Magic Eraser: Fast Image Inpainting with Diffusion Sampling ECE 285: Project Report

Ayush M. Jamdar
ECE
A69032160

Kishore Rajendran
ECE
A69033142

Abstract

In this project, we implement a diffusion-based image inpainting algorithm using the RePaint [4] method. Our work builds upon the guided-diffusion framework, implementing both DDPM and DDIM sampling methods to achieve high-quality inpainting results with arbitrary masks arising from segmented objects. We focus on improving sampling strategies and integrating interactive mask generation using the Segment Anything Model (SAM) [2]. This report includes an overview of the RePaint algorithm, our custom implementation choices, experimental results, and comparison of the DDPM and DDIM sampling approaches. Our final deliverable is a comprehensive pipeline for interactive object removal and context-aware image inpainting in less than 20 seconds – we call it the **Magic Eraser**. Code: https://github.com/Kishore-1R/ECE285-Diffusion-Models

1 Introduction

Image inpainting is the task of filling in missing or corrupted regions in an image. For object removal, inpainting follows segmentation and object masking. Our motivation is to explore state-of-the-art diffusion models for solving this problem, specifically the RePaint algorithm. The key challenge in diffusion-based inpainting is to condition on the known (unmasked) region and generate the unknown (masked) effectively during the sampling process.

We implement from scratch the RePaint method, which introduces a resampling strategy to improve inpainting quality by repeatedly sampling certain timesteps. In addition, we integrate DDIM sampling to speed up inference while preserving visual fidelity. Our results show visually appealing inpainted images with flexible mask handling, interactive control, and batch processing support.

2 Related Work

2.1 Segment Anything Model (SAM)

The Segment Anything Model (SAM) [2] is a recent foundation model for image segmentation developed by Meta AI. It is designed to perform general-purpose segmentation tasks with zero-shot capabilities, making it especially useful for tasks requiring interactive or arbitrary object masking, such as image inpainting.

SAM introduces a flexible architecture that separates the segmentation task into three components:

- **Image Encoder:** SAM uses a Vision Transformer (ViT) as the backbone encoder to produce a dense embedding of the input image. The image encoder operates once per image and outputs a feature map that is reused for multiple segmentation queries. This is important for enabling fast, interactive mask generation.

- **Prompt Encoder:** This module encodes user-provided prompts such as points, bounding boxes, or previous masks. These prompts are embedded into a prompt token representation which guides the segmentation output. The prompt encoder handles both sparse inputs (e.g., points or boxes) and dense inputs (e.g., masks) through different mechanisms.
- **Mask Decoder:** The decoder takes both image features and prompt tokens to generate a segmentation mask. It uses a lightweight transformer decoder that efficiently fuses prompt information with the image embedding. The output includes multiple mask hypotheses with corresponding confidence scores, allowing selection of the best segmentation result.

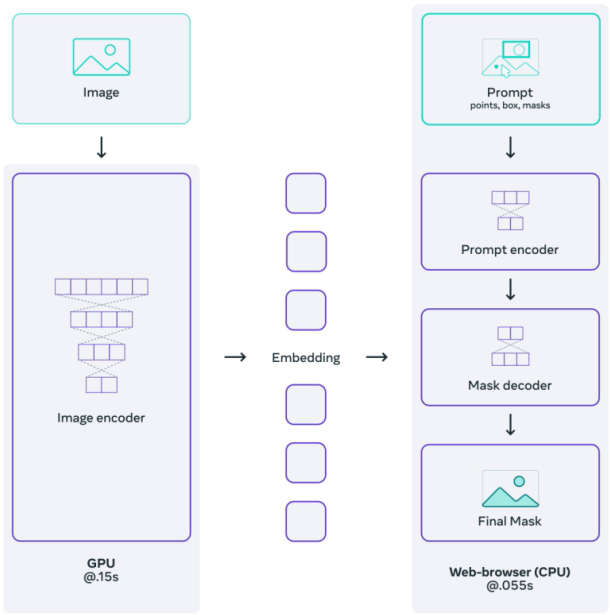

Figure 1: The architecture of the Segment Anything Model (SAM). It consists of an image encoder (ViT), a prompt encoder for user input (points, boxes, masks), and a mask decoder that fuses both to produce segmentation masks.

The key strength of SAM is its ability to generalize across different segmentation tasks without retraining, owing to its pretraining on a large and diverse dataset (SA-1B, containing over one billion masks). In this study, we use SAM to generate high-quality, user-guided masks that define the regions to be inpainted. These masks serve as the conditioning input for the RePaint diffusion process, and their flexibility significantly improves the usability and robustness of our pipeline.

We use the ViT-H model checkpoint for segmenting input images and generating masks in real time.

2.2 Diffusion models: DDPM vs DDIM sampling

Diffusion models are a class of generative models that learn to generate data by reversing a gradual noising process. The foundational idea is to learn a parameterized denoising distribution that maps pure noise back to real data through an iterative refinement process. These models have recently gained popularity for their high generation quality in tasks like image synthesis, inpainting, and editing.

Denoising Diffusion Probabilistic Models (DDPM) [1] form the basis of most diffusion-based generative models. In DDPM, the forward process gradually adds Gaussian noise to an image over a fixed number of steps T, converting the image into nearly pure noise. The model then learns to reverse this process step-by-step using a neural network (typically a U-Net) trained to predict the added noise. The training objective is a simplified variational bound that reduces to a weighted mean squared error (MSE) loss.

The forward process in DDPM is defined as a Markov chain that adds Gaussian noise at each timestep t:

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

where β_t is a variance schedule (typically increasing linearly or cosine spaced). The intermediate noisy images x_t are latent variables. Using closed-form reparameterization, we can sample any x_t directly from x_0:

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{2}$$

with $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$. This reparametrization allows us to re-noise x_0 to any **noise manifold** t. The T^{th} noise manifold is (asymptotically) pure Gaussian random noise while the 0^{th} manifold is pure image signal.

Moreover, we can also reparametrize to re-noise any latent x_{t-s} to a higher noise manifold latent x_t through the distribution:

$$q(x_t \mid x_{t-s}) = \mathcal{N}\left(x_t; \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-s}}}x_{t-s}, \left(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-s}}\right)\mathbf{I}\right), \tag{3}$$

Equations 2 and 3 will be particularly important for our inpainting algorithm.

The reverse process is learned by training a neural network $\epsilon_\theta(x_t, t)$ to predict the noise ϵ added at each step:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right], \tag{4}$$

where $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$.

The sampling process then reverses the forward process via:

$$x_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z, \tag{5}$$

where $z \sim \mathcal{N}(0, \mathbf{I})$, and σ_t can be learned or fixed to match β_t [1].

While DDPM, as used in RePaint [4], produces high-quality results, it requires hundreds to thousands of sequential denoising steps for each test image, making it computationally expensive and slow for inference. With the goal of speeding this process up by avoiding taking all the denoinsing steps, we look at the DDIM sampling strategy.

Denoising Diffusion Implicit Models (DDIM) [5] propose a non-Markovian alternative to DDPM, allowing for faster sampling without retraining. Unlike DDPM, which uses stochastic transitions at each step, DDIM introduces a deterministic update rule that still recovers the same training objective. The key idea is to deterministically compute the predicted clean image $\hat{x}_t$ at each step and then derive the next noisy image using a predefined noise schedule. Think of it as a two-step manifold transition: denoising from the t^{th} manifold to the 0^{th} and then re-noising back to the $(t-s)^{th}$.

This method drastically reduces the number of sampling steps (e.g., from 1000 to 50) by skipping s steps while maintaining similar perceptual quality.

In DDIM, we accelerate 5 with the non-Markovian assumption. Specifically, the sampling iteratively performs the following [3]

$$x_{t-s} = \sqrt{\bar{\alpha}_{t-s}} \cdot \hat{x}_t + \sqrt{1-\bar{\alpha}_{t-s}} \cdot \hat{\epsilon}_t \tag{6}$$

where

$$\hat{x}_t = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_\theta(x_t, t)\right) \tag{7}$$

$$\hat{\boldsymbol{\epsilon}}_t := \frac{\sqrt{1-\bar{\alpha}_{t-s}-\eta^2\tilde{\beta}_t^2}\epsilon_\theta(x_t, t)\,(\boldsymbol{x}_t) + \eta\tilde{\beta}_t\epsilon}{\sqrt{1-\bar{\alpha}_{t-s}}} \tag{8}$$

ϵ is stochastic noise and ϵ_θ is deterministic noise.

Comparison between DDPM and DDIM:

- **Sampling Speed:** DDIM is significantly faster due to fewer sampling steps. DDPM, while slower, can produce slightly better results at very high sampling step counts.
- **Non-Markovian vs Markovian Process:** DDIM creates a non-Markovian process that can “look back" at the entire trajectory. This allows DDIM to better preserve long-range dependencies in complex structures even when it takes fewer steps.
- **Inpainting Use Case:** DDPM allows for masked resampling strategies like RePaint which improve inpainting results. DDIM can also be used for inpainting, often as a faster approximation.

2.3 RePaint Algorithm

RePaint [4] is a diffusion-based image inpainting method that introduces a novel resampling strategy for masked regions during generation. Traditional diffusion-based inpainting methods typically condition the reverse diffusion process on unmasked (known) pixels and progressively denoise the entire image. However, they suffer from boundary artifacts and poor structure coherence within the masked regions. RePaint addresses this by introducing **temporal resampling**: during the reverse diffusion process, it jumps backwards and reprocesses certain timesteps multiple times, enforcing consistency with known pixels while refining the inpainted region. This is depicted and further described in the following illustration taken from the original paper:

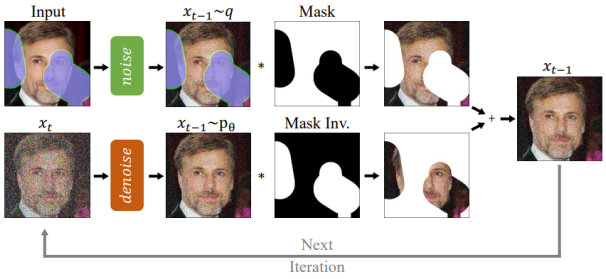

Figure 2: RePaint modifies the standard denoising process in order to condition on the given image content. In each step, the known region (input) is re-noised from the clean image manifold ($t = 0$) to the $(t-1)^{th}$ manifold (x_{t-1}^{known} in 1). Simultaneously, we denoise the latent obtained in the previous step ($x_{t-1}^{unknown}$ in 1) x_t to x_{t-1}. Now that the known and unknown regions are on the same image manifold, we can use their corresponding masks and add them to get x_{t-1}. This is classic denoising in diffusion sampling. The key novelty of RePaint is to renoise this latent x_{t-1} back to x_t and repeat this “resampling" U times before going actually down a noise manifold.

The pseudocode for this algorithm is described in 1. Note that we noticed some inconsistencies in their variance schedule indexing, leading to our DDPM inpainting algorithm implementation in 1 being slightly different from theirs.

Key Components of the RePaint Algorithm

- **Mask Conditioning:** Known regions are preserved throughout the reverse process by overwriting them at every step. The network is only allowed to modify the unknown (masked) regions.
- **DDPM based Resampling Strategy:** The algorithm allows sampling to temporarily go "backward" in time and resample specific steps multiple times (defined by a hyperparameter U). This reintroduces stochasticity and prevents the model from committing too early to suboptimal reconstructions.

RePaint achieves superior inpainting performance compared to its predecessor methods by mixing denoising and stochastic resampling in a carefully balanced schedule, while always enforcing consistency with visible (known) content. We use RePaint as our baseline for inpainting.

Algorithm 1 RePaint: A DDPM based approach

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

OpenAI's Guided Diffusion Repository

We base our implementation on OpenAI's guided-diffusion repository, which provides a highly modular and extensible framework for training and sampling from DDPM-based models. The repository includes many pretrained models (out of which we use the 256x256 unconditioned U-Net), optimized training loops, and sampling code supporting classifier guidance and unconditional generation.

In our work, we adapt this framework to fundamentally re-implement RePaint's resampling-based inpainting, as well as an extended DDIM sampling variant. We implement our own `custom_sample_loop` (without using OpenAI's or RePaint's sampling code) to handle arbitrary binary masks, support the U-step resampling strategy of RePaint, and integrate with Segment Anything Model (SAM) to generate object-aware masks.

3 Method

3.1 Magic Eraser pipeline

Our method implements image inpainting using a diffusion, specifically leveraging DDIM (Denoising Diffusion Implicit Models) for fast and controllable inference while using RePaint's resampling strategy. The pipeline also integrates the Segment Anything Model (SAM) for interactive, user-driven mask generation. Our algorithm is illustrated in 2.

Pipeline Steps:

- **Step 1:** The user provides an image and interactively selects objects to remove using SAM, which generates a binary mask.
- **Step 2:** The generated mask is then dilated (to remove edge effects) and used to remove the selected object from the input image.
- **Step 3:** This empty region is then filled using a pre-trained diffusion model. DDIM sampling is used for faster inference, allowing the user to control the number of steps (T) and iterations (U). We also allow the user to optionally select DDPM if they prefer.

Formulation: Let (x_0) be the original image, (M) the mask, and (x_t) the noisy image at step (t). The DDIM sampling process iteratively denoises (x_t) conditioned on the known (unmasked) pixels and the mask (M), reconstructing the missing region.

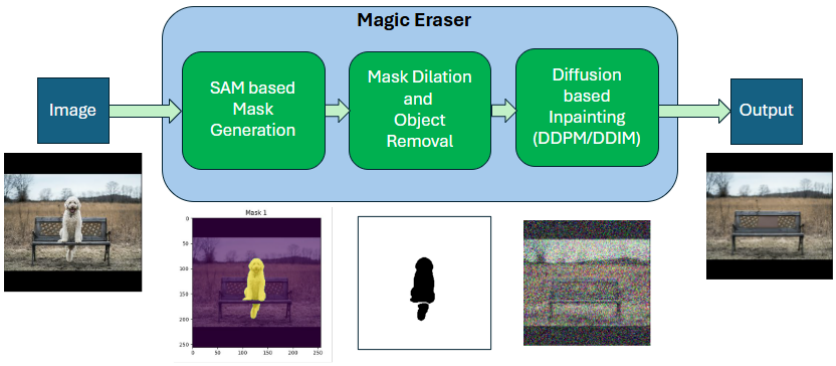

Figure 3: A block diagram illustration of our magic eraser pipeline. Input Image → SAM Mask Generator → Mask → Diffusion Model (UNet backbone) → DDIM Sampler (configurable steps/iterations) → Output: Inpainted Image

3.2 Training & Testing Algorithms

Training Algorithm:

The diffusion model is pre-trained on large-scale datasets (e.g., ImageNet) using standard diffusion training: Add noise to images over many steps. Train the UNet to predict the noise at each step. No additional training is performed for inpainting; we use the pre-trained model.

Testing/Inference Inpainting Algorithm:

User selects an image and mask interactively. The masked image and mask are input to the DDIM-based inpainting pipeline. The model iteratively denoises the masked region, guided by the known pixels and mask, for T steps and U iterations. The final inpainted image and an evolution GIF are saved.

3.3 Novelty & Strengths Compared to Previous Work

Proposed Techniques & Improvements:

- **Fast Inference**: DDIM sampling reduces inference time by up to 50x compared to DDPM, with minimal quality loss.
- **Interactive Masking**: Integration with SAM allows precise, user-friendly mask selection, improving usability and result quality.
- **Configurable Speed/Quality Tradeoff**: Users can adjust T (steps) and U (iterations) for their needs.
- **Progress Visualization**: Evolution GIFs provide insight into the inpainting process.

Reasons & Strengths:

- **Efficiency**: DDIM enables practical, real-time inpainting applications.
- **Flexibility**: Interactive masking and parameter control adapt to diverse user needs.
- **Usability**: The pipeline is accessible via a simple command-line interface.

We base our approach on the RePaint algorithm with custom modifications:

- **Sampling Loop:** Implemented from scratch, supporting both DDPM and DDIM strategies. The DDPM loop includes the RePaint resampling mechanism controlled by a parameter U.
- **Mask Handling:** We maintain the known regions throughout the sampling steps and apply the RePaint strategy only to masked regions. We also add dilation to handle boundaries better.

- **SAM Integration:** We use the Segment Anything Model to allow for interactive and arbitrary mask selection.
- **DDIM Sampling:** Offers a faster alternative to DDPM. Supports the same masking mechanism with much fewer steps.
- **Batch Processing:** Allows processing of multiple images with different U values and sampling strategies in a single run.

Algorithm 2 DDIM Inpainting with RePaint's Resampling

$x_T \sim \mathcal{N}(0, \mathbf{I})$
for $t = T, \ldots, 1$ in skips s **do**
  for $u = 1, \ldots, U$ **do**
    $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $\epsilon = 0$
    $x_{t-s}^{\text{known}} = \sqrt{\bar{\alpha}_{t-s}} x_0 + \sqrt{1 - \bar{\alpha}_{t-s}} \epsilon$
    $\hat{x}_t = \left(x_t - \epsilon_\theta \sqrt{1 - \bar{\alpha}_t}\right) / \sqrt{\bar{\alpha}_t}$
    $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = 0$
    if $t > 1$ **then**
      $x_{t-s}^{\text{unknown}} = \sqrt{\bar{\alpha}_t} \hat{x}_t + \sqrt{1 - \bar{\alpha}_{t-s}} \eta \epsilon + \epsilon_\theta \sqrt{1 - \bar{\alpha}_{t-s}} \sqrt{1 - \eta^2}$
    else
      $x_{t-s}^{\text{unknown}} = \hat{x}_t$
    end if
    $x_{t-s} = m \odot x_{t-s}^{\text{known}} + (1 - m) \odot x_{t-s}^{\text{unknown}}$
    if $u < U$ and $t > 1$ **then**
      $x_t = \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-s}}} x_{t-s} + \sqrt{1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-s}}} \epsilon$
    end if
  end for
end for
return x_0

4 Experiments

- **Datasets:** We use images from the LaMa inpainting dataset for benchmarking, along with custom images for testing diverse scenarios. Images are in PNG format with corresponding binary masks.
- **Implementation Details:**
  - Sampling loop implemented in `gaussian_diffusion.py`.
  - Half-precision (fp16) inference used for faster computation.
  - Configurable sampling steps and U values.
  - All experiments were performed on an NVIDIA A40 GPU.
- **Results:** Our method successfully inpaints large masked regions, preserving structure and texture. DDIM sampling provides similar quality with significantly reduced inference time (T=20 U=10 in under 20 seconds). While T=1000 with U=10 takes around 15 minutes with DDPM. Increasing U improves the smoothness and coherence of inpainted regions. All results are shown in Figures 4 and 5.
- **Ablation Study:** We evaluate the effect of different U values on final image quality. Lower values (e.g., $U = 1$) result in blurrier results, while higher values ($U = 10$) offer sharper, more realistic outcomes. DDPM vs DDIM comparison also highlights the trade-off between speed and quality.

Figure 4: DDPM results for varying number of resampling steps U (for T=1000). We note that for all examples, fewer resampling steps result in incoherent inpainting while the results are the best for more resampling steps U=10 when inpainting is in harmony with the scene context. This experiment shows that resampling is crucial for diffusion inpainting. It took 2 minutes to 15 minutes on varying U from 1 to 10.

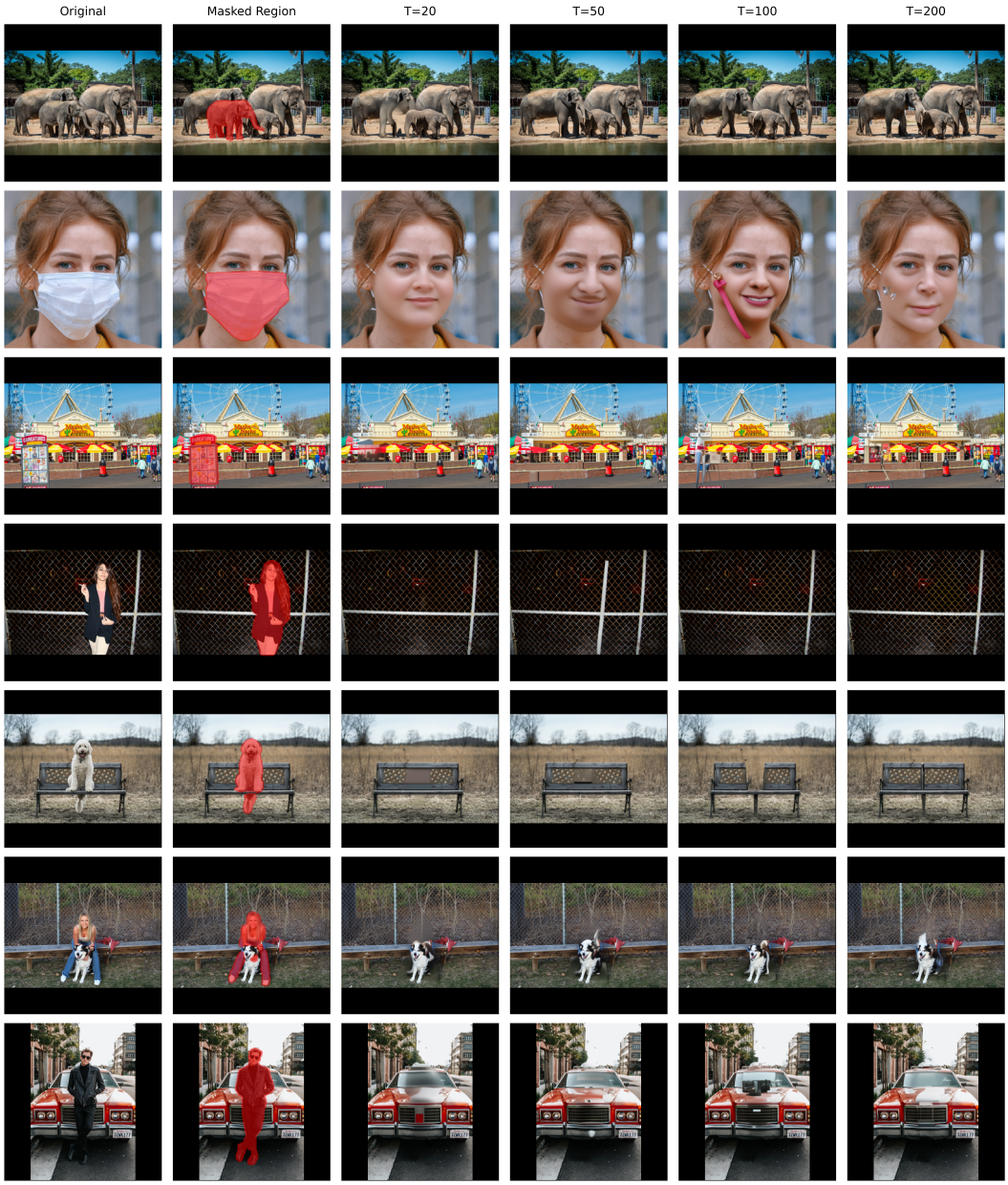

Figure 5: DDIM results for varying number of denoising steps T (for U=10). Here we see that inpainting is great even for the least sampling steps and the results don't show an improving trend on increasing those steps. This study shows that with resampling, DDIM does the job 50x faster. Resampling steps seem to affect the result very strongly, not the denoising steps.

4.1 Discussion

From the results illustrated in Figures 4 and 5, we note that the resampling steps play a critical role in coherent inpainting. This is visible in the DDPM experiment (Fig. 4) where we vary U. For smaller U, diffusion seems to be completely unaware of the scene context and this can be explained using Fig. 2. The diffusion denoising step always remains unaware of the known scene regions as the known and unknown regions are simply added together. Resampling offers a chance to repeat the scene merging over and over again to so that relevant content is generated in the unknown region.

The DDIM experiment shows that the number of diffusion timesteps are not really important – complicated scenes like the face and the mesh were generated in 20 diffusion steps (compare this with 1000 steps in DDPM). Rather the number of resampling steps (U) is crucial.

However, we must note that being a generative model, it hallucinates details. For instance, the face inpainting example has very different and even strange face content. We must remember that diffusion models only promise realistic (not exact or sensible) reconstruction by sampling from the probability distribution of real images. In scenes like the wire-mesh, the reconstructions were exact and sensible. Hence, hallucination is context dependent.

4.2 Conclusion

1. **Efficiency-Quality Trade-off:**
   - DDIM achieves comparable results in just 20 iterations (under 20 seconds, for U=10)
   - DDPM requires 1000 iterations (>15 minutes for U=10) for similar or better results
   - Massive 50x speedup with DDIM with a minimal trade-off in quality
2. **Resampling:**
   - For $U = 1, 2$ the inpainted scene is quite incoherent with the rest of the image
   - But for higher values $U = 10$, the inpainted region blends with the surrounding
   - This indicates that resampling steps help contextualize diffusion to fill relevant information
   - The benefits saturate after $U = 10$, in congruence with the observation in RePaint
   - Thus, the resampling algorithm is a crucial aspect of diffusion inpainting
3. **Performance Comparison:**
   - DDPM generally produces superior results for most scenes owing to the several iterations of sampling
   - DDIM too performs at par or only slightly worse than DDPM
   - The quality-speed trade-off heavily favors DDIM for practical applications
4. **Technical Innovation:**
   - Successfully adapted RePaint's DDPM-based algorithm to work with DDIM
   - Demonstrated the flexibility of diffusion frameworks
   - Showed that RePaint's inpainting strategy is sampling-method-agnostic
5. **Practical Implications:**
   - DDIM makes real-time inpainting feasible
   - User can choose between quality (DDPM) and speed (DDIM) based on needs
   - Interactive applications become possible with DDIM's speed
6. **Limitations and Observations:**
   - Both methods sometimes struggle with highly structured patterns
   - Performance varies based on the size and location of the masked region
   - Edge cases exist where DDPM's slower, more detailed sampling is preferable
   - Being generative modelling, this process is not free from hallucination. We also note that hallucination is context-dependent – the face was strange sometimes but the red car and the wire-mesh was always inpainted correctly.
7. **Future Directions:**
   - Optimizing the pipeline to have fast inpainting on edge devices

- Room for optimization in DDIM sampling parameters
- Distill the model to consume less memory for inference
- Possibility of further speed improvements while maintaining quality

This project demonstrates the significant potential of DDIM for practical image inpainting applications, while also highlighting the continued relevance of DDPM for specific use cases where quality is paramount over speed.

5 Supplementary Material

This is the link to a video recording a presentation (with motivation, approach, results) for this project: Video Presentation Link.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[2] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[3] Taesung Kwon and Jong Chul Ye. Solving video inverse problems using image diffusion models. *arXiv preprint arXiv:2409.02574*, 2024.

[4] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[5] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.