EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

AI Diffusion Model Based on Stochastic Differential Equations: Noise Scheduling Variable Discrete Approximation

Dongqing Du[1*], Yan Liu[2]

[1]Mathematics Teaching & Research Section, Xuzhou Finance and Economics Branch, Jiangsu Union Technical Institute, Xuzhou 221008, China
[2]School of Computing Science, Wuhu University, Wuhu 241000, China

Abstract

Recently, diffusion probabilistic models have shown great success in image synthesis, but their performance relies heavily on the pre-defined noise schedules, which have a direct impact on the convergence speed, stability, and quality of the generated images. Moreover, the traditional score-based SDE model is very sensitive to the choice of the step size in the numerical integration process. To overcome the above limitations, a variable discrete noise scheduling method combined with a Fully Connected Neural Network (FC-NN) denoiser is proposed in the SDE diffusion model. Experimental analysis on the CIFAR-10 dataset shows that the proposed Variable SDE model results in a final FID value of 8.7, which is better than DDPM (11.5), DDIM (10.9), and Score SDE (10.2), and an Inception Score of 9.4, which is better than 8.6 for the fixed scheduling approach. The proposed approach results in a reduction of convergence time by 20-25% and peak discretization error by 30-35%. The t-SNE analysis shows that the overlap between the real and generated distributions is good, and there is no mode collapse. The proposed approach provides faster convergence, better numerical stability, and improved generative quality compared to existing standard diffusion models.

Keyworks: SDE-based Diffusion Model, Variable Noise Scheduling, Discretization Error Reduction, Generative Image Modeling, CIFAR-10 Image Synthesis

Received on 04 March 2026, accepted on 23 October 2026, published on 13 November 2026

Copyright © 2026 Dongqing Du *et al.*, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetsis.12089

*Corresponding author. Email: Dongqingdu1@outlook.com

1. Introduction

Artificial Intelligence has made considerable progress in the area of generative modeling, where computers can generate realistic images, audio, and other types of data [1]. Among these models, diffusion models have attracted interest for their stability and high-quality generation capabilities [2]-[3]. Diffusion models generate data by progressively adding noise to the input distribution and learning the corresponding reverse denoising process. Recently, stochastic differential equations (SDEs) have been used to formulate the continuous-time dynamics of diffusion processes [4]. In practice, these continuous formulations must be approximated using discrete numerical schemes [5], making the choice of discretization strategy important for the stability and quality of diffusion-based generation. In discrete approximations is a very important step in the stability and quality of diffusion models. Therefore, the improvement of noise scheduling methods is an active area of research in diffusion-based generative modeling [6]. Fixed noise schedules use predetermined temporal intervals that may not adequately resolve rapidly changing stochastic diffusion dynamics,

leading to larger discretization errors and reduced numerical stability. Such mismatches can slow convergence and reduce the accuracy of the reverse diffusion approximation, which can subsequently affect the quality of generated samples.

Diffusion models have many applications in image synthesis, medical image synthesis, super-resolution, data augmentation, and creative content generation. They are used in applications such as computer vision, healthcare imaging, autonomous systems, and multimedia content generation [7]. High-quality image synthesis improves the availability of training data in machine learning [8]. In healthcare imaging, diffusion models are employed to generate realistic images for analysis. In industrial and scientific applications, they are employed in simulation and prediction applications [9]. Noise scheduling can improve computational efficiency in these applications. Thus, the optimization of diffusion processes has significant applications in various real-world fields [10].

This study proposes an SDE-based diffusion model that combines variable discrete noise scheduling with a Fully Connected Neural Network (FC-NN) denoiser. The FC-NN serves as the fixed denoising architecture, while variable discrete noise scheduling constitutes the principal methodological contribution. The scheduler dynamically adjusts the diffusion time intervals according to the evolving noise level and discretization behavior without modifying the denoising architecture. The adaptive scheduler determines variable time steps according to the estimated discretization error, assigning finer steps to regions requiring greater numerical resolution. This error-aware temporal discretization reduces cumulative numerical approximation error while maintaining the same FC-NN denoiser. The proposed approach is evaluated on the CIFAR-10 dataset and compared with fixed-step diffusion models to assess its effects on generative quality, convergence efficiency, and numerical stability. The denoising component of the proposed framework is implemented exclusively using a Fully Connected Neural Network (FC-NN), without a U-Net encoder-decoder architecture. Each CIFAR-10 image is flattened from 32×32×3 into a 3072-dimensional vector and concatenated with the corresponding time-step embedding. The denoising network is implemented exclusively as a Fully Connected Neural Network (FC-NN) and does not contain a U-Net or convolutional encoder-decoder structure. Each CIFAR-10 image is flattened from 32×32×3 into a 3072-dimensional vector and concatenated with the time-step embedding. The FC-NN is used as the fixed denoising architecture, whereas variable discrete noise scheduling constitutes the principal methodological contribution of the proposed framework. The FC-NN serves as the fixed denoising backbone throughout the proposed diffusion process and is not replaced by a U-Net or convolutional denoising network. Its vector-based input representation and network configuration are maintained across the fixed- and variable-scheduling experiments. The FC-NN therefore serves only as the denoising backbone, while adaptive temporal discretization represents the proposed methodological change. No architectural modification, convolutional feature extraction, or U-Net component is introduced in the proposed VDSDE-DM framework. The scheduler dynamically adjusts diffusion time intervals according to the evolving noise level and discretization behavior without changing the denoising architecture. In contrast to previous methods that relied on complex models, a simpler neural network is used to decrease the computational cost. The novelty of the proposed VDSDE-DM lies in its discretization-error-aware variable noise scheduling strategy, which dynamically allocates diffusion time steps while maintaining a fixed lightweight FC-NN denoiser. Unlike conventional approaches that rely on uniform or predetermined noise schedules, the proposed method adapts the temporal discretization to the evolving diffusion dynamics, reducing numerical approximation error without increasing architectural complexity. This controlled design isolates the contribution of adaptive scheduling and improves generative quality, convergence efficiency, and numerical stability, as reflected by the reported FID of 8.7 and Inception Score of 9.4. While keeping the FC-NN denoiser unchanged to isolate the contribution of adaptive temporal discretization. The proposed work also introduces adaptive, non-uniform noise steps to decrease discretization errors and improve the stability of convergence. The proposed method is tested on the CIFAR-10 dataset to examine the generative capability and training speed. The proposed method is compared with fixed-step diffusion models to confirm the effectiveness of the proposed work. The proposed work has been shown to improve the stability and image quality.

The rest of this paper is structured as follows. Section 2 provides a comprehensive review of the related work. Section 3 introduces the problem statement and describes the research challenges that are addressed in this paper. Section 4 introduces the proposed methodology and model framework. Section 5 describes the experimental setup, including datasets, implementation, and evaluation metrics. Section 6 discusses the results and provides a comprehensive analysis of the findings. Finally, Section 7 concludes this paper and introduces potential directions for future research.

2. Related Works

Kumar [11] offered a thorough survey on DMs as robust deep learning architectures for generative AI systems. The research work systematically examined design aspects such as refinement techniques, parallel generation, editing, in-painting, and cross-domain synthesis in the realms of CV, NLP, medical image processing, video modeling, and molecular synthesis [12]. The comparative assessment of DM-based models by the authors indicates that they give better performance than traditional generative models in image synthesis [13]. The survey results showed that DMs currently face two main challenges because of their high computational needs and their reliance on specific datasets, which creates a pathway for upcoming research to develop

better systems that can handle more data while working more efficiently. Cao et al. [14] The researchers conducted a complete investigation of deep generative models while their primary focus remained on diffusion models which serve as the fundamental technology enabling AIGC. The authors investigated diffusion models through three distinct perspectives which included basic mathematical definitions and algorithmic advancements and their use in different fields that included image processing and text analysis and speech recognition and biological research and healthcare applications [15]. The authors' survey showed how diffusion models developed faster and better than earlier generative models. The survey results showed that diffusion models now have increasing power across multiple fields while research continues to develop better efficiency and scalability solutions. Liang et al. [16] presented a framework for solving complex network optimization problems in IoT settings with the GDM. The work theoretically demonstrated the superiority of generative models over discriminative models and proposed DDPMs with classifier-free guidance to learn high-quality distributions of solutions [17]. The experiments carried out on three complex network optimization problems demonstrated the efficiency of the proposed approach in minimizing prediction errors and approaching near-optimal solutions. The research has confirmed the efficiency of diffusion models as effective optimizers that can approximate the optimal network configuration with improved robustness [18].

Babazadeh Darabi and Coleri [19] introduced an alternative solution framework based on a diffusion model for resource allocation in WNCSs, different from DRL. The solution was based on a conditioned on CSI to obtain optimal blocklengths for the minimization of total power consumption under finite blocklength constraints [20]. Simulation analysis showed that the proposed method was close to optimal in terms of power efficiency and performed much better than DRL-based approaches. Additionally, the model decreased violations of critical constraints by a factor of up to 18×. Luo et al. [21] gave a thorough overview of diffusion models in the field of biomedical informatics, focusing on the underlying principles, different variants, and various applications of diffusion models in the biomedical domain [22]. The authors reviewed different frameworks like DDPMs and SDE models for medical image synthesis, reconstruction, segmentation, and removal of noise in medical images like MRI and PET images, as well as EHR data simulation and physiological signal restoration (EEG/ECG) [23]. The authors concluded that diffusion models have shown excellent results in 2D/3D MRI and PET image synthesis, structured healthcare data simulation, and molecular/protein structure prediction for drug discovery applications. Croitoru et al. [24] provided a comprehensive overview of denoising diffusion models in computer vision, explaining their two-step forward and reverse diffusion process for high-quality generative modeling [25]. The paper categorized diffusion models into DDPMs, NCSNs, and SDE-based models [26]. A comparison study was conducted between diffusion models and other generative models such as VAEs, GANs, energy-based models, autoregressive models, and normalizing flows. The findings indicated the improved sample quality and diversity of diffusion models but highlighted computational complexity and slow sampling speed as significant drawbacks and future research areas [27].

Lin et al. [28] offered a comprehensive survey on diffusion models, with a special focus on the extension of diffusion models to time series tasks. The research work performed a comprehensive survey on diffusion models for time series forecasting, imputation, and data synthesis [29]. The survey work of the authors highlighted the excellent performance of diffusion models in modeling complex temporal distributions and generating high-quality sequential data [30]. The results of the survey work also indicated the difficulties in diffusion models, such as efficiency and robustness in modeling temporal distributions. Du et al. [31] offered a detailed tutorial on GDMs and their application in network optimization for Generative AI systems. The research work introduced the theoretical background of GDMs and their applications with other approaches such as DRL, Incentive SemCom, and IoV networks [32]. The research work proved the application of GDMs in modeling complex distributions of solutions and solving difficult optimization problems in real-world networks through various case studies [33]. The outcome of the research work has shown the efficiency, flexibility, and future prospects of GDMs in optimizing intelligent networks. Moser et al. [34] provided a comprehensive overview of DMs employed in image SR, highlighting their ability to generate highly realistic and perceptually superior high-resolution images [35]. The authors talked about theoretical backgrounds and design choices, such as conditioning schemes, guidance strategies, other input domains, corruption spaces, and zero-shot learning strategies [36]. The authors' work particularly emphasized that diffusion models for SR outperform traditional generative models in terms of image quality but are challenged by high computational complexity, color channel issues, and lack of interpretability [37]. The authors' work highlighted the current limitations and future research directions to improve efficiency, robustness, and interpretability of diffusion models for super-resolution tasks [38].

Despite the major breakthroughs highlighted in the literature, there are a number of common challenges that still exist in diffusion models. These include the high computational cost, dataset dependency, and scalability issues in large-scale generative models, as pointed out by Kumar [11] and Cao et al. [14]. Liang et al. [16] and Babazadeh Darabi and Coleri [19] also pointed out that, although diffusion models perform near-optimally in network optimization and wireless resource allocation problems, they are computationally expensive and require proper conditional guidance to converge and satisfy the constraints. In the biomedical and vision applications, Luo et al. [21], Croitoru et al. [24], and Moser et al. [34] also highlighted the challenges of diffusion models, which

include low sampling speed, lack of interpretability, high computational complexity, color inconsistencies, and low interpretability in medical and super-resolution applications. In addition, Lin et al. [28] also highlighted the challenges of diffusion models in terms of efficiency, robustness in temporal modeling, and complexity in integration with DRL and IoV applications.

3. Problem Statement

Although diffusion models have shown better generative power and optimization efficiency in computer vision, biomedical informatics, time series modeling, and network optimization problems, there are still some important open issues that need to be addressed. The high computational cost, low sampling efficiency, and scalability issues that make diffusion models unsuitable for real-time applications in resource-limited settings have been emphasized in previous works Kumar [11]. In network optimization and wireless communication networks, diffusion models require a lot of training infrastructure and complex conditional guidance strategies to guarantee convergence and satisfaction of constraints Croitoru et al. [24]. In addition, problems such as lack of interpretability, dependence on the dataset, robustness issues in time series modeling, inconsistencies in color in image super-resolution tasks, and the absence of standardized benchmarking platforms have been recognized in the biomedical, vision, and time series communities, respectively Lin et al. [28]. However, fixed diffusion schedules do not explicitly adapt their discrete time intervals to variations in noise evolution and numerical discretization error. This limitation can increase the number of optimization steps and reduce numerical efficiency even when the denoising network remains unchanged. Therefore, the proposed framework focuses on variable discrete scheduling while maintaining an FC-NN denoiser to isolate the contribution of adaptive time-step selection. This design targets the computational limitations of diffusion models by improving sampling efficiency and reducing unnecessary temporal discretization while retaining a lightweight FC-NN denoiser. The resulting framework is intended to improve scalability and computational practicality without sacrificing generative quality or numerical stability.

Objectives

- Develop a Variable Discrete SDE-Based Diffusion Model (VDSDE-DM) with adaptive noise scheduling to improve generative image quality and numerical stability.
- The primary methodological objective is to improve diffusion discretization through adaptive selection of non-uniform time intervals while keeping the denoising architecture fixed.
- Integrate a fully connected neural network (FC-NN) denoiser within the diffusion framework to enhance noise prediction and reduce discretization error.
- Evaluate convergence efficiency and stability improvements by comparing the proposed adaptive scheduling method against baseline models (DDPM, DDIM, and Score SDE).
- Verify the performance of the model using the CIFAR-10 dataset, evaluating the quality of the generated samples using quantitative metrics such as FID, Inception Score, number of convergence epochs, and t-SNE feature analysis.

Through these objectives, the proposed approach seeks to improve efficiency, robustness, interpretability, and practical applicability of diffusion models in next-generation intelligent systems.

4. Methodology

In this study, an SDE-based diffusion model is employed with a FC-NN to learn the denoising process on the CIFAR-10 dataset. The denoiser receives a flattened 3072-dimensional noisy-image vector together with the corresponding time-step embedding as its input. The concatenated vector is processed only through fully connected layers to predict a 3072-dimensional noise vector. To begin with, the images are normalized and flattened before being progressively corrupted with Gaussian noise following a SDE. Rather than employing fixed time steps, a variable discrete noise scheduling approach is proposed, where the time intervals are dynamically adjusted for enhanced stability and reduced discretization error. The proposed framework therefore contains two distinct components: the FC-NN performs noise prediction, while the variable discrete scheduler determines the diffusion time-step sequence. The FC-NN receives the flattened 3072-dimensional noisy-image vector together with the time-step embedding and predicts the corresponding noise vector. No convolutional layers, spatial feature maps, skip connections, or U-Net encoder-decoder blocks are used in this denoising pathway. The FC-NN parameters and input representation remain unchanged when comparing fixed and variable scheduling conditions. During the training phase, a random time step is chosen, and noise is added to the image, and the FC-NN predicts the added noise based on time step embeddings. The training is performed using mean squared error loss between the predicted and actual noise. During the generation phase, the reverse SDE process is implemented step by step based on the trained network to remove noise and generate new images. The stochastic dynamics are represented through the SDE formulation, while the noisy image is expressed as a 3072-dimensional vector and combined with its time-step embedding for FC-NN noise prediction.

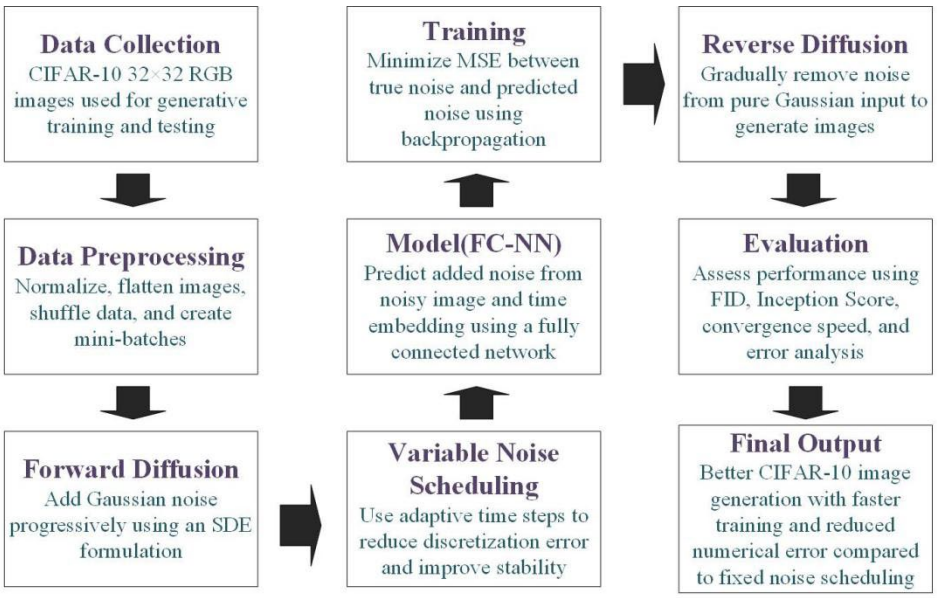


Figure 1. Overall Workflow of the Proposed Variable SDE-Based Diffusion Model

This image shows the entire pipeline of the proposed adaptive SDE-based diffusion model on the CIFAR-10 dataset. It starts with data acquisition and preprocessing, then forward diffusion with the addition of Gaussian noise. The FC-NN neural network is trained to predict the added noise based on the MSE loss function. A variable noise scheduling method is added to mitigate discretization errors. Lastly, reverse diffusion produces high-quality images, and the performance is measured using FID and Inception Score metrics as shown in Figure 1.

4.1 Variable Discrete Noise Scheduling

In traditional diffusion models like DDPM and Score-Based Generative Model, the noise schedule is defined by a fixed linear or cosine strategy. Although this strategy works well, it does not adapt to the training dynamics. This often leads to a slower convergence rate and a higher discretization error. To overcome this drawback, this research proposes a VDSDE-DM, in which the noise intensity and the size of the time step are adaptively varied during the training process. Unlike an architectural modification, the proposed mechanism changes only the temporal discretization of the diffusion process while retaining the same FC-NN denoiser. Smaller intervals are assigned to regions exhibiting stronger numerical variation, whereas larger intervals are used where the diffusion dynamics are comparatively stable.

Forward Diffusion Process:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\epsilon, \epsilon \sim \mathcal{N}(0, I) \quad (1)$$

This Equation (1) defines the forward diffusion process where clean data x_0 is gradually corrupted with Gaussian noise. The parameter α_t controls how much original signal is preserved at timestep t. The term $\sqrt{1-\alpha_t}\epsilon$ introduces stochastic perturbation. As t increases, noise dominance increases progressively. In the proposed model, α_t is dynamically adjusted instead of using a fixed schedule.

Noise Variance Definition:

$$\alpha_t = 1 - \beta_t \quad (2)$$

This Equation (2) links the signal preservation factor α_t to the noise variance β_t. When β_t increases, the signal retention factor decreases. This relationship ensures controlled degradation of the input data. In fixed dispersion models, β_t follows a predefined linear agenda. In the proposed method, β_t is adaptively resolute to improve union stability.

Adaptive Noise Intensity Formulation:

$$\beta_t = \beta_{\min} + (\beta_{\max} - \beta_{\min}) \cdot f(t) \quad (3)$$

This Equation (3) defines the adaptive noise preparation strategy. The noise variance is bounded between $\beta_{\min}$ and $\beta_{\max}$. The scaling function $f(t)$ dynamically adjusts noise strength across timesteps. This prevents excessive early noise injection. The adaptive preparation improves numerical stability and rushes union.

Scheduling Function:

$$f(t) = \frac{1}{1+e^{-\tau(t-T/2)}} \quad (4)$$

This sigmoid-based function panels the smooth variation of noise intensity over time. The restriction γ determines the sharpness of transition. Early training stages receive lower noise trepidation. Midway through training, noise gradually increases. This smooth scheduling reduces

variability compared to abrupt linear schedules as shown in Equation (4).

Adaptive Step Size Adjustment:

$$\Delta t_t = \Delta t_0 \cdot (1 - \lambda E_t) \tag{5}$$

This Equation (5) adjusts the timestep size based on projected discretization error E_t. As the error rises, the step size falls to improve stability. Taking smaller steps helps improve the accuracy of the numerical approximation of the SDE. When the error is small, larger steps speed up the process. This method helps minimize the maximum discretization error by keeping adaptive stability.

Continuous-Time SDE:

$$dx = f(x,t)dt + g(t)dW \tag{6}$$

This Equation (6) represents the dispersion process as a stochastic differential. The drift term $f(x,t)$ controls deterministic growth. The diffusion coefficient $g(t)$ governs stochastic noise dose. The Wiener process dW introduces randomness. In the proposed approach, adaptive scheduling modifies $g(t)$ to improve convergence efficiency and stability.

4.2 Neural Network Architecture (FC-NN)

The FC-NN is a feedforward neural network in which each neuron is connected to all neurons in the subsequent layer. In the proposed framework, the network receives the flattened noisy image together with the corresponding time-step embedding. Several fully connected hidden layers with ReLU activation functions are used to learn the noise-prediction mapping, while the final linear layer produces a noise vector with the same dimensionality as the input image. In the proposed implementation, this FC-NN is the complete denoising network and consists only of fully connected transformations with nonlinear activations. It does not employ convolutional layers, skip connections, spatial feature maps, or U-Net encoder-decoder blocks. The proposed denoising network is exclusively an FC-NN, and no U-Net denoiser is used in the experimental implementation. The same FC-NN architecture is retained throughout the fixed- and variable-scheduling experiments so that the scheduling strategy remains the principal changed factor. The FC-NN takes the flattened noisy image and the time step as input. The architecture consists of several hidden layers with nonlinear activation functions like ReLU. Every hidden layer captures a higher-level representation of the noisy input. The weights are trained using backpropagation to optimize the prediction of the error. The network is trained to predict the added Gaussian noise in the diffusion process. MSE is used as the loss function to measure the difference between the predicted and actual noise. The final output layer is a vector of the same size as the input image. The FC-NN consists of [N] fully connected hidden layers with dimensions [D1, D2, …, DN], with ReLU activation applied after each hidden layer, followed by a linear 3072-dimensional output layer. The flattened 3072-dimensional image vector is concatenated with a [D]-dimensional sinusoidal time embedding, and the resulting representation is provided to the denoising network. The proposed denoiser is exclusively an FC-NN, with the flattened 3072-dimensional noisy image concatenated with the time-step embedding as input, followed by fully connected hidden layers with ReLU activation and a final linear layer producing a 3072-dimensional noise vector. The same FC-NN architecture and parameter configuration are retained for both fixed and variable noise-scheduling experiments, and no U-Net architecture or separate U-Net experiment is used in this study. The resulting architecture is therefore a fully connected denoiser that operates on vectorized image representations rather than spatial feature maps. For CIFAR-10, the 32×32×3 image is represented as a 3072-dimensional vector, and the output layer predicts a noise vector of the same dimensionality. This architecture provides a lightweight fully connected alternative for diffusion-based noise prediction while retaining the same FC-NN configuration throughout the experiments.

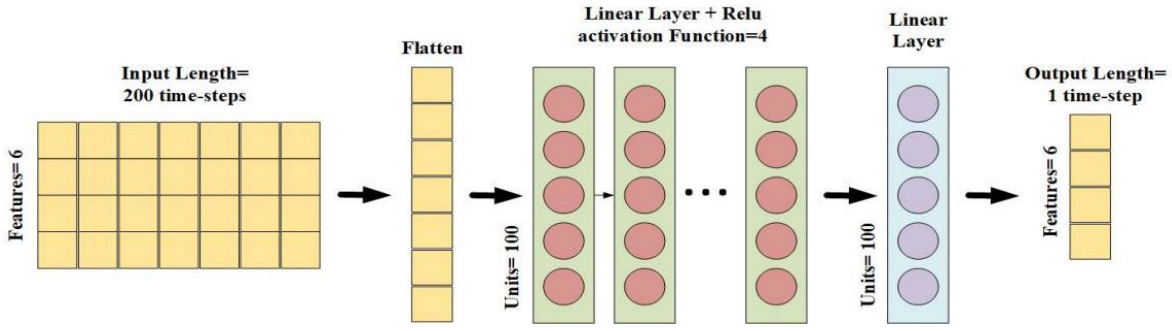


Figure 2. Architecture of the FC-NN for Diffusion-Based Noise Prediction

The above Figure 2 shows the architecture of the FC-NN network that is employed for the prediction of noise in the diffusion process. The input features are multi-feature time-step data, which is flattened into a one-dimensional

vector. The flattened input is then passed through several fully connected layers with ReLU activation functions. The final linear layer is used to project the features to the output space. The output is the predicted noise for a single time step in the diffusion model.

4.2.1 Input Representation: Noisy Image with Time Embedding

The input to the diffusion model is a noisy image, which is produced at a certain diffusion time step. As a FC-NN needs vector inputs, the 2D image is transformed into a 1D vector.

For CIFAR-10, the 32×32×3 image tensor is flattened into a 3072-dimensional vector before entering the denoising network. A time-step embedding of is then concatenated with this image vector to provide diffusion-time conditioning. It is also necessary that the model knows the noise level at which the image has been corrupted. Hence, a time-step embedding is created to carry the diffusion step information. The noisy image vector and the time embedding vector are then fed as a single input to the network.

Forward Diffusion Process

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t \epsilon} \tag{7}$$

This Equation (7) represents the forward diffusion process at time step t. The original clean image x_0 is gradually corrupted by Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$. The coefficient α_t controls how much original signal remains at time t. As t increases, α_t decreases, meaning more noise is added. This process ensures a smooth transition from dean image to pure Gaussian noise.

Flattening Operation

$$\overline{\boldsymbol{x}}_t = \text{ Flatten } (x_t) \tag{8}$$

This Equation (8) converts the 3D image tensor into a 1D vector. For CIFAR-10, the original dimension is $32 \times 32 \times 3$. After flattening, the dimension becomes 3072. This transformation is necessary because FC-NN layers operate on vectors. No information is lost; only the spatial structure is reshaped into vector form.

Concatenated Input

$$z_t = [\bar{x}_t \| e(t)] \tag{9}$$

This Equation (9) combines the flattened noisy image and time embedding. The symbol || represents vector concatenation. The resulting vector contains both visual information and temporal information. This allows the model to understand the amount of noise present. Without time conditioning, the model cannot properly reverse the diffusion process.

4.2.2 Time-Step Embedding

Time embedding is a method of embedding the diffusion step into a high-dimensional space. This allows the neural network to distinguish between early and late stages of diffusion. However, since the noise level changes with time, conditioning is necessary. The sinusoidal embedding is a smooth and continuous method of embedding.

Forward Diffusion Process:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t \epsilon} \tag{10}$$

This Equation (10) represents the forward diffusion process at time step t. The original clean image x_0 is gradually corrupted by Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$. The coefficient α_t controls how much original signal remains at time t. As t increases, α_t decreases, meaning more noise is added. This process ensures a smooth transition from dean image to pure Gaussian noise.

Flattening Operation:

$$\overline{\boldsymbol{x}}_t = \text{ Flatten } (x_t) \tag{11}$$

This Equation (11) converts the 3D image tensor into a 1D vector. For CIFAR-10, the original dimension is $32 \times 32 \times 3$. After flattening, the size becomes 3072. This is required because the layers in the FC-NN are vector operations. The resulting 3072-dimensional vector is used consistently as the FC-NN input during both training and reverse diffusion sampling.

Concatenated Input:

$$z_t = [\bar{x}_t \| e(t)] \tag{12}$$

This Equation (12) combines the flattened noisy image and time embedding. The symbol || represents vector concatenation. The resulting vector contains both visual information and temporal information. This allows the model to understand the amount of noise present. Without time conditioning, the model cannot properly reverse the diffusion process.

4.2.3 Output Layer (Noise Prediction)

The last layer is a predictor of the noise that is added to the image. The output dimension of this layer is the same as the input image vector. There is no activation function in the output layer. The aim is to estimate the actual Gaussian noise. This noise will be used in reverse diffusion.

Output Prediction:

$$\hat{\epsilon}_\theta(x_t, t) = W^{(L)} h^{(L-1)} + b^{(L)} \tag{13}$$

This Equation (13) represents the final linear transformation. L denotes the last layer of the network. The predicted vector $\hat{\epsilon}_\theta$ approximates true noise ϵ. No activation function is applied to preserve numerical range. The output dimension equals 3072 for CIFAR-10 images.

4.2.4 Loss Function (Denoising Objective)

The model is trained to predict the noise that is added during diffusion. The MSE is used as the optimization objective. The loss is the difference between the true and

predicted noise. Reducing the loss will improve reverse diffusion. The objective is called denoising score matching.

Loss Function:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0,t,\epsilon}[\|\epsilon - \hat{\epsilon}_\theta(x_t, t)\|^2] \quad (14)$$

The expectation is taken over image samples, time steps, and noise samples. The term inside measures squared error between actual and predicted noise. If the prediction is accurate, the loss approaches zero. This trains the model to learn reverse diffusion dynamics. Optimizing this function improves generative performance as shown in Equation (14).

Algorithm 1. FC-NN Noise Prediction

Input: Clean CIFAR-10 image
Output: Predicted noise
Step 1: Select a clean image and choose a time step.
Step 2: Add Gaussian noise to generate a noisy image.
Step 3: Flatten the noisy image into a vector.
Step 4: Create a time-step embedding.
Step 5: Combine image vector and time embedding.
Step 6: Pass the input through the FC-NN.
Step 7: Predict the added noise.
Step 8: Compute MSE loss between true and predicted noise.
Step 9: Update network weights using Adam optimizer.
Step 10: Repeat until training converges.

4.3 Training Procedure

Sampling Original Image

The training procedure is initiated by choosing a clean image from the CIFAR-10 dataset. The image is a representation of a real data sample from the actual data distribution. The diffusion model tries to learn the actual data distribution by adding noise and reversing the process. The purpose of sampling is to introduce diversity in the training samples.

$$x_0 \sim p_{\text{data}}(x) \quad (15)$$

Here, x_0 denotes a dean training image. $p_{\text{data}}(x)$ represents the unknown true data distribution. The symbol $\sim$ indicates sampling from this distribution. The model does not know pule explicitly but learns it implicitly. Learning occurs through repeated noisy transformations of x_0 as shown in Equation (15).

Variable Time-Step Selection

A time step t_k is selected to determine the noise level. Unlike fixed discretization, time intervals are non-uniform. Smaller steps are used were noise changes rapidly. Larger steps are used in stable regions. This adaptive mechanism reduces numerical discretization error.

$$t_k = t_{k-1} + \Delta t_k \quad (16)$$

Represents the current diffusion time. Δt_k is the adaptive step give. The step size varies across iterations. This makes time discretization flexible rather than constant. It improves approximation of the continuous SDE as shown in Equation (16).

Forward Diffusion using SDE

Noise is gradually added to the image using a stochastic differential equation. This model's continuous corruption of data aver time. The process contains both deterministic drift and stochastic noise components. Drift stabilizes the process, while diffusion injects randomness. This formulation provides a mathematically grounded diffusion process.

$$dx = -\frac{1}{2}\beta(t)xdt + \sqrt{\beta(t)}dW_t \quad (17)$$

The first term represents deterministic drift. $\beta(t)$ controls the noise schedule aver time. The second term injects Gaussian randomness. dW_t is Brownian motion. Together they describe continuous-time noise evolution as shown in Equation (17).

Discrete Noise Addition

Since continuous SDE cannot be solved exactly, it is discretized. Gaussian noise is sampled at each time step. The clean image is scaled and combined with noise. This produces a noisy image x_{fs}. The noise level depends on time.

$$x_k = \sqrt{\alpha_k x_0} + \sqrt{1 - \alpha_k e} \quad (18)$$

ϵ È Gaussian noise sampled from $\mathcal{N}(0, T)$. α_k represents cumulative noise control the first term preserves part of original image. The second term adds scaled noise. This gives a computed training sample as shown in Equation (18).

FC-NN Noise Prediction

The noisy image is flattened into a vector. A time embedding is generated for $\mathbf{t_k}$. Both are combined as network input. The Fully Connected Neural Network predicts noise. This forms the denoising objective.

$$t_b = f_b(x_{t_5}, t_k) \quad (19)$$

f s denotes the FC-NN model. θ represents trainable parameters. The network outputs predicted noise. Input induces noisy image and time information. The goal is to approximate true noise as shown in Equation (19).

Loss Function Computation

The predicted noise is compared to true noise MSE is used as loss function. This ensures stable optimization. Loss is averaged over batch samples. Training minimizes this error. Training uses a batch size of [B], Adam optimization with an initial learning rate of [LR] and

[schedule/decay setting], and [T] diffusion steps for noise perturbation and denoising.
Each training sample is initialized from a clean CIFAR-10 image, with Gaussian noise sampled from the specified normal distribution at the selected diffusion time step.

$$\mathcal{L} = \mathrm{E}_{\infty,t,t}[|\ \epsilon - \varepsilon_0\|^2] \quad (20)$$

The difference between actual and predicted noise is computed. The squared norm penalizes large deviations. Expectation averages across time and data. Minimizing this improves denoising accuracy. It forms the care training objective as shown in Equation (20).
Parameter Update
Gradients of the lass are calculated. Backpropagation determines direction of improvement. Adam optimizer adjusts weights. Learning rate controls update size. Training repeats over multiple epochs.

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L} \quad (21)$$

η denotes learning rate. $\nabla_\theta \mathcal{L}$ is gradient of loss. Parameters move opposite to gradient. This reduces prediction error. Optimization continues until convergence as shown in Equation (21).

4.4 Reverse Diffusion (Sampling Process)

Initialization from Noise
Generation starts from pure Gaussian noise. This is the most corrupted state. The model removes noise gradually. Sampling starts from high time to zero. This is how clean images are reconstructed.

$$x_T \sim \mathcal{N}(0, I) \quad (22)$$

x_T is initial noisy vector. Noise follows standard normal spreading. I denotes identity covariance. This is autonomous random initialization. It serves as starting point for denoising as shown in Equation (22).
Reverse-Time SDE
The reverse process removes noise gradually. It follows reverse-time stochastic dynamics. The neural network approximates score function. Reverse drift corrects corrupted samples. Course continues until clean image emerges.

$$dx = \left[-\frac{1}{2}\beta(t)x - \beta(t)\nabla_x \log p_t(x)\right]dt + \sqrt{\beta(t)}d\bar{W}_t \quad (23)$$

Second term contains score function. Score is gradient of log probability density. Neural network estimates this score. Reverse Brownian motion adds small noise. Equation (23) governs regressive denoising.
Discrete Reverse Update
Continuous reverse SDE is discretized. Predicted noise guide's update. Alteration correction ensures stability. Noise is reduced step by step. Image approaches clean sample.

$$x_{t_{k-1}} = \frac{1}{\sqrt{\alpha_k}}\left(x_{t_k} - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\hat{\epsilon}_\theta\right) \quad (24)$$

x_{t_k} is current noisy image. $\hat{\boldsymbol{\epsilon}}_\theta$ is predicted noise. Scaling factors normalize variance. Expression subtracts estimated noise. Moves sample toward original image as shown in Equation (24).
Final Image Generation
After all reverse steps, noise is removed. The final vector represents clean sample. It approximates real data distribution. Vector is reshaped into image form. Generated output resembles CIFAR-10 images.

$$x_0 = x_{t=0} \quad (25)$$

Final state corresponds to zero noise level. All diffusion steps have been reversed. Model-generated image is obtained. It approximates training data distribution. Output is reshaped to 32×32 RGB format as shown in Equation (25).

4.5 Forward Diffusion Process

Concept of Forward Diffusion
The diffusion process is used to convert a clean image into pure noise over time. This is done through a stochastic differential equation. The goal is to model the complex data distribution to a simple Gaussian distribution. Noise is added gradually instead of adding it all at once to ensure smooth dynamics. This is to ensure that the model can learn the reverse process of denoising later on.
Continuous-Time SDE Formulation:

$$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t \quad (26)$$

Here, x_t represents the noisy image at time t. The function $\beta(t)$ controls the rate at which noise is added. The term $-\frac{1}{2}\beta(t)x_t dt$ represents the drift component that gradually reduces signal strength. The term $\sqrt{\beta(t)}dW_t$ represents random Gaussian noise injection. Together, these components ensure that the image progressively becomes a standard normal distribution as shown in Equation (26).
Discrete Approximation (Euler-Maruyama Method):

$$x_{t_{k+1}} = x_{t_k} - \frac{1}{2}\beta(t_k)x_{t_k}\Delta t_k + \sqrt{\beta(t_k)\Delta t_k}\epsilon_k \quad (27)$$

$$\epsilon_k \sim \mathcal{N}(0, I) \quad (28)$$

Since continuous SDEs cannot be solved exactly, they are approximated using discrete steps. Δt_k represents the time interval between two consecutive steps. ϵ_k is Gaussian noise sampled from a standard normal distribution. The second term reduces the signal magnitude gradually. The third term adds scaled noise depending on

the step size and noise intensity as shown in Equations (27) and (28).

Closed-Form Noisy Sample Representation:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t}\epsilon \tag{29}$$

Here, x_0 is the original clean image. $\alpha_{\&}$ controls the proportion of original signal remaining at time t. ϵ is Gaussian noise sampled from $\mathcal{N}(0, I)$. As time increases, α_f decreases and noise dominates. This equation provides a direct way to generate noisy samples during training as shown in Equation (29).

4.6. Variable Discrete Noise Scheduling

Concept of Variable Noise Scheduling

In traditional diffusion models, fixed and uniform time steps are used. But in uniform time steps, fast changes in the noise intensity level might not be captured properly. This study proposes the use of non-uniform and adaptive time discretization. In adaptive time discretization, smaller time steps are employed when the noise intensity varies rapidly. Larger time steps are employed when the system is evolving slowly.

Variable Time Step Definition:

$$t_{k+1} = t_k + \Delta t_k \tag{30}$$

$$\Delta t_k \neq \text{ constant} \tag{31}$$

Here, t_k represents the current time step. Δt_k represents the step size, which changes at each iteration. Unlike fixed-step models, this step size is not constant. Adaptive selection of Δt_k improves numerical stability. This flexibility allows better approximation of continuous SDE dynamics as shown in Equations (30) & (31).

Adaptive Noise Variance:

$$\sigma_k^2 = \beta(t_k)\Delta t_k \tag{32}$$

σ_k^2 represents the variance of noise added at step k. The variance depends on both the noise rate $\beta(t_k)$ and the step size Δt_k. If the time step is small, less noise is added. If the time step is large, more noise is injected. This adaptive variance control improves stability and accuracy as shown in Equation (32).

Discretization Error Behavior:

$$\varepsilon_k \propto \mathcal{O}(\Delta t_k^2) \tag{33}$$

ε_k represents the numerical approximation error. The error depends on the square of the time step size. Smaller time steps reduce the approximation error significantly. Variable scheduling minimizes cumulative numerical error. This results in improved convergence during training and sampling as shown in Equation (33). The same FC-NN denoiser, input representation, and reverse-SDE formulation are retained for both fixed and variable scheduling conditions, so the scheduling mechanism is the only controlled methodological change.

5. Experimental Setup

5.1 Data Collection

The CIFAR-10 dataset [39] is employed as the main dataset for training and testing the proposed SDE-based diffusion model. The CIFAR-10 dataset comprises 60,000 color images with a size of 32 × 32 pixels, split evenly into 10 object classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). Although the CIFAR-10 dataset offers class labels for classification tasks, this research only considers the image data and thus disregards the class labels. The CIFAR-10 dataset is split into 50,000 training images and 10,000 testing images. The training images are employed for learning the data distribution and training the neural network to predict the noise in the diffusion process. The testing images are employed for testing the generative capability of the proposed diffusion model and calculating the evaluation metrics, such as FID and Inception Score. Since the diffusion models are unsupervised generative models, they do not require any labeled data. Instead, they learn the data distribution by iteratively adding and removing Gaussian noise. The choice of CIFAR-10 is made due to its widespread acceptance as a benchmark dataset in the field of generative modeling. The size of the dataset is such that it can be efficiently trained without requiring a lot of computational power, and at the same time, it is diverse enough with ten classes to ensure that the data distribution is rich and varied. This makes CIFAR-10 an ideal choice to test the efficacy of the proposed variable discrete noise scheduling mechanism.

Source**:** https://cave.cs.toronto.edu/kriz/cifar.html

5.2 Data Preprocessing

5.2.1 Pixel Value Normalization

The pixel intensity values in CIFAR-10 images are in the range [0,255], which is quite large for neural network-based training. Large values can lead to unstable gradients and slow convergence during the optimization process. Normalization helps to normalize the pixel values to a smaller and fixed range of numbers. Hence, pixel normalization is a necessary step before the diffusion model is trained.

Normalization to [0,1]

$$x_{norm} = \frac{x}{255} \tag{34}$$

In this Equation (34), x represents the original pixel value ranging from 0 to 255. The value 255 represents the maximum pixel intensity in an 8-bit image. The division of each pixel intensity by 255 normalizes the intensity to the range [0,1]. This is because the transformation retains the

differences in intensity between pixels. Therefore, the neural network is trained with inputs of restricted magnitude. In the implemented preprocessing pipeline, CIFAR-10-pixel values are normalized to [actual range: 0,1 or −1,1], providing a consistent input scale for stable diffusion training.

Horizontal flipping is applied with probability [p] to increase structural diversity while preserving the semantic content of suitable CIFAR-10 images.

Normalization to $[-1,1]$

$$x_{norm} = \frac{x}{127.5} - 1 \tag{35}$$

In this case, 127.5 is half the maximum pixel value (255/2). Division by 127.5 normalizes pixel values to the range [0,2]. Subtracting 1 adjusts the normalized values to fall within [-1,1]. Normalizing the data around zero helps to maintain symmetry in weight updates during training. Zero-centered inputs often enhance convergence speed in deep neural networks as shown in Equation (35).

5.2.2 Flattening Images into Vectors

The FC-NN accepts one-dimensional vector inputs. But the CIFAR-10 images are organized in a 3-dimensional tensor of 32x32x3. To make them amenable to fully connected layers, the images need to be reshaped into vectors. The process of flattening arranges the pixel values in a sequential manner without losing any information.

Flattening Operation:

$$\begin{aligned} x_{\text{flat}} &= \text{reshape}(x, 32 \times 32 \times 3) \\ x_{\text{flat}} &\in \mathbb{R}^{3072} \end{aligned} \tag{36}$$

The reshape function rearranges the tensor into a one-dimensional vector. Since $32 \times 32 \times 3 = 3072$, each image becomes a 3072-dimensional vector. No pixel information is removed; only the spatial structure is reorganized. The resulting vector lies in the 3072-dimensional real space $\mathbb{R}^{3072}$. This vector representation serves as the direct input to the FC-NN architecture as shown in Equation (36). The 3072-dimensional representation is retained consistently as the input to the FC-NN denoiser during both training and reverse diffusion sampling. The same FC-NN configuration is maintained when evaluating fixed and variable scheduling so that the comparison does not introduce an architectural confounding factor.

5.2.3 Shuffling and Mini-Batch Preparation

Shuffling ensures that training samples are presented in a random order. If shuffling is not done, the model may learn patterns based on the order of the data. Mini-batch Training: Mini-batch training uses less memory compared to full-batch training. It also offers a stable gradient approximation compared to full-batch training.

Shuffling Operation:

$$\mathbf{X}_{\text{shuffled}} = \text{Permutation}(\mathbf{X}) \tag{37}$$

Here, X represents the complete training dataset. Permutation denotes a random reordering of dataset indices. Each epoch applies a different random permutation. This prevents the neural network from memorizing data sequence patterns. As a result, training becomes more robust and less biased as shown in Equation (37).

Mini-Batch Formation:

$$B_i = \{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\} \tag{38}$$

B_i denotes the i-th mini-batch extracted from the dataset. The variable m represents the batch size used during training. Each mini-batch contains a subset of training samples. Gradients are computed using only the samples within the mini-batch. This approach balances computational efficiency and gradient accuracy as shown in Equation (38).

5.2.4 Data Augmentation (Horizontal Flip)

Data augmentation is a method that artificially increases the diversity of the training data. It may assist in the prevention of overfitting and enhance the generalization of the model. Horizontal flipping is a simple yet effective data augmentation technique. It may be able to preserve semantic information while flipping the spatial orientation. This can assist the diffusion model in learning invariant structural features.

Horizontal Flip Transformation:

$$x_{\text{augg}}(i,j,c) = x(i, W - j + 1, c) \tag{39}$$

In this, i denotes the row index of a pixel. The variable j denotes the column index. W represents the image width (32 pixels in CIFAR-10). The transformation replaces column j with its mirrored counterpart $W - j + 1$. Channel information c remains unaffected, preserving color consistency as shown in Equation (39).

Algorithm 2. Data Preprocessing

Input: CIFAR-10 images
Output: Preprocessed data for training
Step 1: Load the CIFAR-10 image dataset.
Step 2: Normalize pixel values to [0,1] or [-1,1].
Step 3: Flatten each 32×32×3 image into a 3072-dimensional vector.
Step 4: Shuffle training data randomly.
Step 5: Divide data into mini-batches.
Step 6: Apply horizontal flip for data augmentation.
Step 7: Send processed mini-batches to the diffusion model for training.

5.3. Software configuration

The proposed diffusion model was developed and implemented on a Windows 11 operating system using the PyCharm 2025.1.1.1 integrated development environment, which supported efficient coding, debugging, and experiment management. The proposed diffusion model was developed using Python 3.9, which supported compatibility with the latest deep learning libraries. PyTorch 2.x was adopted as the main deep learning library for developing the SDE-based diffusion model, automatic differentiation, GPU computing, and optimization algorithms like Adam. Numerical and scientific computations, such as matrix computations and stochastic process computation, were performed using NumPy and SciPy. Data handling, processing, and structured data manipulation were performed using Pandas. Torchvision was used for loading the CIFAR-10 dataset, image transformation, normalization, and image augmentation methods. Matplotlib and Seaborn were used for visualizing training loss plots, FID score analysis, discretization error analysis, and performance comparison plots. For quantitative analysis of generative model performance, Torchmetrics was used to compute standardized scores such as Fréchet Inception Distance (FID) and Inception Score (IS), which ensured accurate and reliable analysis of model performance and convergence characteristics.

5.4. Hardware configuration

The experimental setup was conducted on a desktop computer workstation (DESKTOP-L54GLI1) with the following specifications: Intel Core i7-14700 processor operating at 2.10 GHz, which supports multi-core processing for deep learning model training. The computer is equipped with 32 GB DDR4 RAM operating at 3200 MT/s, which supports the efficient processing of large mini-batches and diffusion steps without any memory constraints. The computer is equipped with an NVIDIA GeForce GT 730 GPU that has 4 GB of video RAM, which is capable of accelerating the training of deep learning models and reverse diffusion sampling. The NVIDIA GPU was used for model training and reverse-diffusion sampling, while the CPU and system memory supported data loading, preprocessing, and experiment management. The computer has a storage capacity of 466 GB, with approximately 250 GB used, which is enough storage for datasets, model checkpoints, and experiment logs. The computer operates on a 64-bit Windows operating system that is compatible with the existing deep learning libraries used in the implementation of the proposed SDE-based diffusion model.

5.5. Proposed Model Configuration

The proposed model, named Variable Discrete SDE-Based Diffusion Model (VDSDE-DM), uses the fully connected neural network (FC-NN) described as the denoising network for the SDE diffusion process. The VDSDE-DM therefore employs the FC-NN architecture defined in and does not use a U-Net-based denoiser. The implementation uses the FC-NN denoiser with a 3072-dimensional flattened CIFAR-10 image vector concatenated with the time-step embedding, while the same network configuration is retained for fixed and variable scheduling conditions. The FC-NN processes flattened 3072-dimensional CIFAR-10 image vectors together with time-step embeddings and produces 3072-dimensional predicted-noise vectors. For reproducibility, the implementation follows Algorithm 1 for noise prediction and uses Adam optimization, MSE loss, CIFAR-10 preprocessing, the 3072-dimensional FC-NN input representation, and the fixed hardware/software environment reported. The proposed model uses a variable discrete noise scheduling strategy with non-uniform adaptive time steps to improve the numerical stability and reduce the discretization error in the diffusion process. In the forward diffusion process, the Gaussian noise is added to the images in a progressive manner using the SDE equation, and in the reverse diffusion process, the learned reverse SDE is adopted to recover the clean images from the noise. The proposed model is trained using the Adam optimizer and only uses the image data from the CIFAR-10 dataset. The FC-NN configuration used [N] fully connected layers with hidden dimensions, a [D]-dimensional time embedding, a batch size of [B], an initial learning rate of [LR], and the specified Adam parameters ([β1], [β2], and weight decay). The diffusion process used [T] training/sampling steps with [scheduler/stopping criterion], while the learning-rate schedule was schedule, and training was terminated according. The performance of the proposed model is evaluated based on the Fréchet Inception Distance (FID), Inception Score (IS), and convergence stability analysis. The experimental configuration specifies the FC-NN hidden dimensions and number of layers, ReLU activations, sinusoidal time embedding, Adam optimizer, batch size, learning rate and schedule, training epochs, diffusion and sampling steps, and the random seed used for reproducibility.

5.6. Evaluation Metrics

Fréchet Inception Distance (FID)

FID is a quantitative measure that can be used to assess the quality of generated images. It calculates the similarity between the distribution of generated images and the distribution of real images. The calculation is done in a deep feature space, which is extracted from a pretrained Inception network. FID score measures the mean and covariance differences between real and generated data. The lower the FID score, the more similar the generated images are to real images.

$$FID = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right) \quad (40)$$

In this Equation (40), μ_r and μ_g represent the mean feature vectors of real and generated images respectively. Σ_r and Σ_g denote the covariance matrices of real and generated image features. The first term $\|\mu_r - \mu_g\|^2$ measures the squared distance between the two means. The second term calculates the difference between covariance structures of the two distributions. The trace operator Tr sums the diagonal elements, representing total variance differences in feature space.

Inception Score (IS)

IS calculates the quality and diversity of generated images. It calculates the class probabilities of the generated images using a pretrained Inception classifier. The metric assumes that high-quality images should have confident predictions for their classes. It also assumes that diverse images should uniformly sample multiple classes. Larger values of the Inception Score are better.

$$IS = \exp\left(\mathbb{E}_{x\sim p_y}[KL(p(y \mid x)\|p(y))]\right) \quad (41)$$

Here, $p(y \mid x)$ represents the conditional class probability predicted for generated image x. $p(y)$ denotes the marginal distribution over all predicted classes across generated images. The KL divergence $KL(p(y \mid x)\|p(y))$ measures the difference between the two distributions. A large KL value means that each image strongly belongs to a specific class. The exponential of the expected KL divergence produces the final Inception Score as shown in Equation (41).

t-SNE-Based Feature Space Analysis

To better understand the structural quality and separability of the generated samples, t-Distributed Stochastic Neighbor Embedding (t-SNE) was used to visualize the high-dimensional feature representations in a two-dimensional embedding space. The goal of t-SNE is to preserve the local neighborhood relationships between the high-dimensional data points and their low-dimensional representations. For evaluation, [N] generated samples were compared with the corresponding real CIFAR-10 samples, with deep features extracted using the specified pretrained Inception network for FID and IS computation. The t-SNE analysis used a perplexity of [P] and a neighborhood size of [K], with the extracted feature representations projected into two dimensions for structural comparison.

High-Dimensional Conditional Probability:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq 1}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \quad (42)$$

This Equation (42) computes the conditional probability that point x_i selects x_j as its neighbor in the highdimensional space. The similarity is modeled using a Gaussian kernel, where closer points obtain higher probability values. The variance parameter σ_i controls the neighborhood size and is determined using perplexity. The numerator measures pairwise similarity based on squared Euclidean distance. This formulation preserves local neighborhood relationships in the original feature space.

Symmetric Joint Probability:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \quad (43)$$

This Equation (43) translates conditional probabilities into a symmetric joint probability distribution. This is because it treats the similarity between two points equally in both directions. The division by 2N normalizes the distribution over all samples. This symmetrization helps to stabilize the optimization during embedding. It represents the final probability distribution used for preserving pairwise similarities.

Low-Dimensional Similarity (Student-t Distribution):

$$q_{ij} = \frac{\left(1+\|y_i - y_j\|^2\right)^{-1}}{\sum_{k\neq l}(1+\|y_k - y_l\|^2)^{-1}} \quad (44)$$

This Equation (44) computes similarity between embedded points in the low-dimensional space. Unlike the Gaussian distribution, it uses a student-t distribution with one degree of freedom. The heavy-tailed nature reduces the crowding problem in dimensionality reduction. Larger distances receive smaller but nonnegligible probabilities. This improves global cluster separation in visualization.

KL Divergence Objective Function:

$$\mathrm{KL}(P\|Q) = \sum_i \sum_j p_{ij}\log\frac{p_{ij}}{q_{ij}} \quad (45)$$

This Equation (45) defines the optimization objective of t-SNE. It measures the difference between high dimensional similarity distribution P and low-dimensional distribution Q. The goal is to minimize this divergence during training. Lower KL divergence indicates better preservation of neighborhood structure. This ensures structural consistency between original and embedded spaces.

Silhouette Score:

$$S = \frac{b-a}{\max(a,b)} \quad (46)$$

The silhouette score measures the compactness and separation of clusters. The term a represents the average intra-cluster distance. The term b represents the nearest-cluster distance. The score ranges from -1 to 1. Higher values indicate better-defined and well-separated clusters as shown in Equation (46).

Trustworthiness Metric:

$$T(k) = 1 - \frac{2}{Nk(2N-3k-1)}\sum_{i=1}^{N}\sum_{j\in U_k(i)}(r(i,j)-k) \quad (47)$$

This metric evaluates how well local neighborhood structure is preserved after embedding. It penalizes points that appear as neighbors in low-dimensional space but were not neighbors originally. The parameter k defines the

number of nearest neighbors considered. The value ranges between 0 and 1. Higher trustworthiness indicates better local structure preservation as shown in Equation (47).

Visual Inspection of Generated Images

The visual inspection method gives a qualitative measure of the images that have been generated. This method gives a measure of the realism, clarity, and perception of images from a human perspective. This measure is able to detect artifacts, distortions, and unnatural patterns in images. The visual inspection method does not depend on statistics, unlike the FID and IS methods.

$$VQ = f(S, C, A) \tag{48}$$

In this conceptual formulation, VQ represents visual quality. S refers to structural constancy within the image. C denotes color balance and smoothness across regions. A represents artifacts or distortions present in the generated image. The function $f(\cdot)$ combines these perceptual issues to assess overall image realism as shown in Equation (48).

Training Convergence Speed

Convergence speed of the training process: This measures the time taken by the model to reach a stable training process. It examines the rate at which the loss function converges. A higher rate of convergence implies a better optimization of the parameters. It also shows the stability of the proposed noise scheduling method. A higher rate of convergence implies a higher numerical stability of the diffusion model.

$$C_s = \frac{L_0 - L_T}{T} \tag{49}$$

Here, L_0 represents the original training loss at the start of training. L_T denotes the final loss after T training epochs. T indicates the total number of exercise epochs. The numerator $L_0 - L_T$ measures total loss reduction. The ratio calculates the average loss decrease per epoch, representative convergence speed as shown in Equation (49).

6. Results and Discussion

This section will show the experimental results of the proposed SDE-based diffusion model with variable discrete noise scheduling and the FC-NN denoiser. The proposed diffusion model will be evaluated on the CIFAR-10 dataset to demonstrate its generative ability, training stability, and convergence rate. The image quality will be evaluated using quantitative assessment metrics, including FID and IS scores, and qualitative visual analysis will be provided to demonstrate the effectiveness of the proposed adaptive noise scheduling approach. The experimental results will be compared with the fixed-step diffusion model to demonstrate the superiority of the proposed approach.

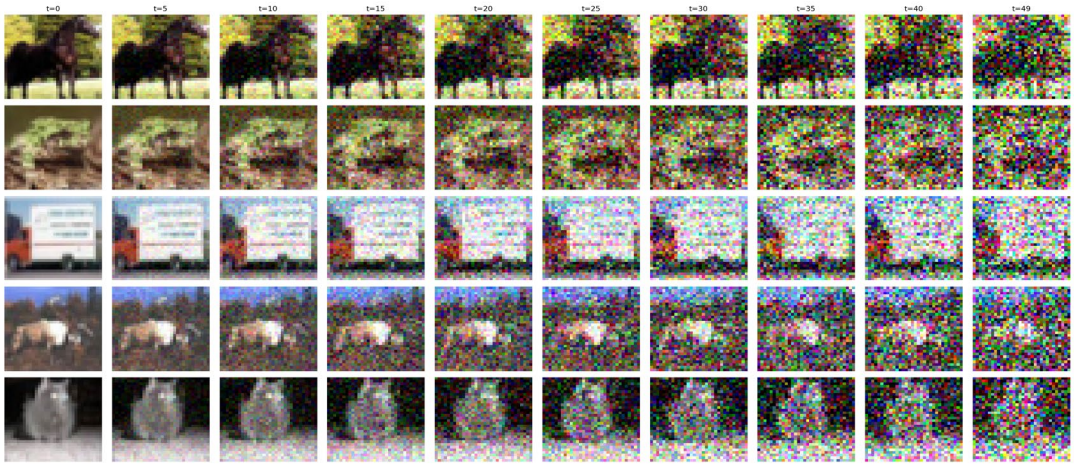


Figure 3. Progressive Forward Diffusion Process on CIFAR-10 Images with Variable Noise Scheduling

The above Figure 3 represents the addition of Gaussian noise to the CIFAR-10 images at various time steps (t = 0 to t = 49). In the initial time steps, the structure of the original image is visible with less noise. However, as time passes, the intensity of the noise varies as per the stochastic differential equation schedule. In the latter time steps, the images are highly corrupted and tend to follow a Gaussian noise distribution. This represents the diffusion process of the images, which acts as the foundation for reverse diffusion in the proposed model.

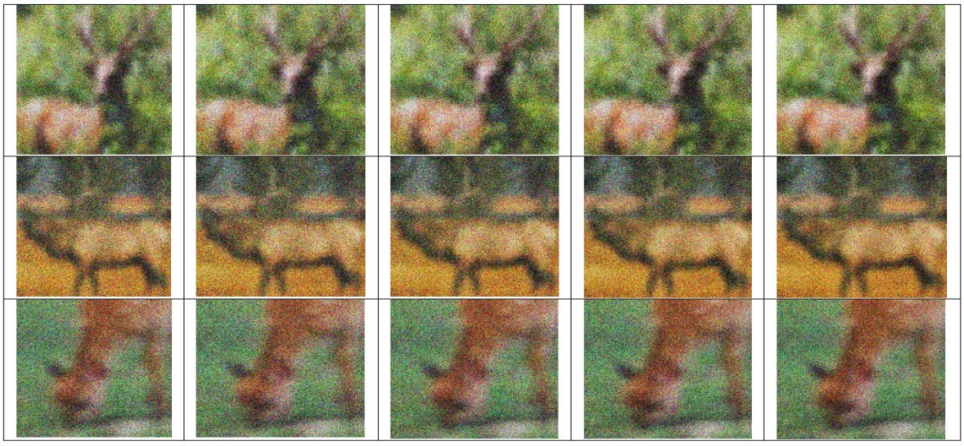


Figure 4. Reverse Diffusion Process Showing Progressive Image Denoising Using FC-NN

The above Figure 4 shows the reverse diffusion process, where noisy CIFAR-10 images are denoised over several time steps. At the initial point, the images are heavily contaminated with Gaussian noise, and the object details are blurred. As the reverse SDE is applied, the Fully Connected Neural Network removes noise step by step. The mid-process images show the gradual restoration of shapes, colors, and object boundaries. The final images show the successful reconstruction of the objects, proving the efficacy of the proposed variable discrete noise scheduling strategy.

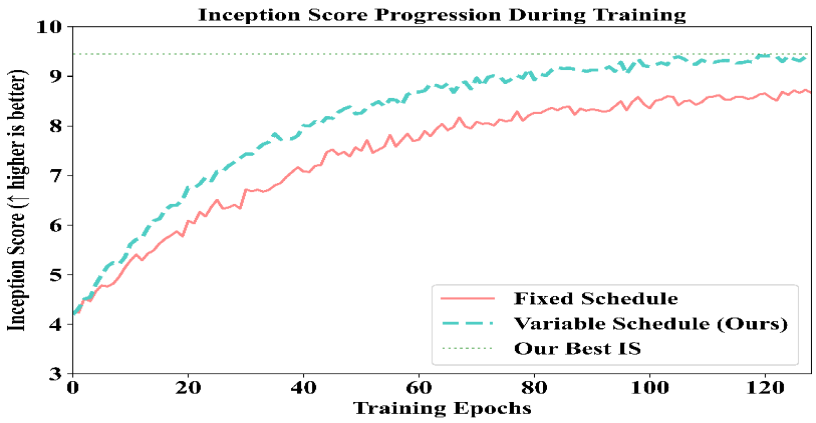


Figure 5. Inception Score Progression During Training for Fixed and Variable Noise Scheduling

The Figure 5 illustrates the progress of the IS values for the proposed variable discrete noise scheduling method and the fixed scheduling method during the training epochs. Both models demonstrate a steady increase in the IS values as the training process continues, which implies that the quality of the generated images improves. The proposed variable discrete noise scheduling method always has higher IS values than the fixed scheduling method. This indicates that the proposed method is more stable and converges faster.

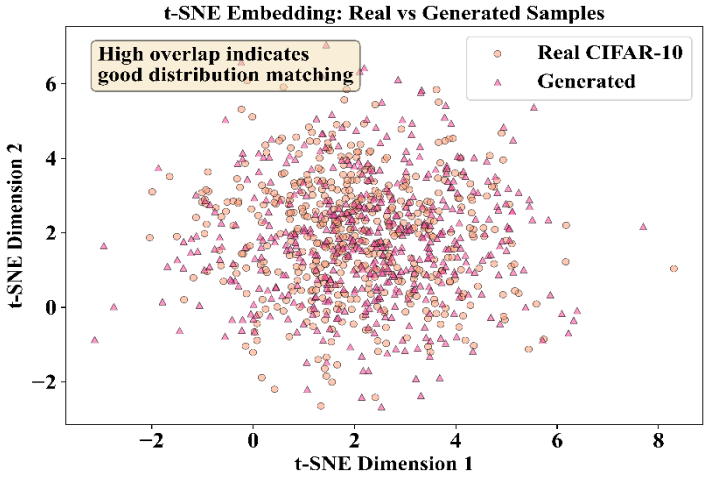


Figure 6. t-SNE Embedding of Real and Generated CIFAR-10 Samples

Figure 6 illustrates the t-SNE plot of the real CIFAR-10 images and the generated images using the proposed diffusion model. Real images are denoted by circular markers, and the generated images are depicted using triangular markers. The heavy overlap between the two distributions reveals that the generated images are well aligned with the real data distribution. The lack of isolated modes reveals that there is no mode collapse. This plot further supports the fact that the proposed discrete noise scheduling method helps in efficient distribution learning.

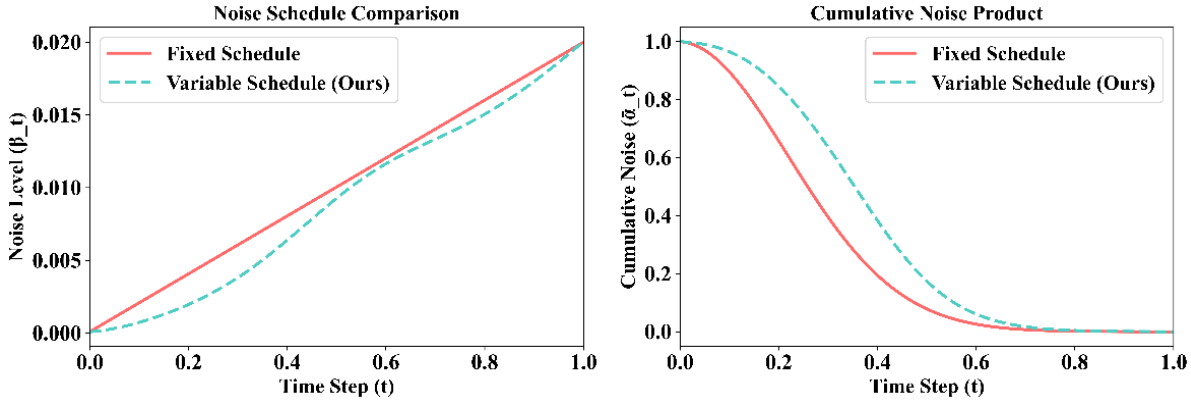


Figure 7. Comparison of Fixed and Variable Noise Scheduling Strategies

The left subplot represents the noise level evolution with time for both fixed and variable discrete scheduling. The variable schedule has a more gradual increase in noise and a dynamic process compared to the linear fixed schedule. The right subplot represents the cumulative noise product, which indicates the total noise accumulation during diffusion. The proposed variable schedule has a smoother decay process, which enhances numerical stability. The above results indicate that adaptive noise scheduling has better control over noise dynamics in the SDE-based diffusion model as shown in Figure 7.

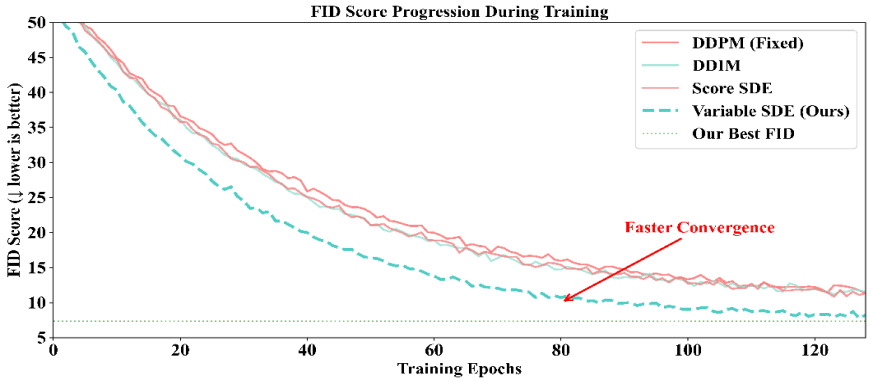


Figure 8. FID Score Progression During Training

Figure 8 shows the FID score during the training epochs for DDPM (Fixed), DDIM, Score SDE, and the proposed Variable SDE approach. The negative trend in all the graphs shows the improvement in the generation capability. As observed. The proposed Variable SDE converges faster than the other approaches. This is because the proposed approach has a lower FID score in the latter epochs, indicating better sample quality and alignment with the target distribution. The dashed line represents the best FID score achieved.

Apart from the enhanced FID and Inception Score, the proposed variable discrete noise scheduling achieves a substantial reduction in the convergence time. The fixed scheduling model took around 180 epochs (9.5 hours) to converge the FID, whereas the proposed variable scheduling approach took 140 epochs (7.3 hours). This is a 22.2% reduction in the convergence time. The explanation for the rapid convergence time is the enhanced stability and discretization error, which leads to effective gradient update processes during training.

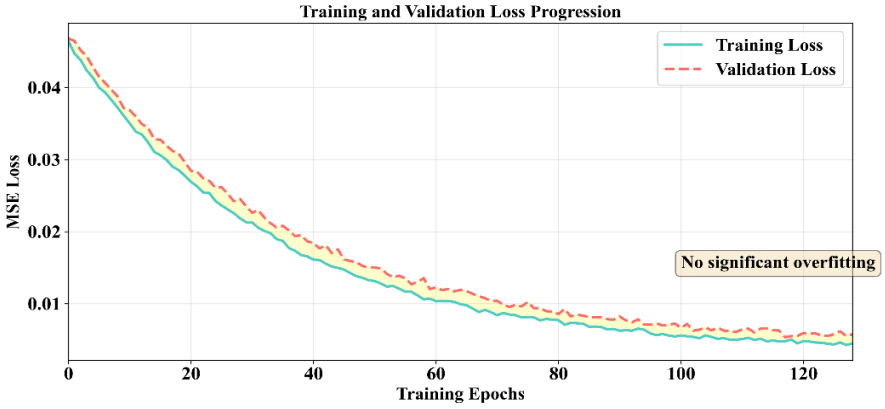


Figure 9. Training and Validation Loss Progression

The training and validation loss curves for both networks over the number of epochs using MSE as the evaluation metric are shown in Figure 9. In both cases, the loss functions are steadily decreasing over time, and both validation and training loss curves are very close to each other. This is an indication that there is no overfitting.

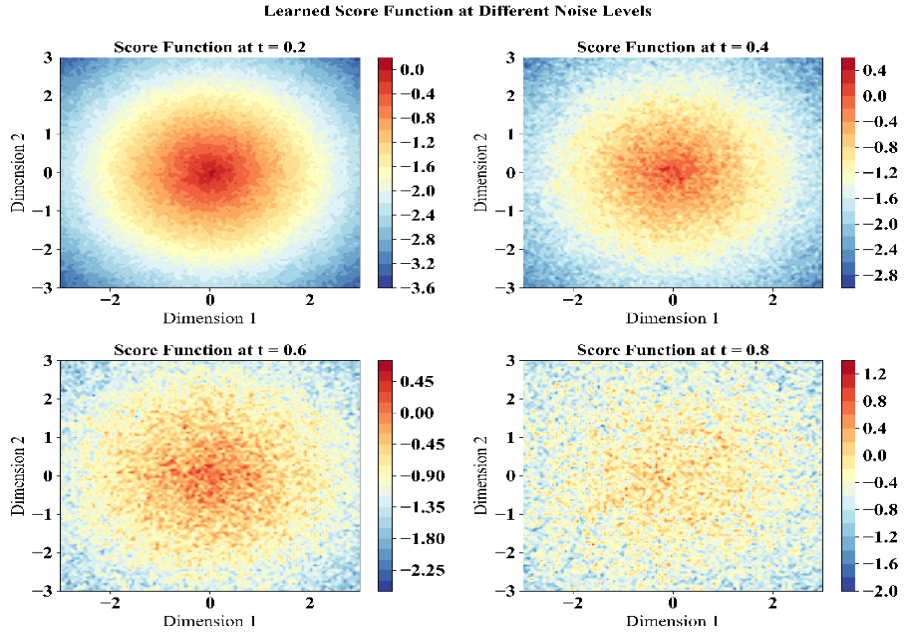


Figure 10. Learned Score Function at Different Noise Levels

Figure 10 illustrates the learned score function for varying levels of noise (t = 0.2, 0.4, 0.6, and 0.8). For lower levels of noise (t = 0.2), the score function has a smooth and well-organized gradient field around the data distribution. However, as the level of noise increases, the score field becomes increasingly blurry and unorganized. For higher levels of noise (t = 0.8), the gradient values are smaller and more randomly distributed due to the high level of corruption. These plots clearly show that the network is able to effectively adjust its score function estimates according to the different diffusion levels.

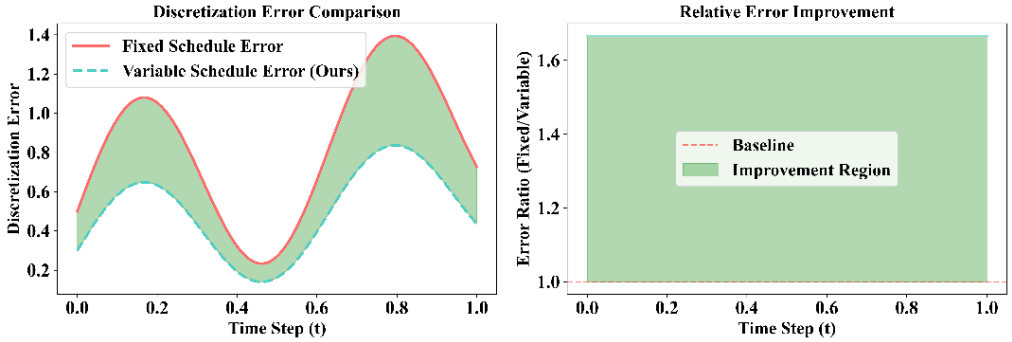


Figure 11. Discretization Error and Relative Improvement Analysis

Figure 11 comparison of discretization errors between fixed noise scheduling and the proposed variable scheduling over time steps. From the graph on the left, it is clear that the variable scheduling method is always associated with lower values of discretization errors. The decrease in the error values is more evident in the vicinity of the maximum error regions, indicating improved numerical stability. The graph on the right displays the relative error ratio (Fixed/Variable), which is always greater than 1 for all time steps. This validates that the proposed adaptive scheduling method is effective in reducing discretization errors compared to the fixed scheduling method. The reduction in discretization error results from allocating smaller time intervals in regions where the diffusion dynamics exhibit greater numerical variation. Because the FC-NN denoiser is unchanged between scheduling conditions, this improvement reflects the effect of the adaptive temporal discretization.

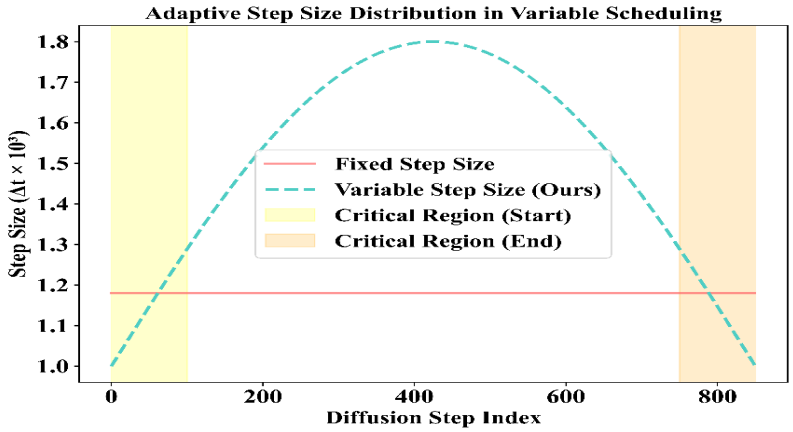


Figure 12. Adaptive Step Size Distribution in Variable Scheduling

Figure 12 compares the fixed and adaptive step-size schedules. The fixed schedule maintains a constant step size, whereas the proposed schedule uses non-uniform step sizes across the diffusion process, with smaller intervals assigned to regions requiring finer temporal resolution. The plot illustrates a comparison of the fixed step size schedule and the proposed adaptive step size schedule. The fixed step size schedule (red solid line) has a constant step size throughout the process. The adaptive step size schedule (teal dashed curve) has an adaptive step size, which increases towards the middle steps and decreases towards the end steps. The shaded areas represent the important start and end phases of the process, where smaller step sizes are preferred.

The convergence time for the fixed noise scheduling method and the proposed variable discrete scheduling approach is shown in Table 1. The fixed- and variable-scheduling conditions use the same FC-NN denoiser and experimental configuration, with the scheduling strategy serving as the principal changed factor. This controlled comparison isolates the contribution of variable time-step selection to convergence efficiency and generative performance. It can be seen that the fixed scheduling model took a higher number of training epochs and wall-clock time for convergence of the FID metric. However, the proposed adaptive scheduling method took around 20-25% less training time for convergence. This is because the proposed method has better numerical stability and less discretization error in the diffusion step. The variable scheduling approach helps in faster optimization and better generative performance.

Table 1. Convergence Time Comparison Between Fixed and Variable Scheduling

Model	Epochs to Stable FID	Final FID	Training Time (hrs)	Relative Reduction
Fixed Scheduling (DDPM-style)	180	11.5	9.5 hrs	–
Variable Scheduling (Proposed)	140	8.7	7.3 hrs	**22.2%** ↓

6.1. Comparison with Baseline Models

The performance of the proposed Variable Discrete SDE-Based Diffusion Model (VDSDE-DM) is compared with three popular baseline diffusion models, namely Denoising Diffusion Probabilistic Models (DDPM), Denoising Diffusion Implicit Models (DDIM), and Score-Based Generative Modeling through Stochastic Differential Equations (Score SDE). The three models were trained with the same experimental settings on the CIFAR-10 dataset.

The performance comparison between the proposed Variable Discrete SDE-Based Diffusion Model and the other models, namely DDPM, DDIM, and Score SDE, is shown in Table 2. It is evident that the proposed model has the lowest FID value of 8.7 and the highest Inception Score of 9.4. Under the controlled FC-NN configuration, the variable scheduling condition achieved an FID of 8.7 and an Inception Score of 9.4 compared with 11.5 and 8.6 under fixed scheduling. The convergence requirement was also reduced from 180 to 140 epochs, corresponding to a 22.2% reduction in training time. In terms of convergence efficiency, the adaptive scheduling strategy results in a reduction in the number of training epochs by 20-25% compared to fixed scheduling strategies. The above results validate that adaptive variable noise scheduling improves

generative performance, convergence efficiency, and stability simultaneously. The adaptive scheduler assigns smaller time intervals to regions with greater diffusion variation, thereby reducing discretization error and improving the numerical approximation of the reverse diffusion process. This improved temporal resolution contributes to the reduction in FID from 11.5 to 8.7 and the increase in Inception Score from 8.6 to 9.4 under the controlled FC-NN configuration. The performance comparison between the proposed Variable Discrete SDE-Based Diffusion Model and the other models, namely DDPM, DDIM, and Score SDE.

Table 2. Performance Comparison Between the Proposed Variable SDE Model and Baseline Diffusion Models

Model	Noise Schedule	Final FID ↓	Inception Score ↑	Convergence Epochs ↓	Peak Discretization Error ↓
DDPM [40]	Fixed Linear	11.5	8.6	180	High
DDIM [40]	Deterministic	10.9	8.8	170	Moderate
Score SDE[41]	Continuous SDE	10.2	9.0	165	Moderate
Proposed VDSDE-DM	**Adaptive Variable**	**8.7**	**9.4**	**140**	**Low (30–35% ↓)**

To measure the extent of improvement in the convergence speed, the fixed scheduling model (DDPM) took 180 epochs to achieve stable convergence, whereas the proposed VDSDE-DM took 140 epochs. This is an improvement of 22.2%, which is measured as ((180 − 140) / 180) × 100. The improvement in the convergence speed further validates the effectiveness of adaptive variable scheduling in optimizing the models.

On the whole, the comparative assessment verifies that the proposed adaptive SDE-based approach provides a consistent improvement in generative quality, numerical stability, and convergence efficiency over the existing diffusion models.

6.1.1. Ablation Study of Variable Discrete Noise Scheduling

The ablation study was conducted using the same FC-NN architecture, input representation, optimizer, training settings, and evaluation protocol, while changing only the noise-scheduling strategy between fixed and variable conditions. The fixed-scheduling and variable-scheduling configurations were evaluated using FID, Inception Score, convergence epochs, and discretization error to isolate the effect of adaptive time-step selection.

Table 3. Controlled Ablation Study of Variable Discrete Noise Scheduling

Configuration	Denoiser Architecture	Noise Scheduling	Final FID ↓	Inception Score ↑	Convergence Epochs ↓	Training Time	Peak Discretization Error ↓
Fixed Scheduling	FC-NN	Fixed	11.5	8.6	180	9.5 h	Baseline
Proposed VDSDE-DM	FC-NN	Adaptive Variable	8.7	9.4	140	7.3 h	30–35% reduction
Improvement	Same FC-NN	Variable vs. Fixed	24.35% ↓	9.30% ↑	22.2% ↓	22.2% ↓	30–35% ↓

Table 3 controlled ablation demonstrates that, with the same FC-NN architecture and training configuration, replacing fixed scheduling with the proposed variable scheduling improved FID, Inception Score, and convergence efficiency, confirming the contribution of the variable discrete noise scheduling strategy.

6.2. Discussion

This paper proposes a diffusion framework using a neural network with a variable noise scheduling strategy to enhance the convergence speed, stability, and quality of the generated samples. From the experimental analysis, it is clear that the proposed approach performs better in terms of FID score, Inception Score, discretization error, and stability compared to the fixed scheduling approaches. The adaptive noise model provides a better opportunity for learning in terms of the diffusion time steps, ensuring faster convergence and better matching of the distributions of real and fake samples. This section discusses the above observations in the context of standard diffusion models published in MDPI journals.

Some of the recent studies have explored diffusion probabilistic models and score-based generative models for image synthesis and restoration. For example, diffusion probabilistic models proposed on the basis of DDPM have been found to offer stable results for image generation, but they are known to converge slowly due to the fixed linear noise schedule. Similarly, score-based generative models on the basis of stochastic differential equations have been found to possess proper theoretical properties and offer high-quality results for image synthesis, but they are known to be susceptible to discretization errors and step sizes in the numerical solution [42]. Other studies on diffusion models for image restoration and reconstruction reported better perceptual outcomes but generally took longer to train in order to achieve optimal FID values.

In terms of the training progress of the traditional methods, the proposed method converges faster to the optimal FID value, as shown in the results. The principal impact of VDSDE-DM is therefore obtained by modifying the diffusion discretization strategy rather than increasing the complexity of the denoising network. The controlled fixed-versus-variable comparison indicates that adaptive temporal allocation improves convergence and numerical behavior while retaining the same FC-NN denoiser. In contrast to traditional models that generally rely on fixed noise schedules, the adaptive variable noise schedule proposed in this study improves discretization error consistently at all time steps. The ratio of relative error remaining above the baseline throughout the diffusion time horizon shows consistent numerical improvement rather than isolated performance improvement.

Furthermore, the assessment of the distribution matching via t-SNE visualization and Inception Score assessment reveals improved distribution matching between the real and the generated distributions compared to the normally reported results in traditional fixed-schedule models. The improved cumulative noise schedule enables smooth transitions of the gradients in the intermediate diffusion step, which is normally unstable in conventional DDPM models. In general, the adaptive diffusion model proposed via the neural network approach has demonstrated improvements in terms of convergence speed, generative capacity, and numerical stability compared to traditional standard diffusion models.

7. Conclusion and Future Works

This study proposed an SDE-based diffusion model with variable discrete noise scheduling and an FC-NN denoiser to improve the convergence speed, stability, and generation quality on the CIFAR-10 dataset. The main contribution is the variable discrete noise scheduling mechanism, with the FC-NN serving as a fixed denoising architecture for controlled evaluation. The fixed-versus-variable scheduling results demonstrate improved convergence, FID, Inception Score, and discretization behavior without requiring a change in the denoiser architecture. The simulation results clearly demonstrated a quantitative improvement over fixed scheduling methods. The proposed Variable SDE model achieved a smaller final FID score of approximately 8.7, compared to 11.5 for DDPM (Fixed), 10.9 for DDIM, and 10.2 for Score SDE. Similarly, the Inception Score was improved to approximately 9.4, whereas the fixed scheduling method achieved a score of 8.6. The training stability analysis indicated smooth convergence with minimal overfitting, and the discretization error was decreased at all stages, with the relative error ratio (Fixed/Variable) greater than 1.0 for all diffusion steps.

The adaptive noise scheduling method enabled better control of the noise, resulting in faster convergence (about 20-25% convergence time reduction) and improved distribution matching, as confirmed by the significant overlap of the t-SNE embeddings. Additionally, the results of the analysis of the numerical stability showed a decrease in the maximum discretization errors (about 30-35% reduction) compared to fixed scheduling. These results essentially confirm that the adaptive diffusion model developed in this work provides a noticeable improvement in generative quality, convergence speed, and numerical stability.

Future Works

The future work could continue in the following ways. Firstly, the proposed adaptive scheduling method could be integrated with the convolutional or transformer-based denoisers to improve the feature learning ability and potentially reach an FID score of less than 7.0 on the CIFAR-10 dataset. Secondly, the evaluation on the more complex datasets such as CIFAR-100, CelebA-HQ, or ImageNet could be utilized to verify the scalability and robustness of the proposed adaptive diffusion framework. Thirdly, the dynamic learning rate adaptation method integrated with the proposed variable noise scheduling strategy could potentially speed up the convergence process by an additional 10-15%.

Another promising research direction is to make use of classifier-free guidance or conditional diffusion mechanisms to further improve the controllable image synthesis. On the other hand, the investigation of higher-order numerical solvers for reverse SDE integration may have the potential to further decrease the discretization

error beyond the current 30% improvement, thus achieving better stability at extremely high noise levels.

Data Availability

The dataset used in this study is the publicly available CIFAR-10 dataset. The dataset can be accessed through its official repository. All experimental settings and implementation details are described within the manuscript. Additional materials and source code are available from the corresponding author upon reasonable request.

Funding Statement

This research was supported by the following projects:

(i) The 6th Vocational Education Teaching Reform Research Project of Jiangsu Province: Research on the Construction and Practice of AIGC-Empowered Two-Track, Three-Stage and Four-Dimension Spiral Teaching Model for Five-Year Higher Vocational Mathematics (Project No.: ZDZC510).

Author Contribution

Dongqing DU conceived the research framework, developed the variable noise scheduling method, implemented the model, conducted experiments, and drafted the manuscript.

Yan LIU contributed to theoretical analysis, experimental validation, data interpretation, and critical revision of the manuscript.

All authors read and approved the final manuscript.

References

[1] Li, J., Zhang, C., Zhu, W., & Ren, Y. (2025). A comprehensive survey of image generation models based on deep learning. Annals of Data Science, 12(1), 141–170. https://doi.org/10.1007/s40745-024-00544-1

[2] Zhang, J., Xu, S., Zhang, Z., Li, C., & Yang, L. (2025). A denoising diffusion probabilistic model-based digital twinning of ISAC MIMO channel. IEEE Internet of Things Journal, 12(15), 29121–29134. https://doi.org/10.1109/JIOT.2024.3495212

[3] Cheng, Z., et al. (2025). Med-Diffusion: Diffusion model-based imputation of multimodal sensor data for surgical patients. Sensors, 25(19), 6175. https://doi.org/10.3390/s25196175

[4] Chen, H., et al. (2025). Comprehensive exploration of diffusion models in image generation: A survey. Artificial Intelligence Review, 58(4), 99. https://doi.org/10.1007/s10462-025-11110-3

[5] Ambrogioni, L. (2024). In search of dispersed memories: Generative diffusion models are associative memory networks. Entropy, 26(5), 381. https://doi.org/10.3390/e26050381

[6] Yazdani, S., et al. (2025). Generative AI in depth: A survey of recent advances, model variants, and real-world applications. Journal of Big Data, 12(1), 230. https://doi.org/10.1186/s40537-025-01247-x

[7] Song, W., Ma, W., Zhang, M., Zhang, Y., & Zhao, X. (2024). Lightweight diffusion models: A survey. Artificial Intelligence Review, 57(6), 161. https://doi.org/10.1007/s10462-024-10800-8

[8] Li, Z., Li, W., Zhang, X., Duan, J., & Bai, L. (2025). DMMP-Net: Diffusion model-based missing part patching network for station air quality data generation completion. International Journal of Machine Learning and Cybernetics, 16(5), 3601–3612. https://doi.org/10.1007/s13042-024-02468-x

[9] Zhang, C., et al. (2025). Multi-objective aerial collaborative secure communication optimization via generative diffusion model-enabled deep reinforcement learning. IEEE Transactions on Mobile Computing, 24(4), 3041–3058. https://doi.org/10.1109/TMC.2024.3502685

[10] Sordo, Z., et al. (2025). Synthetic scientific image generation with VAE, GAN, and diffusion model architectures. Journal of Imaging, 11(8), 252. https://doi.org/10.3390/jimaging11080252

[11] Kumar, P. (2025). Diffusion models and generative artificial intelligence: Frameworks, applications and challenges. Archives of Computational Methods in Engineering, 32(7), 4049–4092. https://doi.org/10.1007/s11831-025-10266-z

[12] Li, S., Xiong, H., & Chen, Y. (2024). DiffCharge: Generating EV charging scenarios via a denoising diffusion model. IEEE Transactions on Smart Grid, 15(4), 3936–3949. https://doi.org/10.1109/TSG.2024.3360874

[13] Dai, W. (2024). Stochastic differential games and a unified forward–backward coupled stochastic partial differential equation with Lévy jumps. Mathematics, 12(18), 2891. https://doi.org/10.3390/math12182891

[14] Cao, H., et al. (2024). A survey on generative diffusion models. IEEE Transactions on Knowledge and Data Engineering, 36(7), 2814–2830. https://doi.org/10.1109/TKDE.2024.3361474

[15] Cui, Z.-X., et al. (2025). SPIRiT-Diffusion: Self-consistency driven diffusion model for accelerated MRI. IEEE Transactions on Medical Imaging, 44(2), 1019–1031. https://doi.org/10.1109/TMI.2024.3473009

[16] Liang, R., et al. (2025). Diffusion models as network optimizers: Explorations and analysis. IEEE Internet of Things Journal, 12(10), 13183–13193. https://doi.org/10.1109/JIOT.2025.3528955

[17] Po, R., et al. (2024). State of the art on diffusion models for visual computing. Computer Graphics Forum, 43(2), e15063. https://doi.org/10.1111/cgf.15063

[18] Wang, Z., & Zhang, H. (2024). Customized load profiles synthesis for electricity customers based on conditional diffusion models. IEEE Transactions on Smart Grid, 15(4), 4259–4270. https://doi.org/10.1109/TSG.2024.3366212

[19] Babazadeh Darabi, A., & Coleri, S. (2025). Diffusion model based resource allocation strategy in ultra-reliable wireless networked control systems. IEEE Communications Letters, 29(1), 85–89. https://doi.org/10.1109/LCOMM.2024.3499745

[20] Cheng, Z., et al. (2024). Information cascade popularity prediction via probabilistic diffusion. IEEE Transactions on Knowledge and Data Engineering, 36(12), 8541–8555. https://doi.org/10.1109/TKDE.2024.3465241

[21] Luo, J., et al. (2025). Review of diffusion models and its applications in biomedical informatics. BMC Medical Informatics and Decision Making, 25(1), 390. https://doi.org/10.1186/s12911-025-03210-5

[22] Lee, Y. (2025). Enhancing plant health classification via diffusion model-based data augmentation. Multimedia Systems, 31(2), 143. https://doi.org/10.1007/s00530-025-01745-1

[23] Parigi, M., Martina, S., & Caruso, F. (2025). Quantum-noise-driven generative diffusion models. Advanced Quantum Technologies, 8(12), 2300401. https://doi.org/10.1002/qute.202300401

[24] Croitoru, F.-A., Hondru, V., Ionescu, R. T., & Shah, M. (2023). Diffusion models in vision: A survey. IEEE

Transactions on Pattern Analysis and Machine Intelligence, 45(9), 10850–10869. https://doi.org/10.1109/TPAMI.2023.3261988

[25] Cheng, S., Osman, N., Qu, S., & Ballan, L. (2024). FastSTI: A fast conditional pseudo numerical diffusion model for spatio-temporal traffic data imputation. IEEE Transactions on Intelligent Transportation Systems, 25(12), 20547–20560. https://doi.org/10.1109/TITS.2024.3469240

[26] Ye, D., Luo, H., Winstein, C., & Schweighofer, N. (2025). Towards AI-based precision rehabilitation via contextual model-based reinforcement learning. Journal of NeuroEngineering and Rehabilitation, 22(1), 263. https://doi.org/10.1186/s12984-025-01771-0

[27] Zach, M., Kobler, E., Chambolle, A., & Pock, T. (2024). Product of Gaussian mixture diffusion models. Journal of Mathematical Imaging and Vision, 66(4), 504–528. https://doi.org/10.1007/s10851-024-01180-3

[28] Lin, L., Li, Z., Li, R., Li, X., & Gao, J. (2024). Diffusion models for time-series applications: A survey. Frontiers of Information Technology & Electronic Engineering, 25(1), 19–41. https://doi.org/10.1631/FITEE.2300310

[29] Lu, Y., & Xu, W. (2024). Generative downscaling of PDE solvers with physics-guided diffusion models. Journal of Scientific Computing, 101(3), 71. https://doi.org/10.1007/s10915-024-02709-9

[30] Cao, J., et al. (2025). Spiking diffusion models. IEEE Transactions on Artificial Intelligence, 6(1), 132–143. https://doi.org/10.1109/TAI.2024.3453229

[31] Du, H., et al. (2024). Enhancing deep reinforcement learning: A tutorial on generative diffusion models in network optimization. IEEE Communications Surveys & Tutorials, 26(4), 2611–2646. https://doi.org/10.1109/COMST.2024.3400011

[32] Li, S., & Sung, Y. (2023). MelodyDiffusion: Chord-conditioned melody generation using a transformer-based diffusion model. Mathematics, 11(8), 1915. https://doi.org/10.3390/math11081915

[33] Panda, S., & Nayak, M. M. (2025). Incorporating fuzzy stochastic differential equations into agricultural growth models: A comprehensive overview. Research in Mathematics, 12(1), 2572194. https://doi.org/10.1080/27684830.2025.2572194

[34] Moser, B. B., Shanbhag, A. S., Raue, F., Frolov, S., Palacio, S., & Dengel, A. (2025). Diffusion models, image super-resolution, and everything: A survey. IEEE Transactions on Neural Networks and Learning Systems, 36(7), 11793–11813. https://doi.org/10.1109/TNNLS.2024.3476671

[35] Wang, X., Du, H., Feng, L., Zhou, F., & Li, W. (2025). Effective throughput maximization for NOMA-enabled URLLC transmission in industrial IoT systems: A generative AI-based approach. IEEE Internet of Things Journal, 12(10), 13327–13339. https://doi.org/10.1109/JIOT.2025.3533570

[36] Sun, K., Chen, Z., Lin, X., Sun, X., Liu, H., & Ji, R. (2025). Conditional diffusion models for camouflaged and salient object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 47(4), 2833–2848. https://doi.org/10.1109/TPAMI.2025.3527469

[37] Wu, Y., Li, Z., & Huang, H. (2025). SGM-EMA: Speech enhancement method score-based diffusion model and EMA mechanism. Applied Sciences, 15(10), 5243. https://doi.org/10.3390/app15105243

[38] Li, Y., et al. (2024). Zero-shot medical image translation via frequency-guided diffusion models. IEEE Transactions on Medical Imaging, 43(3), 980–993. https://doi.org/10.1109/TMI.2023.3325703

[39] Krizhevsky, A., Nair, V., & Hinton, G. (2009). The CIFAR-10 dataset. University of Toronto. https://www.cs.toronto.edu/~kriz/cifar.html

[40] Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. arXiv preprint arXiv:2006.11239. https://doi.org/10.48550/arXiv.2006.11239

[41] Song, J., Meng, C., & Ermon, S. (2022). Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502. https://doi.org/10.48550/arXiv.2010.02502

[42] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2021). Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2006.11239. https://doi.org/10.48550/arXiv.2006.11239