# FEDNet: A Feature-Enhanced Diffusion Network for Efficient and Universal Texture Synthesis

Haichuan Song
East China Normal University
hcsong@cs.ecnu.edu.cn

Xinyi Chen
East China Normal University
51265901075@stu.ecnu.edu.cn

## Abstract

**Texture synthesis aims to generate larger textures resembling an input exemplar, which is crucial for applications like scene rendering, 3D model texturing, and virtual environment design. Despite great advances in the field, texture synthesis remains a challenging task. This is due to the rich space of appearances covered by textures, with varied statistical and spatial characteristics, sometimes varying within the texture itself (e.g. *non-stationary* textures). This paper introduces a feature-enhanced diffusion network (FEDNet), which is trained on pairs of large and small patches cropped from the input. FEDNet ensures both fidelity and diversity through two key techniques: 1) A frequency-aware residual block (FARB) enhances feature extraction during the down-sampling step of the UNet; 2) A feature fusion connection (FFC) integrates spatial histogram layers into skip connections. By capturing the overall structure and texture attributes of the input example, FEDNet can expand the input and its sub blocks after training. Additionally, our model is capable of continuously extending textures through cascading. Extensive experiments demonstrate our method's improved performance over state-of-the-art approaches in both stationary and non-stationary texture synthesis tasks.**

***Keywords:*** ***Image manipulation, Texture synthesis, Diffusion-based generation, Non-stationary texture.***

## 1. Introduction

Texture synthesis refers to the task of generating novel textures that accurately capture the visual features and structural attributes of a given texture exemplar. In the field of computer graphics, this task has been explored in the past two decades [7, 34, 6, 19, 16, 36, 15, 17, 35, 33, 22]. However, existing methods are each suited to specific types of textures, and there is currently no general and effective texture synthesis model for arbitrary textures.

Overall, textures can be categorized as stationary or non-stationary. A texture is considered stationary if its observed appearance remains generally the same within a small moving window, as shown in Figure 1. Conversely, non-stationary textures exhibit varying appearances in different locations at the scale of the same window. The challenge of texture synthesis lies in preserving the pattern and appearance of the exemplar texture while avoiding exact replication and unnatural artifacts. This challenge is particularly evident when dealing with non-stationary textures characterized by irregularity and a lack of translation invariance. As revealed in Figure 1, current methods cannot achieve satisfactory synthesis results for both stationary and non-stationary textures simultaneously and still suffer from the following issues affecting the synthesis quality: 1) inability to preserve the original texture structure; 2) the presence of boundary issues; and 3) the emergence of artifacts.

To tackle these issues, we introduce a texture synthesis model named *Feature Enhanced Diffusion Network* (FEDNet), which demonstrates notable synthesis capabilities across a wide range of texture exemplars. To begin with, we propose frequency-aware residual blocks (FARBs) to enhance the encoder's ability to capture high-frequency content. Inspired by [21], we also redesign the skip connections in the UNet architecture by introducing a trainable histogram layer that incorporates statistical information into the fusion of low-level and high-level feature maps. This effectively combines the traditional feature extraction method with diffusion models to improve the synthesis results. Given a texture exemplar, the network learns the expansion of arbitrary texture blocks extract from the exemplar. After training, the model can quickly generate extended textures that closely resemble the input. Moreover, through cascading, it is possible to continuously produce larger textures.

We evaluate our texture synthesis model using both stationary and non-stationary textures as input exemplars. In the experiments, our method demonstrates an increased versatility in terms of supported appearances as well as a higher quality compared to the state-of-the-art methods. It produces satisfactory texture synthesis results for both stationary textures and more challenging non-stationary textures.

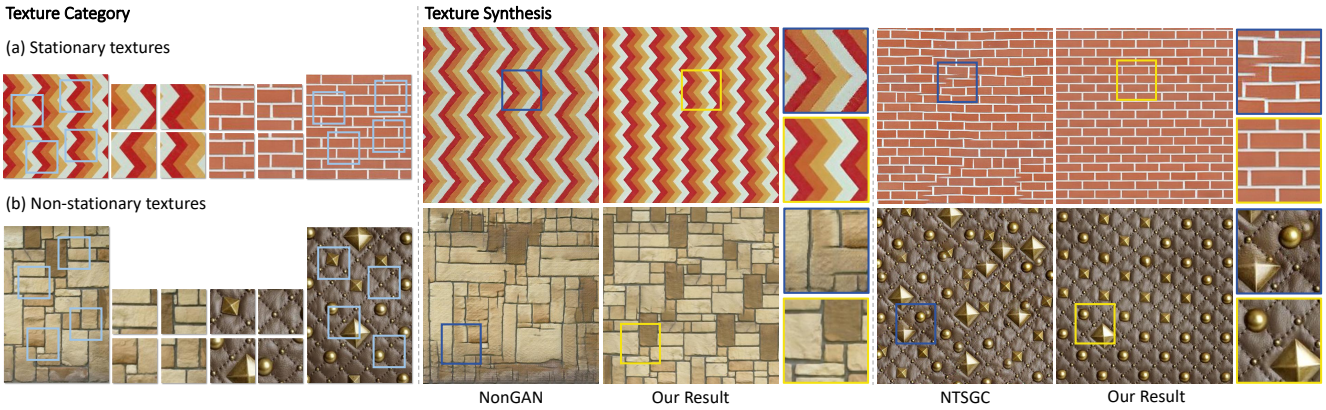


Figure 1. Left: Differences between stationary and non-stationary textures. Middle: Comparison between our results and those of the most advanced GAN-based texture synthesis method [44]. Right: Comparison between our results and those of the state-of-the-art texture synthesis method [42]. It can be observed that the existing deep learning-based methods cannot handle stationary textures well. At the same time, NonGAN suffers from boundary issues, while NTSGC may introduce artifacts. Additionally, both methods fail to preserve the original structure of some textures. In contrast, our method effectively preserves the original texture structure and details, demonstrating excellent visual quality.

## 2. Related Work

**Classical approaches.** The classical texture synthesis approaches can be classified into three categories: pixel-based methods [7, 34], stitching-based methods [6, 19, 16, 22] and optimization-based methods [15, 35, 14]. Self-tuning texture optimization [14] is the state-of-the-art classical method. It uses smart initialization and an additional edge map as the guidance for achieving more coherent structures. Although this method produces plausible results for almost all examples, but it cannot handle textures containing large but unpronounced features. Hu et al. [12] proposed an improved statistical-based method for texture synthesis. This method is capable of extending both structured and stochastic materials without suffering from artifacts seen in traditional texture synthesis approaches, such as structure errors. However, traditional non-parametric methods often limit randomness in the generated texture to preserve regularity. In addition, while classical methods excel at reproducing small-scale structures, they often fail to capture and reproduce the large-scale or global structures, leading to their inadaptability for non-stationary texture synthesis.

**Deep learning-based approaches.** With the development of deep learning, researchers have proposed some methods based on it to address the limitations of classical methods. Gatys et al. [8] introduced pioneering work on texture synthesis using neural networks. This method characterizes textures by learning features from natural images and minimized the difference with the input using optimization procedures, which requires numerous steps to modify the synthesized counterparts, typically taking several minutes even on a modern GPU. Since, several approaches have been introduced to accelerate the inference process. For instance, Ulyanov [31] constructed a feed-forward network for synthesizing target textures, leveraging texture loss defined in [8]. Heitz et al. [9] proposed a textural loss $L_{SW}$, capturing complete feature distributions and proving more robust than the Gram-matrix loss $L_{Gram}$ used in [8].

In the past few years, GAN-based methods [13, 2, 44, 27, 26] have been proposed for texture synthesis tasks and achieved improved performance. Jetchev et al. [13] introduced Spatial GAN (SGAN), which represents the pioneering application of unsupervised GANs in texture synthesis. Based on SGAN, Bergmann et al. [2] developed Periodic Spatial GAN (PSGAN), specifically designed to learn periodic textures from individual texture or datasets. Besides, NonGAN [44] enables the generator to learn how to extend small sample windows to larger texture windows during adversarial training, thereby acquiring the ability to faithfully reproduce local patterns and global structures. SinGAN [26] captures the spatial distribution and interrelationships of patches in texture samples, thereby generating textures with similar visual content. Although these methods are able to handle some challenging non-stationary textures, they often suffer from visual artifacts, especially near boundary areas.

Recently, Zhou et al. [42] proposed an improved CNNMRF model with versatile guided correspondence loss ($\mathcal{L}_{GC}$), denoted as NTSGC throughout this paper. NTSGC combines the Markov random field (MRF) optimization framework with a pre-trained neural network VGG-19. It demonstrates the capability to produce high-quality texture results under both uncontrolled and controlled conditions, surpassing other deep learning-based methods, making it the state-of-the-art in texture synthesis. Deep learning-based methods have also achieved promising results in other texturing tasks, such as guiding texture synthesis [43] or

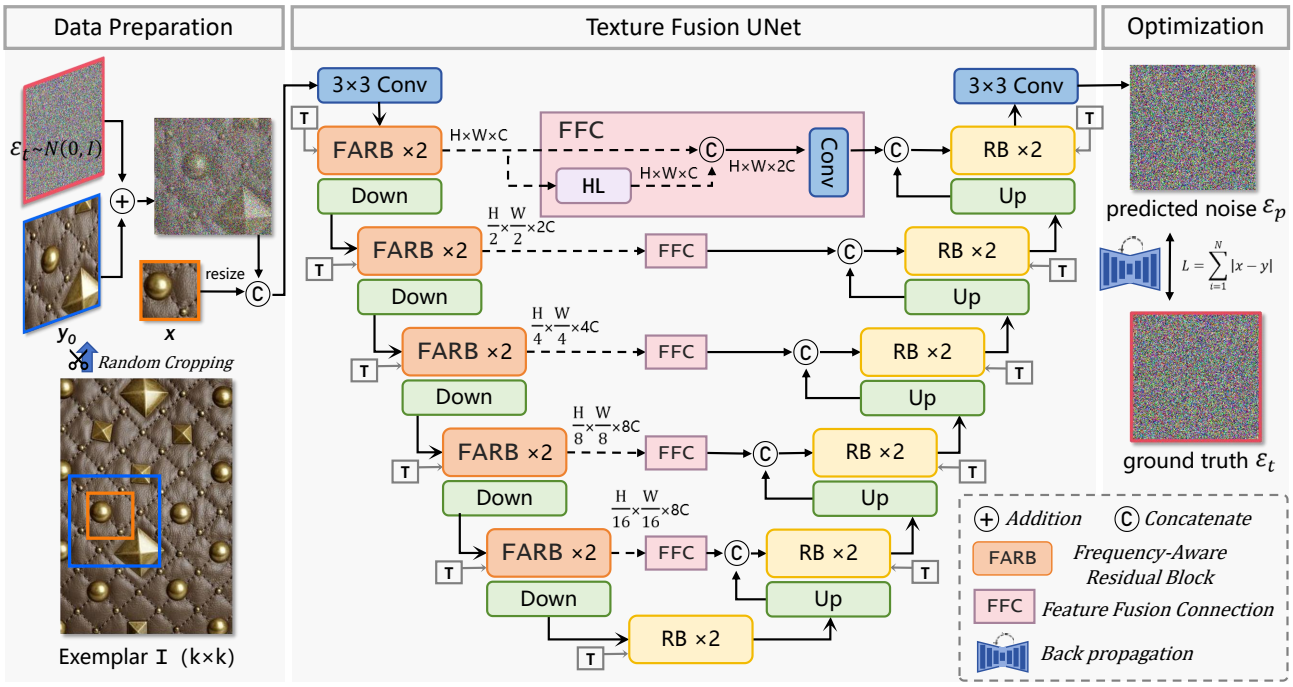


Figure 2. Overall framework: The left part represents the data preparation module, which continuously crops texture pairs from exemplar image $I$. The middle part is the texture fusion UNet structure, designed for the texture synthesis task and used to predict noise. The right part corresponds to the optimization phase, where the model parameters are optimized through back-propagation.

painting texture on a 3D mesh surface [11].

In general, deep learning-based methods are effective at the texture synthesis task. However, when faced with the task of synthesizing non-stationary textures existing methods still tend to produce undesirable results. Meanwhile, compared with classical methods, the reliability of stationary textures synthesis is lower, such as producing noticeable visual seams, as shown in Figure 1 and Figure 7.

**Denoising diffusion probabilistic model.** DDPM was initially introduced by Sohl-Dickstein et al. [28] and promoted by Ho et al. [10]. Due to its robust statistical theoretical foundation and powerful feature processing capabilities, it has been widely applied in various fields, including image generation and restoration [3, 38], 3D reconstruction [30, 37], and natural language processing [1] [5].

## 3. Method

We begin this section with an overview of texture synthesis with denoising diffusion probabilistic model, followed by a detailed explanation of the data preparation process, the designed network architecture and the specifics of model inference.

Our purpose is to construct an end-to-end texture synthesis model for a single texture. Once the model is trained, it can indefinitely generate images that exhibit diversity while retaining the original texture visual characteristics of the input exemplar.

As illustrated in Figure 2, the overall framework comprises three main parts: 1) *Data preparation:* Initially, we utilize two cropping windows to extract blocks from texture $I$, which forms the basis for our training dataset. 2) *Noise prediction:* Subsequently, the UNet predicts the added noise based on the input data. 3) *Optimization:* During this phase, the distance between the predicted noise $\varepsilon_p$ and the ground truth noise $\varepsilon_t$ is calculated using the loss function, and the model parameters are then optimized through back-propagation.

Denoising Diffusion models [10] consist of two fundamental steps: First, during the training phase, noise is progressively added to the input in a process called the noising process. Second, during the inference phase, the model iteratively denoises the Gaussian sample to generate the result.

In our method, the model undergoes training using pairs of data $(x, y_0)$ extracted from the texture $I$. This training process follows a forward diffusion mechanism, wherein Gaussian noise is progressively introduced into the initial input through a fixed Markov chain represented as $q(y_t|y_{t-1})$. Throughout this phase, the model learns to predict noise patterns. Subsequently, during sampling stage, starting with a pure noise image $y'_S$, the model iteratively

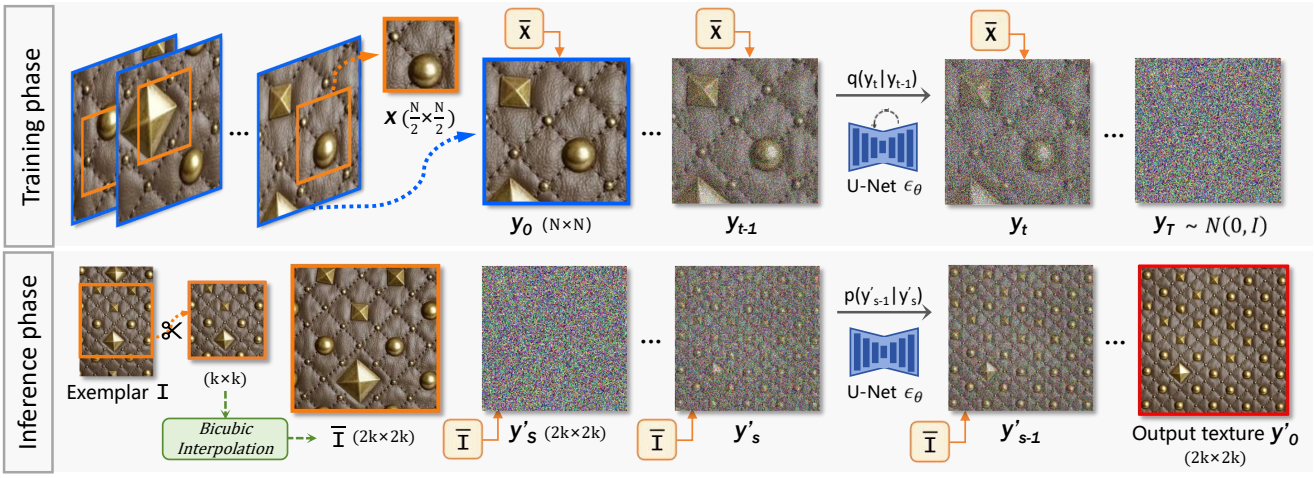


Figure 3. Training and Inference of Conditional Image Generation: The upper part corresponds to the *training phase*, during which the model optimizes its parameters to develop the capability to predict noise. The lower part represents the *inference phase*, during which the model iteratively removes predicted noise and obtains the output texture.

recovers the signal from it, thereby generating the target texture $y'_0$ conditioned on the input texture $\bar{I}$. Texture $\bar{I}$ is derived by upscaling texture $I$ using bicubic interpolation, as illustrated in Figure 3.

### 3.1. Data Preparation

Conditional diffusion models build upon traditional diffusion models by incorporating specific conditions or constraints to guide the generation process. When applied to the texture synthesis task, a small texture block is utilized as conditional information, while a larger texture block serves as the initial image onto which noise is added. Through training, the model gains the capability to perform denoising based on the conditional information. During the inference stage, the model can generate larger textures with similar perceptual qualities to the input texture exemplar.

As illustrated in Figure 2, to obtain the training image pair $(Y, X)$, we randomly crop a patch $y_0$ of size $N \times N$ from the original texture $I$, denoted by the blue square. Subsequently, we crop a smaller patch $x$ of size $\frac{N}{2} \times \frac{N}{2}$ from this initial patch, denoted by the orange square. Here, $N$ represents a configurable hyperparameter. This method is consistent with the approach used in [44]. A large number of diverse training image pairs can be prepared before model training, or can be generated on the fly during the training process.

After the model is trained, we crop the largest central portion of the texture $I$ and use this square image as the input for synthesis. Through model inference, the output texture, i.e., the texture synthesis result, is obtained.

### 3.2. Network Architecture

We adopt a similar network structure as Saharia et al. [24]. This network was proposed to perform super-resolution, a task that is related to texture synthesis as it also increases resolution. We employ a denoising diffusion probabilistic model for conditional image generation, achieving texture synthesis through a random iterative denoising processes.

**Injecting information.** We utilize small texture blocks to act as conditional information for generating larger textures. As shown in SR3 [24], a simple concatenation operation is sufficient to introduce the initial texture into the noise addition and denoising processes. We initially attempted to integrate conditions by incorporating attention mechanisms; however, the generated results were unsatisfactory and this introduced computational overheads.

During the training phase, given a temporal step $t$, the noisy image $y_t$ is obtained by adding Gaussian noise to the original image $y_0$ according to $y_t = \sqrt{\gamma_t} y_0 + \sqrt{1-\gamma_t}\varepsilon_t$, where $\varepsilon_t$ denotes noise sampled from a Gaussian distribution and $\gamma_t$ is the noise scaling factor. We concatenate the guidance information $x$ with the noisy image $y_t$ before utilizing the UNet architecture to predict noise. Throughout the training process, the UNet model continually updates its parameters via the back-propagation, guided by the loss computed between the predicted noise $\varepsilon_p$ and the ground truth noise $\varepsilon_t$.

During the inference phase, we use the resized input exemplar $\bar{I}$ as the guidance information, enabling the model to iteratively denoise from pure noise to produce a result texture $y'_0$.

**Frequency-Aware Residual Block.** In textures, edges and

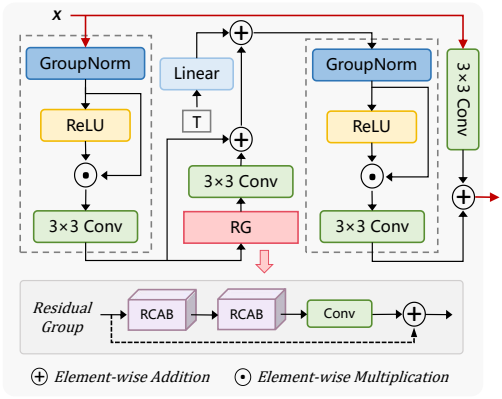


Figure 4. Structure of the frequency-aware residual block.

repeating patterns introduce distinct high-frequency signatures in texture synthesis. Inspired by this, We enhance the model's sensitivity to these high-frequency features and seek to improve the overall synthesis process.

In diffusion models, the encoder in the UNet architecture extracts and compresses features from the input image, playing a pivotal role in accurately predicting noise. Since the network needs to learn the expansion of texture blocks, perceiving and capturing high-frequency information is crucial for the reconstruction of structures and details. However, the high-frequency information in images gradually becomes blurred or lost during the network downsampling, which is disastrous for texture synthesis tasks. To address this issue, we design a specialized frequency-aware residual block to help the encoder better capture high-frequency content.

The FARB module primarily consists of two basic convolution blocks and a residual group (RG) [41], as depicted in Figure 4. The convolution blocks, enclosed within dashed lines, are adapted from the implementation in BigGAN [4]. We implement a self-gating mechanism to enhance features using the Swish function [23], formulated by:

$$f(x) = x \cdot sigmoid(x) = x \cdot \frac{1}{1+e^{-x}}. \quad (1)$$

The residual group consists of residual channel attention blocks (RCABs) and a skip connection. The RCAB draws inspiration from the success of channel attention (CA) and residual block (RB), rescaling the extracted residual features based on the significance among channels, thereby enabling the network to focus on more valuable channels. Meanwhile, the skip connection facilitates faster transmission of low-frequency information through the network, ensuring sufficient network depth to extract important but more scarce high-frequency information.

In our approach, the feature tensor $x$ undergoes processing through the first basic module to yield output $x_{shallow}$, which is then fed into a RG module. The resulting output undergoes a 3 × 3 convolution layer and is summed with $x_{shallow}$ to yield $x_{RG}$.

$$x_{RG} = x_{shallow} + Conv(f_{RG}(x_{shallow})). \quad (2)$$

Subsequently, temporal step, which is an important information in diffusion models, is introduced and fed into the second basic module, resulting in output $x_{deep}$. Finally, the feature tensor $x$ is summed with $x_{deep}$ after passing through another 3 × 3 convolution layer to obtain the output of the FARB module $x_{FARB}$.

$$x_{FARB} = Conv(x) + x_{deep}. \quad (3)$$

FARB enhances the model's ability to capture fine-grained details and complex structures, thereby improving the accuracy of noise prediction and achieving higher-quality synthesized textures.

**Feature Fusion Connection.** Traditionally, handcrafted techniques like local binary patterns (LBP) and histogram of oriented gradients (HOG) are used for feature extraction based on the spatial distribution. With the advancement of deep learning, automated feature learning has replaced handcrafted feature extraction, leading to widespread success. However, deep architectures require a greater number of parameters to capture spatial distribution compared to traditional features, and sometimes cannot demonstrate better or comparable performance in texture analysis. To further enhance deep learning networks, studies [18] [40] [39] have incorporated traditional texture extraction techniques into deep learning models, thereby providing additional intuitive feature information and references. This integration represents a promising direction for the future development of deep learning in texture analysis.

Inspired by this, to further adapt to the texture synthesis task and enhance the model's data expression capabilities, we introduce statistical texture features into skip connections to effectively characterize the distribution of local spatial features. This is achieved by taking into account the histogram of the content.

The histogram is a successful and efficient method for aggregating information. The standard histogram operation counts values within a specified range. However, as an indicator function, its non-differentiable nature prevents the back-propagation of parameter updates in neural networks. To overcome the limitation, Wang et al. [32] employed linear basis functions to enable back-propagation in histogram operations, and Sedighi et al. [25] used radial basis function (RBF) modeling to adapt to steganalysis task. Then Peeples et al. [21] introduced a histogram layer that combines the properties of both methods. They utilized smoother RBFs to represent the histogram structure, allowing the model to estimate bin centers and widths through back-propagation. Their proposed histogram layer effectively captures texture

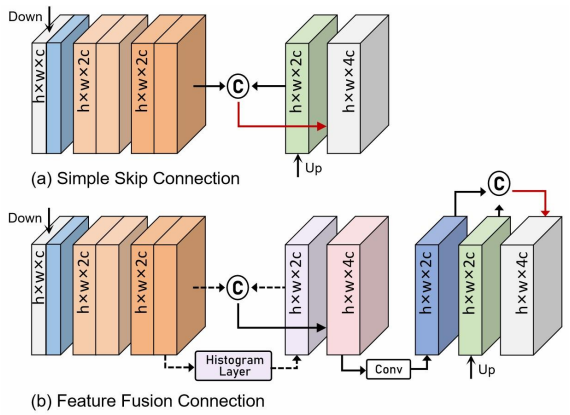


Figure 5. Flowcharts of (a) simple skip connection and (b) feature fusion connection.

information, significantly outperforming other texture encoding methods.

Skip connections in UNet refer to the mechanism of connecting corresponding layers between the encoder (downsampling path) and the decoder (upsampling path). Unlike traditional methods, we do not simply connect the feature maps of encoder and decoder layers. Instead, we introduce a feature fusion connection (FFC) that incorporates the trainable histogram layer [21], aiding the decoder in better utilizing the features extracted from the encoder, as illustrated in Figure 5. The FFC can be defined by

$$f^i_{FFC} = Conv(Concat(f^i_e, HisLayer(f^i_e))), \quad (4)$$

where $f^i_e$ represents the output feature of the i-th layer of the encoder, and $f^i_{FFC}$ represents the feature generated by the FFC.

Experimental results demonstrate that this improvement significantly enhances the detail expression of synthesized textures, while introducing negligible additional computational overhead, as shown in Figure 10.

### 3.3. Sampling Acceleration

In DDPM, inference is defined as a reverse Markovian process, proceesing opposite to the forward diffusion process. This results in the number of steps required to generate the image being the same as T steps in training, leading to a long sampling time.

To reduce the time needed for image generation, we build on the idea of DDIM [29] during the sampling phase. DDIM shares the same training objective as DDPM, but it no longer limits the diffusion process to be a Markov chain, enabling faster generation process with fewer sampling steps. Therefore, by specifying a deterministic step size $S(S \leq T)$, the model can iteratively denoise from a pure noise image $y'_S$ using $S$ steps to achieve the target result $y'_0$, defined as:


Figure 6. Using DDIM significantly improves the sampling efficiency of the model, resulting in a reduction in the time needed to generate a 512×512 texture image from the original 4 minutes to approximately 6 seconds.

$$y'_{s-1} = \sqrt{\gamma_{s-1}}\left(\frac{y'_s - \sqrt{1-\gamma_s}\epsilon_\theta(y'_s, \bar{I}, s)}{\sqrt{\gamma_s}}\right) + \sqrt{1-\gamma_{s-1}-\sigma_s^2}\cdot\epsilon_\theta(y'_s, \bar{I}, s) + \sigma_s\varepsilon, \quad (5)$$

where

$$\sigma_s = \eta\sqrt{\frac{1-\gamma_{s-1}}{1-\gamma_s}}\sqrt{1-\frac{\gamma_s}{\gamma_{s-1}}}. \quad (6)$$

Here, $\gamma$ represents the cumulative $\alpha$ defined in [29], and $\epsilon_\theta$ is the noise prediction network conditioned on the current input $y'_s$, control condition $\bar{I}$, and timestamp $s$.

Using this method, we reduce the number of sampling steps from 2000 to 50 and the generation time on an NVIDIA GeForce RTX 2080 Ti from the original 4 minutes to 6 seconds. The acceleration effect is depicted in Figure 6.

## 4. Experiments

Our approach is implemented using PyTorch. For comparison with other methods, we collected 50 texture images from [44], resizing all input textures to 256×256 pixels, resulting in textures of 512×512 pixels. Our model is trained on an NVIDIA GeForce RTX 2080 Ti with 11GB memory, and the training time is about three and a half hours. Once the training is completed, the model takes 6 seconds to double the width and height of the 256 x 256 texture through denoising. The input textures include non-stationary texture samples with irregular and non-uniform structures, where our method successfully captures and extends the global structures presented in the input examples. Additionally, our method is also applicable to more stationary textures, including regular or random structured textures.

We compare our method against six representative approaches in Figure 7. Self-tuning texture optimization [14] is the state-of-the-art classical method, employing smart initialization and an additional edge map for guiding more coherent structures; CNNMRF [18] explores a combination

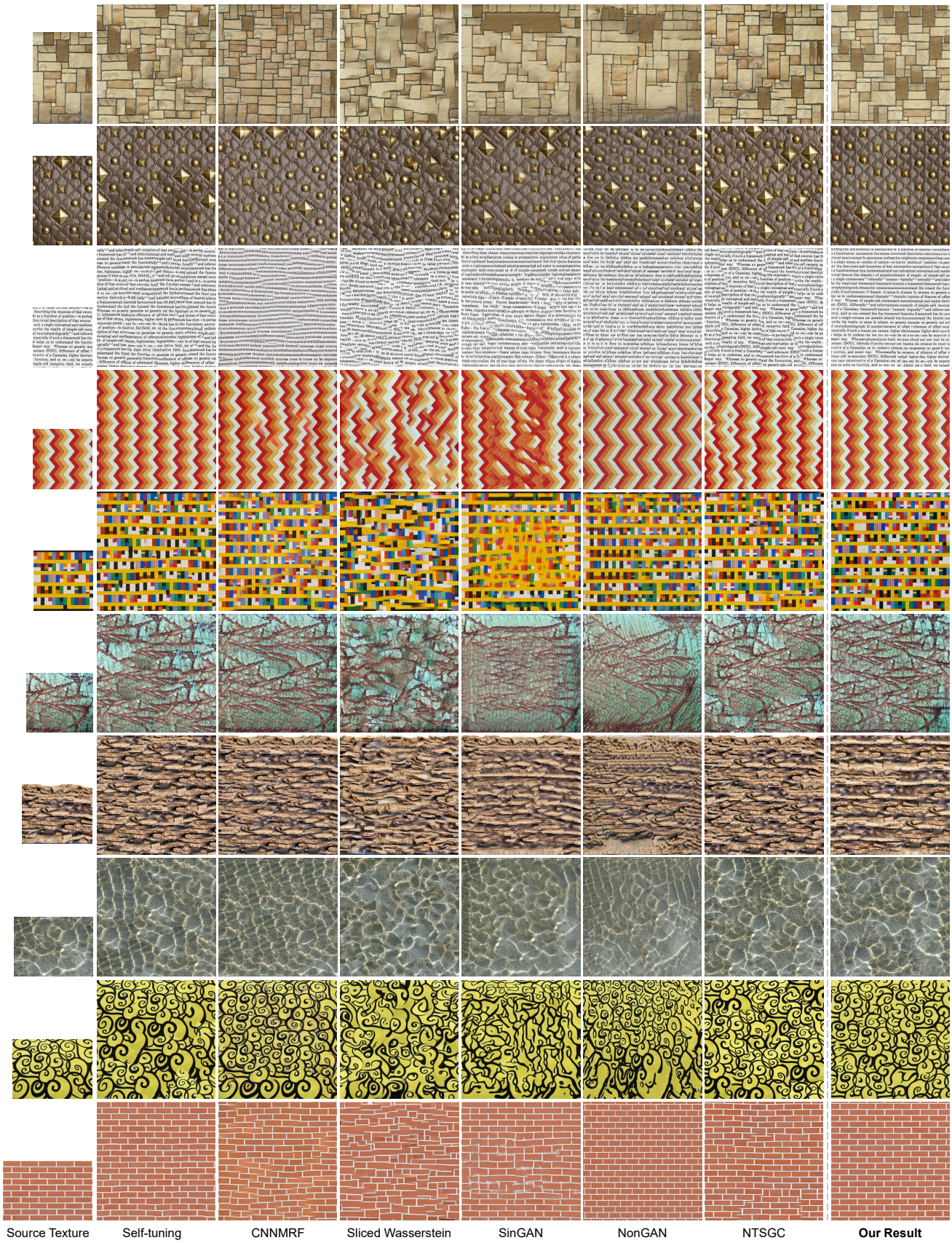


Figure 7. Qualitative comparison between our method and six state of the art methods. For each source texture, the results shown from left to right are respectively generated by Self-tuning [14], CNNMRF [18], Sliced Wasserstein [9], SinGAN [26], NonGAN [44], NTSGC [42] and our method.

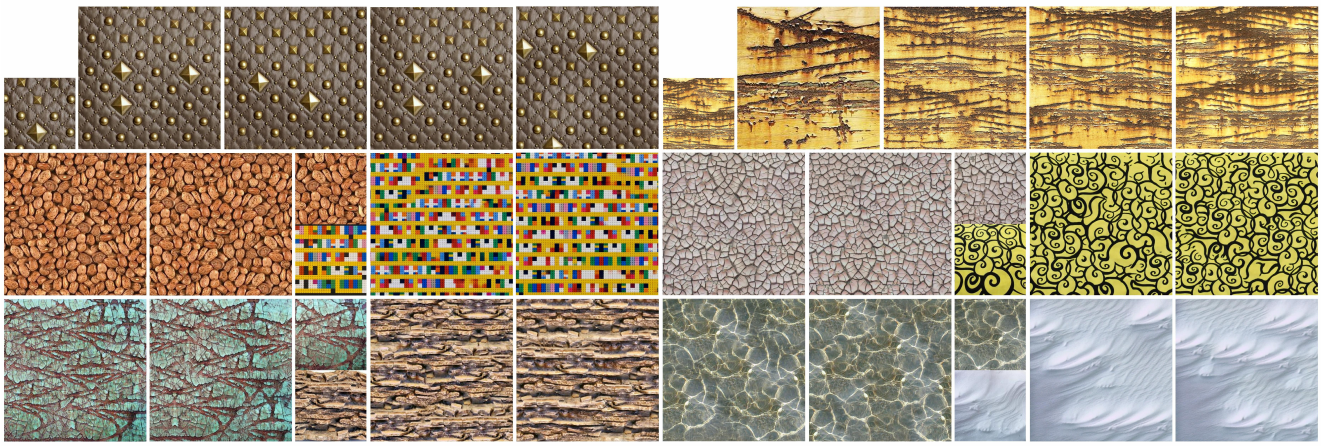

Figure 8. Our model generates texture images that exhibit both high-quality and diversity. In the first row, for the two source textures, four diverse synthesis results are shown respectively. More results are displayed in the second and third rows.

| | Self-tuning | CNNMRF | Sliced Wasserstein | SinGAN | NonGAN | NTSGC | **Ours** |
|---|---|---|---|---|---|---|---|
| Human Eval. | 28.6% | 15.7% | 10.4% | 6.1% | 13.2% | 31.4% | - |
| Color Dis. | 0.649 | 2.305 | 2.331 | 1.566 | 2.273 | 0.928 | 1.586 |

Table 1. Quantitative comparison between our method and six state of the art methods. Human Evaluation and the Color Distance between the source and synthesized textures can serve as reference metrics for assessing the synthesis quality.

of generative Markov random field (MRF) models and discriminatively trained deep convolutional neural networks (dCNNs) for synthesizing 2D images. At the same time, CNNMRF is the baseline model of NTSGC; Sliced Wasserstein loss [9], the state-of-the-art statistic-based textural loss, captures complete feature distributions; SinGAN [26] and NonGan [44] are two state-of-the-art GAN-based models for single-image texture synthesis tasks. NTSGC [42] combines MRFs and neural networks and achieves remarkable synthesis results.

### 4.1. Generation Quality

As shown in Figure 8, our model can synthesize diverse texture images while maintaining the structure of the input exemplar. This shows the strong ability of the model to produce diverse and high-quality textures, thereby better meeting the wide range of user needs.

In Figure 7, it can be observed that our results exhibit favorable visual outcomes across all examples versus other approaches. Compared to CNNMRF and NTSGC, our synthesis results consistently exhibit greater clarity, fewer artifacts, and, importantly, better texture structures. The results synthesized by Sliced Wasserstein appear highly disordered. SinGAN and NonGAN, on the other hand, suffer from boundary issues, where repetition and poor synthesis effects occur at the boundaries, as observed in the sixth row of Figure 7. While self-tuning yields reasonable results in almost all examples, ours are comparatively improved in terms of reproducing small scale features (less wavy) and large scale structures and regularities.

In the field of texture synthesis, the quality of synthesis is evaluated primarily by visual observation, and there is currently no universal quantitative evaluation metric or standard benchmark. We refer to the quantitative comparison methods used in [42], and similarly conduct subjective image quality assessment and color distance calculation.

In subjective quality evaluation, the forced-choice pairwise comparison method is considered to be more reliable [20]. In our study, we presented three images simultaneously to the participants: the source texture, the synthesized texture from our method, and the synthesized texture from one of the comparison methods. Participants were required to choose the synthesized texture they perceived to be more plausible. This approach allows for a direct and intuitive comparison between different synthesis methods. We invited 50 participants from different fields to take part in the experiments. The human evaluation metric presented in Table 1 represents the proportion of the results of the comparison model that were selected. For example, 28.6% indicates that when comparing textures generated by the self-tuning method and our method, participants considered the quality of the former to be better in 28.6% of cases, while the percentage of our method being preferred is 71.4%. The results of the Human evaluation indicate that our method consistently outperformed others in terms of visual quality. This can be attributed to our method's ability to generate plausi-

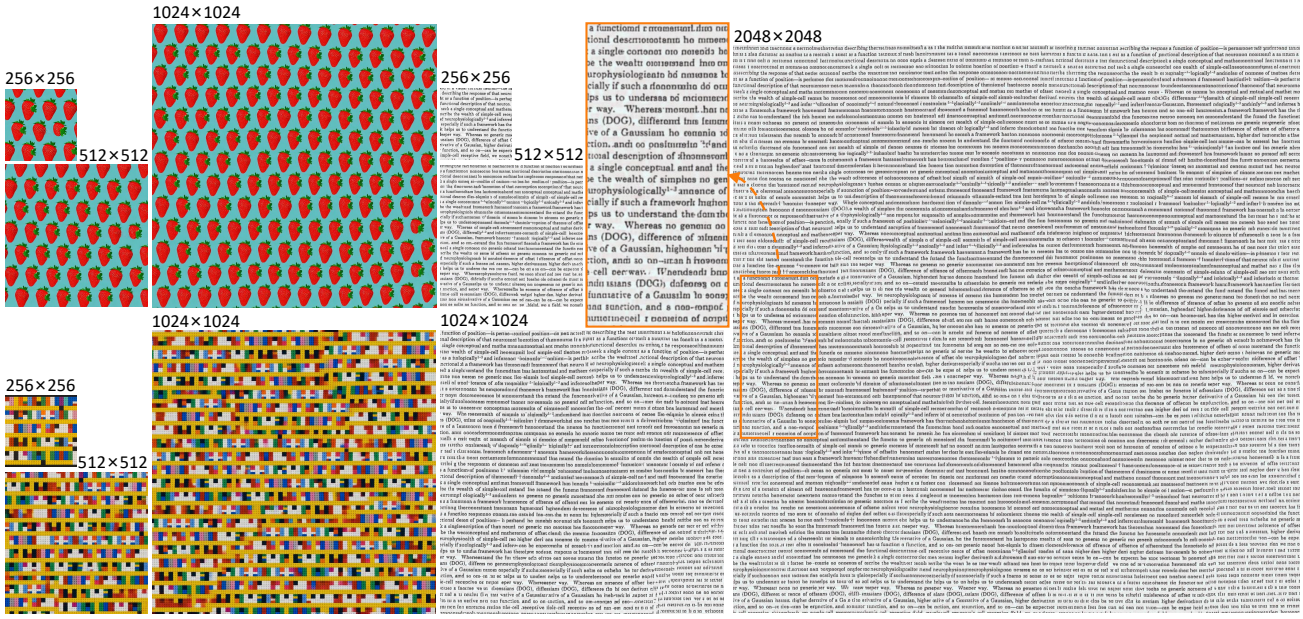


Figure 9. Synthesizing 2×, 4× and 8× textures through model cascading.

ble texture structures while preserving the original texture style, ensuring that the synthesized results are visually appealing and structurally coherent.

Traditionally, texture synthesis techniques used a best-matching neighborhood objective to optimize for textures from examples [34]. We thus use this metric as an additional indicative measure of quality. Specifically, the output texture was partitioned into small blocks of size 10x10, denoted as $R_i$, where $i = 1, 2, ..., N$ and $N$ represents the total number of blocks. For each resulting block $R_i$, we traversed the small blocks of size 10x10 in the source texture, denoted as $S_j$, where $j = 1, 2, ..., M$ and $M$ is the total number of blocks in the source texture. We then computed the squared Euclidean distances between the corresponding pixels of $R_i$ and each $S_j$:

$$D_{ij} = \sum_{(x,y)\in R_i} \|R_i(x,y) - S_j(x,y)\|^2. \quad (7)$$

Here, $R_i(x,y)$ and $S_j(x,y)$ represent the color values at position $(x,y)$ in $R_i$ and $S_j$, respectively. The minimum value of $D_{i,j}$ for each $R_i$ was selected as the nearest neighbor distance:

$$D_i^* = min_j D_{ij}. \quad (8)$$

Finally, the average of these nearest neighbor distances was calculated to serve as the overall color distance metric between the source and synthesized textures:

$$\tilde{D} = \frac{1}{N}\sum_{i=1}^{N} D_i^*. \quad (9)$$

This method effectively evaluates the color distance between the two textures, and the experimental results are shown in Table 1. It can be observed that our method demonstrates a certain improvement over some existing approaches. While the color distance is related to synthesis quality, it is merely a statistical measure of pixel color values and does not account for the patterns and structural coherence of the generated textures. Therefore, it can only serve as an indication in evaluating the quality of the generated textures.

### 4.2. Efficiency Assessment

As illustrated in Figure 6, the utilization of DDIM during the inference phase notably reduces the time required to generate textures while preserving both diversity and high quality.

In terms of processing time, the self-tuning method [14] takes approximately 4 minutes per result. NTSGC[42] combines MFR with deep learning and utilizes pre-trained networks to extract multi-layer features. Therefore, it does not require self-training of the network and takes only 5 minutes for each sample. Among deep learning-based methods, Ulyanov et al. [31] spend 1 hour training each sample using texture networks; PSGAN [2], NonGAN [44] and InGAN [27] require 12, 5 and 4.5 hours, respectively, to train on an image of the same size. The training time of our approach is amongst the fastest of the deep learning-based methods, even though it is significantly slower than that of classical methods and methods that use pre-trained models. Additionally, its ability to handle a wide range of textures and produce high quality results compensates for its slower speed.

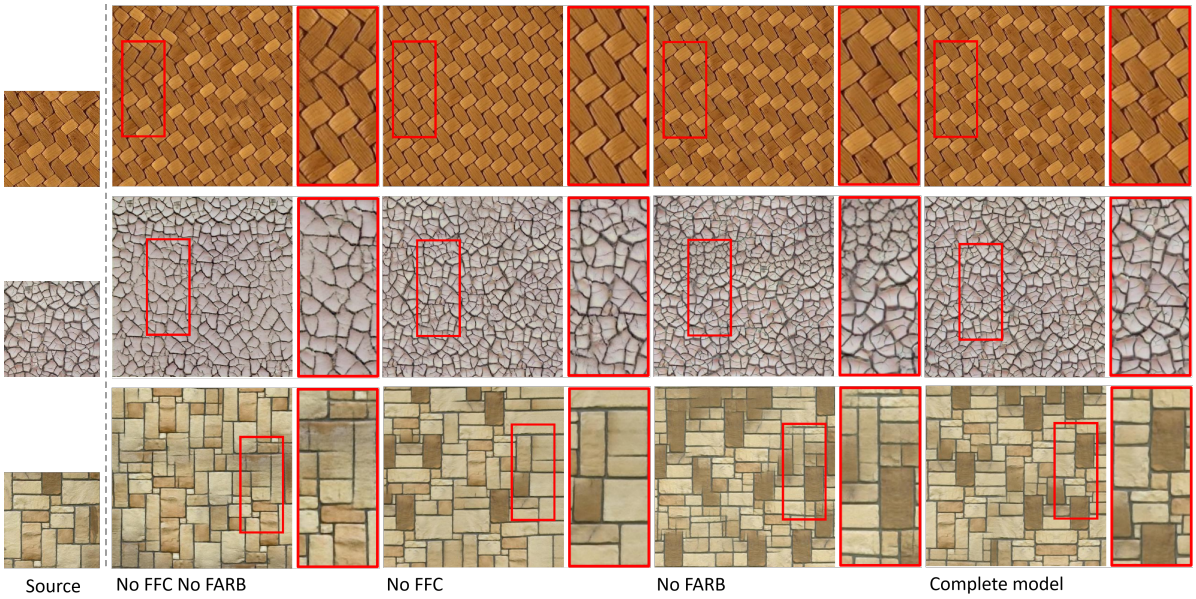


Figure 10. Qualitative results of ablation study on different network components.

### 4.3. Texture Expansion

Our model demonstrates a robust capability for continuous texture expansion. For 4× synthesis, we use the trained model for 2× synthesis twice. Beginning with a 256×256 input image, we initially generate a 512×512 texture, which is subsequently used to produce a 1024×1024 texture.

Given the increased memory demands for generating larger textures, we use an NVIDIA GeForce RTX 3090 GPU with 24GB memory for this experiment. The sampling time for 512×512, 1024×1024 and 2048×2048 images are 4 seconds, 15 seconds and 1 minute, respectively. Moreover, this approach enables us to achieve 8× synthesis and 16× synthesis. The generated results are presented in Figure 9. It can be observed that for the texture containing text, the model consistently produces desirable and high-quality outputs, even when magnified by a factor of 64 (8 times longer and 8 times wider).

### 4.4. Ablation Study

To further demonstrate the effectiveness of each proposed component, we conduct ablation studies using four configurations: 1) the initial network architecture; 2) the network without the FFC module; 3) the network without the FARB module; and 4) the complete network (including both FARB and FFC) to obtain synthesis results.

As shown in Figure 10, the initial network architecture is less suitable for the texture synthesis task, failing to capture the global structure and local patterns of the input. The generated results exhibit obvious artifacts and visual seams. The introduction of the FARB enhances the model's capabilities in extracting and optimizing features, particularly in discerning high-frequency features, thereby improving the pattern and appearance of the results. The integration of FFC captures the local spatial distribution of shallow features in the encoder, optimizing feature connection between the encoder and decoder. As a result, the model generates images with more accurate texture details. Finally, the complete model effectively integrates the capabilities of both methods, resulting in the highest quality across overall structure, element distribution, and detail expression.

## 5. Discussion and Conclusion

In this work, we propose the feature-enhanced diffusion network (FEDNet), which tackles the complex challenges of synthesizing both stationary and non-stationary textures. The texture fusion UNet is designed with effective frequency-aware residual blocks and feature fusion connections. The former avoids the high-frequency information being blurred during downsampling, while the latter incorporates traditional histogram feature into the skip connections. Experiments show that our method outperforms existing state-of-the-art techniques in the synthesis of challenging cases of both stationary and non-stationary textures.

The limitation of our technique is the influence of the cropping window size. We observed that different textures have different ideal cropping window sizes, and that using too large or too small sizes can be detrimental to the results as shown in Figure 11. This variance can be attributed to the diverse element sizes and structures inherent to textures. A common pre-processing step before training texture images involves resizing them to conform to the model's specifications. However, this step may introduce additional infor-

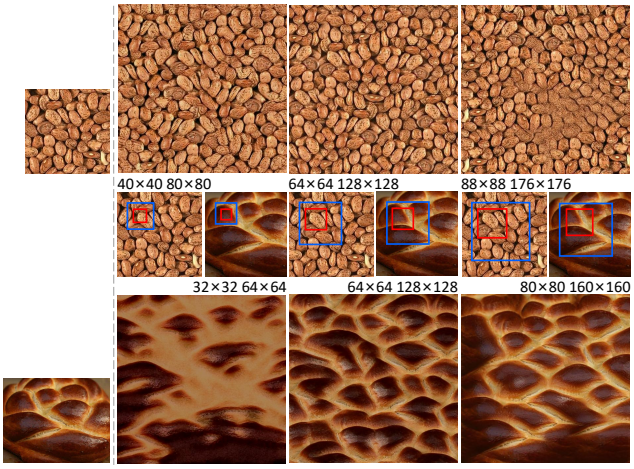


Figure 11. Results obtained using different cropping window sizes for texture images. It can be observed that using excessively large or small cropping windows leads to undesirable outcomes.

mation or lead to information loss. Therefore, we advocate using different cropping window sizes to adapt to the size and structure of individual textures. Although we currently use the fixed window size, which is 64 x 64 and 128 x 128 pixels, our framework still demonstrates excellent texture synthesis performance. Determining the most appropriate cropping window size for diverse textures will be the focal point of our future research.

## References


[1] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021. 3

[2] U. Bergmann, N. Jetchev, and R. Vollgraf. Learning texture manifolds with the periodic spatial gan. *arXiv preprint arXiv:1705.06566*, 2017. 2, 9

[3] J. Betker, G. Goh, L. Jing, T. Brooks, J. Wang, L. Li, L. Ouyang, J. Zhuang, J. Lee, Y. Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023. 3

[4] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 5

[5] T. Chen, R. Zhang, and G. Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022. 3

[6] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001. 1, 2

[7] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999. 1, 2

[8] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, 28, 2015. 2

[9] E. Heitz, K. Vanhoey, T. Chambon, and L. Belcour. A sliced wasserstein loss for neural texture synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9412–9420, 2021. 2, 7, 8

[10] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3

[11] A. Hu, N. Desai, H. Abu Alhaija, S. W. Kim, and M. Shugrina. Diffusion texture painting. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024. 3

[12] Y. Hu, C. He, V. Deschaintre, J. Dorsey, and H. Rushmeier. An inverse procedural modeling pipeline for svbrdf maps. *ACM Transactions on Graphics (TOG)*, 41(2):1–17, 2022. 2

[13] N. Jetchev, U. Bergmann, and R. Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016. 2

[14] A. Kaspar, B. Neubert, D. Lischinski, M. Pauly, and J. Kopf. Self tuning texture optimization. In *Computer Graphics Forum*, volume 34, pages 349–359. Wiley Online Library, 2015. 2, 6, 7, 9

[15] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. In *ACM SIGGRAPH 2005 Papers*, pages 795–802. 2005. 1, 2

[16] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *Acm transactions on graphics (tog)*, 22(3):277–286, 2003. 1, 2

[17] S. Lefebvre and H. Hoppe. Appearance-space texture synthesis. *ACM Transactions on Graphics (TOG)*, 25(3):541–548, 2006. 1

[18] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2479–2486, 2016. 5, 6, 7

[19] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20(3):127–150, 2001. 1, 2

[20] R. K. Mantiuk, A. Tomaszewska, and R. Mantiuk. Comparison of four subjective methods for image quality assessment. In *Computer graphics forum*, volume 31, pages 2478–2491. Wiley Online Library, 2012. 8

[21] J. Peeples, W. Xu, and A. Zare. Histogram layers for texture analysis. *IEEE Transactions on Artificial Intelligence*, 3(4):541–552, 2021. 1, 5, 6

[22] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *2009 IEEE 12th international conference on computer vision*, pages 151–158. IEEE, 2009. 1, 2

[23] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 5

[24] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. Image super-resolution via iterative refinement.

*IEEE transactions on pattern analysis and machine intelligence*, 45(4):4713–4726, 2022. 4

[25] V. Sedighi and J. Fridrich. Histogram layer, moving convolutional neural networks towards feature-based steganalysis. *Electronic Imaging*, 29:50–55, 2017. 5

[26] T. R. Shaham, T. Dekel, and T. Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4570–4580, 2019. 2, 7, 8

[27] A. Shocher, S. Bagon, P. Isola, and M. Irani. Ingan: Capturing and retargeting the" dna" of a natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4492–4501, 2019. 2, 9

[28] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 3

[29] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 6

[30] J. Tang, T. Wang, B. Zhang, T. Zhang, R. Yi, L. Ma, and D. Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22819–22829, 2023. 3

[31] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *arXiv preprint arXiv:1603.03417*, 2016. 2, 9

[32] Z. Wang, H. Li, W. Ouyang, and X. Wang. Learnable histogram: Statistical context features for deep neural networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 246–262. Springer, 2016. 5

[33] L.-Y. Wei, J. Han, K. Zhou, H. Bao, B. Guo, and H.-Y. Shum. Inverse texture synthesis. In *ACM SIGGRAPH 2008 papers*, pages 1–9. 2008. 1

[34] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, 2000. 1, 2, 9

[35] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Transactions on pattern analysis and machine intelligence*, 29(3):463–476, 2007. 1, 2

[36] Q. Wu and Y. Yu. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics (TOG)*, 23(3):364–367, 2004. 1

[37] R. Wu, B. Mildenhall, P. Henzler, K. Park, R. Gao, D. Watson, P. P. Srinivasan, D. Verbin, J. T. Barron, B. Poole, et al. Reconfusion: 3d reconstruction with diffusion priors. *arXiv preprint arXiv:2312.02981*, 2023. 3

[38] S. Xie, Z. Zhang, Z. Lin, T. Hinz, and K. Zhang. Smartbrush: Text and shape guided object inpainting with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22428–22437, 2023. 3

[39] Y. Xu, F. Li, Z. Chen, J. Liang, and Y. Quan. Encoding spatial distribution of convolutional features for texture representation. *Advances in Neural Information Processing Systems*, 34:22732–22744, 2021. 5

[40] W. Zhai, Y. Cao, Z.-J. Zha, H. Xie, and F. Wu. Deep structure-revealed network for texture recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11010–11019, 2020. 5

[41] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 286–301, 2018. 5

[42] Y. Zhou, K. Chen, R. Xiao, and H. Huang. Neural texture synthesis with guided correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18095–18104, 2023. 2, 7, 8, 9

[43] Y. Zhou, R. Xiao, D. Lischinski, D. Cohen-Or, and H. Huang. Generating non-stationary textures using self-rectification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7767–7776, 2024. 2

[44] Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang. Non-stationary texture synthesis by adversarial expansion. *arXiv preprint arXiv:1805.04487*, 2018. 2, 4, 6, 7, 8, 9