GradDiff-N-Tab: Gradient Noise Tabular Data Diffusion Model Imputation

Ari Wibisono
Faculty of Computer Science, Universitas Indonesia
Indonesia
ari.w@cs.ui.ac.id

Denny
Faculty of Computer Science, Universitas Indonesia
Indonesia
denny@cs.ui.ac.id

Petrus Mursanto
Faculty of Computer Science, Universitas Indonesia
Indonesia
santo@cs.ui.ac.id

Simon See
NVIDIA, Senior Member IEEE
Singapore
ssee@nvidia.com

***Abstract*—Imputation is one of the methods to improve the quality of a dataset. The imputation problem can be solved using statistical techniques, machine learning algorithms, and generative models. This research proposes improving the standard imputation algorithm based on the Diffusion Model. We propose to use Perlin noise or gradient noise generation to generate noise at each step of the diffusion mode and propose a scheduler to improve the performance of the diffusion model-based imputation algorithm. Perlin noise generation and cosine scheduler have a positive influence on improving the performance of non-normal data imputation. Four real-world datasets are used to evaluate our proposed methods. Based on the evaluation tests of the RMSE value, our proposed method produces a 10% lower RMSE value than the baseline imputation algorithms based on diffusion models, GAN, and VAE.**

***Index Terms*— Tabular data, Imputation, Cosine Scheduler, Stable Diffusion Models, Gradient Noise**

I. Introduction

Data processing should be improved [1]. Missing data can occur due to incomplete form filling and questionnaire questions needing to be filled in [2]. Other causes of missing data include sensors that fail to record data [3]. The missing data will affect the statistics and data analysis [4]. To accommodate the problem of missing data, the primary reference that can be used is to categorize data based on the MCAR (Missing Completely at Random), MNAR (Missing not at random), and MAR (missing at random) categories [5].

The simplest way to deal with missing data is by not using the data. Disposing of the data will cause a new bias in the data. So, the researchers tried to impute artificial values into the data. The simplest imputation method is by using statistics [7]. Some of the machine learning algorithms used are K-Nearest Neighbor (KNN) [6], MICE [9], and Missforest [8].

Furthermore, researchers also try to use neural network techniques in imputing data. One of the algorithms is Multi-Layer Perceptron [18]. Deep-learning techniques perform imputations based on Auto Encoder [30]. The development of generative modeling also supports solutions to imputation problems. The generative algorithm used for imputation is GAIN [43]. Testing several imputation techniques against various datasets has been carried out by Miao et al. [10]. The diffusion model technique is also used to overcome the problem of missing data. The diffusion model technique used is CSDI (Conditional Score Diffusion Model Imputation) [11].

Score-based generative models have shown competitive outcomes compared to state-of-the-art methods across various implementation tasks. These tasks include generative images [20][21], audio processing [22], and shape generation [23] Song. et al. contrast the performance of score-based diffusion models with vanilla Continuous Normalizing Flows and state-of-the-art methods [20]. The framework of score-based diffusion models proposes the gradual diffusion of distributions based on noise distribution. Stochastic differential equations are employed to learn from the distribution of sampled data. Score-based diffusion models exhibit advantages during training compared to the vanilla Continuous Normalizing Flows (CNF) method [20]. The reason is that the maximum likelihood objective for CNF training requires an expensive Ordinary Differential Equation (ODE) solver for each optimization step. In contrast, score-based diffusion models utilize a weighted score-matching loss combination for score matching, which is less computationally expensive. Evaluation results of score-based models show likelihood outcomes that compete favorably with recent autoregressive models, with minor degradation observed in Fréchet Inception Distances [24].

Algorithms developed to impute data are numerous, but most real-world data have an abnormal distribution. This anomalous data causes the results of some algorithms to fail to work well. The process of machine learning, deep learning, and generative models makes the conversion of abnormal data into a normal distribution so that machine learning calculations can get the best results. In this research, the author will propose an imputation method in the diffusion models algorithm for non-normal or extreme data. The improvement process involves noise generation, loss function, and noise scheduler changes.

II. Background & Problem Formulation

A. Diffusion models imputation

Assume a sample x_0 that has missing values. Based on Song. et al. [11], we formulate to produce imputation target $x_t^{ta} \epsilon x_0^{co}$ by using known data $x_0^{co} \epsilon x_{\square}^{ta}$. Then, we can assume that the goal of the probabilistic model is to estimate the distribution of $q\ (x_0^{ta} \mid x_0^{co}\ )$ by modeling $p_\theta(x_0^{ta} \mid x_0^{co}\ )$. So, we can notate all known values as x_0^{co} and missing values as x_0^{ta}, so the backward imputation process will be as in formula 2.

$$q_\theta(x_{0:T}^{ta} \mid x_0^{co}) := p(x_{0:T}^{ta}) \prod_{t=1}^{T} p_\theta(\ x_{t-1}^{ta} | x_t^{ta}, x_0^{co}),$$
$$x_T^{ta} \sim \mathcal{N}\ (0, \mathrm{I}). \tag{1}$$

$$p_\theta(x_{t-1}^{ta} \mid x_t^{ta}, x_0^{co}) :$$
$$= \mathcal{N}\ (x_{t-1}^{ta};\ \mu_\theta\ (x_t^{ta}, t \mid x_0^{co}), \sigma_\theta\ (x_t^{ta}, t \mid x_0^{co})\mathrm{I}). \tag{2}$$

B. Gaussian Noise

Gaussian noise or noise based on normal distribution is made based on a probability density function (pdf) with normal distribution parameters [29].

$$\varphi\ (z) = \frac{1}{\sigma\ \sqrt{2\pi}}\ e^{\frac{-(z-\mu)^2}{2\sigma^2}} \tag{3}$$

In formula 3, the notation σ is the standard deviation, z represents the gray level, μ is the average value. Gaussian noise can be minimized in image processing with several filtering and smoothing techniques [31].

C. Perlin Noise / Gradient Noise

To create noise, we generate pseudo-random numbers [12][13]. This generator is a form of smooth function. We can add noise functions with different frequencies and amplitudes in Perlin noise [49]. We try to create functions with increasing frequency variations: $noise(t)$, $noise(2t)$, $noise(4t)$, ... $noise(2it)$. In this case, we do not change the function; we only change the frequency parameter.

High frequencies usually have low amplitudes. In Perlin noise, we can configure the amplitude of each frequency. The value that helps to configure this is persistence. $p^{\square}$. The persistence value is between 0 and 1. The higher the persistence value, the more visible the high frequencies will be. In formula 4, p^i is the amplitude at the i^{th} stage. The final noise is the sum of the persistence defined by $perlin(t)$.

The user can change or control the octave value by changing the value of 2^i and the persistence value p^i.

$$perlin(t) = \sum_{i=0}^{k} p^i\ noise(2^i.t) \tag{4}$$

III. Proposed Method

We proposed Perlin noise imputation & scheduling for diffusion models. A good noise is a noise that has a homogeneous pattern visualization random characteristics, and its value changes slowly. Generally, noise features are similar in size between one noise and another. Perlin noise can have better noise characteristics than normal noise. In normal noise, we try to interpolate a random value along the line with a random integer. These values are randomly degenerate; some deals have the same pattern, but the noise changes quickly in some parts. The place where the noise changes very fast is said to be high frequency. While the place where the noise changes slowly is said to be low frequency. We can conclude that normal noise is built on high and low frequencies.

Perlin noise also does the same thing in producing random integer values. However, Perlin noise uses a gradient tangent to the 1D noise function to create noise values. No matter the direction of the gradient produced by Perlin noise, the value will increase and decrease along with the previous value. If two lattice points have opposite gradient directions, the noise function will produce an S-shape. By using this construction, the frequencies of Perlin noise have a similar size. Thus, the frequency spectrum of Perlin noise is more generalized than normal noise with high and low frequencies. Perlin noise uses gradient measurements to generate noise, while normal noise only uses random values.

A. Perlin Noise Imputation

In Gaussian noise, we have a forward process function.

$$x_t = \sqrt{\tilde{\alpha}_t}\ x_0 + \sqrt{1 - \tilde{\alpha}_t}\ \epsilon \tag{5}$$

Where ϵ is Gaussian noise with an objective function $\min L_{simple}$

$$\min L_{simple} = \mathbb{E}_{t,x_0,\epsilon}\left\|\epsilon - \epsilon_\theta(\sqrt{\tilde{\alpha}_t}\ x_0 + \sqrt{1 - \tilde{\alpha}_t}\ \epsilon, t)\right\|^2 \tag{6}$$

We try to introduce Perlin noise into the forward process equation of diffusion models,

$$x_t^p = \sqrt{\tilde{\alpha}_t}\ x_0 + \epsilon_{perlin}(\sqrt{1 - \tilde{\alpha}_t}) \tag{7}$$

Following the standard diffusion model objective function, we can denote the forward process of Perlin noise in this form.

$$x_t^p = \sqrt{\tilde{\alpha}_t}\ x_0 + \epsilon_{perlin}(\sqrt{1 - \tilde{\alpha}_t}) \tag{8}$$

This approach will prevent noisy data from degenerating by multiplying. $\tilde{\alpha}_t$ with ϵ, but the noise fusion process is produced when the Perlin function generates the noise directly $\epsilon_{perlin}(\sqrt{1 - \tilde{\alpha}_t})$ With this approach, the noise fraction at each t is no longer rigidly degenerate but has a softer specification. With the approach of input noise with a more stable frequency, the input of noisy data into the network also becomes stable. So, the distribution of input data also has implications for the stability of the data. Noisy data using Perlin noise is closer to normal distribution characteristics than Gaussian normal. As we know, to produce perfect features as input into the network, researchers try to make the input data into the network have a normal distribution. One of the ways used is by using a logarithmic function so that the values entering the deep

TABLE I. DIFFUSION MODELS EXPERIMENT PARAMETERS

Parameters	Value	Parameters	Value
epochs	200	Diffusion Embedding	128
batch_size	36	beta_start	0.0001
Learning rate:	0.001	beta_end	0.5
Diffusion layers:	4	num_steps	100
Diffusion channels:	64		

learning network can be optimally trained for feature representation.

B. Scheduling

The diffusion model process [14] is a process where the input given is a noise sample. The noise sample is slowly input into the diffusion model based on the fraction schedule. The noise fraction will be input into the diffusion model process in the forward and backward processes.

The forward process is the training process, where the data is slowly given noise and inputted with an alpha value that increases with each step. At the same time, the backward process is a generative process, where the input into the diffusion model process is the opposite of the forward or training process. So, the diffusion model that has been trained will be inputted with the full noise and iterated in as many steps as previously initialized. If in the previous example using step 10 and fraction between 0 and 1, then the backward process will do scheduling from alpha 1, 0.9, 0.8 . . , 0.1. So that after this backward process, the data generated from the model with full noise input will be obtained, which will slowly decrease the full noise fraction scheduling.

Chent. et al. tried to evaluate various noise scheduling variations to be affiliated with the diffusion models technique [15]. Lin et al. have also tried to contribute to their findings by scheduling weaknesses at the beginning and end of scheduling in the diffusion model, where the improvements made can increase the image contrast in the generative model results [16].

The scheduler in the diffusion model is represented in the notation $\tilde{\alpha}_t$ in Equation (8). The value of $\tilde{\alpha}_t$ will be produced as a parameter for the formation of noisy data with Perlin noise x_t^p. In this research, we propose to use the cosine function to maximize and improve the
performance of Perlin noise. Using the cosine function can maximize the input forwarding information during training.

$$q\ (x_{1:T}|x_0) \tag{9}$$

Changing data from the original data x_0 to data with maximum noise $x_{1:T}$ can be slowed down by the cosine function. With a slower step process, the deep learning network will obtain information about the noisy data at step t before it becomes maximum noise. So, the deep learning network obtains more information about the data before it becomes noise compared to the standard approach.

In this research, we propose to use a cosine scheduler to maximize our proposed Perlin noise generation. This approach is chosen because the adapted step cosine function allows the deep learning model to obtain more data information before it slowly becomes noise. There are 100 steps to reach maximum alpha or full noise (1). Information from the data can still be maximized very slowly up to step 85 by the cosine scheduler. After step 85, cosine starts adding alpha with a value of 0.2 and continues to increase until full noise one at step 100. So, at steps 0 to 85, the network model can learn data with minimal noise, so the deep learning network model obtains information about the data better than other schedulers. If we compare quadratic and linear at step 60, the alpha values are 0.2 and 0.6. With these values, the amount of noise incorporated into the data becomes quite large. While in cosine, the alpha value at step 60 is only 0.03, so the deep learning network model still has enough information compared to the influence of noise.

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, f_t = \cos^2\left(\frac{t\ +\ s}{1+s}.\ \frac{\pi}{2}\right) \tag{10}$$

t is a step, s is a floating number parameter to form the cosine function.

In looking at the loss diffusion model perspective, we need to see how the model learns from complete noise to degenerate data. So, the perspective of thinking is that at the beginning of the epoch, the model does not know about the data (high loss), but at the end of the epoch iteration, it manages to understand the data (low loss).

IV. RESULTS AND DISCUSSIONS

There are four datasets used in this experiment. The datasets were obtained from the OpenMl repository [27] and the UCI machine learning dataset [26]. Each dataset is normalized to a value between 0 and 1. Each scenario is tested ten times to get the consistency of the value obtained. The properties of each dataset can be seen in Table 1.

To prove the consistency of our proposed method against non-normal data, we tested our proposed method algorithm with two types of dataset categories. The first dataset category distribution is a dataset that has a normal distribution (frog, satellite). The second dataset category distribution is a non-normal dataset (bike and credit card dataset). To evaluate our proposed method, we compare our proposed algorithm with GAIN (GAN)-based imputation [17], VAE [28], and CSDI (Imputation based on Diffusion models [11]. Based on Table. I. Datasets are normal and non-normal based on the Shapiro-Walk test with alpha 0.05 [31].

We use three scenarios to simulate the missing rate of data [30]: Missing Completely at Random (MCAR): Data absences are identified as MCAR (Missing Completely At Random) when the occurrence of missing data is entirely unrelated and independent of the available dataset. Missing Not At Random (MNAR): Incomplete data is considered MNAR (Missing Not At Random) when, despite having access to all available observed information, the probability of missing data is contingent upon the unobserved values. Missing at Random (MAR): In instances of missing data being categorized as MAR (Missing At Random), this classification indicates that the likelihood of data absence is unrelated to the missing values, given the already observed data.

TABLE II. (RMSE) NON-NORMAL DATASET BIKE | CPU

dts	sce	mod	Missing Ratio				dts	sce	mod	Missing Ratio			
			20%	*40%*	*60%*	*80%*				*20%*	*40%*	*60%*	*80%*
bike	*mcar*	enh	**0.0792**	**0.1000**	**0.1246**	**0.1799**	**credit-card**	*mcar*	enh	**0.0854**	**0.0926**	**0.1080**	**0.1293**
		std	0.0979	0.1208	0.1393	0.1870			std	0.0968	0.1027	0.1203	0.1376
		GAIN	0.2054	0.2270	0.2650	0.2934			GAIN	0.1789	0.1887	0.2062	0.2376
		VAE	0.2179	0.2566	0.2876	0.3102			VAE	0.1537	0.1697	0.1879	0.2045
	mnar	enh	**0.0723**	**0.1094**	**0.1433**	**0.1768**		*mnar*	enh	**0.0720**	**0.1081**	**0.1422**	**0.1770**
		std	0.0921	0.1220	0.1576	0.1879			std	0.0925	0.1241	0.1567	0.1880
		GAIN	0.2356	0.2640	0.2822	0.2973			GAIN	0.2463	0.2562	0.2460	0.2824
		VAE	0.2304	0.2559	0.2873	0.3183			VAE	0.1528	0.1735	0.1848	0.2038
	mar	enh	**0.0726**	**0.1027**	**0.0852**	**0.1443**		*mar*	enh	**0.0878**	**0.0897**	**0.0979**	**0.1024**
		std	0.0893	0.1105	0.1016	0.1601			std	0.0960	0.1024	0.1091	0.1096
		GAIN	0.2172	0.2539	0.2712	0.2937			GAIN	0.1952	0.1946	0.2268	0.2381
		VAE	0.2380	0.2484	0.2884	0.2733			VAE	0.1590	0.1842	0.1816	0.1666

TABLE III. (RMSE) NORMAL DATASET – FROG | SATE

dts	sce	mod	Missing Ratio				dts	sce	mod	Missing Ratio			
			20%	*40%*	*60%*	*80%*				*20%*	*40%*	*60%*	*80%*
frog	*mcar*	enh	**0.0559**	**0.0568**	**0.0618**	**0.0771**	**sate**	*mcar*	enh	**0.0392**	**0.0419**	**0.0486**	**0.0615**
		std	0.0590	0.0622	0.0682	0.0832			std	0.0406	0.0427	0.0491	0.0624
		GAIN	0.0757	0.1079	0.1977	0.2005			GAIN	0.0515	0.0864	0.1553	0.1958
		VAE	0.1118	0.1199	0.1229	0.1267			VAE	0.1541	0.2438	0.3346	0.4233
	mnar	enh	**0.0585**	**0.0570**	**0.0605**	**0.0761**		*mnar*	enh	**0.0416**	**0.0422**	**0.0481**	**0.0597**
		std	0.0621	0.0636	0.0661	0.0812			std	0.0430	0.0425	0.0487	0.0614
		GAIN	0.1163	0.1380	0.1832	0.2248			GAIN	0.1023	0.1564	0.2508	0.2776
		VAE	0.1166	0.1172	0.1161	0.1283			VAE	0.1458	0.2353	0.3293	0.4101
	mar	enh	**0.0544**	**0.0493**	**0.0533**	**0.0604**		*mar*	enh	**0.0367**	**0.0390**	**0.0418**	**0.0454**
		std	0.0569	0.0536	0.0589	0.0662			std	0.0372	0.0399	0.0426	0.0458
		GAIN	0.1102	0.1095	0.1801	0.2224			GAIN	0.1062	0.1276	0.1718	0.2421
		VAE	0.1174	0.1157	0.1288	0.1328			VAE	0.1497	0.2247	0.3086	0.3921

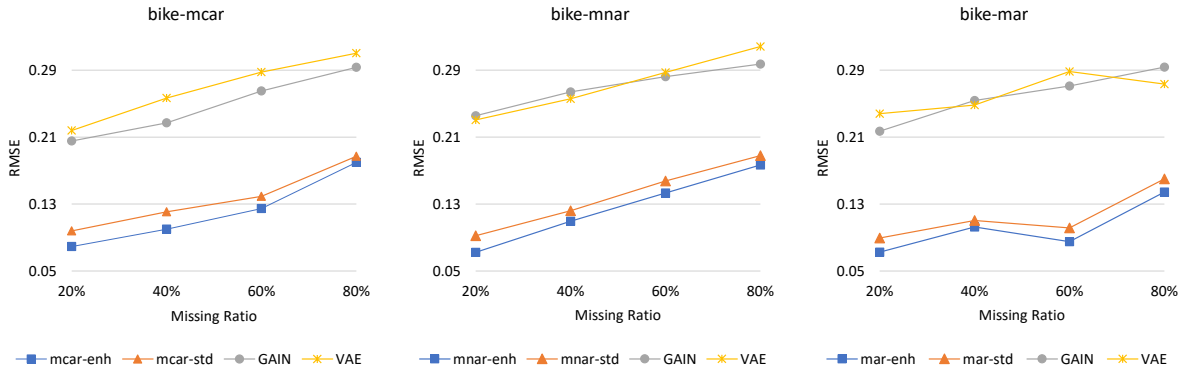

Fig 1. Bike Dataset Visualization (Result)

Each scenario is treated equally, with various missing rates of 20%, 40%, 60%, and 80%. We identified that Perlin noise could make noisy data more stable on non-normal data than normal noise. Furthermore, to validate the results of the proposed method, we conducted experiments with several datasets. The experiment was conducted ten times for each scenario with the standard scheduler CSDI algorithm (std). We take the average value to be displayed in Table.III and Table. IV.

Table. III and Table. IV compare the proposed method (Perlin noise generation and cosine scheduler), standard CSDI (normal noise generation), GAN's method for imputation (GAIN), and imputation based on Variational Auto Encoder (VAE). Denote by enh, std, GAIN, and VAE, respectively. We tried to compare our proposed method with the state-of-the-art imputation method with CSDI. CSDI imputation is an imputation algorithm based on a diffusion model. We modified and utilized the implementation of CSDI in TabCSDI [25].

The results in Table.III shows that the RMSE results in the proposed method have consistently smaller values than those RMSE values in the CSDI and GAIN—the test in Table.III is on non-normal data. Using the normal dataset, we also tested the proposed method (Perlin noise generation). The results of this test are shown in Table IV. The test is conducted in an identical environment. On the normal dataset, the performance of Perlin noise is almost the same as that in the standard method. Thus, the effect of Perlin noise is not very influential on the imputed normal dataset.

We descriptively and visually examine the comparative error performance, specifically the Root Mean Square Error (RMSE), in experimental scenarios involving different types of missing data mechanisms: Missing Completely at Random (MCAR), Missing Not at Random (MNAR), and Missing at Random

TABLE IV. ABLATION STUDY | BIKE, CREDIT CARD

No	Perlin Noise	Cosine Scheduler	RMSE (20% missing ratio - MCAR)	
			Bike Dataset	Credit Card Dataset
1	*x*	*x*	0.0930	0.0951
2	√	*x*	**0.0870**	**0.0916**
3	*x*	√	**0.0835**	**0.0849**
4	√	√	0.0747	0.0817

TABLE V. ABLATION STUDY | FROG, SATELLITE

No	Perlin Noise	Cosine Scheduler	RMSE (20% missing ratio - MCAR)	
			Frog Dataset	Satellite Dataset
1	*x*	*x*	0.0593	0.0402
2	√	*x*	0.0598	0.0461
3	*x*	√	0.0544	0.0391
4	√	√	0.0562	0.0416

(MAR). Each missing data scenario involved varying levels of missingness at rates of 20%, 40%, 60%, and 80%. These analyses were conducted on non-normally distributed data, exemplified by the Bike dataset as visualized in Fig 1.

The RMSE values obtained from the proposed method (enh) consistently demonstrated the lowest RMSE values compared to those of the baseline algorithm (std), GAIN, and VAE. This consistent trend across different missing data scenarios suggests the ability of the proposed method (enh) to minimize errors within these datasets. Visual representations in Fig 1 elucidate that Perlin noise generation consistently yields lower RMSE values than the standard algorithm (std), GAIN, and VAE across various missing mechanisms and rates. This consistency underscores the efficacy of Perlin noise in reducing error values.

This advantageous characteristic of Perlin noise can be attributed to its high slope normal distribution within the dataset, compelling non-normally distributed data to adopt a form closer to normality through noisy data contamination. However, this effect is not observed in datasets with a normally distributed structure. In cases where the dataset is normally distributed, Perlin Noise does not confer any advantage in normalizing the data, as the dataset already exists in a normal distribution state. Through testing the RMSE value, we can conclude that the proposed method (enh) has a lower RMSE value than the baseline (std) on data with non-normal values.

A. Ablation Analysis.

Table.VI and Table.VII shows the effect of each of our proposed noise generation schedulers. We compare the proposed method with the standard condition on the non-normal, categorical, and normal datasets. It compares the result of the 20% missing rate average on MCAR, MNAR, and MAR scenarios.

In Table. VI rows 1 and 2, we show the results without using the cosine scheduler, but we added Perlin Noise in row 2. In rows 3 and 4, we compare the algorithm with the enhancement of the Cosine scheduler in Table. VI, Bike dataset using Perlin noise produces RMSE = 0.0870, smaller than the RMSE at row 1 (without Perlin noise) with RMSE = 0.0930. Also, in row 2 of the Credit Card dataset (CC), Perlin noise produces RMSE = 0.0916, smaller than the RMSE at row 1 (without Perlin noise) with RMSE = 0.0951. In summary, the difference RMSE value of Perlin Noise Components: Bike dataset = 6.4 % and Credit card dataset: 3.6 %. For the Cosine scheduler components: Bike dataset = 10.2 % and Credit card dataset = 10.7%. Additionally, when incorporating the cosine scheduler component, the RMSE value becomes 10% lower than the baseline method. Here, we can conclude that Perlin noise can improve the performance of RMSE on non-normal datasets.

The result of the ablation study on the normal dataset is explained in the Frog and Satellite dataset. The Perlin noise component's effect cannot help improve the performance of the RMSE value for a normal dataset. On the contrary, in the cosine scheduler, the component can help improve the RMSE performance consistently on the normal dataset. In row 3 of Table VII, the resulting RMSE value is 0.391 on the Satellite Dataset. For this value, if we compare row 1 (without cosine scheduler), the RMSE value is greater, which is 0.0402.

The ablation study test for the Frog and Satellite dataset (normal data distribution) Perlin noise generation cannot help increase RMSE performance on diffusion model imputation. However, a cosine scheduler can improve RMSE performance in normal and non-normal data. The differences in RMSE values for datasets incorporating Cosine scheduler components were as follows: the Frog dataset exhibited a decrease of 8.2%, and the Satellite dataset experienced a decline of 2.7%

In row 4 of Table.VI, we can see that adding Perlin noise generation and cosine scheduler components results in the smallest RMSE value compared to those without these components—the RMSE value of the Credit Card dataset in row 4 of Table.VI has a value of 0.0817. This value is smaller than the value of the Credit Card dataset in row 1 without the Perlin noise and cosine scheduler components, with an RMSE value of 0.0951. From the ablation study test in Table.VI and Table.VII, it can be explained that adding the Perlin noise generation component with a cosine scheduler consistently lowers the RMSE imputation value on non-normal datasets.

V. CONCLUSION

Perlin noise generation and cosine scheduler have a positive influence on improving the performance of non-normal data imputation. Based on the Perlin noise generation evaluation, our tests have decreased the RMSE error value. Our scheduler component changes also improved performance compared to the standard baseline on non-normal data. Based on the evaluation, analysis, and experiments conducted, the characteristic of Perlin noise can be attributed to its high slope normal distribution within the dataset, compelling non-normally distributed data to adopt a form closer to normality through the imposition of noisy data contamination. Perlin noise distribution makes the noisy data added with noise more normal when entering the deep learning network in the imputation diffusion model. Loss function calculation in normal noise generation makes calculations based on the fraction of noise with data. However, with the change of loss

function in Perlin noise, the noise portion is given directly when the noise is generated in the Perlin noise formula. Thus, Perlin noise can be utilized to improve imputation on non-normal data. The ablation study of non-normal data explains that Perlin noise generation coupled with a cosine scheduler can consistently have a lower RMSE value of 10.7 % than the standard CSDI algorithm. Our plan for future research is to explore how continual learning can improve the performance of deep learning algorithms. However, whether the improvement is practical or can be applied in general needs to be examined more carefully.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the Faculty of Computer Science Unviersitas Indonesia and DIKTI-AI-Center for providing the computational resources and support. The access to advanced significantly contributed to the successful completion of this research.

REFERENCES

[1] X. Miao, Y. Gao, B. Zheng, G. Chen, and H. Cui, “Top-k dominating queries on incomplete data,” IEEE Trans. Knowl. Data Eng., vol. 28, no. 1, pp. 252–266, Jan. 2015.

[2] A. A. Qahtan, A. Elmagarmid, R. C. Fernandez, M. Ouzzani, and N. Tang, “FAHES: A robust disguised missing values detector,” in Proc. SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 2100–2109.

[3] H. Song and D. A. Szafir, “Where’s my data? Evaluating visualizations with missing data,” IEEE Trans. Vis. Comput. Graphics, vol. 25, no. 1, pp. 914–924, Jan. 2019.

[4] B. Zhao, B. Wu, T. Wu, and Y. Wang, “Zero-shot learning posed as a missing data problem,” in Proc. Int. Conf. Comput. Vis., 2017, pp. 2616–2622.

[5] D. B. Rubin, “Inference and missing data,” Biometrika, vol. 63, no. 3, pp. 581–592, 1976.

[6] N. S. Altman, “An introduction to kernel and nearest -neighbor nonparametric regression,” Amer. Statistician, vol. 46, no. 3, pp. 175–185, 1992.

[7] B. Twala, M. Cartwright, and M. Shepperd, “Comparison of various methods for handling incomplete data in software engineer-ing databases,” in Proc. Int. Symp. Empir. Softw. Eng., 2005, pp. 234–239.

[8] D. J. Stekhoven and P. B€uhlmann, “MissForest non parametric missing value imputation for mixed-type data,” Bioinformatics, vol. 28, no. 1, pp. 112–118, 2011.

[9] P. Royston and I. R. White, “Multiple imputation by chained equations (MICE): Implementation in stata,” J. Statist. Softw., vol. 45, no. 4, pp. 1–20, 2011.

[10] X. Miao, Y. Wu, L. Chen, Y. Gao and J. Yin, "An Experimental Survey of Missing Data Imputation Algorithms," in IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 7, pp. 6630-6650, 1 July 2023, doi: 10.1109/TKDE.2022.3186498.

[11] Tashiro, Yusuke, et al. "Csdi: Conditional score-based diffusion models for probabilistic time series imputation." Advances in Neural Information Processing Systems 34 (2021): 24804-24816.

[12] Perlin, Ken. "An image synthesizer." ACM Siggraph Computer Graphics 19.3 (1985): 287-296.

[13] Hugo Elias, Mount.D, Eastman.R, CMSC 425: Lecture 12 : Procedural Generation: 1D Perlin Noise, Lecture Notes, https://www.cs.umd.edu /class/spring2018/c msc425/Lects/lect12-1d-perlin.pdf, Accessed on : 2023, June,30

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. NeurIPS, 2020.

[15] Chen, Ting. "On the importance of noise scheduling for diffusion models." arXiv preprint arXiv:2301.10972 (2023).

[16] Lin, Shanchuan, et al. "Common Diffusion Noise Schedules and Sample Steps are Flawed." arXiv preprint arXiv:2305.08891 (2023).

[17] J. Yoon, J. Jordon, and M. Schaar, “GAIN: Missing data imputation using generative adversarial nets,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 5675–5684.

[18] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Let a biogeography-based optimizer train your multi-layer perceptron,” Inf. Sci., vol. 269, no. 1, pp. 188–209, 2014.

[19] A. Majumdar, “Blind denoising autoencoder,” IEEE Trans. Neu-ral Netw. Learn. Syst., vol. 30, no. 1, pp. 312–317, Jan. 2019.

[20] Song, Yang, et al. "Score-based generative modeling through stochastic differential equations." arXiv preprint arXiv:2011.13456 (2020).

[21] Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." Advances in neural information processing systems 34 (2021): 8780-8794.

[22] Chen, Jianfei, et al. "Vflow: More expressive generative flows with variational data augmentation." International Conference on Machine Learning. PMLR, 2020.

[23] Cai, Ruojin, et al. "Learning gradient fields for shape generation." Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. Springer International Publishing, 2020.

[24] Heusel, Martin, et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." Advances in neural information processing systems 30 (2017).

[25] Zheng, Shuhan, and Nontawat Charoenphakdee. "Diffusion models for missing value imputation in tabular data." arXiv preprint arXiv:2210.17128 (2022).

[26] UC Irvine Machine learning Repository, https://archive.ics.uci.edu/, Accessed on :2023, October, 2023.

[27] OpenMl Dataset, https://www.openml.org/, Accessed on :2023, May, 20.

[28] Gondara, Lovedeep, and Ke Wang. "Mida: Multiple imputation using denoising autoencoders." Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III 22. Springer International Publishing, 2018.

[29] Fisher, Robert, et al. "Image Synthesis—Noise Generation."(2013)

[30] Van Buuren, Stef. "Multiple imputation of multilevel data." Handbook of advanced multilevel analysis 10 (2011): 173-196.

[31] Shapiro, S. S.; Wilk, M. B. (1965). "An analysis of variance test for normality. Biometrika. 52 (3–4): 591–611. JSTOR 2333709.