectively. While the generated samples $G(z)$ have given as input to $\mathcal{D}$ through $\mathcal{G}$. Each model i.e., $\mathcal{G}$ and $\mathcal{D}$ will have parameters for a stabilized performance. The aim of $\mathcal{D}$ is to label original data samples x as TRUE (or one) while the generated samples $G(z)$ as FALSE (or zero). This can be done by using Binary cross entropy loss function as

$$L(\hat{y}, y) = [y * \log(\hat{y}) + (1 - y) * \log(1 - \hat{y})],$$

where $\hat{y}$ and y are probability of predicting label y and true label respectively. Thus, loss function for the $\mathcal{D}$ can be considered as:

$$L_D = \log(D(x), 1) + \log(D(G(z)), 0).$$

While the aim of $\mathcal{G}$ is to generate samples similar to $D(x)$ i.e., $D(G(z))$ have to be labeled as one. It is given in mathematical form as:

$$L_G = \log(D(G(z)), 1).$$

Hence, based on the above description, $\mathcal{D}$ needs to label original samples x as 1, and generated samples $G(z)$ have to be labeled as 0 i.e., to maximize the $\mathcal{D}$ loss function. Whereas in case of $\mathcal{G}$, $D(G(z))$ has to be labeled as 1 i.e., the generated samples have to be labeled as original samples. Hence, the error function has to be minimized. In other words, $\mathcal{D}$ seeks to maximize the average of log probability of the original samples and log of the inverted probabilities of fake samples. Whereas $\mathcal{G}$ seeks to minimize the average of the log of inverse probability following two-player min-max game with value loss function $V(G; D)$ as follows:

$$\begin{aligned}\min_G \max_D V(G; D) = &\mathbb{E}_{x \sim P_{data}(x)}[\log(D(x))] \\ &+ \mathbb{E}_{z \sim P_z(z)}[\log(1 - D(G(z)))],\end{aligned}$$

where $\mathbb{E}[\cdot]$ is an expected value over data instances corresponding to the real and random data. $P_{data}(x)$ and $P_z(z)$ are the probability distribution of original data and random noise samples, respectively.

GAN learns the distribution of the data provided to the discriminator, and the generator generates similar synthetic data with the help of adversarial training.

B. Hyper-parameters of GAN

In deep learning, hyper-parameters are often used to control the learning process of a model. The model performance is uncertain, and a set of hyper-parameters is selected in order to optimize the desired output. Generally, the GAN has various hyper-parameters, such as learning rate, batch size, number of iterations, optimizer, number of layers, activation function, and loss function, which affect the model performance. In this paper, we have mainly considered loss function and optimizer to tune the GAN model, whereas the remaining hyper-parameters are kept fixed. These hyper-parameters are selected as learning rate = 0.001, number of iterations = 6000, number of layers = 3, activation function = Leaky_ReLU. The batch size is dependent on the number of missing values (here, we have selected batch size = 192). In the following, we briefly discuss the selection of loss function and optimizer.

1) Loss function: The loss function is the main hyper-parameter of the GAN model that may influence the performance majorly. In our case, we have considered seven different loss functions, namely, original GAN, least square, Kullback-Leibler divergence (KLD), Jenson-Shannon divergence (JSD), Pearson X^2, Square Hellinger, and Reverse KLD. For the detailed discussion on the various loss functions discussed above we refer to [22].

2) Optimizer: The optimization algorithm is an important hyper-parameter of GAN on which the speed and reliability of the model is dependent. We have considered RMSProp optimizer and Adam optimizer in our tuning case in which Adam (Adaptive moment estimation), a first-order gradient-based optimization of stochastic objective function based on adaptive lower-order moments, works faster and more reliable when the value function is tuning [23].

C. Comparative methods for data imputation

In this subsection, comparative data imputation methods that are used in this paper are described. These methods contain simple imputation methods, MICE and machine learning based KNN imputation method.

1) Simple imputation methods: Traditionally statistical values such as mean, median, etc., have been used in the data imputation. These are the easiest and direct imputation methods that do not need to learn the data distribution. However, in the case of imbalanced data, these imputation methods provide inaccurate imputation because the distribution is skewed to either side. Mean and/or median may alter correlations between the variables. These methods generally are used when data is missing at random and less than 5% of all the data.

2) MICE: The method is based on fully conditional specifications, where each incomplete sample is imputed by a separate model. The imputation is done in multiple steps to get the best imputed ones which indicates some measure of convergence.

3) Machine learning imputation methods: Instead of statistical imputation methods, machine learning based methods such as k-nearest neighbor (KNN), random forest, etc., are used as data imputation methods. Among them, KNN is the most popular method to predict the missing values over the dataset. The KNN imputation algorithm works based on the selection of distance measures (e.g., Euclidean, robust, etc.) and a number of nearest neighbors for the prediction of missing value. The missing value is imputed with an average value depicted from the group classified using KNN.

D. Metrics for evaluation

There are several metrics used for evaluation of performance of the data imputation methods. Root mean square error (RMSE) value is a default metric to evaluate the similarity between the original incomplete data and imputed complete data. While the Frechet inception distance evaluates performance of GAN (FID score [21]) between real (training) samples and generated samples. Definition of the metrics are given as follows:

1) Frechet inception distance (FID) score: FID score is one of the most important metrics to estimate GAN's performance between real and synthetic data. FID score is a distance value calculated between real and synthetic feature vectors. A lower value of the score indicates a more similar data generation. The ideal value of the score is zero.

$$d^2 = \|\mu_1 - \mu_2\|^2 + \mathrm{Tr}(C_1 + C_2 - 2 \times \sqrt{C_1 \times C_2}), \quad (1)$$

where μ_1 and μ_2 (C_1 and C_2) are mean values (covariance matrices) of real samples and generated samples, respectively and Tr denotes trace of the matrix.

2) Root mean square error (RMSE): RMSE is the most common metric to estimate the difference between the imputed dataset with test methods. A lower value of RMSE indicates the more stable imputation method, whereas a larger value indicates the lesser stability.

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}}, \quad (2)$$

where
x_i = real sample in time-series;
$\hat{x}_i$ = imputed/predicted sample in time-series;
N = Total number of data samples.

E. Gearbox dataset

The dataset considered in the paper is a real-time multivariate dataset obtained from a steel manufacturing plant situated in the south part of Italy. The plant produces steel bars used in the construction. The steel making process is a multistage process including steel making, continuous casting, and rolling stages. Billets (samples) are sequentially rolled in multiple rolling mills, i.e., a chain of rotating machines that progressively stretches the bars and reduces their diameters to desired forms. The mill drive train supplies mechanical energy to the top and bottom driven rollers of the rolling mill stand; it is configured with an electrical motor, a gear drive, and a pinion stand, all connected by couplings; the combination of these elements is also referred to as a cage. In the dataset, we consider the features that are closed to the cage (refer to Fig. 2). It contains six sensors mounted around the gearbox to record the signals showing the gearbox's performance. These features include different types of signals such as vibrations (Acc1, Acc2, Acc3), temperature (Temp1, Temp2), pressure (Press), etc., recorded at different sampling frequencies. The process undergoes maintenance regularly, which prevents any occurrence of an unexpected event in the process. Hence, the dataset holds most normal operation samples while very few faulty samples have diverse behavior. Thus the dataset is an imbalanced class dataset. Due to sensor failure and dimension mismatching of the dataset, missing values are present. The dataset consists of 3590 samples, including 5% of missing samples. The missing values mainly occur into three features out of six, i.e., due to temperature and pressure sensors failure. The GAN is used to learn the distribution of each feature distinctly and generate similar synthetic samples for the imputation.

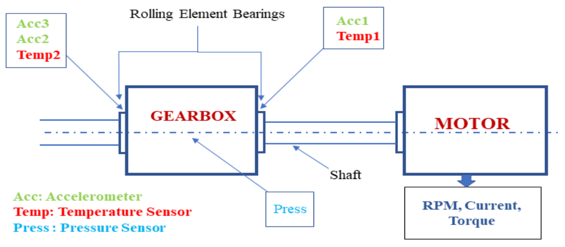


Fig. 2: Cage in production process [20]

III. Proposed Method

In the paper, we proposed a methodology to impute the incomplete data $X_{incomplete}$ using GAN. Since GAN performance is unpredictable concerning the dataset and hyper-parameters, we split the proposed approach into two stages. In the first stage, GAN model training is performed to get the best result over the time-series data in terms of a set of hyper-parameters $H_i = (V_j(G;D), O_k)$, where $V_j(G;D)$ and O_k are different loss functions and optimizers. Here we have used $j = 7$ loss functions, and $k = 2$ optimizers. Secondly, using the best H_i, we generate the synthetic samples for each feature. For every 1000 iteration, the synthetic samples are stored in an array, and then the best synthetic samples are chosen by using a minimum FID score. Finally, we impute the synthetic samples into the incomplete dataset $X_{incomplete}$. This two-stage procedure is summarized in the following pseudo-algorithm (Algorithm 1). The input at second stage to generate synthetic samples is given as $X_{Train} = M \cdot X_{incomplete}$, where " $\cdot$ " is an element-wise matrix product.

The performance of Algorithm 1 is compared with various state-of-the-art data imputation techniques, such as simple imputation methods and KNN based imputation methods. In

Algorithm 1 Data Imputation using GAN

```
Stage one: GAN model training
    Input: GAN training set $X_{incomplete}$
    Compute mask matrix $M$
    for i = 1: j*k do
        Select hyper-parameters $H_i = (V_j(G;D), O_k)$
        for iteration/1000==0 do
            Compute FID score for $H_i$
        end for
    end for
    Select best performed $H_i$
    Output: Trained hyper-parameters $H_i$ for GAN
Stage two: Synthetic data generation using GAN
    Input: $H_i$ from Stage one and $X_{Train}$
    Select number of samples equal to batch size from $X_{Train}$
    for iteration/1000 == 0 do
        Compute FID score for each feature
        Save generated samples to array for each feature
    end for
    Select minimum FID score (FID$_{min}$) for each feature
    Select corresponding generated samples array to (FID$_{min}$)
    Impute generated samples into the missing dataset to get complete dataset
        $X_{complete} = M \cdot X_{incomplete} + (1 - M) \cdot X_{imputed}$
    Output: Complete dataset $X_{complete}$ using GAN
```

the next section, we discuss the effectiveness of Algorithm 1 in detail.

IV. Results And Discussions

This section confers outcomes of the proposed method concerning the other comparative methods. We also discuss the problems that occurred during the training of hyper-parameters of GAN, which are uncertain over the dataset.

A. *Impact of hyper-parameters on GAN training*

The training of the GAN model is still a challenging task to remain. This subsection will discuss the problems for hyper-parameters, such as loss functions, optimizers, and GAN performance. In our case, we have trained the GAN model over the various set of hyper-parameters to get the best-performed set of hyper-parameters for the given time-series data. Some common issues encountered during the training of the GAN model are discussed in the following.

1) Mode collapse: The generator aims to produce a wide variety of output similar to real samples. However, it happens to limit the variety of samples to the same kind of samples (e.g., *nan*), and the generator is stuck in further iterations. This kind of issue is known to be a mode collapse. In our case, we have obtained the mode collapse problem for two loss functions, JSD and KLD. In contrast, the best performance of GAN has been obtained for the least square loss function. Hence, in the final model of GAN, the least square loss

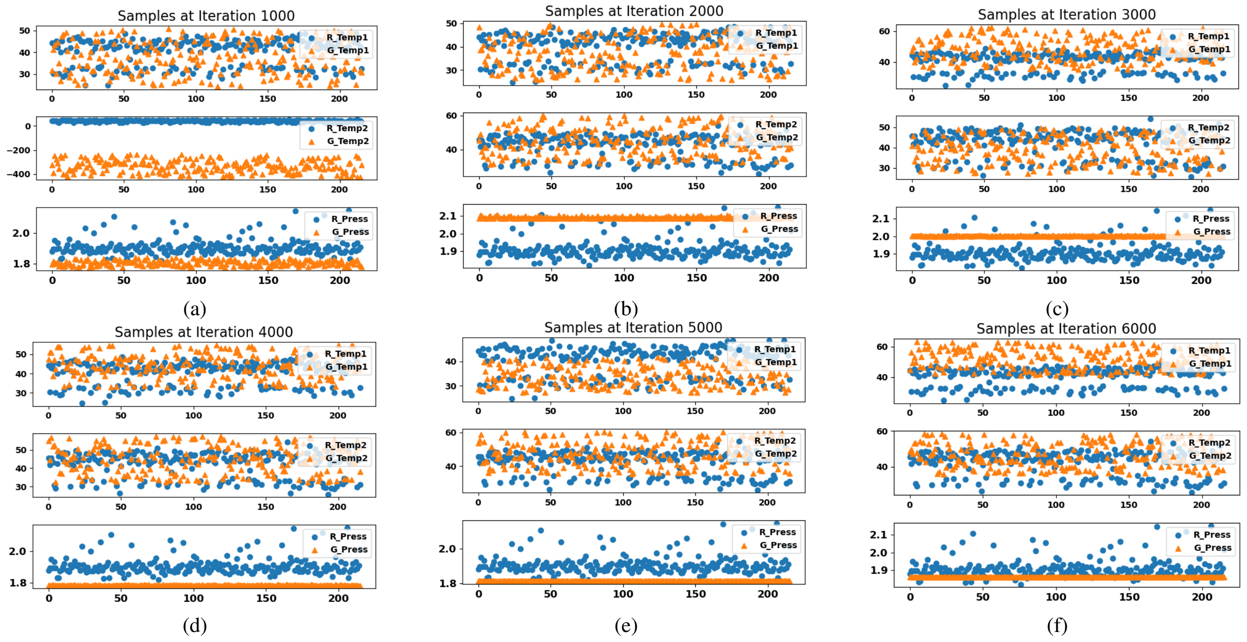


Fig. 3: GAN training for every 1000 iterations

function is used. However, Fig 3a notices that the generated samples for Temp2 are far from the real samples. A similar situation is obtained for JSD and KLD in which the generated samples provide nan values.

2) Convergence: The second problem in GAN training is non-convergence. We cannot guarantee the convergence of the generated samples after fixed iterations. The generation of the samples always do not show convergence, whereas the best generated samples can be taken with the help of the FID score. Fig. 3 demonstrates that the best generated samples are not at the end of the training, i.e., after 6000 iterations, while for Temp1, the best generation is after 1000 iteration. For Temp2 and Press, we obtain the best generated samples after 3000 and 6000 iterations, respectively. Finally, we select the minimum values FID score FID$_{min}$ for the generated samples. The evolution of FID score during the training is also depicted in Fig. 4.

B. GAN model training

This subsection describes the performance of GAN for the selected set of hyper-parameters H_i in stage one of Algorithm 1. The best set of H_i is obtained with the least square loss function, learning rate 0.001, and Adam optimizer for the given data. Fig 4 illustrates the FID scores for every 1000 iteration for H_i. The selection of generated samples is made by using the minimum FID score among real and generated samples. Thus, FID$_{min}$ will provide the best distributed generated samples against the real samples.

C. Data imputation results

Data imputation is a substitution process of the missing values with their imitated values. The imitated values may be statistical values or generated values from its distribution. We generate the synthetic samples using GAN (Algorithm 1) and

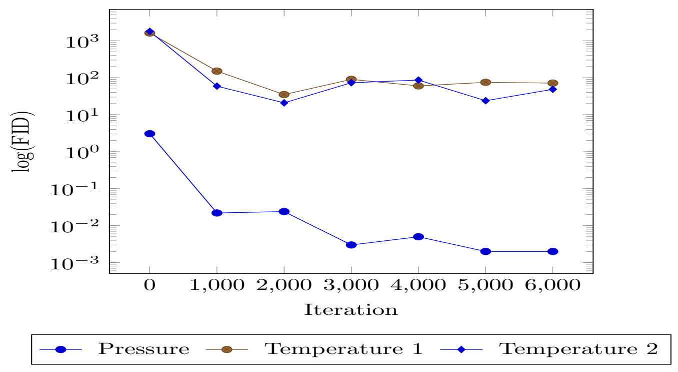
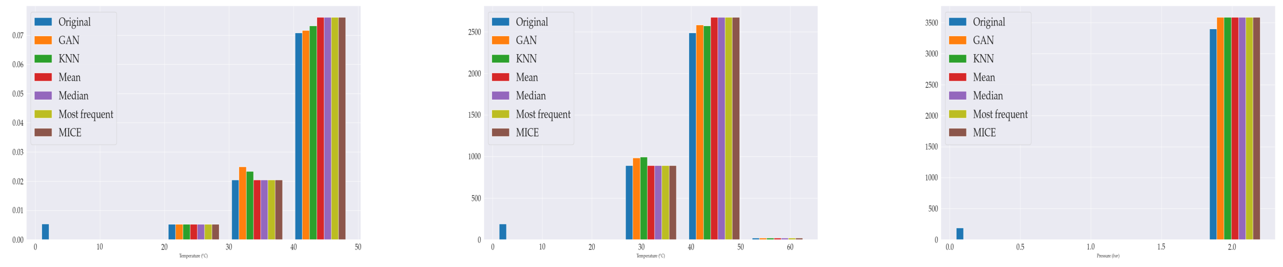


Fig. 4: FID scores for every 1000 iterations

replace them at the place of the missing values. The imputation provides the complete dataset by which we can enhance the analytical results over the data. Comparison results of the outcome of Algorithm 1 with different imputation methods are reported in Fig. 5 and Table I. Fig. 5 depicts the histogram over the original incomplete data (missing valued data) and imputed data obtained by different methods. We can recognize that the original data has 190 missing samples for three features each. Instead of having the variability among the features, the GAN-based imputed data shows a similar distribution with the original data and imputes the generated samples equally distributed in place of the missing values. Besides other methods, the KNN based imputation method imputes samples only from few ranges of the original distribution. While simple imputation methods and MICE impute the single computed value, i.e., mean, median, most frequent value over the missing values. Thus, the GAN-based imputation performs over the distribution of the original data. In the numerical computation,



(a) (b) (c)

Fig. 5: Histogram plots comparison with all methods

the RMSE value ensures that the difference between real samples and imputed samples. Table I demonstrates that GAN outperforms the imputation of the missing values between all the considered imputation methods.

TABLE I: RMSE value for different imputation methods

Method	Temp1	Temp2	Press	Combined
KNN	8.831	9.074	0.439	7.315
Mean	9.312	9.661	0.438	7.751
Median	9.804	10.202	0.436	8.173
Most frequent	9.731	10.766	0.434	8.382
MICE	9.305	9.637	0.438	7.735
GAN	**8.043**	**8.959**	**0.434**	**6.955**

V. Conclusion and Future Scope

In this paper, we have presented an efficient data imputation technique based on GANs. Several experimental validations have been performed to show the effectiveness of the proposed method. The GAN based data imputation outperforms the state-of-the-art data imputation methods.

To extend the applications of GAN to Industrial data, generating the fault samples identifying the distribution of the fault event could be one future direction. Hence, the problem of imbalanced data can be resolved.

References

[1] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," *Neural Computing and Applications*, vol. 19, no. 2, pp. 263–282, 2010.

[2] J. M. Jerez, I. Molina, P. J. García-Laencina, E. Alba, N. Ribelles, M. Martín, and L. Franco, "Missing data imputation using statistical and machine learning methods in a real breast cancer problem," *Artificial intelligence in medicine*, vol. 50, no. 2, pp. 105–115, 2010.

[3] A. Kazemi and H. Meidani, "Igani: Iterative generative adversarial networks for imputation applied to prediction of traffic data," *arXiv preprint arXiv:2008.04847*, 2020.

[4] M. Saad, M. Chaudhary, F. Karray, and V. Gaudet, "Machine learning based approaches for imputation in time series data and their impact on forecasting," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 2621–2627.

[5] P. Keerin, W. Kurutach, and T. Boongoen, "Cluster-based knn missing value imputation for dna microarray data," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012, pp. 445–450.

[6] S. Van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, vol. 45, no. 1, pp. 1–67, 2011.

[7] M. Przewięźlikowski, M. Śmieja, and Ł. Struski, "Estimating conditional density of missing values using deep gaussian mixture model," in *International Conference on Neural Information Processing*. Springer, 2020, pp. 220–231.

[8] M. Śmieja, Ł. Struski, J. Tabor, B. Zieliński, and P. Spurek, "Processing of missing data by neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 2719–2729.

[9] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[10] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[11] Y. Li, S. Liu, J. Yang, and M.-H. Yang, "Generative face completion," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3911–3919.

[12] N. Torres-Reyes and S. Latifi, "Audio enhancement and synthesis using generative adversarial networks: A survey," *International Journal of Computer Applications*, vol. 182, no. 35, pp. 27–31, 2019.

[13] W. Jiang, Y. Hong, B. Zhou, X. He, and C. Cheng, "A gan-based anomaly detection approach for imbalanced industrial time series," *IEEE Access*, vol. 7, pp. 143 608–143 619, 2019.

[14] M. Zamorski, A. Zdobylak, M. Zięba, and J. Świątek, "Generative adversarial networks: recent developments," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2019, pp. 248–258.

[15] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (gans): A survey," *IEEE Access*, vol. 7, pp. 36 322–36 333, 2019.

[16] Y. Luo, X. Cai, Y. Zhang, J. Xu *et al.*, "Multivariate time series imputation with generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 1596–1607.

[17] J. Yoon, J. Jordon, and M. Van Der Schaar, "Gain: Missing data imputation using generative adversarial nets," *arXiv preprint arXiv:1806.02920*, 2018.

[18] W. Zhang, Y. Luo, Y. Zhang, and D. Srinivasan, "Solargan: Multivariate solar data imputation using generative adversarial network," *IEEE Transactions on Sustainable Energy*, vol. 12, no. 1, pp. 743–746, 2020.

[19] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, "A review on generative adversarial networks: Algorithms, theory, and applications," *arXiv preprint arXiv:2001.06937*, 2020.

[20] K. Sarda, A. Acernese, V. Nolè, L. Manfredi, L. Greco, L. Glielmo, and C. Del Vecchio, "A multi-step anomaly detection strategy based on robust distances for the steel industry," *IEEE Access*, vol. 9, pp. 53 827–53 837, 2021.

[21] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *arXiv preprint arXiv:1706.08500*, 2017.

[22] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2794–2802.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.