# Inpainting with Diffusion Models

## Understanding the RePaint approach

Gauthier Multari
ENS Paris-Saclay
gauthier.multari@ens-paris-saclay.fr

Florent Pollet
ENS Paris-Saclay
florent.pollet@ens-paris-saclay.fr

January 13, 2024

## Contents

# 1 Introduction

Inpainting is a computer vision task that involves the restoration or completion of missing or damaged regions within an image. This task is crucial in various applications, such as image editing, object removal, and image restoration, where the goal is to intelligently fill in the gaps in an image while preserving its visual coherence and realism. The inpainting task poses a unique set of challenges, as it requires algorithms to understand the context of the surrounding pixels and generate visually plausible content in the missing regions.

Approaches in inpainting have evolved significantly over the years, driven by advancements in deep learning and neural network architectures. Traditional inpainting methods often relied on handcrafted features and patch-based techniques, which had limitations in handling complex structures and diverse image content. The advent of deep learning, particularly generative models, has revolutionized inpainting, enabling more sophisticated and context-aware completion of missing information. During the last years State-of-the-Art approaches were based on Generative Adversarial Networks (GANs) [3] and Autoregressive modeling [9] but recently Denoising Diffusion Probabilistic Models (DDPM) appeared to be a powerful alternative to such methods [1].

The core idea of the DDPMs is to train a model to denoise images iteratively by reversing a diffusion process. The generation then starts with a randomly sampled noise and the model is applied for a given number of steps in order to generate an image.

The method proposed in the REPAINT paper [8] uses an unconditionally (relative to inpainting) trained DDPM, meaning that it is not trained to generate images for a given mask. The core idea is to use a model that was not trained explicitly for the inpainting task. This yields several advantages. The first being that any mask can be used during inference, the second being that it improves the model generation capabilities thanks to an image synthesis prior that was trained on very large amounts of data and finally it allows the method to be used with different DDPM priors.

The early approaches to image inpainting primarily relied on low-level cues within the input image or its neighboring regions in a large dataset to fill missing regions. Deterministic image inpainting, following the introduction of GANs, adopted a standard configuration with an encoder-decoder architecture, adversarial training, and tailored losses for achieving photo-realistic results. Various architectural designs, such as Dilated Convolutions, Partial Convolutions, Gated Convolutions, Contextual Attention, Edges maps, Semantic Segmentation maps, and Fourier Convolutions, were introduced to address high-level semantic context in image inpainting. However, GAN-based methods struggled with semantic synthesis.

To address deterministic limitations, diverse image inpainting methods introduced VAE-based networks, co-modulation layers, and autoregressive approaches for handling irregular masks and improving diversity. Some methods exploited image priors, like StyleGAN and non-trained generator network structures.

In our work, we start by quickly reviewing diffusion models and explaining the RePaint method, before performing many experiments to understand the potential of this new method and the advantages of diffusion models for inpainting, like diversity. Finally, we suggest some extensions to adapt RePaint for other diffusion models. Our code is available on Github `https://github.com/florian6973/RePaint` and it was forked from the original RePaint repository. Tests were made using a Nvidia RTX 4080 (Laptop) and a Nvidia V100S (OVH Cloud) to have reasonable sampling times.

# 2 Method

Our current project is based on the algorithm introduced in [8].

## 2.1 Diffusion model

Diffusion models sample from a distribution by reversing a gradual noising process. In particular, sampling starts with noise $x_T$ and produces gradually less-noisy samples $x_{T-1}$, $x_{T-2}$,... until reaching a final sample $x_0$. Each timestep $t$ corresponds to a certain noise level, and $x_t$ can be thought of as a mixture of a signal $x_0$ with some noise $\epsilon$ where the signal to noise ratio is determined by the timestep $t$. The process is illustrated in 1.

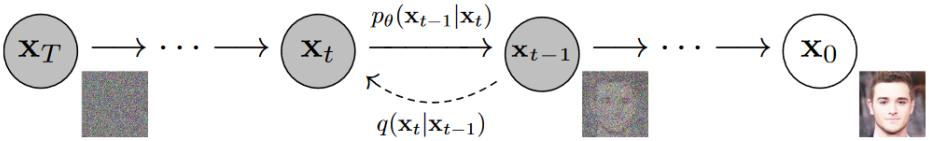



Figure 1: Illustration of the DDPM process. Figure from [4]

Let the data distribution $x_0 \sim q(x_0)$ and $q$ a Markovian process that gradually generates noised samples through $x_T$. The sample $x_t$ is obtained by adding i.i.d. Gaussian noise with variance $\beta_t$ at timestep $t$ and scaling the previous sample $x_{t-1}$ with $\sqrt{1-\beta t}$:

$$q\left(x_t \mid x_{t-1}\right) := \mathcal{N}\left(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t \mathbf{I}\right) \tag{1}$$

Ho, Jain, and Abbeel noted that $q(x_t|x_0)$ can be expressed as a Gaussian distribution by using the independence property of the noise added at each step. Let $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s$:

$$\begin{aligned} q\left(x_t \mid x_0\right) &= \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t} x_0, \left(1-\bar{\alpha}_t\right) \mathbf{I}\right) \\ &= \sqrt{\bar{\alpha}_t} x_0 + \epsilon \sqrt{1-\bar{\alpha}_t}, \epsilon \sim \mathcal{N}(0, \mathbf{I}) \end{aligned} \tag{2}$$

Using the Bayes theorem, we can deduce the posterior $q(x_{t-1}|x_t, x_0)$. It is a Gaussian with mean $\tilde{\mu}_t(x_t, x_0)$ and variance $\tilde{\beta}_t$ defined as follows:

$$q\left(x_{t-1} \mid x_t, x_0\right) = \mathcal{N}\left(x_{t-1}; \tilde{\mu}\left(x_t, x_0\right), \tilde{\beta}_t \mathbf{I}\right) \tag{3}$$

The diffusion model learns to produce a slightly more "denoised" $x_{t-1}$ from $x_t$. The model is parameterized as a function $\epsilon_\theta(x_t, t)$ which predicts the noise component of a noisy sample $x_t$. To train these models, each sample in a minibatch is produced by randomly drawing a data sample $x_0$, a timestep $t$, and noise $\epsilon$, which together give rise to a noised sample $x_t$.

This reverse process is modeled by a Neural Network trained to predict the parameters $\mu_\theta(x_t, t)$ and $\Sigma_\epsilon(x_t, t)$ of a Gaussian distribution:

$$p_\theta\left(x_{t-1} \mid x_t\right) = \mathcal{N}\left(x_{t-1}; \mu_\theta\left(x_t, t\right), \Sigma_\theta\left(x_t, t\right)\right) \tag{4}$$

The learning objective of the model is defined using the variational lower bound:

$$L = \mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right] = \mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t \geq 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}\right] \tag{5}$$

This loss can be decomposed as three terms, $L_T$, $L_t - 1$ and $L_0$ [4]:

$$\mathbb{E}_q[\underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right)}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}] \tag{6}$$

The term $L_{t-1}$ is the one used to train the network for the reverse diffusion steps. The simplifed training objective is :

$$L_{\text{simple}} := E_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_0\left(x_t, t\right)\|^2\right] \tag{7}$$

Given that the sampling process consists of repeatedly predicting $x_{t-1}$ from $x_t$, starting from $x_T$. Ho, Jain, and Abbeel [4] show that, under reasonable assumptions, we can model the distribution $p_\theta(x_{t-1}|x_t)$ of $x_{t-1}$ given $x_t$ as a diagonal Gaussian $\mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\epsilon(x_t, t))$, where the mean $\mu_\theta(x_t, t)$ can be calculated as a function of $\epsilon_\theta(x_t, t)$. By doing so, we get the following parametrization of the predicted mean $\mu_\theta(x_t, t)$:

$$\mu_\theta\left(x_t, t\right) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta\left(x_t, t\right)\right) \tag{8}$$

**Conditioning of the know region of the image** The goal of the method presented of the paper is to do image inpainting using a mask region as a condition. Given that the forward process is defined by a Markov Chain of added gaussian noise, it is possible to sample the intermediate image $x_t$ at any point in time, and therefore to sample the known regions at any time step. The method using in REPAINT is the following:

$$\begin{aligned} x_{t-1}^{\text{known}} &\sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t}x_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right) \\ x_{t-1}^{\text{unknown}} &\sim \mathcal{N}\left(\mu_\theta\left(x_t, t\right), \Sigma_\theta\left(x_t, t\right)\right) \\ x_{t-1} &= m \odot x_{t-1}^{\text{known}} + (1-m) \odot x_{t-1}^{\text{unknown}} \end{aligned} \tag{9}$$

With $x$ the ground truth image, $m \odot x$ the unknown pixels and $(1-m) \odot x_t$ the known pixels. The idea is the following, $x_{t-1}^{\text{known}}$ is sampled using the known pixels from the image $x_t$ and $x_{t-1}^{\text{unknown}}$ is sampled by the model given the previous iteration $x_t$. The two images are finally merged together using the mask.

## 2.2 Resampling

The authors of the REPAINT paper claim that applying the method described precedently produces poor results, the image generated from the mask not being coherent with the original image. We tested these claims in Section 3.2. The model manages to generate images that are more or less matching the neighboring regions but they are not semantically meaningful. Their conclusion is that the model is using the context of the known region of the image but does not harmonize the generated part with the known one. The most probable reason for this is that when predicting $x_{t-1}$ using $x_t$, the model does take into account both the output of the DDPM and the sampling of the known region, but when sampling the known pixels, the model does not consider the generated parts of the image. The model tries, by construction to harmonize the image at every step, but it cannot converge because the issue arises at every step. The idea of the resampling method presented by the authors is to "give more time" to the model in order to help it harmonize the conditional information $x_{t-1}^{\text{known}}$ with the generated information $x_{t-1}^{\text{unknown}}$.

To do so, at each start of the resampling steps, the output $x_{t-1}$ is diffused back to $x_t$ by sampling from 1. While this step adds noise to the image, it helps to preserve information incorporated in the generated region therefore improving the semantic quality of the generated output. The process is illustrated in 2.

The inpainting algorithm used by the authors of the paper is given in Algorithm 1.

## 2.3 Metrics

The authors of the original REPAINT paper used user feedback as their evaluation metric. The participants were presented with an input image featuring hidden areas. Alongside this image, two distinct inpainting solutions are showcased. The participants were then prompted to choose the more realistic image between the two options. To prevent any bias towards a particular approach, the methods were anonymized and displayed in a random order for each image. Additionally, to ensure consistency, each user responded to each question twice and could only submit their answer if they remained consistent with themselves in at least 75% of their responses. Given that we could not use the same method, we used the LPIPS metric [14] to try and quantitatively evaluate the results of the model.

LPIPS, or Learned Perceptual Image Patch Similarity, is a metric designed to quantify the perceptual similarity between two images. Unlike traditional metrics such as PSNR or SSIM, LPIPS focuses on capturing human perceptual differences. LPIPS utilizes a pre-trained neural network based on architectures like VGG or AlexNet.

**Algorithm 1** Repaint algorithm

$x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
**for** $t = T, \ldots, 1$ **do**
  **for** $u = 1, \ldots, U$ **do**
    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ **if** $t > 1$, **else** $\epsilon = \mathbf{0}$
    $x_{t-1}^{\text{known}} = \sqrt{\bar{\alpha}_t} x_0 + (1 - \bar{\alpha}_t)\, \epsilon$
    $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ **if** $t > 1$, **else** $\mathbf{z} = \mathbf{0}$
    $x_{t-1}^{\text{unknown}} = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta \left( x_t, t \right) \right) + \sigma_t z$
    $x_{t-1} = m \odot x_{t-1}^{\text{known}} + (1 - m) \odot x_{t-1}^{\text{unknown}}$
    **if** $u < U$ and $t > 1$ **then**
      $x_t \sim \mathcal{N}\left( \sqrt{1 - \beta_{t-1}} x_{t-1}, \beta_{t-1} \mathbf{I} \right)$
    **end if**
  **end for**
**end for**
**return** $x_0$

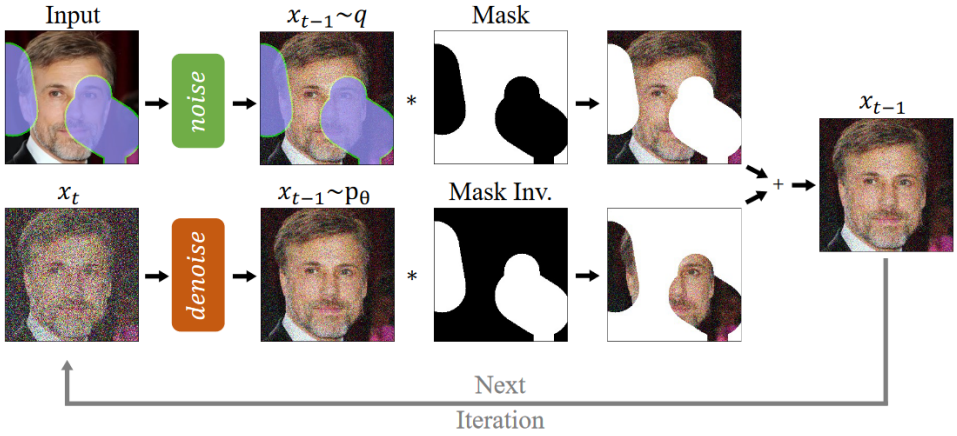



Figure 2: RePaint modifies the standard denoising process in order to condition on the given image content. In each step, we sample the known region (*top*) from the input and the inpainted part from the DDPM output (*bottom*). (Figure from the original paper [4])

To compute LPIPS, images are divided into patches, which may overlap to capture local details. These patches are then processed through the pre-trained neural network, extracting high-dimensional feature vectors that encapsulate perceptual information. The distance between corresponding feature vectors of patches in the two images is computed, often using metrics like Euclidean distance or cosine similarity. Aggregation of these distances, typically by mean or sum, yields an overall perceptual similarity score for the image pair. This final score is commonly normalized for interpretability. The general formula is the following:

$$\text{LPIPS}\left(I_1, I_2\right) = \frac{1}{N} \sum_{i=1}^{N} d\left(f\left(I_{1i}\right), f\left(I_{2i}\right)\right)$$

Where $I_1$ and $I_2$ are the input images. $N$ is the total number of patches. $f$ represents the feature extraction function of the pre-trained neural network. $d$ is the distance metric used to compute the difference between corresponding feature vectors.

LPIPS offers perceptual relevance by closely aligning with human perception, capturing subtle nuances that traditional metrics might miss. Its flexibility allows application to various image types, including natural scenes, artwork, and textures. But it has important limitations. One notable constraint is its limited generalization, as it is trained on a specific dataset, potentially leading to less accurate assessments for images outside that scope. The subjectivity involved in human perceptual judgments during training introduces biases, affecting the metric's overall reliability. Additionally, LPIPS is not universally applicable, varying in effectiveness depending on image characteristics and specific applications. It may not adequately penalize certain artifacts or handle larger structural changes, emphasizing local features and potentially overlooking global or contextual perceptual aspects.

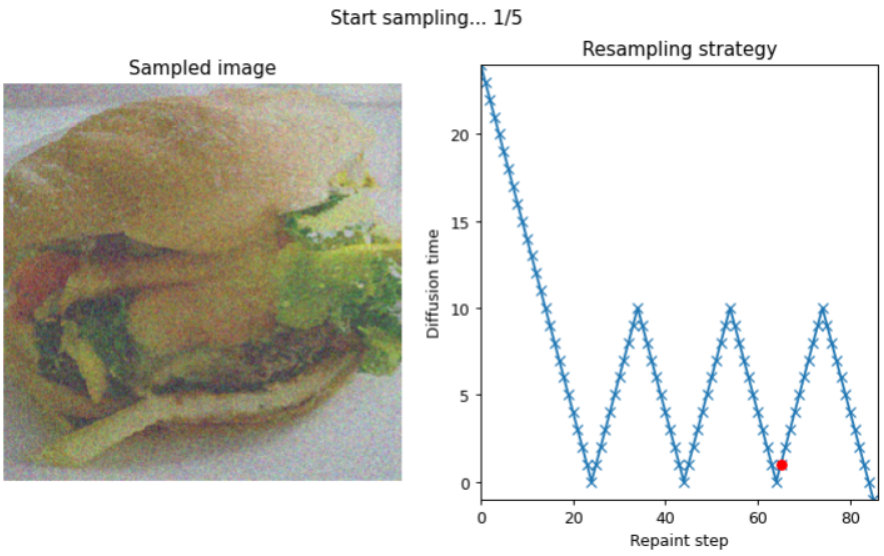



Figure 3: RePaint live interface

# 3 Experiments with RePaint

In this section, since resampling is the core of the RePaint method which allows you to control the inpainting process, we perform a series of experiments to analyze it. We therefore set up a test pipeline for RePaint, with automatic logging and an easy visualization of the intermediate steps, as shown in 3.

## 3.1 Time performance depending on image dimensions

We start with a very generic experiment: we wanted to perform experiments while saving time, since diffusion model inference is very computationally costly. We downloaded and created a new configuration for the Guided Diffusion model for images 64x64, which is not included in the RePaint code, but available on OpenAI Guided Diffusion repository.

We define a short resampling schedule (see 3.2) and we compare the time performance between sampling from a 64x64 model and a 256x256 model in Figure 4. We can deduce two main remarks from the results:

- The first iteration is always slightly longer, which indicates that Python/Pytorch may be loading some resources.
- There is no large time difference (roughly 10 %)between sampling from 64x64 and 256x256. This can be surprising at first but it can be understood when we look at the UNet architecture, which is very similar between the two models.

Therefore, there is no easy way to speed up the experiments with lower resolution images, and we are going to stick to 256x256 images in the coming subsections.

## 3.2 Behavior of the model without resampling

The first experiment conducted was to test the inpainting capabilities of the diffusion model without the method presented in the paper. To do so, we completely removed the resampling step from the diffusion process. The diffusion model is trained to denoise images in a 1000 steps, but we can skip steps in the process to test the impact of a different amount of diffusion steps. We conducted tests with $\{2, 5, 10, 25, 50, 100, 250, 500, 1000\}$ steps. The tests were performed using an NVIDIA RTX 4080 laptop GPU. For each test run, 10 images were generated, with the same starting image but with a different mask. The experiments were run on CelebA-HQ [7], ImageNet [13] and Places2 [15].

The quantitative results of these experiments are presented in Figure 5. The computational time scales linearly for the CelebA-HQ and Places2 dataset. As for the iNet256 model the higher the amount of steps the higher the computation time. We could not find an explanation for this behavior. Regarding the LPIPS metric, it plumets as the number of steps increases from 2 to 10 but then remains stable for any higher number of steps. This can be explained by the fact that the model generates images that are close to the original image and the LPIPS metric does not take into account the semantical validity of the inpainted part. This results in similar LPIPS value even though the qualitative results are different.

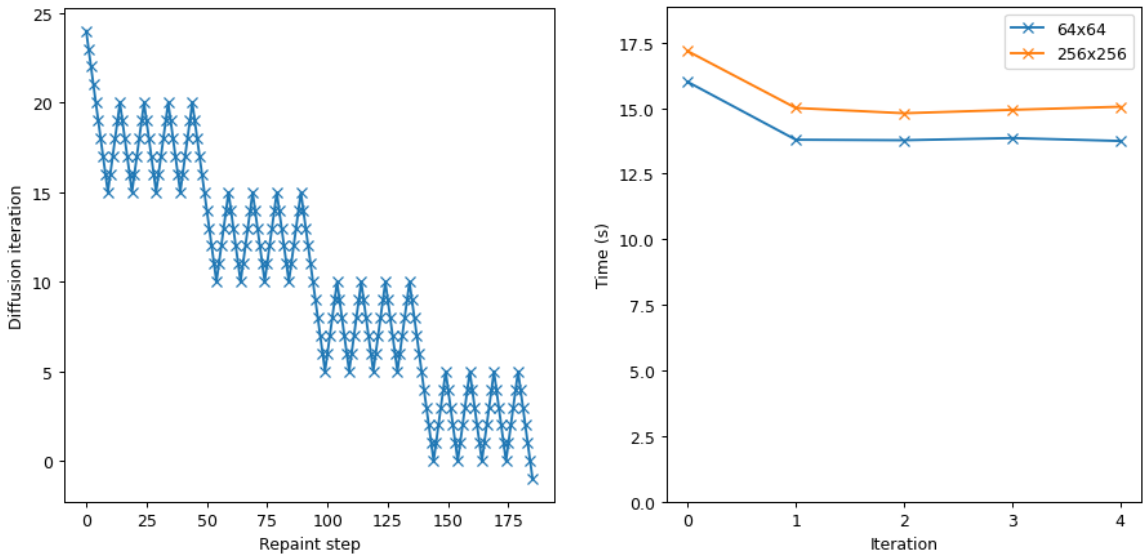
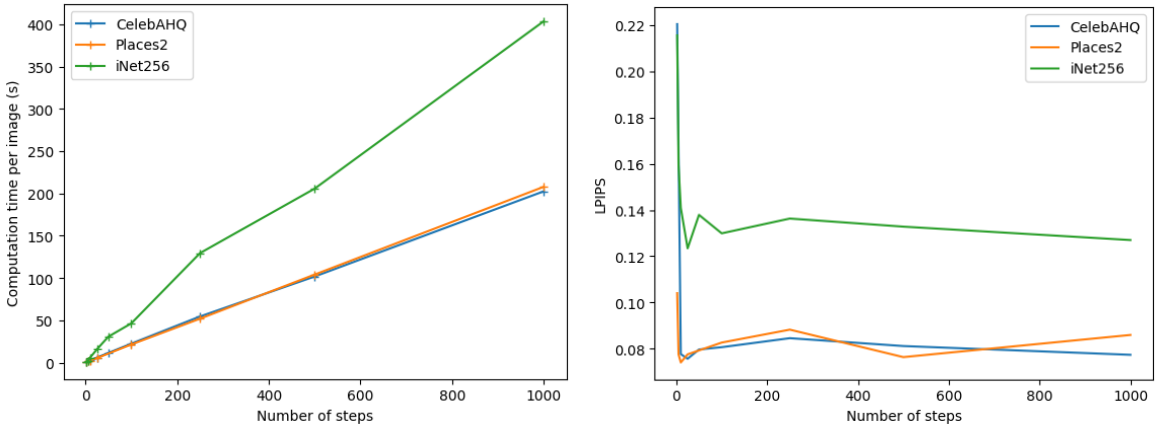



Figure 4: Execution time comparing Guided Diffusion 64x64 vs Guided Diffusion 256x256 [Intel Xeon Gold 6226R, Nvidia V100S]




(a) Average computation time for the three models

(b) Average LPIPS for the three models

Figure 5: Average computation time and LPIPS for the three models

We present a few examples of qualitative results in Table 3. We notice that even with an extremely low amount of steps (2), the model already managed to get a pixel color distribution that is not very different from the original image. This is especially true for the ImageNet and Places2 datasets. There is a visible difference in quality when the number of steps increases up until a 100 steps. After this, a higher amount of steps does not yield noticeable difference. The flaws of the models, namely the inconsistent borders of the inpainted parts are comparable.

### 3.3 Impact of the amount of resampling

Now that we are confident that we need resampling to improve the quality of the generations of our model, we perform experiments for a fixed amount of time steps with a variable amount of resampling steps. We fixed the number of timesteps at 50 to reduce the computational burden, considering the improvement of quality with a higher number of steps is not as important as the computational burden that comes with it. Once again we performed each test on 10 images for each dataset with a different mask for each image. The numbers of resampling steps tested are: {1, 2, 5, 10, 25, 50}. We see an increase in quality as the number of resampling steps increase. Even the gap between 25 resampling steps and 50 resampling steps yields a very noticeable bump in quality. For the face, the mask in the examples 3 is quite difficult, removing more than 50% of the original image, and cutting many different zones. When the amount of resampling is low, the lines of the mask is very noticeable. With 10 steps of resampling the quality of the inpainting zones becomes great, even for difficult zones like hair, some artifacts are still visible. At 50 resampling steps, the reconstructed image is of comparable

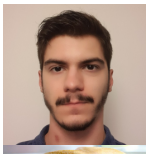
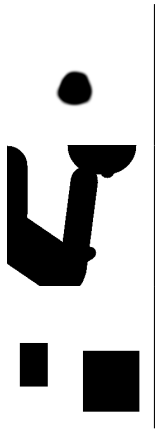
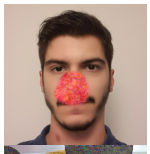
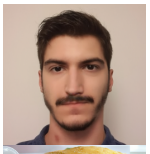
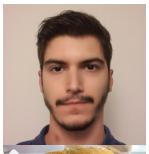
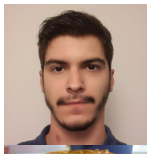
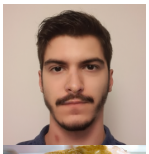
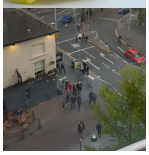
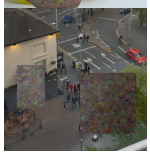
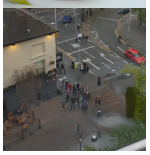
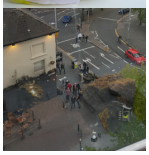
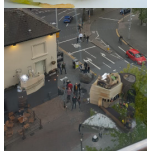
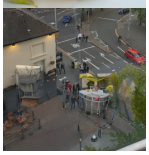
| Dataset | Original | Mask | 2 it | 50 it | 100 it | 500 it | 1000 it |
|---|---|---|---|---|---|---|---|
| CelebA-HQ |  |  |  |  |  |  |  |
| ImageNet |  | |  |  |  |  |  |
| Places2 |  | |  |  |  |  |  |

Table 1: Examples of generated images for a given number of iteration using the same mask on the three datasets

quality as the original image.

On the ImageNet dataset, the example image is a fountain. One very noticeable impact of the amount of resampling is the removal of details. For instance, without resampling the model generated on most images a water jet on the right side of the fountain, but it removed as soon as we add resampling, as if the model was biased to blend to the background of the sides of the mask (of the known the region). It struggles to reproduce the patterns of the fountain, but the generated part is very close to the rest of the pattern so much that it is nearly indistinguishable from the original. Another interesting behavior is the way the model reconstructs the lower part of the image. As the number of resampling steps increase, the amount of details of the flower hedge becomes very close to the original image. The model struggles to generate a proper central pillar and it even starts to vanish with high amount of resampling.

Finally, the Places example image is a complex scene with many details. The low resampling results are very poor in terms of quality, and even make absolutely no semantic sense. As the amount of resampling gets bigger the results get better. The model manages to close the road lines that were removed, but it struggles to reproduce the patterns of both the rooftop of the building on the right and of the pavement of the alley. An interesting behavior can be seen on the image at 25rs, where the model generated ghost-like people on the alley.

To conclude this part, from our experiments, the amount of resampling has a great impact on the final quality of the generated image. But it has notable drawbacks when too high, the first is that if the model lacks information on the structure to reconstruct, it will choose to reproduce the patterns that are predominant around the borders of the mask. Another example of a too high amount of resampling is given in Table 2. Here the masks is very restrictive on the known region, leading to only the center of an apple to be available for the model. The model has been class conditioned to the 'Pelican' class of the iNet model. Without resampling, the apple doesn't blend at all with the rest of the image. With 10 resampling steps, the image generated is of great quality, even though the background doesn't seem natural. On the other hand with 50 resampling steps, the image generated does look like a pelican it is not realistic at all. We are not sure as to why this result was generated by the model, it seems very far from the distribution of natural images.

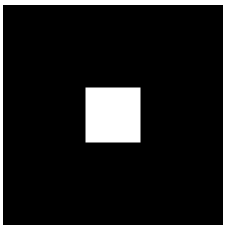
| Original | Mask | no rs | 10 rs | 50 rs |
|---|---|---|---|---|
|  |  |  |  |  |

Table 2: Examples generated images using an apple as source image and 'Pelican' as model conditioning

The second main drawback is the massive increase in computational cost. With 50 times steps the process required 130 iteration with 2 resampling steps, with 50 resampling steps it requires 3970 iterations. To address

this second issue, we tried to modify the resampling strategy proposed by the authors by only applying it at the start or at the end of the diffusion process.

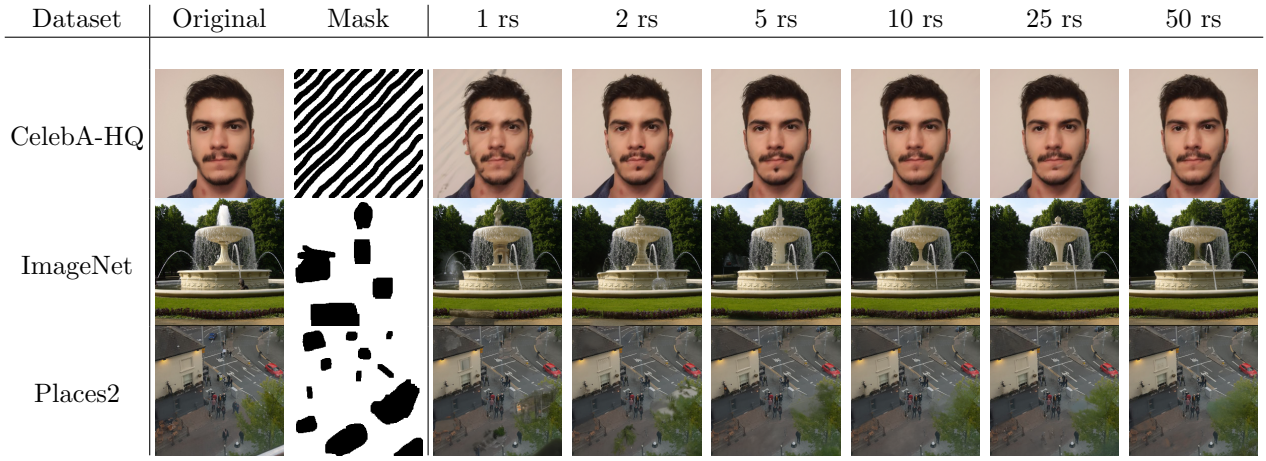



Table 3: Examples of generated images for a given number of resampling step (rs) on the three datasets

## 3.4 Evolutive resampling

In this experiment, we investigate the impact of resampling on image quality at different times of the denoising process. For this, we define two schedulers, plotted in 6, one performing resampling only for early/noisy steps, and another only for late steps, but with the same number of total steps. We then sample 5 images based on each scheduler, as shown in 7. We measure the performance with three metrics: LPIPS based on VGG, LPIPS based on Alex, and SSIM (we show $1 - SSIM$ to plot all the metrics together).

We can see that no images are perfect, since the schedulers are very short. There is also no clear winner between early and late resampling. SSIM suggests that late resampling is slightly better. This can be understood if we consider that late resampling will superficially homogenize the limits of the inpainting region and therefore they will be a better integration, whereas early resampling leads to more global changes and is more like exploration in the space of images.

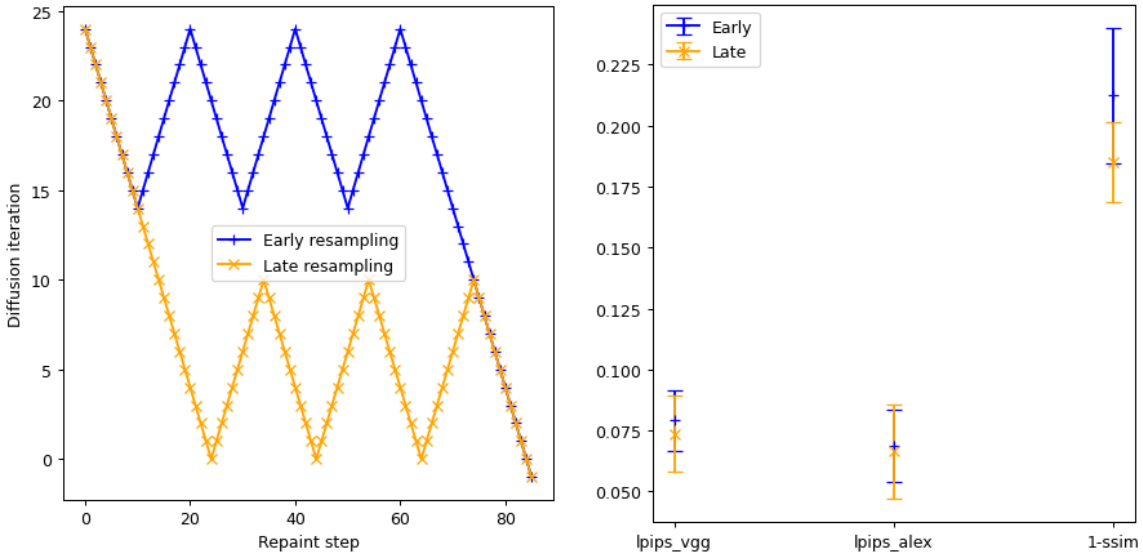



Figure 6: Evolutive resampling strategy and metrics (lower is better)

## 3.5 Inpainting diversity

In this subsection, we pursue the investigation of 3.4 with a special focus on face generation and its diversity. We assume the use case of someone looking to inpaint a face, and who would like to have a precise control on both new suggestions and integration. Of course changing seeds can work, but you need to recompute all the

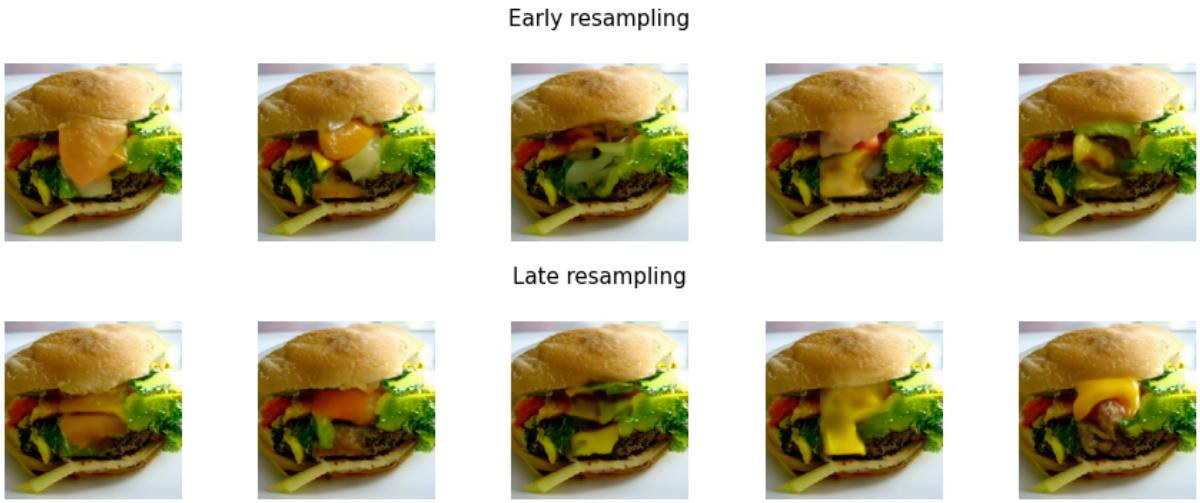



Figure 7: Evolutive resampling results

steps, which is expensive. Therefore, we check if it is possible through resampling to explore many possibilities and manually adjust the homogenization at a lower cost.

In this regard, we denoise an image up to the Repaint step 14, we save the image at this step and then we consider different scenarii: between one and three resampling patterns, and between small, medium and large jumps. Schedules are shown in 4) and generated images in 5.

As expected, more resampling patterns give smoother images, but with small jumps, you cannot get a real change in the facial expression. It is only with large resampling jumps that the expression and the look of the face change. However, if a large jump changes the face, it is not enough to homogenize, and a few late small resampling steps are needed in this regard. That is why the best image is probably the middle-left one, with enough late resampling to homogenize.

Therefore, adding more late resampling can improve integration and doing one large resampling step can make a new suggestion, without recomputing everything and still improving globally the quality of the image. These observations can lead to a manually guided diffusion process to get an image close to what someone can look for.

### 3.6 Generation of texture

One of the possible uses of generative models can be the generation of textures. We could imagine use cases like a 3D environment where the texture near the camera view is generated in real time, bypassing the the repetitiveness of the usual textures all while having a low impact on VRAM. We tried to evaluate the capability of this method to extend existing textures. Two of the results can be seen in Table 6. The results are very poor, both of the models are too conditioned to generate scenes and do not try to simply extend the known part of the image even with high amounts of resampling. We still think that this use of DDPM is a possibility, as long as the model used is trained to generate textures. We did not find any pre-trained model to test that theory.

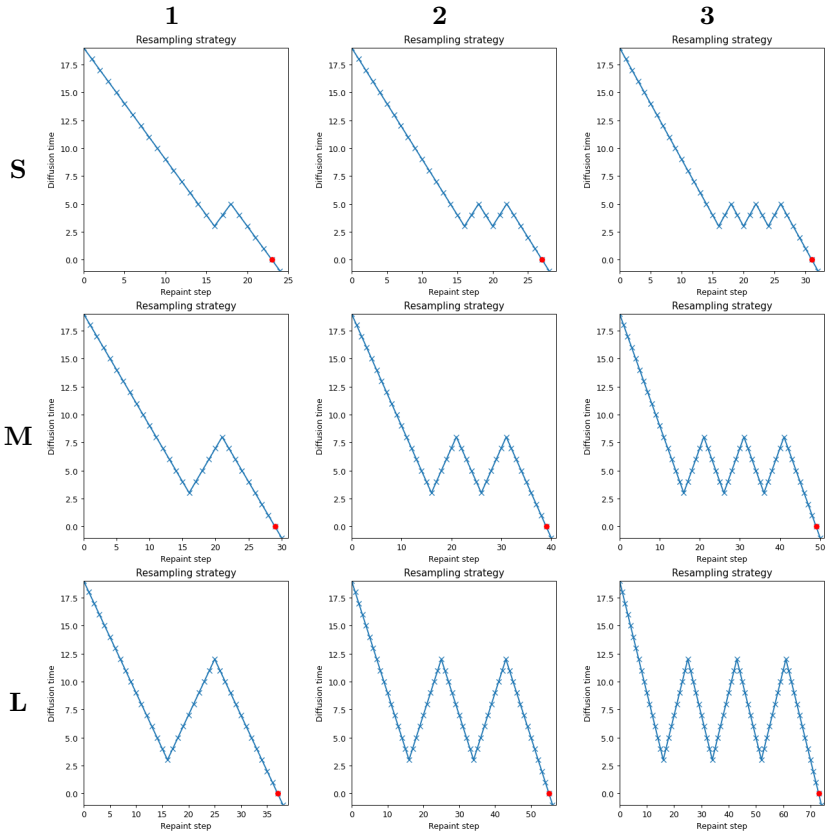



Table 4: Resampling schedules (same seed)


Table 5: Faces generated with different resampling 4

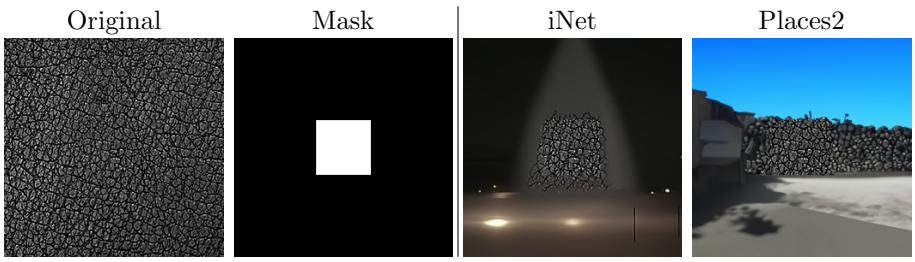



Table 6: Examples of generated images starting from a texture

# 4 Extensions of RePaint

In this section, we discuss some potential generalization and evolution of the RePaint method, since the field of diffusion models for image generation is very active.

## 4.1 Reimplementing RePaint with HuggingFace Diffusers

HuggingFace has developed in [10] an open-source *Python package* to provide a common interface for diffusion models with an online library of open access pretrained models. We can therefore easily test different schedulers and different denoising model in a streamlined manner.

Unfortunately, HuggingFace library does not contain the guided diffusion ImageNet 256 dataset from [2]. We did a quick test with the model *google/ddpm-cat-256*, based on [4] and trained with cat images. Compared to the guided diffusion model used by the original RePaint paper [8], this represents a simpler diffusion model, with less refinement.

Regarding the implementation, using the Diffusers library is very straightforward, since you just need to modify the denoising loop, and rely on the scheduler parameters to add the required noise. You can therefore focus much more on the logic and it makes the code significantly more readable than the original RePaint code.

We test it on our traditional hamburger in Table 7, and we compare with and without resampling to highlight the importance of this technique to get better quality image. Without resampling, we clearly see the bias of the model, because a cat is in the inpainted region, whereas this is not the case in the images with resampling. Note that the results cannot be exceptional, since the model has never been trained with hamburger images.

## 4.2 Extending RePaint for text-to-image Latent Diffusion Models (LDM)

Now that the RePaint method can work with the Diffusers library, we can test the method with more complex models. For instance, instead of using unconditional or class-conditional architectures, like in the RePaint paper, we decided to test it on a text-conditional model such as Stable Diffusion 1.5 *runwayml/stable-diffusion-v1-5* from [12].

Stable Diffusion uses the CLIP tokenizer and text model *openai/clip-vit-large-patch14*, introduced in [11], and a conditional UNet. The denoising step is therefore modified to take into account the text guidance via a classifier-free guidance [5] approach, unlike RePaint Guided Diffusion models which have explicit classifiers.

Moreover, Stable Diffusion is a latent diffusion model, so there is a supplementary Variational AutoEncoder (VAE), as shown in 8, which is used for preprocessing and post-processing, to convert the input high-resolution image to a lower-resolution representation in a latent space. Then, the traditional diffusion model workflow is performed in a latent space. The main advantages of latent diffusion models are a faster sampling and a smaller memory usage for better resolution images.

Therefore, we modify the RePaint pipeline to downsample the mask 3x512x512 to the latent dimension 4x64x64, since at first glance the pixel to latent space mapping seems to preserve spatial information. We also encode the original image through the VAE and we will use this encoded version to perform the merge at each iteration. Except these changes, we stick to the original RePaint pipeline, meaning that we update the latent representation taking into account the downsampled mask and encoded original image after every denoising step and for resampling we noise the latent representation by a factor computed by the scheduler. We also keep the same parameters for the scheduler and the resampling.

We present some results we have in 7. As we can see, resampling is necessary to match the same style as the original image, and the prompt can efficiently guide the inpainting task. However, despite the resampling steps, there are still some cases for which inpainting completely fails: there is no semantic meaning or the prompt

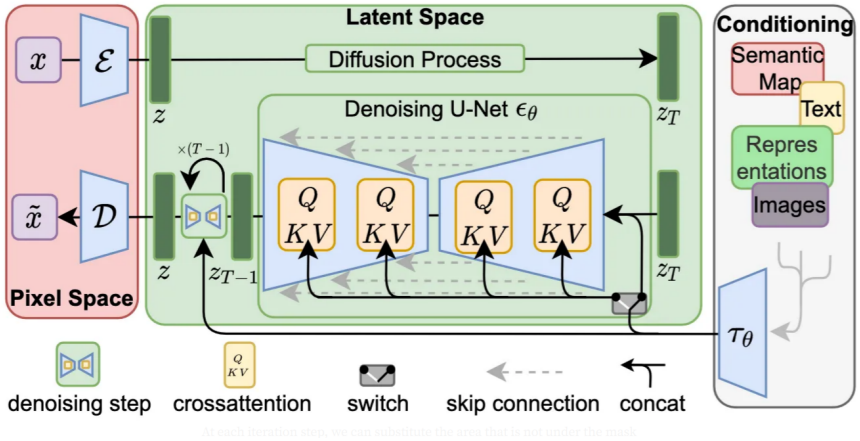



Figure 8: Stable Diffusion architecture (Figure from [12])

seems to be ignored, as shown in Table 8. A potential reason could be that the classifier-free guidance with is fighting against the resampling homogenization and this does not lead to satisfactory results. Moreover, downsampling the mask may lead to more imprecision and worst matching.

In the Stable Diffusion paper [12], they use a specific UNet network with added channels for the original image and the mask; they fine-tune the original Stable Diffusion model for inpainting. Given the high resampling computational cost of the RePaint method, this method may be more practical to use for traditional application. Nevertheless, if this method always gives very nice images, it often ignores the prompt to prefer semantic meaning during our tests with *runwayml/stable-diffusion-inpainting*, shown in the last column of 8. Moreover, based on the examples we consider, using a prompt to guide generation does not seem significantly more powerful than a simpler class-conditional approach, because inpainting fails when suggesting specific and original situations.

| **Model** | DDPM-RePaint | DDPM-RePaint | SD-RePaint | SD-RePaint | SD-RePaint |
|---|---|---|---|---|---|
| **Prompt** | No prompt | No prompt | Burger | Burger | Flowers |
| **Mask** | Center | Center | Corner | Corner | Center |
| **Resampling** | Without | With | Without | With | With |
| **Result** |  | | | | |

Table 7: RePaint with Hugging Face Diffusers

| **Model** | SD-RePaint | SD-RePaint | SD-RePaint | SD-Inpainting |
|---|---|---|---|---|
| **Prompt** | Cutlery | Bite in burger | Ants | Ants |
| **Result** |  | | | |

Table 8: Inpainting can fail

### 4.3 Limitations and further work

During our tests and our exploration, we noticed several current important limitations to inpainting with RePaint or other Deep-Learning methods, that should be overcome before such methods can be widely used by the public or for professional applications.

**Resolution** So far, generative models have been very computationally intensive and only tackles low-resolution images. Without a graphic card, it is clearly not possible to use a diffusion model, just for inference, and even with powerful ones, it takes a few minutes to generate a 256x256 image. If we consider that we often want several samples to decide which is the best, RePaint is far from being a useful method for inpainting.

**Evaluation metric** A major issue in evaluating inpainting methods is the lack of mathematical metrics. Whether it is for RePaint [8] or for Stable Diffusion [12], the performance is assessed through user study. Recent metrics like LPIPS have been promising in filling this gap, however as we have seen in our tests, it can indicate when an inpainting is completely wrong, but it cannot distinguish between relatively good inpainting results, like we had when we compared with and without resampling.

**RePaint for other diffusion models** Following 4.2 on RePaint for latent diffusion models, it could be interested to investigate more in the adaptations of RePaint to other recent diffusion models, and consider new improvements to help homogenize prompt and context for instance. [6] could be a promising idea for better inpainting.

# References


[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *arXiv preprint arXiv:1809.11096* (2018).

[2] Prafulla Dhariwal and Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis*. 2021. arXiv: 2105.05233 [cs.LG].

[3] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *arXiv preprint arXiv:2006.11239* (2020).

[5] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: 2207.12598 [cs.LG].

[6] Susung Hong et al. *Improving Sample Quality of Diffusion Models Using Self-Attention Guidance*. 2023. arXiv: 2210.00939 [cs.CV].

[7] Ziwei Liu et al. "Deep learning face attributes in the wild". In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Dec. 2015.

[8] Andreas Lugmayr et al. *RePaint: Inpainting using Denoising Diffusion Probabilistic Models*. 2022. arXiv: 2201.09865 [cs.CV].

[9] Jialun Peng et al. "Generating Diverse Structure for Image Inpainting with Hierarchical VQ-VAE". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10775–10784.

[10] Patrick von Platen et al. *Diffusers: State-of-the-art diffusion models*. https://github.com/huggingface/diffusers. 2022.

[11] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].

[12] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].

[13] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.

[14] Richard Zhang et al. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. 2018. arXiv: 1801.03924 [cs.CV].

[15] Bolei Zhou et al. "Places: A 10 million image database for scene recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).