TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A
PROJECT REPORT
ON

ENHANCING ROBUSTNESS OF GENERATIVE ADVERSARIAL IMAGES FOR PREVENTING IMAGE MANIPULATION

SUBMITTED BY:

AKANKSHA GIRI (PUL076BEI004)
ANJU CHHETRI (PUL076BEI005)
ANKIT SHAHI (PUL076BEI006)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

April, 2024

Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance of the project report entitled "**Enhancing Robustness of Generative Adversarial Images for Preventing Image Manipulation**" submitted by **Akanksha Giri**, **Anju Chhetri** and **Ankit Shahi** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

....
Supervisor
Prof. Dr. Basanta Joshi
Assistant Professor
Department of Electronics and Computer
Engineering,
Pulchowk Campus, IOE, TU.

....
External examiner
Mr. Shreeniwas Sharma
Chief Executive Officer
Alternate Technology Pvt. Ltd.

Date of approval:

Copyright

Acknowledgement

We would like to express our gratitude to the Department of Electronics and Communication Engineering for providing us with the incredible opportunity to undertake this major project. This project offered us a platform to develop and enhance our technical and interpersonal skills by working collaboratively as a team. We are grateful for this opportunity, which would not have been possible without the department's support and encouragement. We are highly indebted to **Prof. Dr. Basanta Joshi** for his continuous guidance, constant supervision, and invaluable support throughout the project. His instructions and feedback were instrumental in the successful completion of this project.

Abstract

Text-guided diffusion models, particularly latent diffusion models (LDMs), pose a privacy threat due to their ability to generate highly realistic images conditioned on text prompts. This raises concerns about malicious image editing. Existing methods leverage adversarial perturbations to hinder realistic image generation by LDMs, but these methods lack robustness to JPEG compression, a common image compression algorithm. This work proposes an approach for generating adversarial perturbations that are robust to JPEG compression. We demonstrate the effectiveness of our method and explore how text prompts influence the success of adversarial attacks against text-guided diffusion models.

Keywords: *image manipulation, adversarial perturbations, adversarial attack, diffusion model, differential JPEG*

Contents

List of Figures

List of Tables

List of Abbreviations

DCT	Discrete Cosine Transform
FID	Fréchet Inception Distance
GPU	Graphics Processing Unit
GAN	Generative Adversarial Network
JPEG	Joint Photographic Experts Group
LAION	Large-scale Artificial Intelligence Open Network
LDM	Latent Diffusion Model
LPIPS	Learned Perceptual Image Patch Similarity
PGD	Projected Gradient Descent
PSNR	Peak Signal-to-Noise Ratio
QF	Quality Factor
RGB	Red Green Blue
SSIM	Structural Similarity Index Measure

1. Introduction

1.1 Background

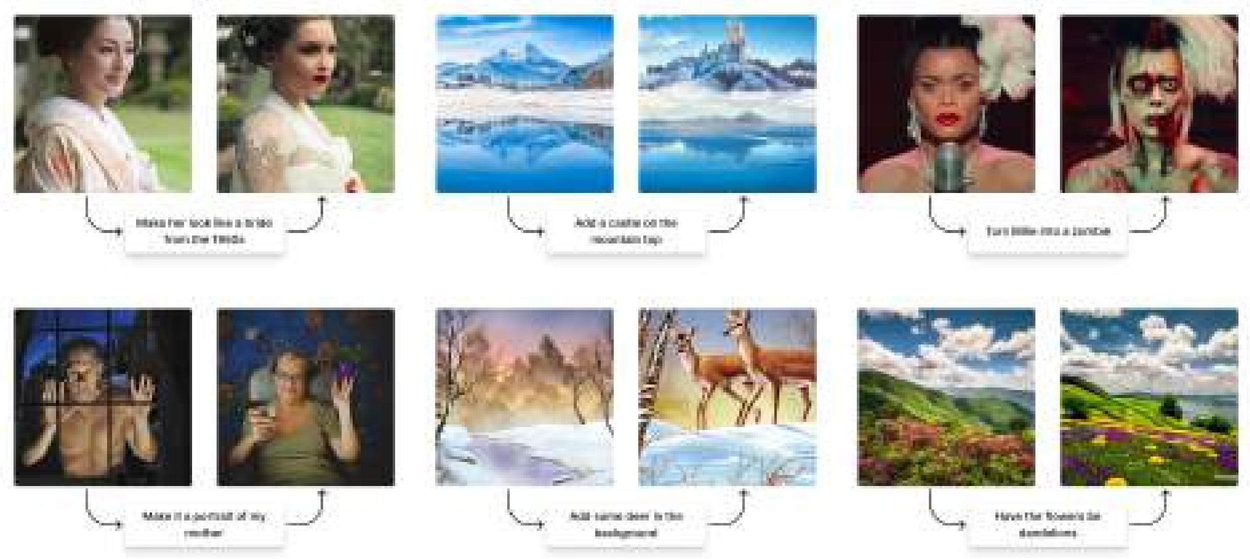

Make her look like a bride from the 1960s
Add a castle on the mountain top
Turn Billie into a zombie
Make it a portrait of my mother
Add some deer in the background
Have the flowers be dandelions

Figure 1.1: The figure represents image editing technique using text-guided diffusion model. The text below each image shows the editing prompt given to the model. The arrow illustrates the changes made to the source image, resulting in an edited image.

The fast growth of artificial intelligence (AI) technology in recent years has resulted in unparalleled capabilities, notably in image generation and manipulation [1, 2, 3, 4, 5]. Unfortunately, this progress has also given rise to an alarming increase in the misuse of AI-generated content for deceptive purposes. Malicious actors can leverage advanced image generation techniques, such as text-to-image diffusion models like Stable Diffusion [5] and generative adversarial network (GAN) like conditional GAN [6], to generate high-quality, realistic images based on textual prompts. These techniques do not necessitate extensive technical expertise from the perpetrator, lowering the barrier to entering the creation of deepfakes.

Instances of manipulated images circulating in the news, social media, and various online platforms have raised serious concerns about the potential consequences of unchecked AI manipulation [7, 8]. The consequences for targeted individuals with the edited images can be severe, including emotional distress, reputational damage, and even social isolation.

To address this problem, several methods have been put forth, many of which make use of adversarial attacks [9, 10, 11, 12] for protecting images against malicious attacks. These adversarial attacks [13, 14, 15, 16] add imperceptible perturbations to the original images. These perturbations imperceptibly alter the pixel data within the image, effectively disrupting the image representation used by the deep learning models. This manipulation disrupts the model's ability to recognize the key features or alter the decision boundaries, rendering the image unusable for deepfakes and significantly hindering malicious manipulations of images.

1.2 Problem Statement

Salman et al. [9] proposed two protection techniques, namely an encoder attack and a diffusion attack, but these protection schemes are not robust to JPEG compression as shown in [17], which is noteworthy considering it is used on 77% of all websites [18]. Das et al. [19] proposed a method for removing the adversarial perturbations where different JPEG compression levels are applied to obtain the original image.

The interaction between text and image prompts occurs through cross-attention layers in text-conditioned diffusion models [20]. To our knowledge, [21] is the only work that explores the impact of text prompts on the effectiveness of immunization methods. While text-conditioning in diffusion models introduces an additional input channel for crafting adversarial examples, its impact on adversarial attacks remains largely unexplored.

1.3 Objectives

In summary, the primary objectives of our project are:

- To develop an effective image immunization pipeline robust to JPEG compression.
- To study the impact of text prompts on immunization methods by categorizing the text prompts into four groups.

2. Literature Review

2.1 Related Work

2.1.1 Existing Techniques for Image Editing/Manipulation

2.1.1.1 GAN-based Image Manipulation Technique

DragGAN [22] is one of the techniques in image manipulation that allows users to "drag" the content of any GAN-generated images. Users only need to click a few handle points and target points on the image, and this approach will move the handle points to reach their corresponding target points precisely.
Another notable technique in GAN-based image manipulation is **MaskGAN** [23]. Given a target image, users are allowed to modify masks of the target images per the source images so that we can obtain manipulated results. Instead of directly transforming images in the pixel space, MaskGAN learns the face manipulation process as traversing on the mask manifold, thus reducing more diverse results concerning facial components, shapes, and poses.
PuppetGAN [24] learns to change specific things in real pictures, like a person's mouth expression, by using examples of these changes in computer-generated images and individual real photos during its training. The proposed PuppetGAN model correctly manipulates real images and uses only unlabeled real images and synthetic demonstrations during training.

2.1.1.2 Diffusion-based Image Manipulation Technique

DiffusionCLIP [25] enables faithful text-driven manipulation of real images by preserving important details when the state-of-the-art GAN inversion-based methods fail. It also makes possible several applications, including multi-attribute transfer, stroke-conditioned picture generation, and image translation between unseen domains.

High-Resolution Image Synthesis with Latent Diffusion Models (LDM) [5] analyzes trained diffusion models in pixel space, aiming to find a perceptually equivalent but computationally more suitable space to train high-resolution image synthesis diffusion models by separating the perceptual compression and semantic compression stages.

DreamBooth [26] an AI-powered photo booth can generate a myriad of images of the subject in different contexts with just a few images (typically 3-5) of a subject, using the guidance of a text prompt. The results exhibit natural interactions with the environment, as well as novel articulations and variations in lighting conditions, all while maintaining high fidelity to the key visual features of the subject.

Common techniques to fine-tune generated images are time-consuming, produce poorly integrated results (inpainting), or result in unexpected changes across the entire image. **Diffusion Brush** [27], a LDM-based tool to efficiently fine-tune desired regions within an AI-synthesized image.

The diffusion models can achieve image sample quality superior to the current state-of-the-art generative models [28]. GANs hold state-of-the-art, but their drawbacks make them difficult to scale and apply to new domains. Diffusion models are a class of likelihood-based models that have recently been shown to produce high-quality images while offering desirable properties such as distribution coverage, a stationary training objective, and easy scalability.

2.1.2 Prevailing Prevention Techniques against AI-powered Image Manipulation

To stop the spread of manipulated images which may spread misinformation and hamper the dignity of an individual, the first step is the prevention of such AI-powered image manipulation itself. In order to disrupt AI-powered image manipulation, unperceptive watermarks are added and watermarked images are made adversary examples for deepfake models [12]. These days, AI is also being employed to copy artistic styles to generate images, resembling those of famous artists. To tackle this style mimicry, Glaze has been developed that enables artists to apply "style cloaks" to their art before sharing online [10]. These cloaks apply barely perceptible perturbations to images, and when used as training data, mislead generative models that try to mimic a specific artist. Nightshade is another powerful tool for content owners to protect their intellectual property by providing strong disincentive against unauthorized data training [11].

Another effective prevention measure against AI-powered editing is image immunization. Photoguard [9] involves "immunizing" images by adding imperceptible adversarial perturbations. These added perturbations disrupt the inner workings of the targeted diffusion model and thus prevent it from producing realistic modifications of the immunized images.

2.1.3 Failure Cases of Prevention Techniques

JPEG compression is the main enemy of image immunization techniques involving adversarial perturbations [17, 19]. Adversarial perturbations are not robust to JPEG compression, which poses a major problem because of the common usage and availability of JPEG [17]. Cao et al. [29] devised a new platform for evaluating the effectiveness of imperceptible perturbations as a protective measure by purifying a perturbed image with consistency-based losses. The Shield defense framework [19] utilizes JPEG compression to effectively “compress away” such pixel manipulation. Shield immunizes DNN models from being confused by compression artifacts by “vaccinating” a model.

2.2 Related Theory

2.2.1 Adversarial Machine Learning

Adversarial Machine Learning is the study of effective machine learning techniques against an adversarial opponent [30]. This field of study focuses on attacks on machine learning algorithms and defenses against such attacks. These attacks are carried out by adding perturbations that manipulate the input-output mapping learned by the neural network [31]. Perturbed inputs are indistinguishable from the original input. The presence of such adversarial examples is the inherent weakness of the neural network which causes it to work well on naturally occurring data but can drastically change a network’s response when the points lie outside of the data distribution [13, 32].

In an adversarial environment, it is anticipated that the adversarial opponent tries to cause machine learning to fail in many ways. An adversary can poison a model’s classifications, often in a highly targeted manner by crafting input data that has similar features to normal data but is capable of fooling the model. For example, Dalvi et al. [33] demonstrated how linear classifiers used in spam filters could be fooled by adding some ”good words” into the spam emails without affecting the readability of the spam message.

Such adversaries have also been used in avoiding the detection of attacks, causing benign input to be misclassified as attack input, launching targeted attacks or searching a classifier to find blind spots in the algorithm. Even non-linear classifiers were found to be susceptible to such attacks when Biggio et al. [34] demonstrated the first gradient-based attacks on such machine-learning models. When deep neural networks were gaining popularity, Szegedy et al. [31] demonstrated that deep neural networks could be fooled by adversaries, again using a gradient-based attack to craft adversarial perturbations. Although adversarial machine learning has been more prevalent in academia, it is being adopted by companies in assessing the robustness of machine learning models and minimizing the risk of adversarial attacks [35].

2.2.2 Diffusion Models

Diffusion models are probabilistic models which learn the data distribution by normally denoising the normally distributed variable. It maps datum(usually image data) to the latent space using the Markov chain [36]. It consists of a forward process(diffusion process) and a backward process(reverse diffusion process). During the forward process, we add noise gradually from the known distribution to the approximate posterior given as q($x_{1:T}$—x$_0$). Ultimately, the x_T is a pure Gaussian noise. During the reverse diffusion processes the network learns to reverse the diffusion process to generate new data. Starting from the pure gaussian noise p(x_T) := $\mathcal{N}(x_T,\ 0,\ I)$, the models learns the joint probability distribution $p_\theta(x_{0:T})$ as

$$p_\theta(\mathbf{x}_{0:T}) \coloneqq p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \coloneqq p(\mathbf{x}_T)\prod_{t=1}^{T}\mathcal{N}(\mathbf{x}_{t-1};\mu_\theta(\mathbf{x}_t,t),\sum_\theta(\mathbf{x}_t,t)) \qquad (2.1)$$

We only learn the time-dependent parameters of the Gaussian transition. Markov chain asserts that the transition to x_{t-1} from x_t depends only on the previous timestep.

$$p_\theta(x_{t-1}|x_t) \coloneqq \mathcal{N}(x_{t-1};\mu_\theta(x_t,t),\sum_\theta(x_t,t)$$

2.2.3 Cross-Attention

Stable diffusion relies on a mechanism called cross-attention to integrate information from textual prompts with the emerging image during the generation process [20]. Stable diffusion takes two inputs i.e. one is the textual prompt that describes the desired image content using words and another is the image, often filled with noise that the model progressively refines. Cross-attention allows the model to analyze the text prompt and understand its meaning. In other words, it examines the current state of the image being generated and identifies the correspondence between the textual element and image features. Based on these connections, the model adjusts the image to better match the description in the prompt.

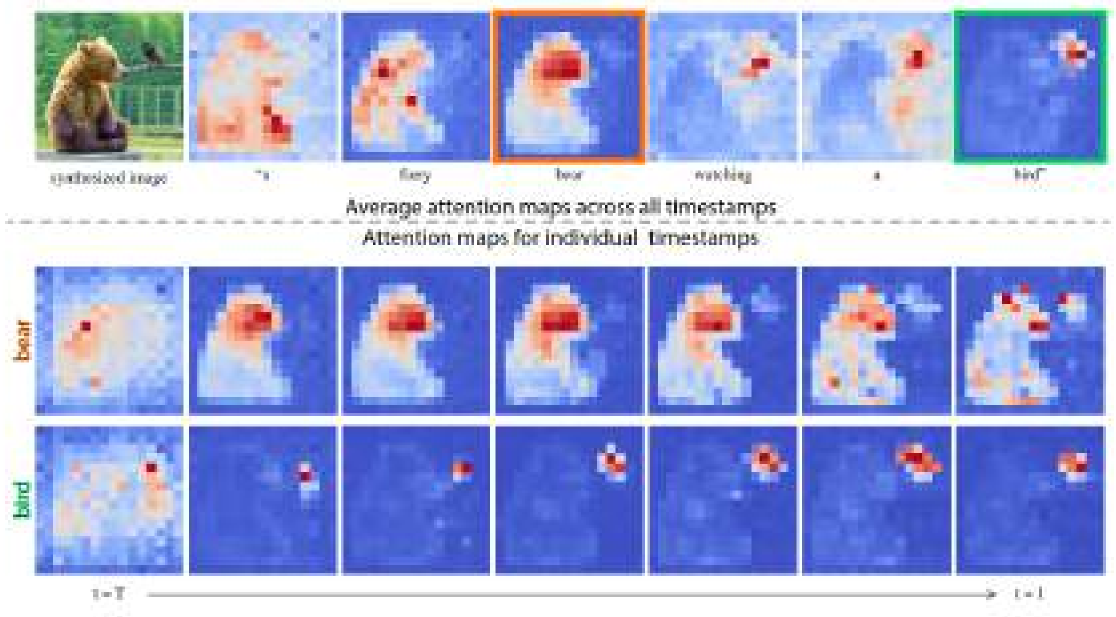

synthesized image
"a
furry
bear
watching
a
bird"
Average attention maps across all timestamps
Attention maps for individual timestamps
bear
bird
t=T
t=1

Figure 2.1: Cross-attention mapping in a text-conditioned diffusion-based image generation. The top row shows the average cross marks for each word in the prompt used to synthesize the image on the left. The bottom row shows the attention maps from diffusion steps concerned with the word 'bear' and 'bird'. Figure from [20]

Cross-attention maps is given by

$$M = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

Hence, cross-attention acts like a bridge enabling the text prompt to guide the image generation step-by-step. This is what allows Stable Diffusion to create realistic and creative images based on textual descriptions. In this case, the cross-attention is used to condition transformers inside a UNet layer with a text prompt for image generation.

2.2.4 Gradient Checkpointing

Gradient checkpointing is a technique used during deep learning, specifically to reduce the memory footprint required for training large neural networks. It comes at the cost of a slight increase in computation time [37].

Training deep neural networks involves a process called backpropagation, where the model calculates gradients (directions for improvement) for each parameter based on the error. Backpropagation requires storing all the intermediate activations (outputs) from the forward pass through the network. For models with many layers or limited memory, storing all these activations can be very memory-intensive. This is where gradient checkpointing comes into the picture. This technique tackles the memory issue by strategically not storing all intermediate activations.

Instead, it selects specific points in the computational graph (the network architecture) and saves the activations at those points, called checkpoints. During backpropagation, the parts of the graph between the saved checkpoints are recomputed instead of relying on the stored activations. By strategically choosing checkpoints, gradient checkpointing significantly reduces memory usage, allowing us to train larger models or overcome memory limitations on our hardware. The downside is a slight increase in computation time because of the additional calculations during backpropagation when recomputing sections of the graph. Overall, gradient checkpointing is a valuable tool for training memory-intensive deep learning models by making efficient use of available memory.

2.2.5 Evaluation Metrics

1. **SSIM (Structural Similarity Index):** SSIM serves as a metric to evaluate the structural similarity between two images. It is a full reference metric that needs a reference image and a processed image, both taken from the same image capture. Usually, the processed image is compressed.

   The Structural Similarity Index, which ranges from -1 to +1, is determined by this system between two provided photos [38]. When the two provided photographs have a value of +1, it means they are almost identical, and when the value is -1, it means the two are significantly different. These numbers are frequently changed to fall between 0 and 1, where the extremes have the same meaning. It measures the similarity between two images based on 3 key features: Luminance, Contrast and Structure.

   (a) Luminance : Luminance is measured by averaging over all pixel values.

   $$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

   Luminance comparison function:

   $$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

   C_1 is a constant to ensure stability.

(b) Contrast: It is the standard deviation of all pixel values.

$$\sigma_x = \left(\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)^2 \right)^{\frac{1}{2}}$$

Contrast comparison function:

$$c(x, y) = \frac{2\sigma_x \sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

(c) Structure: This is obtained by dividing the input signal by its standard deviation to get a result with a unit standard deviation. It ensures a more reliable comparison.

$$\frac{x - \mu_x}{\sigma_x}$$

Structure comparison function:

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}$$

where σ_{xy} is defined as,

$$\sigma_{xy} = \tfrac{1}{N-1} \textstyle\sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y)$$

SSIM score is given by:

$$SSIM(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} \cdot [s(x, y)]^{\gamma}$$

where $\alpha > 0$, $\beta > 0$ and $\gamma > 0$ denote the relative importance of each of the metrics.

2. **LPIPS (Learned Perceptual Image Patch Similarity):** LPIPS, a learned perceptual metric, is utilized to assess the perceptual similarity between the original and adversarial images. This metric goes beyond pixel-level comparisons, capturing perceptual nuances important for human visual understanding. Perceptual similarity is a byproduct of visual representations that have been adjusted to be predictive of significant structural elements in the outside world, rather than a unique purpose unto itself [39].

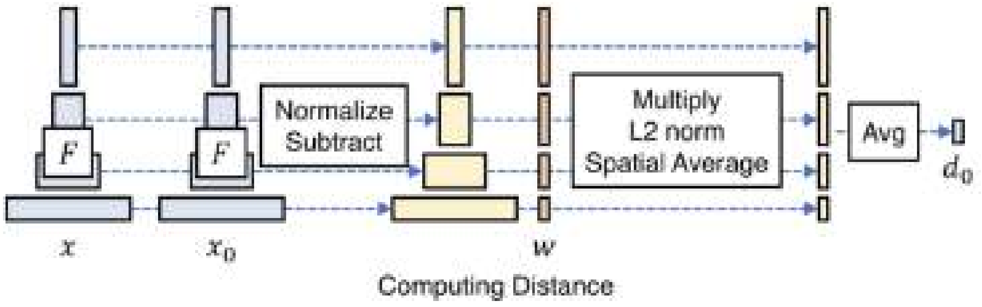


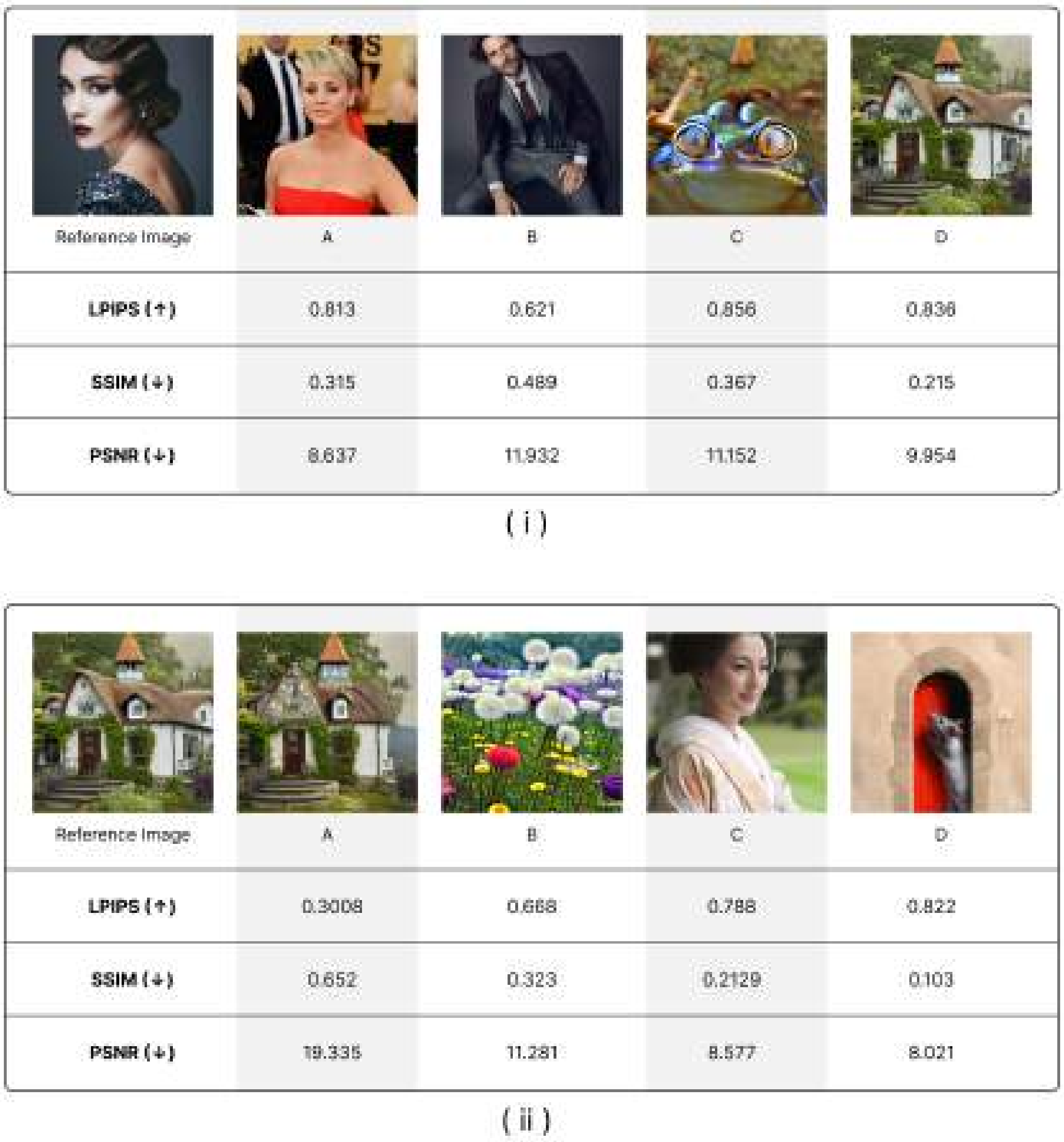
Figure 2.2: Schematic diagram of network computing LPIPS distance from two patches x and x_0 for a given network $\mathcal{F}$. Figure from [39]

The distance between reference x and distorted patch x_0 is calculated using this workflow and the equation below with a network $\mathcal{F}$. As shown in [39], the feature stack is extracted from $\mathcal{L}$ layers and unit-normalize in the channel dimension. Then the activation is scaled channel-wise and the l_2 distance is computed.

$$d(x, xo) = \sum_{l} \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (\hat{y}^l_{hw} - \hat{y}^l_{0hw})||^2$$

For our analysis, we use the Alex Net [40] variant as proposed by [39].

3. **Comparison of Evaluation Metrics**



	Reference Image	A	B	C	D
LPIPS (↑)		0.813	0.621	0.856	0.836
SSIM (↓)		0.315	0.489	0.367	0.215
PSNR (↓)		8.637	11.932	11.152	9.954

(i)

	Reference Image	A	B	C	D
LPIPS (↑)		0.3008	0.668	0.788	0.822
SSIM (↓)		0.652	0.323	0.2129	0.103
PSNR (↓)		19.335	11.281	8.577	8.021

(ii)

Figure 2.3: LPIPS, SSIM, PSNR score of images A, B, C, D in comparison with reference image

The arrow indicates the direction of increasing dissimilarity between images. LPIPS measures the perceptual similarity between two images, with higher scores suggesting greater dissimilarity. Conversely, lower SSIM and PSNR scores indicate more dissimilarity between images.

In (i) of Figure 1.3, because the reference image and B share common features like color palette, their LPIPS score is low, while their SSIM and PSNR scores are high compared to A, C, and D. This implies that the reference image and B are more similar than the other images.

Similarly in (ii) of Figure 1.3, the reference image and C have no common features, resulting in a high LPIPS score and low SSIM and PSNR score compared to A, which shares many features with the reference image. This suggests that A is more similar to the reference image than C.

2.2.6 Technology Used

2.2.6.1 PyTorch

PyTorch [41, 42] is a popular machine learning library, originally developed by Meta AI, based on the Torch library. It provides a wide range of tools that support development in computer vision, NLP, and other areas. It has been specially optimized for deep learning using CPUs and GPUs gaining popularity in the research community. Two important features of PyTorch are tensor computing with strong GPU acceleration which enables efficient computation of large and complex models and deep neural networks based on a taped-based automatic differentiation system that automatically computes gradients for backpropagation during training. PyTorch keeps writing models easy and productive by internally handling the complexities inherent to ML. Another important feature of PyTorch is that, unlike other deep learning frameworks that use static computation graphs, it uses dynamic computation graphs instead providing users with greater flexibility.

2.2.6.2 Latent Diffusion Models

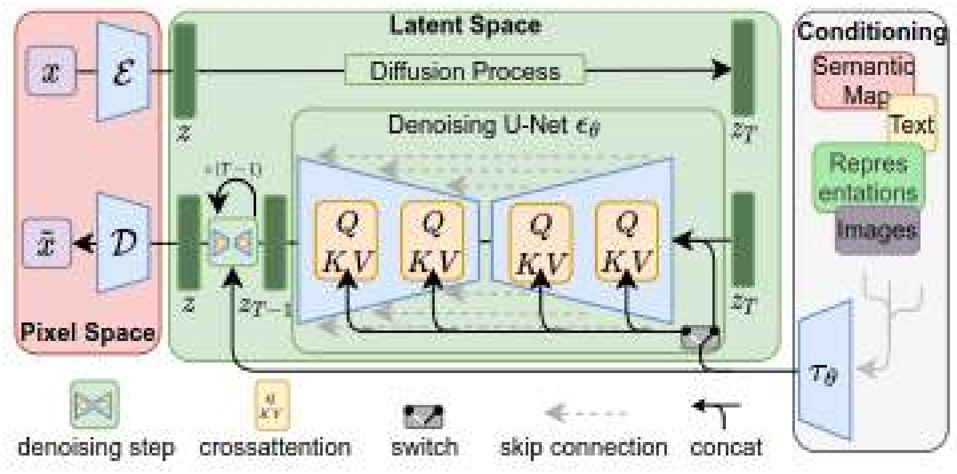


Figure 2.4: Conditioning LDM via cross attention mechanism Figure from [5]

As diffusion models directly work in the image space, their optimization is computationally demanding [5]. To solve this, LDMs were introduced which utilize the diffusion process in the latent space instead of the image space. Input image x_0 is first mapped to latent representation using a powerful pretrained autoencoders [5], $z = \mathcal{E}(x_0)$, where is $\mathcal{E}$ is the encoder. We then apply the standard diffusion process in the z.

Another network is trained for denoising by minimizing the given loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\mathbf{z}_0,\epsilon\sim\mathcal{N}(0,1)} \left(\|\epsilon - \epsilon_\theta(\mathbf{z}_{t+1}, t)\|_2^2\right) \tag{2.2}$$

After the denoising network is trained, latent representation of the new image $\tilde{z}$ is obtained. This representation is decoded using a using a decoder network $\mathcal{D}(\tilde{z})$ to get a new image.

2.2.6.3 Projected Gradient Descent (PGD) for Generating Adversarial Images

The PGD attack falls under the category of a white-box attack, which assumes that the attacker possesses detailed knowledge of the model's gradients. PGD attempts to find the perturbation that maximizes or minimizes the loss of a model on a particular input while keeping the size of the perturbation smaller than a specified amount referred to as epsilon. This constraint is usually expressed as the $L2$ or $L\infty$ norm of the perturbation and it is added so the content of the adversarial example is the same as the unperturbed sample — or even such that the adversarial example is imperceptibly different to humans.

Algorithm 1 Projected Gradient Descent (PGD) Attack

```
1: Input: Input image $\mathbf{x}$, target image $\mathbf{x}_{targ}$, neural network $g$, perturbation budget $\epsilon$, step size $k$, number of steps $N$
2: Initialize adversarial perturbation $\delta \leftarrow 0$ and immunized image $\mathbf{x}_{adv} \leftarrow \mathbf{x}$
3: for $n = 1$ to $N$ do
4:     Generate output: $\mathbf{x}_{out} \leftarrow g(\mathbf{x}_{adv})$
5:     Compute mean squared error: $l \leftarrow \|\mathbf{x}_{targ} - \mathbf{x}_{out}\|_2^2$
6:     Update adversarial perturbation: $\delta \leftarrow \delta + k \cdot sign(\nabla_{\mathbf{x}_{adv}} l)$
7:     $\delta \leftarrow min(max(\delta, -\epsilon), \epsilon)$
8:     Update the immunized image: $\mathbf{x}_{adv} \leftarrow \mathbf{x}_{adv} - \delta$
9: end for
10: Return: $\mathbf{x}_{adv}$
```

3. Proposed Methodology

3.1 Feasibility Study

In our feasibility study, we delved into significant research papers to comprehend the landscape of adversarial image generation. Photoguard [9] stood out, presenting an approach to mitigate risks associated with large diffusion models. The proposed methods, namely encoder attack and diffusion Attack, aimed to immunize images against malicious editing by introducing adversarial perturbations. This foundational research served as a starting point for our project, inspiring us to explore and build upon these immunization techniques.

Furthermore, we explored the insights provided by [13] which shed light on early attempts to understand the generation of adversarial images. The authors argued that the vulnerability of neural networks to adversarial perturbations is rooted in their linear nature, emphasizing the impact of nonlinearity and overfitting. This understanding is crucial as we navigate the landscape of adversarial attacks, guiding our exploration of immunization strategies against potential vulnerabilities identified in neural network models.

3.2 Requirement Analysis

The success of our project hinged on the availability of substantial computational resources, specifically a minimum of 16 GB NVIDIA Tesla T4 GPU [1]. This requirement was deemed essential for using backpropagation algorithm in the diffusion model effectively, ensuring a higher degree of effectiveness in the generation of adversarial images.

Given our computational limitations, we opted for Stable Diffusion version 1, hosted in Hugging Face [2]. It became evident that its performance fell short, particularly in generating realistic images of human subjects. This realization underscored the necessity for a more powerful diffusion model. The identified need for enhanced computational capabilities and a more sophisticated model was imperative to achieve the project's objectives and generate high-quality adversarial images that closely resembled human subjects.

[1] https://www.nvidia.com/en-us/data-center/tesla-t4/

[2] https://huggingface.co/runwayml/stable-diffusion-v1-5

3.3 Dataset

We use InstructPix2Pix Dataset [43] for benchmarking our results. InstructPix2Pix combines two large pretained models, a large language model and a text-to-image model, to generate a dataset for training a diffusion model to follow written image editing instructions. Since our objective is to prevent such AI manipulation, we found this dataset to be most suitable for our use case.

InstructPix2Pix's image editing model is trained on a generated dataset that consists of 454,445 examples. Each example contains an input image, an editing instruction and an output edited image. They provide two versions of the dataset, one in which each pair of edited images is generated 100 times and the best examples are chosen based on CLIP metrics and one in which the examples are randomly chosen. We chose to work with clip-filtered dataset which was about 436GB in size.

3.4 Differentiable JPEG compression

JPEG compression initially changes the image into a luma-chroma color space known as YCbCr. It reduces the resolution of the chroma channels, processes each 8x8 block through a two-dimensional discrete cosine transform, and quantizes each block.

The block diagram for JPEG compression is given below:

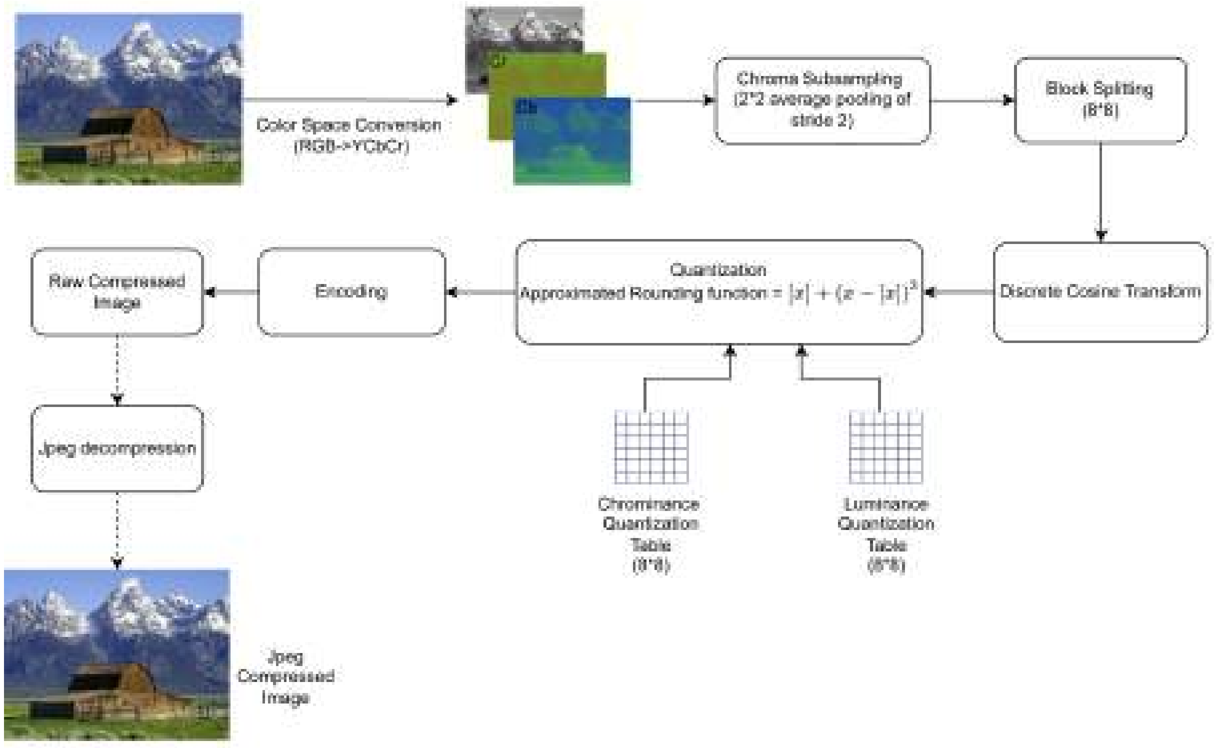

Color Space Conversion (RGB->YCbCr)
Y
Cr
Cb
Chroma Subsampling (2*2 average pooling of stride 2)
Block Splitting (8*8)
Discrete Cosine Transform
Quantization
Approximated Rounding function = $\lfloor x \rceil + (x - \lfloor x \rceil)^3$
Encoding
Raw Compressed Image
Jpeg decompression
Chrominance Quantization Table (8*8)
Luminance Quantization Table (8*8)
Jpeg Compressed Image

Figure 3.1: JPEG Encoding-Decoding Pipeline. The original image undergoes a series of transformations for compressed storage and retrieval. Encoding utilizes lossy compression techniques to discard less perceptible image information, achieving significant file size reduction. Decoding reverses this process to reconstruct an approximation of the original image.

The following section outlines the process for executing each stage utilizing differentiable operations as per [44].

- **Color Space Conversion:**

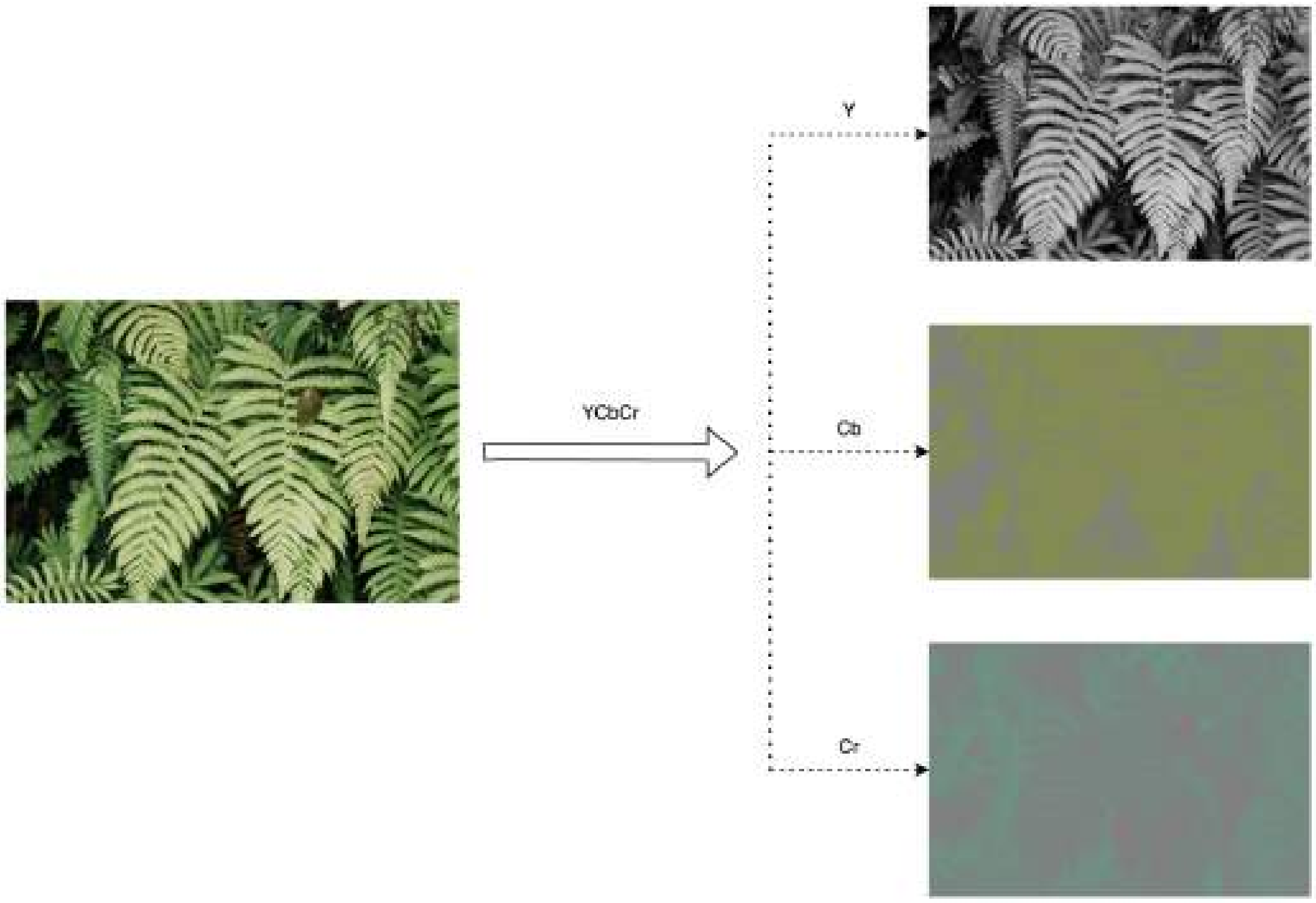


Figure 3.2: Color space conversion from RGB to YCbCr

In Figure 3.2, Y (Luminance) channel presents minute details of the original image while the Cb and Cr (Chrominance) channel does not provide much information to the human. To exploit this property of human visual system and changes are made to color information without affecting quality of image perceived. Most of the images available are present in the RGB (Red, Green, and Blue) color space to convert them into YCbCr space we can use the following formula:

Most images are stored in the RGB (Red, Green, and Blue) color space. For JPEG we need to convert it into YCbCr format. In this format, luminance information is stored as a single component (Y) and chrominance information is stored as two color-difference components (Cb and Cr). For conversion we can utilize the following formula:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

This color space conversion is an affine transformation.

- **Chroma Subsampling:** Human eyes are more sensitive to brightness than color information which can also be viewed in figure 3.2. This characteristic of the human visual system allows us to modify the components without significantly impacting perceived image quality. Y (Luminance) channel preserves the fine details of the original image, while the Cb and Cr (Chrominance) channels contain primarily color information. JPEG downsamples the Cb and Cr channels by a factor of two in each dimension.

- **Block Splitting:** Each channel is divided into 8×8 blocks of 64 pixels. The subsequent steps of the compression algorithm deals with each 8×8 block independently.

- **Discrete Cosine Transformation:** First, each value must be subtracted with 128 to make the value range from -128 to $+127$. DCT transforms an 8×8 block of pixels into linear combination of 64 patterns which are given by Figure 3.3. The DCT yields a weight matrix indicating how much each base image contributes to the formation of the source image.

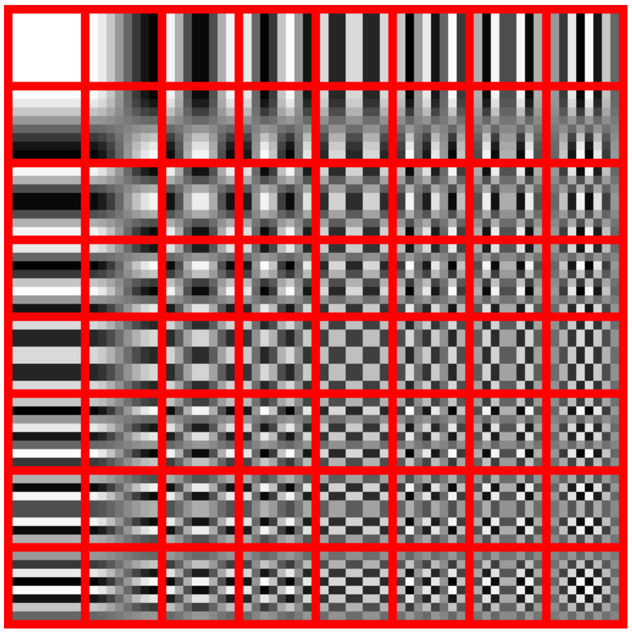


Figure 3.3: Discrete Cosine Transformation in JPEG algorithm

- **Quantization:** Human eyes are not that well equiped to view the high frequency elements in an image. After chroma subsampling this is the step where information is lost. The weight matrix is divided by a precalculated quantization table. *Quantization matrix* $Q \in 8 \times 8$ is used to create $D_{i,j} = \lfloor \frac{C_{i,j}}{Q_{i,j}} \rceil$, $\lfloor . \rceil$ represents rounding to the nearest integer. Within the JPEG quantization matrix, higher values are typically concentrated in the bottom right quadrant. Mathematically, these higher Q values in the lower-right region result in a significant number of elements in the DCT (Discrete Cosine Transform) domain (represented by D) becoming zero. Consequently, this effectively removes high-frequency details from the image, such as sharp edges and intricate textures.

- **Approximating Rounding Function:** The non-differentiability of the rounding function used in JPEG quantization impedes the application of gradient-based optimization techniques for crafting adversarial examples. To address this challenge, differentiable surrogate functions are used that closely approximate the rounding behavior. Examples of suitable surrogate functions include linear, Fourier, or polynomial functions. The specific approximation technique utilized in our experiments is detailed in Section 4.2.

- **Decoding:** For obtaining the RGB image, all the steps are performed in reverse order. Inverse quantization involves element-wise multiplication with the quantization matrix. This process is lossy because original values cannot be recovered perfectly. Inverse discrete cosine transformation reverses the DCT transformation applied during compression. Finally, we perform YCbCr to RGB conversion which is an affine transformation involving matrix multiplication and bias addition.

4. Proposed Experimental Setup

We build upon the work of [9], who proposed two attack strategies i.e. *encoder attack* and *diffusion attack*. This serves as the baseline for our experiments conducted within the PyTorch framework. We use open source version of stable diffusion hosted in the Hugging face[1].

4.1 Dataset

Throughout our experiments, we use the InstructPix2Pix image editing dataset[2]. For preparing our dataset, we use the first two shards of the InstrutPix2Pix dataset which are 15 GB each. We select a total of 800 examples in total from both shard folders using random sampling. Then we extract the image URL, edit prompt, and input caption from the prompt.json file into a text file which will be used as input to our diffusion models. Based on the diffusion results, we select 200 examples from randomly chosen 800 examples. We again filter this number to 60 examples for running our experiments based on the qualitative analysis.

4.2 JPEG-Resilient Adversarial Attack

To make adversarial images robust to JPEG compression we combine encoder attack with differentiable JPEG. [44] provided a detailed explanation on the differentiable implementation of the JPEG compression algorithm. Experimental results provided by [45] show that polynomial approximation of the rounding function yields the best result. So, we implemented our attack using the polynomial function given as:

$$R(x) = \lceil x \rfloor + (x - \lceil x \rfloor)^3 \tag{4.1}$$

As shown in [44], the ensemble of models employing different quality factors gives more robust results. We utilize this method in the existing attack model.

Building on the results of [9], we employ their hyperparameter settings for our study.

[1]https://huggingface.co/docs/diffusers/en/api/pipelines/stable_diffusion/img2img
[2]http://instruct-pix2pix.eecs.berkeley.edu/

Seed	Strength	Guidance	Number of inference steps
9222	0.7	7.5	50

Table 4.1: Stable Diffusion Hyperparameters

To make adversarial images robust to JPEG compression, we combine stable diffusion network with a differentiable JPEG block (diff_jpeg).

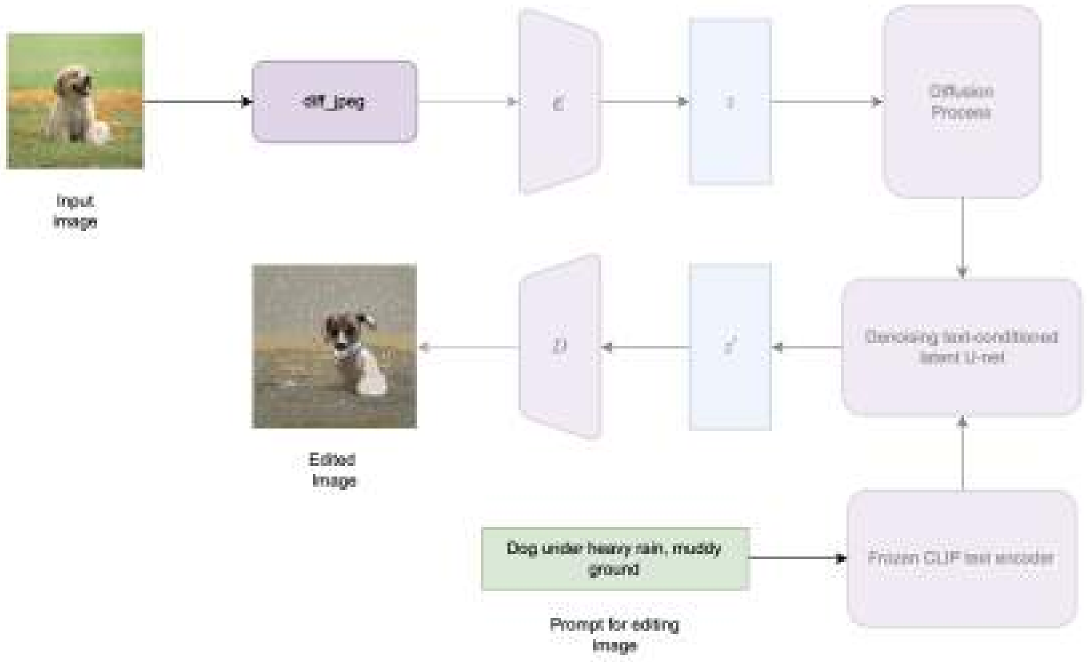


Figure 4.1: Overview of the proposed JPEG-resilient attack. The input image is first compressed using a differentiable JPEG algorithm. The compressed representation is then fed into the LDM to learn perturbations robust to JPEG compression.

The diff_jpeg block performs $\mathcal{F}(\mathbf{JPEG}(x, q))$ where, $JPEG$(x, q) compresses x to q quality factor(QF) and gives it to stable diffusion pipeline. The pipeline then generates a text-guided edited image.

4.2.1 JPEG-Resilient Diffusion Attack

The whole diffusion process is used for optimizing 4.2 to learn adversarial perturbations.

$$\delta = \underset{\|\delta\|_p \leq \epsilon}{\arg\min} \mathcal{L}(JPEG_\theta(\mathcal{F}(x + \delta)), x_{targ}) \tag{4.2}$$

For our experiments, we assign the value zero to every element within x_{targ}.

Epsilon	QFs	Step Size	Number of Steps	Number of Denoising Steps
0.1	[25,50,75,100]	0.006	100	4

Table 4.2: JPEG-Resilient diffusion attack hyperparameters

Algorithm 2 JPEG-Resilient Diffusion Attack on Stable Diffusion Model

```
1: **Input:** Input image $\mathbf{x}$, target image $\mathbf{x}_{targ}$, Differential JPEG $diff\_jpeg$, Stable Diffusion model $f$, perturbation budget $\epsilon$, step size $k$, number of steps $N$, mean $\mu$
2: Initialize adversarial perturbation $\delta \leftarrow 0$ and immunized image $\mathbf{x}_{immu} \leftarrow \mathbf{x}$
3: $QF = [qf_1, qf_2, ..., qf_N]$
4: **for** $n = 1$ to $N$ **do**
5:     grad = 0
6:     **for** qf in QF **do**
7:         Perform JPEG compression: $\mathbf{x}_{jpeg} \leftarrow diff\_jpeg(\mathbf{x}_{immu}, qf)$
8:         Generate image using diffusion model: $\mathbf{x}_{out} \leftarrow f(\mathbf{x}_{jpeg})$
9:         Compute mean squared error: $l \leftarrow \|\mathbf{x}_{targ} - \mathbf{x}_{out}\|_2^2$
10:        Update grad: $grad+ = \nabla_{\mathbf{x}_{immu}} l$
11:    **end for**
12:    Update adversarial perturbation: $\delta_{diffusion} \leftarrow \delta_{diffusion} + k \cdot sign(\mu(grad))$
13:    $\delta_{diffusion} \leftarrow clip(\delta_{diffusion}, -\epsilon, \epsilon)$
14:    Update the immunized image: $\mathbf{x}_{immu} \leftarrow \mathbf{x}_{immu} - \delta_{diffusion}$
15: **end for**
16: **Return:** $\mathbf{x}_{immu}$
```

4.2.2 JPEG-Resilient Encoder Attack

Image representation obtained from encoder block is used for optimizing 4.3 to learn adversarial perturbations.

$$\delta = \underset{\|\delta\|_p \leq \epsilon}{\arg\min} \mathcal{L}(JPEG_\theta(\mathcal{F}(x + \delta)), z_{targ}) \tag{4.3}$$

For our experiments, we have assigned the value zero to every element in z_{targ}.

Epsilon	QFs	Step Size	Number of Steps
0.06	[25,50,75,100]	0.02	1000

Table 4.3: JPEG-Resilient encoder attack hyperparameters

Algorithm 3 JPEG-Resilient Encoder Attack on Stable Diffusion Model

```
1: Input: Input image $\mathbf{x}$, target image $\mathbf{x}_{targ}$, Stable Diffusion model encoder $\mathcal{E}$, Differential JPEG $diff\_jpeg$, perturbation budget $\epsilon$, step size $k$, number of steps $N$, Mean $\mu$
2: Compute the embedding of the target image: $\mathbf{z}_{targ} \leftarrow \mathcal{E}(\mathbf{x}_{targ})$
3: Initialize adversarial perturbation $\delta \leftarrow 0$ and immunized image $\mathbf{x}_{immu} \leftarrow \mathbf{x}$
4: $QF = [qf_1, qf_2, ..., qf_N]$
5: for $n = 1$ to $N$ do
6:     grad = 0
7:     for qf in QF do
8:         Perform JPEG compression: $\mathbf{x}_{jpeg} \leftarrow diff\_jpeg(\mathbf{x}_{immu})$
9:         Compute the embedding of the immunized image: $\mathbf{z} \leftarrow \mathcal{E}(\mathbf{x}_{jpeg})$
10:        Compute mean squared error: $l \leftarrow \|\mathbf{z}_{targ} - \mathbf{z}\|_2^2$
11:        Update grad: $grad+ = \nabla_{\mathbf{x}_{immu}} l$
12:    end for
13:    Update adversarial perturbation: $\delta_{encoder} \leftarrow \delta_{encoder} + k \cdot sign(\mu(grad))$
14:    $\delta_{encoder} \leftarrow clip(\delta_{encoder}, -\epsilon, \epsilon)$
15:    Update the immunized image: $\mathbf{x}_{immu} \leftarrow \mathbf{x}_{immu} - \delta_{encoder}$
16: end for
17: Return: $\mathbf{x}_{immu}$
```

4.3 Impact of Text Prompt on the Attack Performance

During the generation of adversarial images, we can provide different types of text prompts to the network. Fusion of text and image information occurs through cross-attention layers, and the generated cross-attention maps influence the spatial layout of the generated image [20]. To investigate the influence of the text prompt on the generated image, we categorize the prompts into four types as shown in Table 4.4

Prompt Type	Description
Content Prompt	Provides a description of the image content
Irrelevant prompt	Text unrelated to the image content.
Random token prompt	Sequence of tokens randomly sampled from a uniform distribution.
Null prompt	No text prompt is provided.

Table 4.4: Prompt types for adversarial image

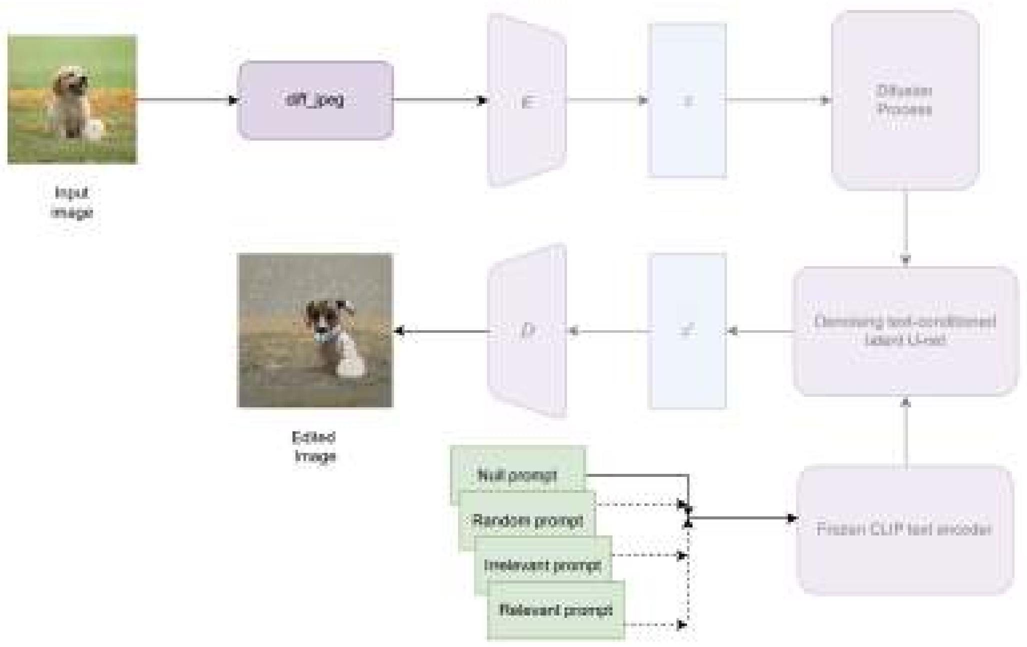


Figure 4.2: Overview of the proposed architecture to test the impact of text prompt on the adversarial attack performance.

For content prompts, we leverage the Large-scale Artificial Intelligence Open Network (LAION) captions used by InstructPix2Pix. Shan et al. [11] provided a feature space visualization of the semantic frequency for all the common concepts. Based on this analysis, we identify "Vase" and "Polyester" to be some of the least frequent words in the LAION-aesthetic, which fit the irrelevant prompt criteria.

Prompt Type	**Prompt Example**
Content Prompt	Red glazed cake with pomegranate and caramel triangle
	Little mouse at a red door
	Traditional hut at sunrise after a snowfall, Grindelwald, Canton of Bern, Switzerland, Europe
	...
Irrelevant Prompt	Vase Polyester
Random Token	[49406 34 2323 343 998 ...]
Null Prompt	" "

Table 4.5: Example of different prompt types

5. Results And Discussion

5.1 Limitation of Photoguard

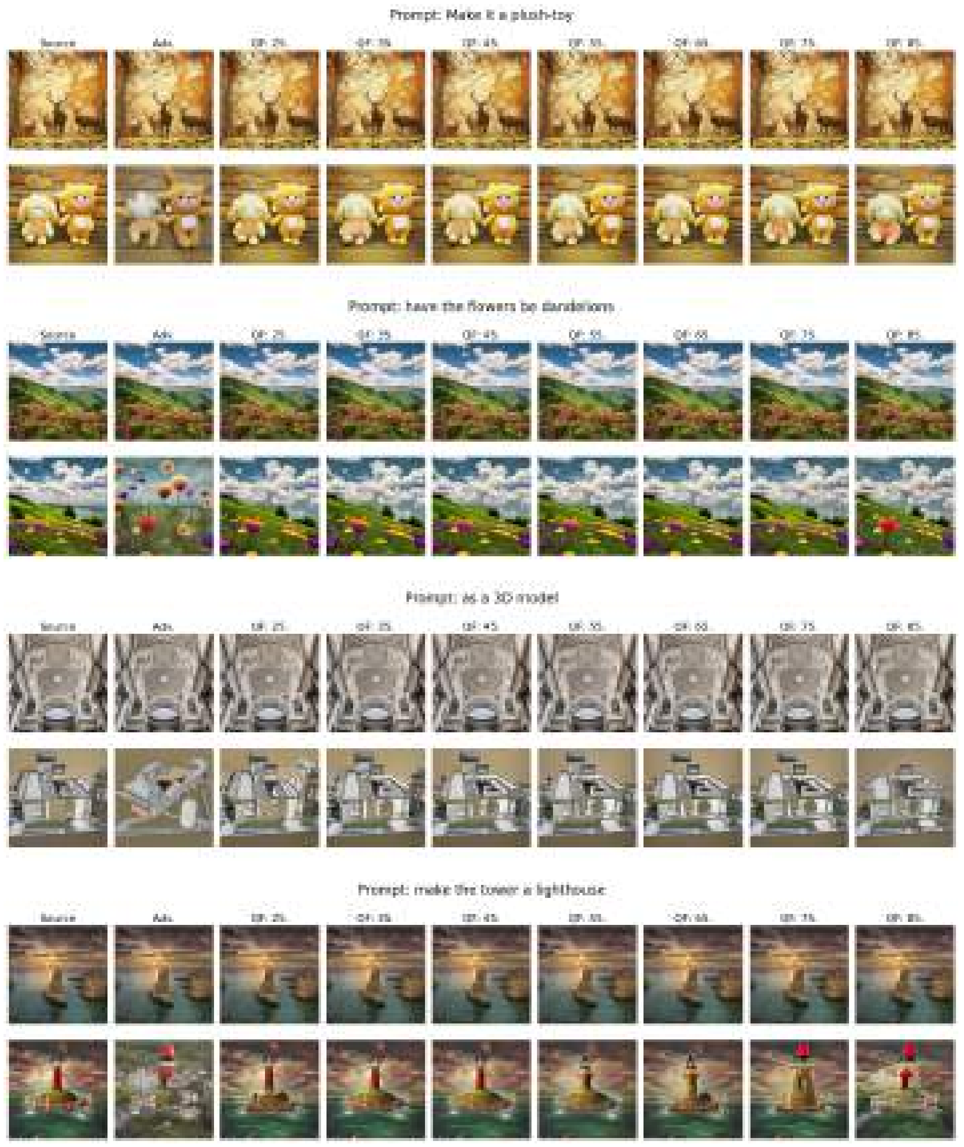

Prompt: Make it a plush-toy
Source
Adv.
QF: 25
QF: 35
QF: 45
QF: 55
QF: 65
QF: 75
QF: 85
Prompt: have the flowers be dandelions
Prompt: as a 3D model
Prompt: make the tower a lighthouse

Figure 5.1: The above figures demonstrate the effects of different levels of JPEG compression on the adversarial images generated using encoder attack. The prompt above each figure acts as a text guide for editing the image. JPEG compression removes the perturbations from compressed adversarial images undoing the whole immunization and making it vulnerable to image manipulations.

JPEG compression weakens the protection mechanisms applied using encoder attack. This vulnerability allows the adversary to potentially remove the perturbations by simply applying JPEG compression algorithm to the protected image. Figure 5.1 visually represents this effect at a QF of 75, where the edits applied on the adversarial image appears similar to the edits applied on the source image. Quantitave analysis in the Table 5.1 further supports this observation. To assess robustness across different QFs, we evaluted QFs ranging from 25 to 85 at an interval of 10.

5.2 Robustness to JPEG Compression

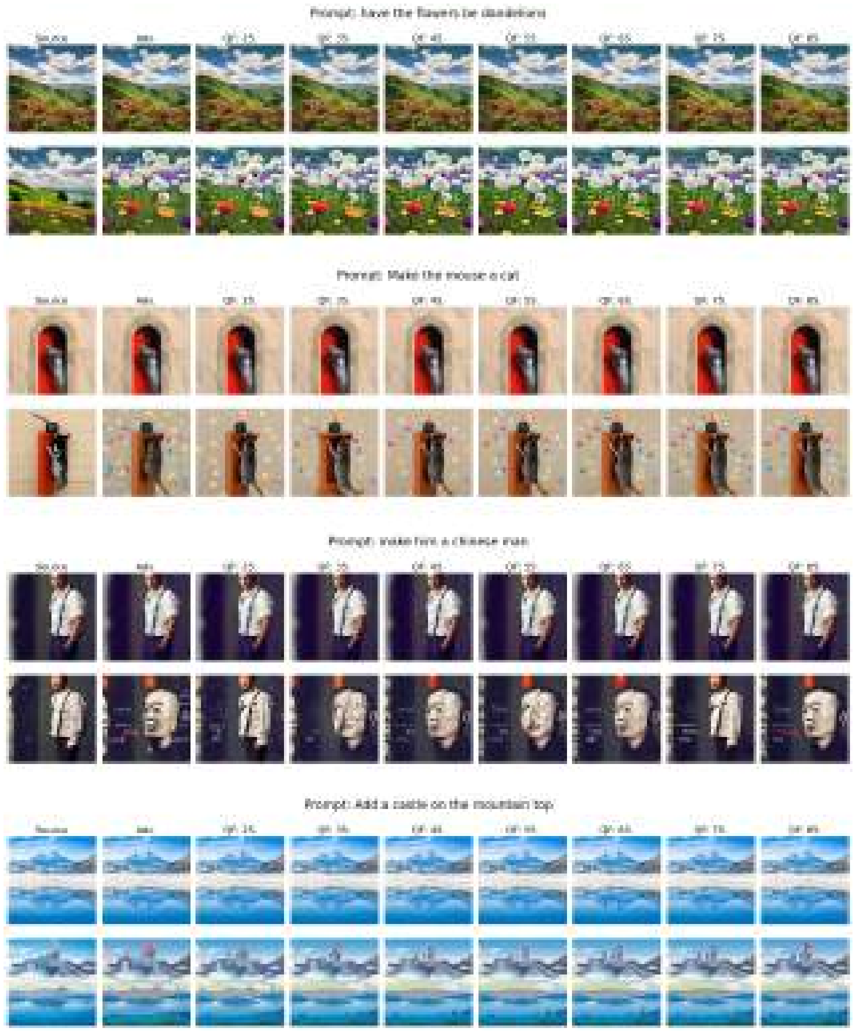

Prompt: have the flowers be dandelions
Source
Adv
QF: 25
QF: 35
QF: 45
QF: 55
QF: 65
QF: 75
QF: 85
Prompt: Make the mouse a cat
Prompt: make him a chinese man
Prompt: Add a castle on the mountain top

Figure 5.2: The above figure demonstrate the robustness of JPEG-Resilient Diffusion Attack against different levels of JPEG compression. Prompts act as a text guide for image editing.

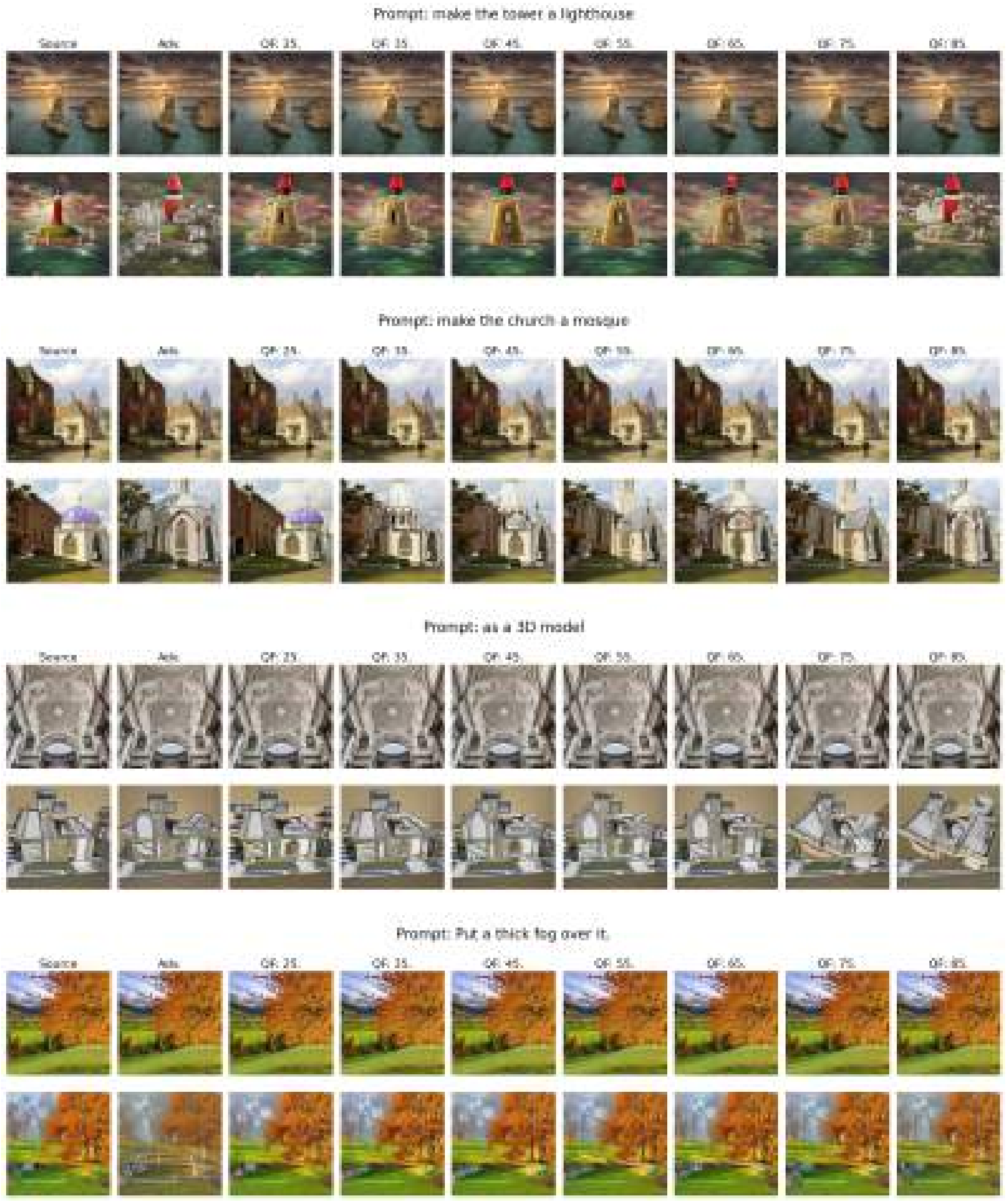

Prompt: make the tower a lighthouse
Prompt: make the church a mosque
Prompt: as a 3D model
Prompt: Put a thick fog over it.
Source
Adv.
QF: 25.
QF: 35.
QF: 45.
QF: 55.
QF: 65.
QF: 75.
QF: 85.

Figure 5.3: The above figure demonstrates the robustness of JPEG-Resilient Encoder Attack against different levels of JPEG compression. The prompts act as a text guide for image editing.

Figure 5.2 and 5.3 present the result of our methods. Edits applied on the compressed adversarial images at different QFs using text-guided diffusion model are similar to those applied on the adversarial image without any compression. This is further supported by the quantitative analysis in Table 5.1.

5.2.1 Quantitative Analysis of Experimental Results

Metric	Method	Quality Factor							
		25	35	45	55	65	75	85	100
LPIPS (↑)	Diffusion attack	0.41 ± 0.11	0.40 ± 0.10	0.41 ± 0.10	0.42 ± 0.10	0.43 ± 0.11	0.44 ± 0.10	0.46 ± 0.10	0.51 ± 0.10
	Encoder attack	0.34 ± 0.12	0.32 ± 0.10	0.32 ± 0.11	0.34 ± 0.11	0.35 ± 0.12	0.40 ± 0.12	0.48 ± 0.11	**0.63 ± 0.13**
	JPEG-Resilient Diffusion attack (Ours)	**0.43 ± 0.10**	**0.44 ± 0.10**	**0.44 ± 0.11**	**0.45 ± 0.10**	**0.45 ± 0.10**	**0.45 ± 0.10**	0.46 ± 0.10	0.50 ± 0.10
	JPEG-Resilient Encoder attack (Ours)	0.33 ± 0.11	0.33 ± 0.11	0.33 ± 0.11	0.36 ± 0.12	0.38 ± 0.12	0.41 ± 0.12	**0.48 ± 0.12**	**0.63 ± 0.13**
SSIM (↓)	Diffusion Attack	0.62 ± 0.11	0.63 ± 0.11	0.61 ± 0.11	0.60 ± 0.11	0.60 ± 0.11	0.58 ± 0.10	0.56 ± 0.10	0.52 ± 0.11
	Encoder Attack	0.68 ± 0.10	0.69 ± 0.09	0.69 ± 0.10	0.66 ± 0.11	0.65 ± 0.11	0.59 ± 0.11	0.51 ± 0.10	**0.40 ± 0.10**
	JPEG-Resilient Diffusion attack (Ours)	**0.59 ± 0.11**	**0.58 ± 0.10**	**0.58 ± 0.12**	**0.57 ± 0.11**	**0.56 ± 0.11**	**0.55 ± 0.12**	0.54 ± 0.12	0.50 ± 0.11
	JPEG-resilient encoder attack (Ours)	0.69 ± 0.10	0.68 ± 0.10	0.67 ± 0.09	0.64 ± 0.10	0.62 ± 0.11	0.59 ± 0.11	**0.50 ± 0.10**	**0.40 ± 0.10**

Table 5.1: LPIPS (↑) and SSIM (↓) score for different attack methods across different JPEG compression levels. Arrows next to the metrics denote the direction of increasing dissimilarity.

Table 5.1 shows the experimental results of different immunization techniques measured across different quality factors. LPIPS (↑) and SSIM (↓) scores are used to quantify the perceptual and structural similarity between the edits generated from unimmunized and immunized images across different JPEG compression levels. Ideally, a greater distance between the images, reflected by higher LPIPS and smaller SSIM values is desired. For QF below 85, JPEG-resilient diffusion attack generates perturbations most robust to JPEG compression as this method targets the entire LDM pipleline which includes encoder, diffusion, and decoder network instead of just encoder network as in JPEG-resilient encoder attack. By using an ensemble of models running on different QFs, we obtain gradients that capture the effects of compression across a wide range (QFs 25-100). This ensemble approach allows us to generate adversarial perturbations that are robust across varying range of QFs.

5.2.2 Impact of Text Prompt on Adversarial Attack Performance

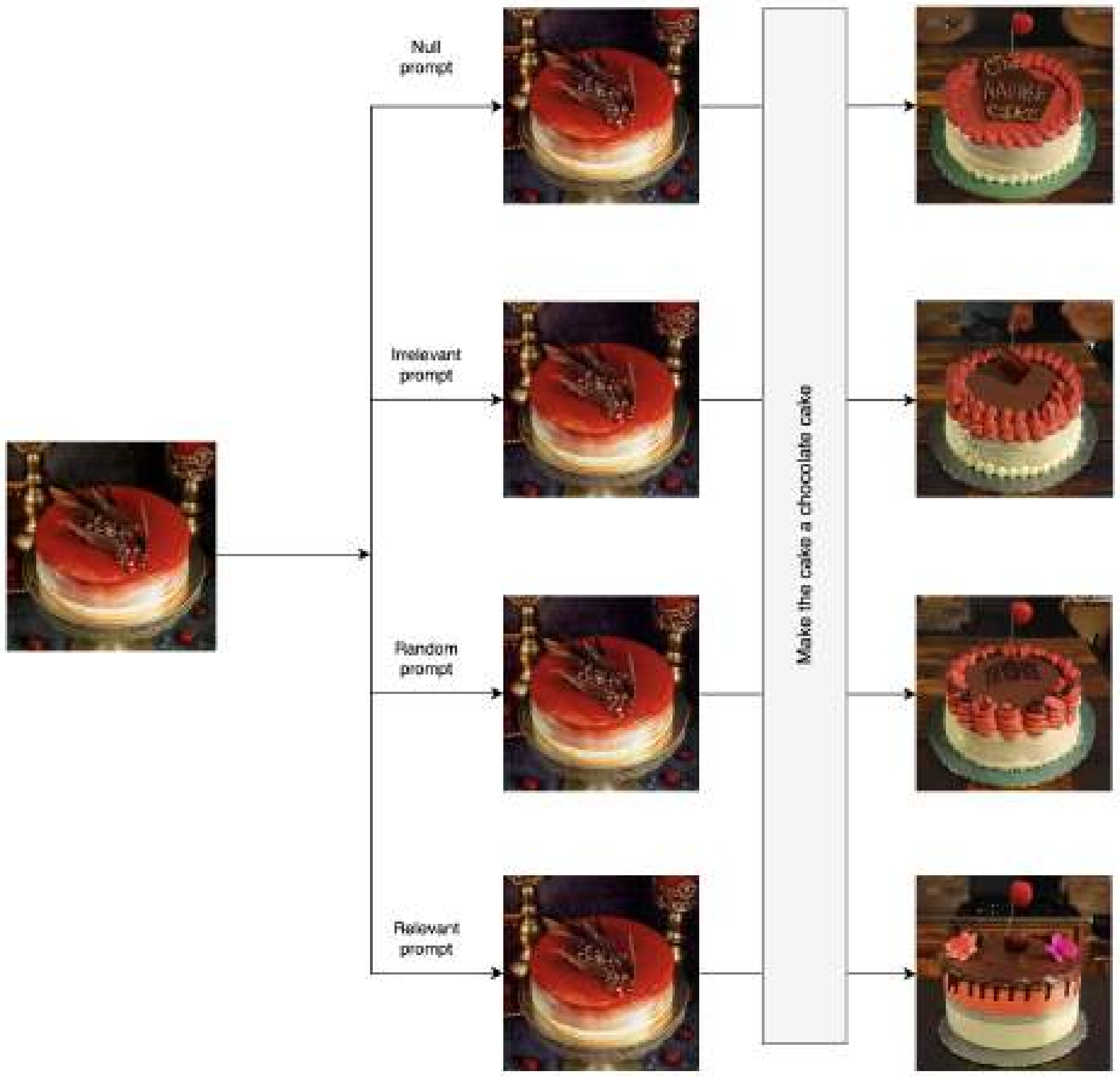


Figure 5.4: This figure demonstrates the influence of prompt type on adversarial image manipulation. We used four prompt types (null, irrelevant, relevant, random token) for eight original images for generating adversarial images. Edits are applied to these adversarial images where the order follows the null prompt result in the first row first column and the following irrelevant prompt in the second image of the same row. This pattern continues for images in the second row for content and random token prompts.

Our adversarial image generation process leverages 4-step denoising steps. Quantitative analysis demonstrates that the null prompt achieves superior performance compared to other prompt types. This dominance might be attributed to the discrepancy between prompts used during adversarial generation and subsequent image editing. The null prompt restricts the process to the vision branch of the stable diffusion pipeline. This allows the network to focus solely on the image information, potentially leading to better results compared to prompts.

Prompt Type	LPIPS (↑)	SSIM (↓)	PSNR (↓)
Relevant	0.47 ± 0.11	0.53 ± 0.10	17.39 ± 1.85
Irrelevant	0.48 ± 0.12	**0.52 ± 0.10**	17.18 ± 1.94
Null	**0.51 ± 0.11**	0.52 ± 0.11	**17.05 ± 2.13**
Random Token	0.47 ± 0.12	0.53 ± 0.12	17.42 ± 2.35

Table 5.2: Different image quality metrics measuring similarity scores between edited images generated from the original image and adversarial image. We used four prompt types (null, irrelevant, relevant, random token) for original images for generating adversarial images.

5.3 Challenges

5.3.1 Computation

We conducted our experiments using online platforms like Google Colab and Kaggle. Most of our experiments are performed in Kaggle which provides 30 hours of compute unit per week.

PyTorch 2.0 offered a memory-efficient attention implementation, scaled dot product attention(SDPA) [1], without requiring any extra dependencies such as xFormers. But the results varied significantly with different GPU architectures and the benchmarking details show that there isn't much performance boost in the NVIDIA Tesla T4 Processor, which we used.

5.3.2 Challenges in Metrics reflecting Human Perception

Determining the optimal metric that accurately captures human perception in image processing is inherently subjective and contingent on the specific context. While metrics like LPIPS, SSIM and PSNR have become popular metrics in generative AI for images, they often fall short in fully capturing human perception. Human perception goes beyond basic similarity. Factors like style consistency, naturalness within the target domain, and preservation of emotional cues are crucial, and these metrics often struggle to account for them. There is also a discrepancy in human perception and perceptual similarity. A high/low metric score might simply indicate the model's ability to mimic existing images within a dataset, not

[1] https://huggingface.co/docs/diffusers/optimization/torch2.0#benchmark

necessarily its ability to generate outputs that humans find aesthetically pleasing or visually interesting.

6. Conclusion

This work presents a technique for generating adversarial examples that are robust to JPEG compression. We achieve this by incorporating the differentiable approximation of the JPEG algorithm in the attack pipeline. This enables the calculation of gradients across the entire pipeline, allowing us to craft adversarial perturbations that are robust to subsequent JPEG compression. To further enhance the robustness across various compression factors, we employ an ensemble of models running on different QFs. Our comprehensive quantitative evaluation, encompassing QFs ranging from 25 to 100, demonstrates that this approach significantly improves the robustness of adversarial examples against JPEG compression.

Furthermore, we investigated the potential influence of text prompts on the generated adversarial images. We categorized the prompts into four groups and conducted a quantitative analysis to assess their impact. Our findings reveal that using no prompt (null prompt) leads to superior image editing results compared to other prompts. This advantage likely stems from the mismatch between prompts used for generating adversarial examples and those used for editing.

7. Limitations and Future Enhancement

7.1 Four Denoising Steps

A potential limitation of our experiments is the use of only 4 denoising steps for adversarial image generation. As suggested by [43], the structure of the image in cross-attention layer is determined in the early stages of the diffusion process. With only four denoising steps we might not be able to fully capture the influence of text prompts on the image. This raises the question of whether 4 denoising steps are sufficient to draw definitive conclusions about the impact of text prompts on adversarial vulnerability. To understand this, a deeper understanding with a wider range of denoising steps might be necessary to fully understand the interplay between text prompt and image.

7.2 Higher Perturbation Budget

Our method requires a larger perturbation budget to generate effective adversarial examples compared to classification models. This can be attributed to the nature of generative models. Unlike classification models, generative models learn by reconstructing training data, which allows them to have a much richer feature space [46]. This could potentially lead to a smaller adversarial space for images. Consequently, a higher perturbation budget is necessary to craft successful attacks within this limited space.

7.3 Attack Transferability to Other Models

We focus on adversarial attacks in white box settings, where the model architecture and gradients are known. While this simplifies the attack process, real-world scenarios often include black-box settings where the model details are not known. Research by [47, 48] demonstrates potential adversarial perturbation transferability. To further explore this concept, we could investigate adversarial attacks on generative models, specifically focusing on both intra-family and inter-family transferability.

References

[1] B. Kawar, S. Zada, O. Lang, O. Tov, H. Chang, T. Dekel, I. Mosseri, and M. Irani, “Imagic: Text-based real image editing with diffusion models,” in *Conference on Computer Vision and Pattern Recognition 2023*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2210.09276

[2] B. Li, X. Qi, T. Lukasiewicz, and P. Torr, “Controllable text-to-image generation,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/1d72310edc006dadf2190caad5802983-Paper.pdf

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Commun. ACM*, vol. 63, no. 11, p. 139–144, oct 2020. [Online]. Available: https://doi.org/10.1145/3422622

[4] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=B1xsqj09Fm

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* Los Alamitos, CA, USA: IEEE Computer Society, jun 2022, pp. 10 674–10 685. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01042

[6] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[7] “AI-generated naked child images shock Spanish town of Almendralejo — bbc.com,” https://www.bbc.com/news/world-europe-66877718, [Accessed 26-03-2024].

[8] “Children making AI-generated child abuse images, says charity — bbc.com,” https://www.bbc.com/news/technology-67521226, [Accessed 26-03-2024].

[9] H. Salman, A. Khaddaj, G. Leclerc, A. Ilyas, and A. Madry, “Raising the cost of malicious AI-powered image editing,” in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML’23. JMLR.org, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.06588

[10] S. Shan, J. Cryan, E. Wenger, H. Zheng, R. Hanocka, and B. Y. Zhao, “Glaze: Protecting artists from style mimicry by Text-to-Image models,” in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 2187–2204. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/shan

[11] S. Shan, W. Ding, J. Passananti, H. Zheng, and B. Y. Zhao, “Prompt-specific poisoning attacks on text-to-image generative models,” *ArXiv*, vol. abs/2310.13828, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:264426741

[12] L. Lv, “Smart watermark to defend against deepfake image manipulation,” in *2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS)*, 2021, pp. 380–384.

[13] I. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *International Conference on Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[14] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, “Adversarial examples are not bugs, they are features,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/e2c420d928d4bf8ce0ff2ec19b371514-Paper.pdf

[15] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=BJm4T4Kgx

[16] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, “Adversarial machine learning,” in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, ser. AISec ’11. New York, NY, USA: Association for Computing Machinery, 2011, p. 43–58. [Online]. Available: https://doi.org/10.1145/2046684.2046692

[17] P. Sandoval-Segura, J. Geiping, and T. Goldstein, "JPEG compressed images can bypass protections against ai editing," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2304.02234

[18] w3techs, "Usage statistics of JPEG for websites," https://w3techs.com/technologies/details/im-jpeg, 2020.

[19] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, "Shield: Fast, practical defense and vaccination for deep learning using JPEG compression," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining.* New York, NY, USA: ACM, 2018, p. 9.

[20] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or, "Prompt-to-prompt image editing with cross attention control," 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.01626

[21] B. Zheng, C. Liang, X. Wu, and Y. Liu, "Improving adversarial attacks on latent diffusion model," 2024.

[22] X. Pan, A. Tewari, T. Leimkühler, L. Liu, A. Meka, and C. Theobalt, "Drag your gan: Interactive point-based manipulation on the generative image manifold," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.10973

[23] C.-H. Lee, Z. Liu, L. Wu, and P. Luo, "MaskGAN: Towards diverse and interactive facial image manipulation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 5548–5557.

[24] B. Usman, N. Dufour, K. Saenko, and C. Bregler, "PuppetGAN: Cross-domain image manipulation by demonstration," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9449–9457.

[25] G. Kim, T. Kwon, and J. C. Ye, "Diffusionclip: Text-guided diffusion models for robust image manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 2426–2435. [Online]. Available: https://doi.org/10.48550/arXiv.2110.02711

[26] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *2023*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Los Alamitos, CA, USA: IEEE Computer Society, jun 2023, pp. 22 500–22 510. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.02155

[27] P. Gholami and R. Xiao, “Diffusion brush: A latent diffusion model-based editing tool for ai-generated images,” *ArXiv*, vol. abs/2306.00219, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2306.00219

[28] P. Dhariwal and A. Nichol, “Diffusion models beat gans on image synthesis,” 2021.

[29] B. Cao, C. Li, T. Wang, J. Jia, B. Li, and J. Chen, “IMPRESS: Evaluating the resilience of imperceptible perturbations against unauthorized data usage in diffusion-based generative AI,” in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. [Online]. Available: https://openreview.net/forum?id=RRSltzPc7w

[30] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, “Adversarial machine learning,” in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, ser. AISec ’11. New York, NY, USA: Association for Computing Machinery, 2011, p. 43–58. [Online]. Available: https://doi.org/10.1145/2046684.2046692

[31] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *International Conference on Learning Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[32] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” 2019.

[33] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, “Adversarial classification,” in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’04. New York, NY, USA: Association for Computing Machinery, 2004, p. 99–108. [Online]. Available: https://doi.org/10.1145/1014052.1014066

[34] B. Biggio, B. Nelson, and P. Laskov, “Poisoning attacks against support vector machines,” in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML’12. Madison, WI, USA: Omnipress, 2012, p. 1467–1474.

[35] “Google Responsible AI Practices – Google AI — ai.google,” https://ai.google/responsibility/responsible-ai-practices/, [Accessed 05-04-2024].

[36] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” 2015. [Online]. Available: https://proceedings.mlr.press/v37/sohl-dickstein15.html

[37] T. Chen, B. Xu, C. Zhang, and C. Guestrin, “Training deep nets with sublinear memory cost,” *CoRR*, vol. abs/1604.06174, 2016. [Online]. Available: http://arxiv.org/abs/1604.06174

[38] P. Datta, “All about Structural Similarity Index (SSIM): Theory + Code in PyTorch — medium.com,” https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e, [Accessed 26-03-2024].

[39] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun 2018, pp. 586–595. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00068

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[41] “PyTorch — pytorch.org,” https://pytorch.org/, [Accessed 26-03-2024].

[42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[43] T. Brooks, A. Holynski, and A. A. Efros, “Instructpix2pix: Learning to follow image editing instructions,” in *CVPR*, 2023.

[44] R. Shin and D. Song, "JPEG-resistant adversarial images," in *NIPS 2017 Workshop on Machine Learning and Computer Security*, vol. 1, 2017, p. 8.

[45] C. Reich, B. Debnath, D. Patel, and S. Chakradhar, "Differentiable JPEG: The Devil is in the Details," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024.

[46] Z. Zhao, J. Duan, K. Xu, C. Wang, R. Z. Z. D. Q. Guo, and X. Hu, "Can protective perturbation safeguard personal data from being exploited by stable diffusion?" 2023.

[47] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," 2017.

[48] J. Gu, X. Jia, P. de Jorge, W. Yu, X. Liu, A. Ma, Y. Xun, A. Hu, A. Khakzar, Z. Li, X. Cao, and P. Torr, "A survey on transferability of adversarial examples across deep neural networks," 2023.

Appendices

A Appendix A

A.1 Demo User Interface

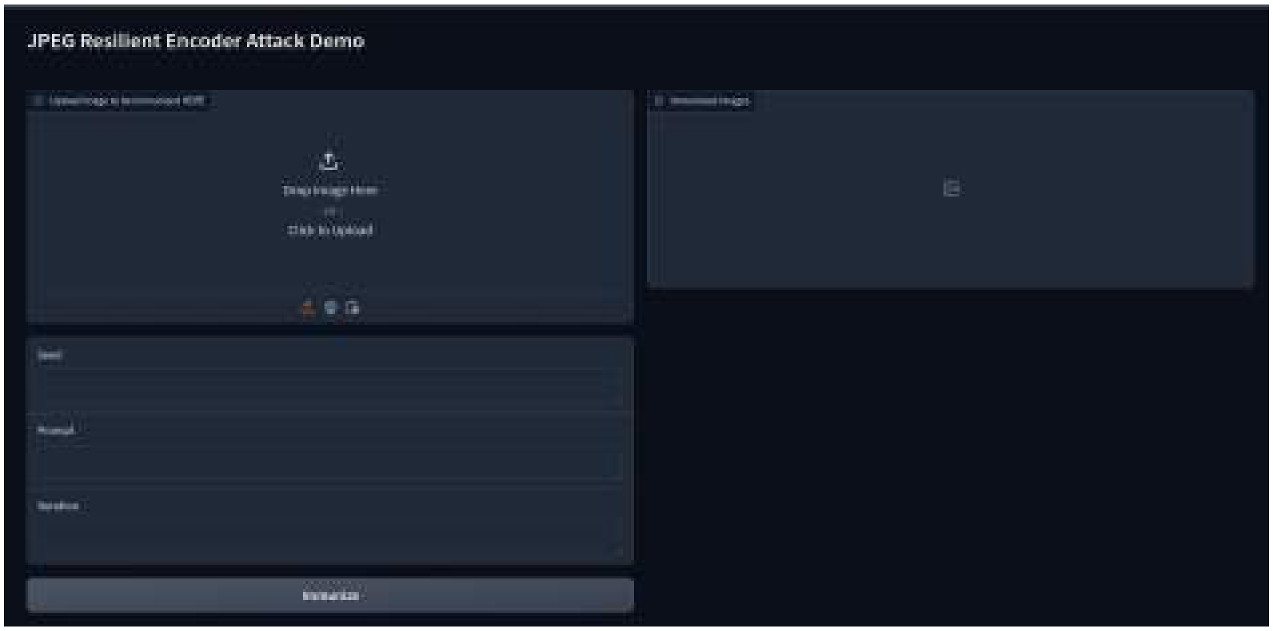


Figure A1: Interactive User Interface for Immunizing Images using JPEG-Resilient Encoder Attack

Figure A1 shows a simple user interface(UI) developed in Gradio [1] to demonstrate the core idea of our project. The UI is easy to use. A user can upload the image that they want to immunize and provide seed value, prompt and number of iterations as input to the model. After pressing the “immunize” button, the model runs the JPEG-resilient encoder attack on the input image and based on the hyperparameters and input parameters, it returns the immunized image as an output which the user can download locally.

The input parameters fed to the model determine the quality of the attack. The pattern of adversarial perturbations added to the input images varies in every run, resulting in a discrepancy in the immunized image. The seed value maintains consistency and ensures the

[1]https://www.gradio.app/

reproducibility of results. Similarly, the strength of the adversarial attack is determined by the number of iterations. The higher the number of iterations, the stronger the attack will be. Additionally, the user can input a prompt to increase the strength of the attack if s/he is sure what prompts the adversary may provide to manipulate the image. Otherwise, the prompt is optional.

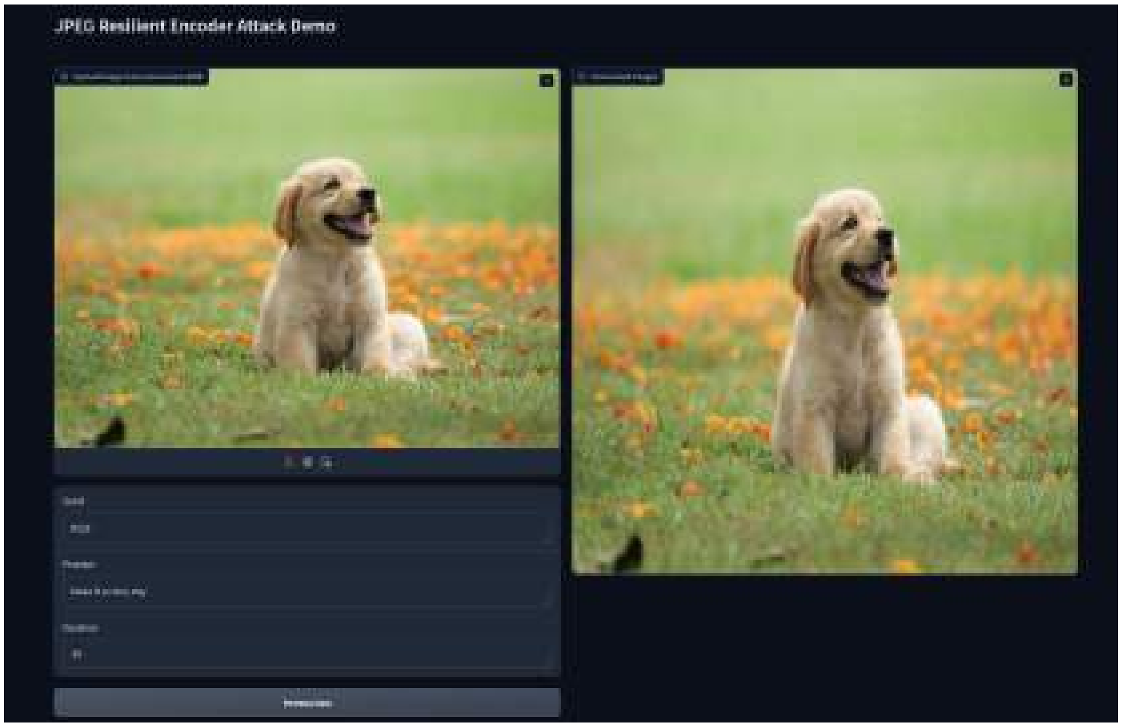


Figure A2: Running demo application for immunizing an image

A.2 Adversarial Image generation in different iteration

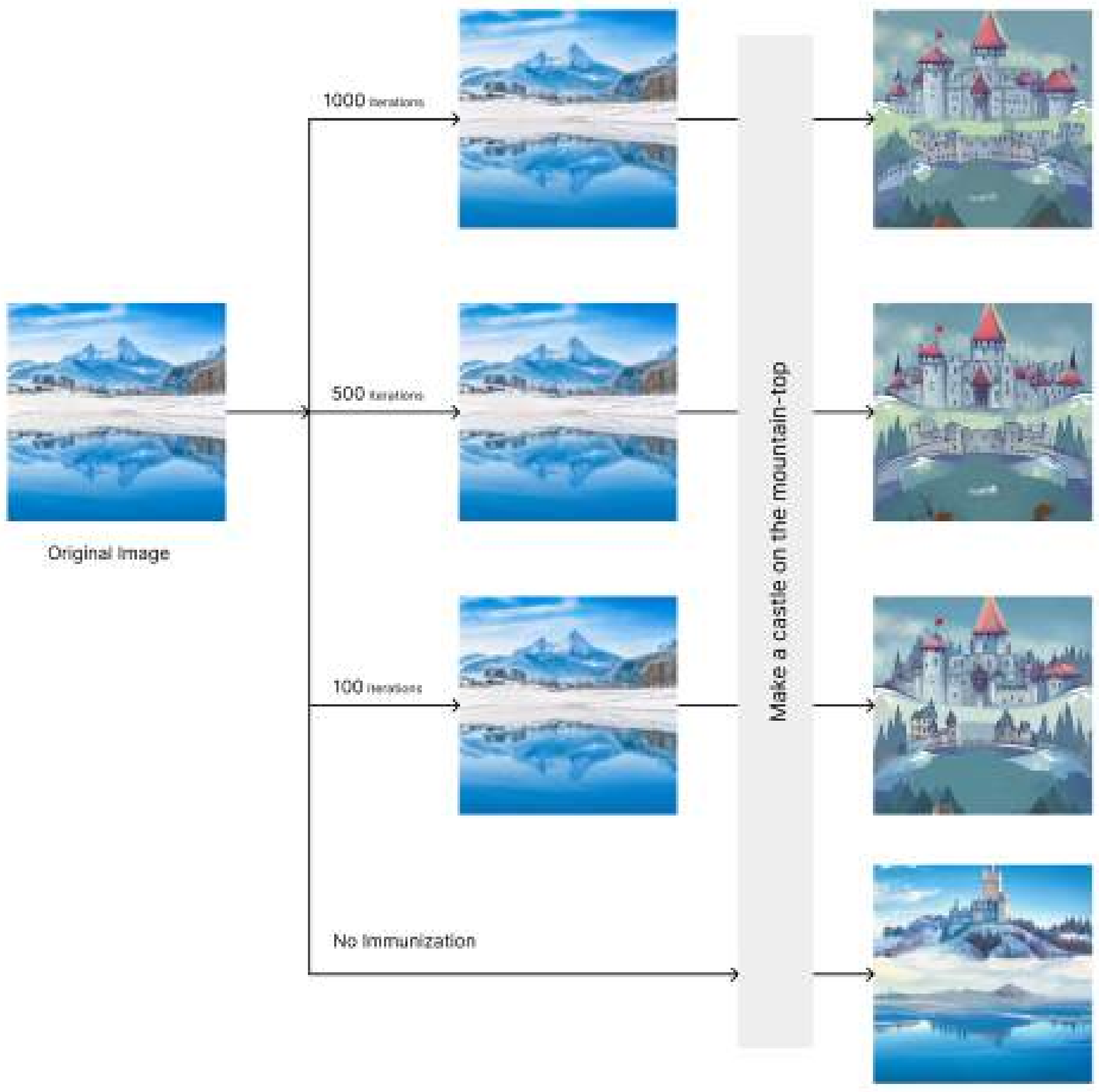


Figure A3: Adversarial image generation with different iteration in encoder attack

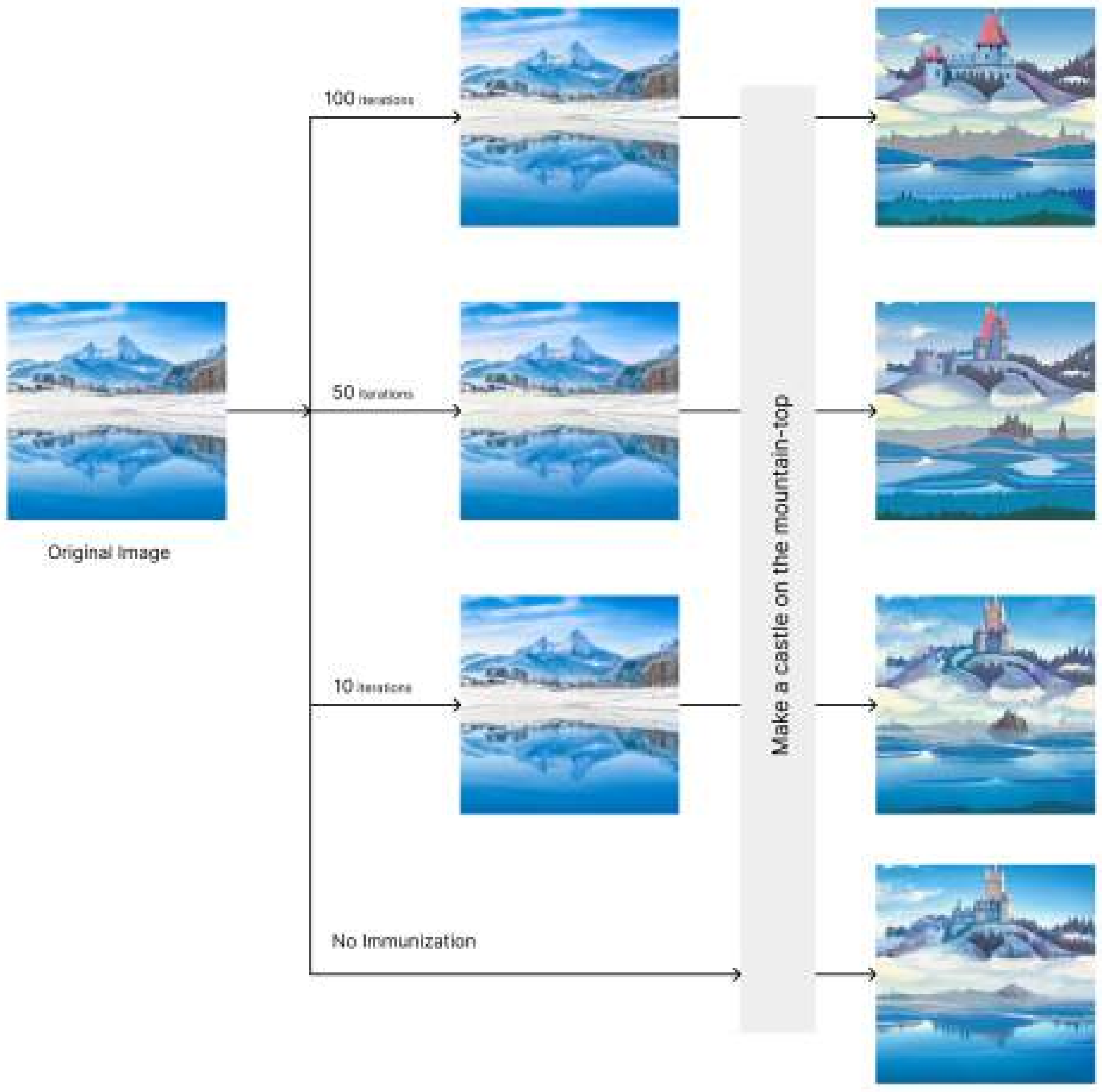


Figure A4: Adversarial image generation with different iteration in diffusion attack

A.3 Image Variations using Different Textual Prompts

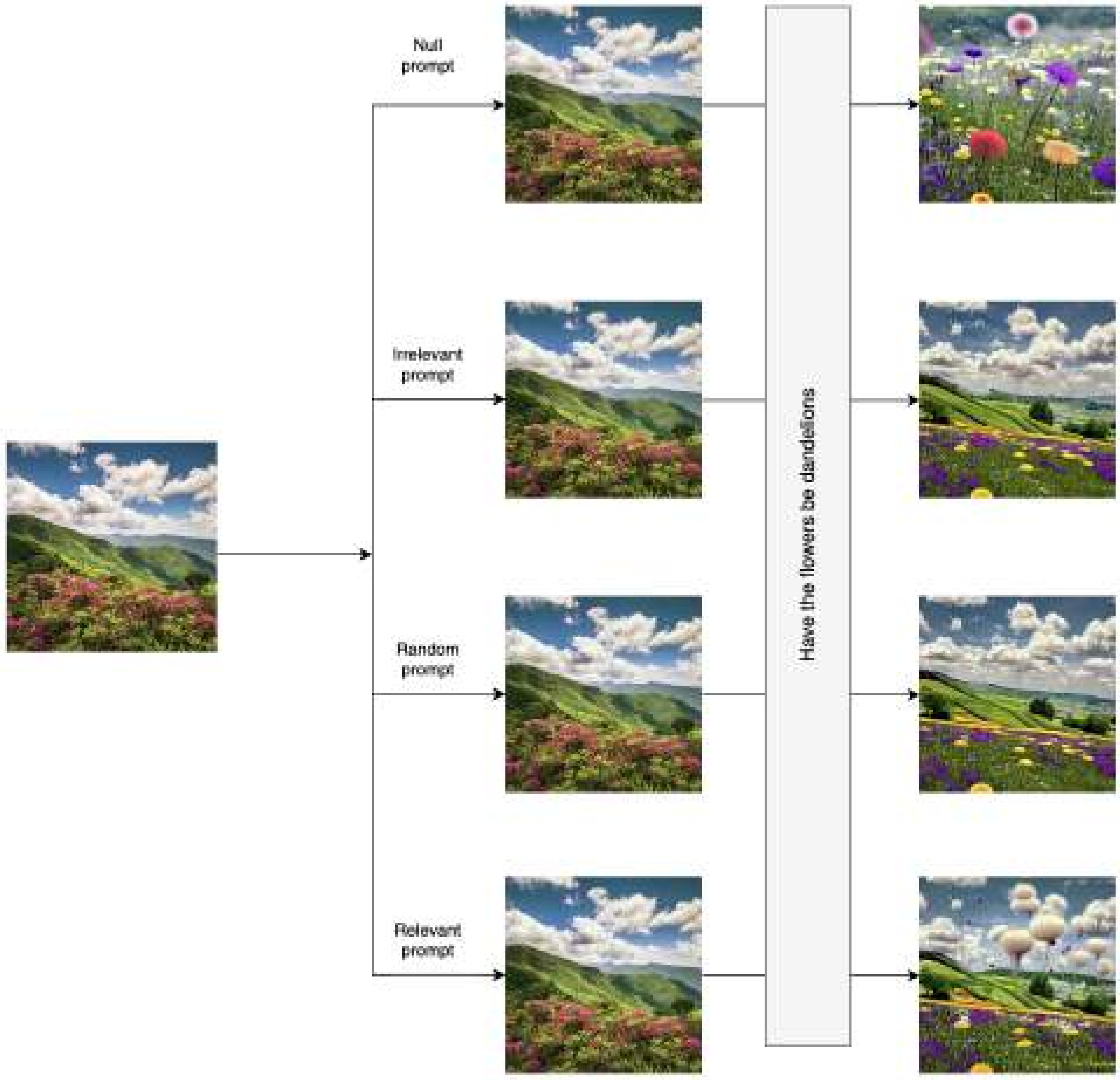


Figure A5: Influence of prompt type on adversarial image manipulation.

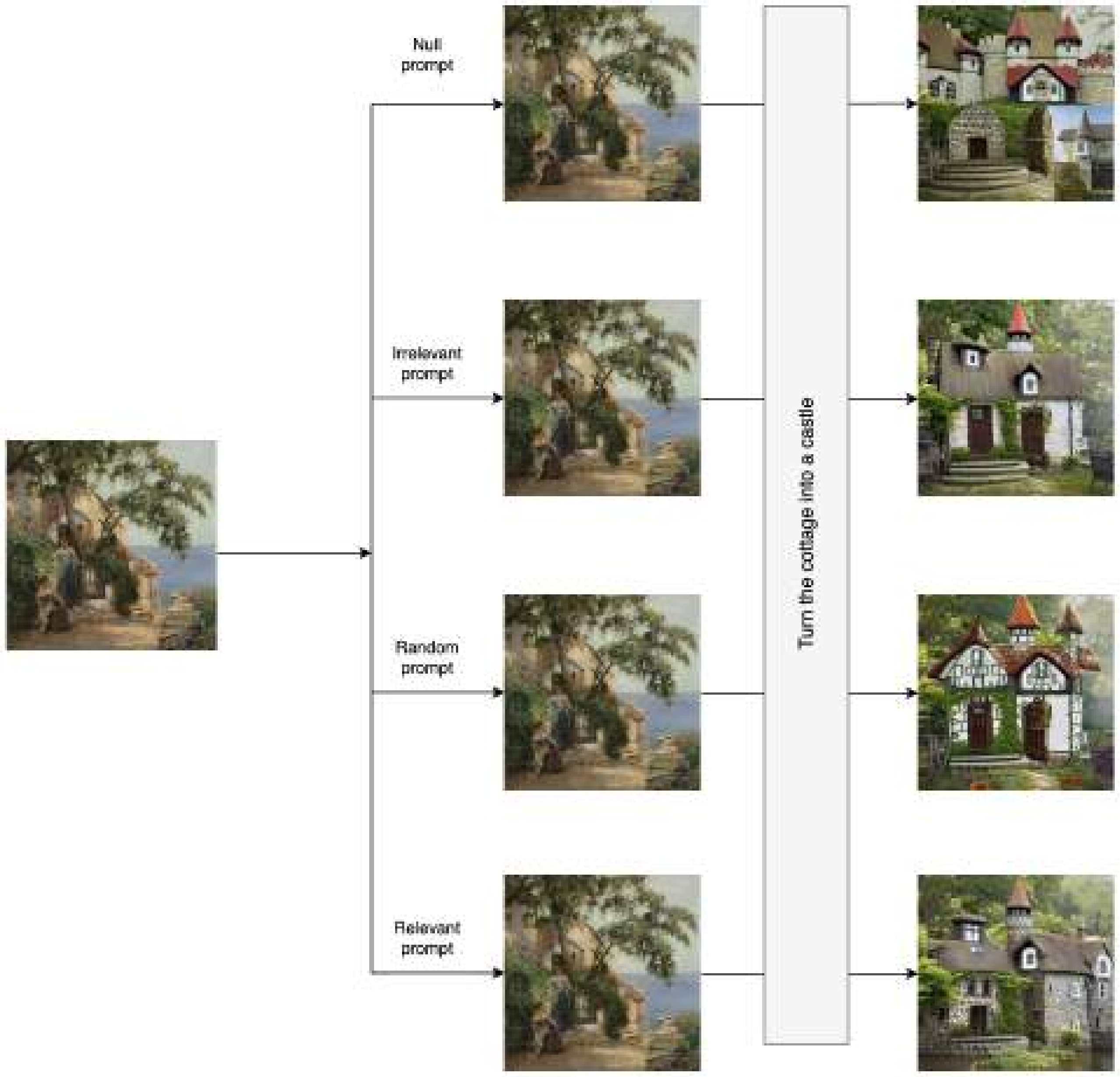


Figure A6: Influence of prompt type on adversarial image manipulation.

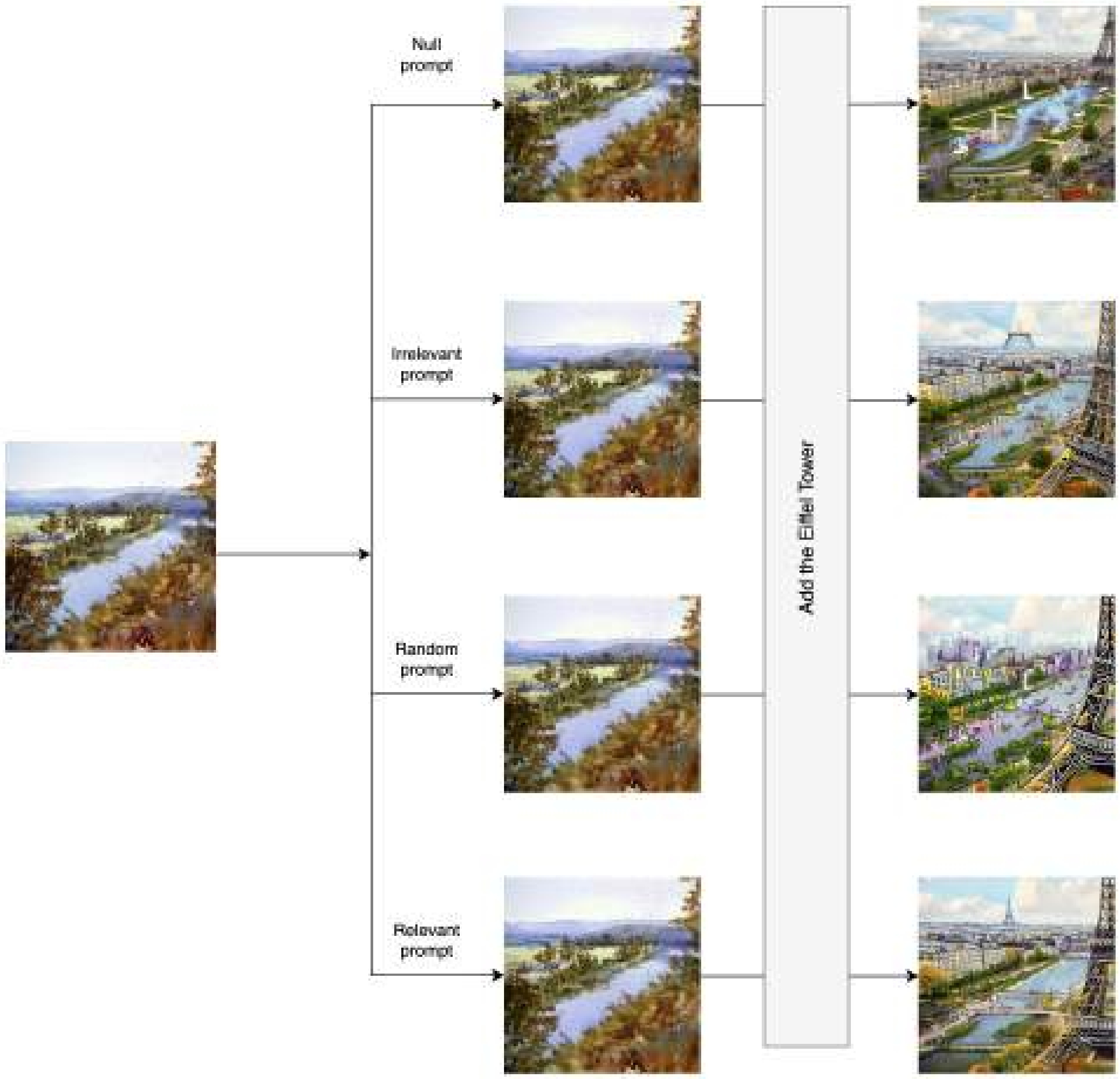


Figure A7: Influence of prompt type on adversarial image manipulation.

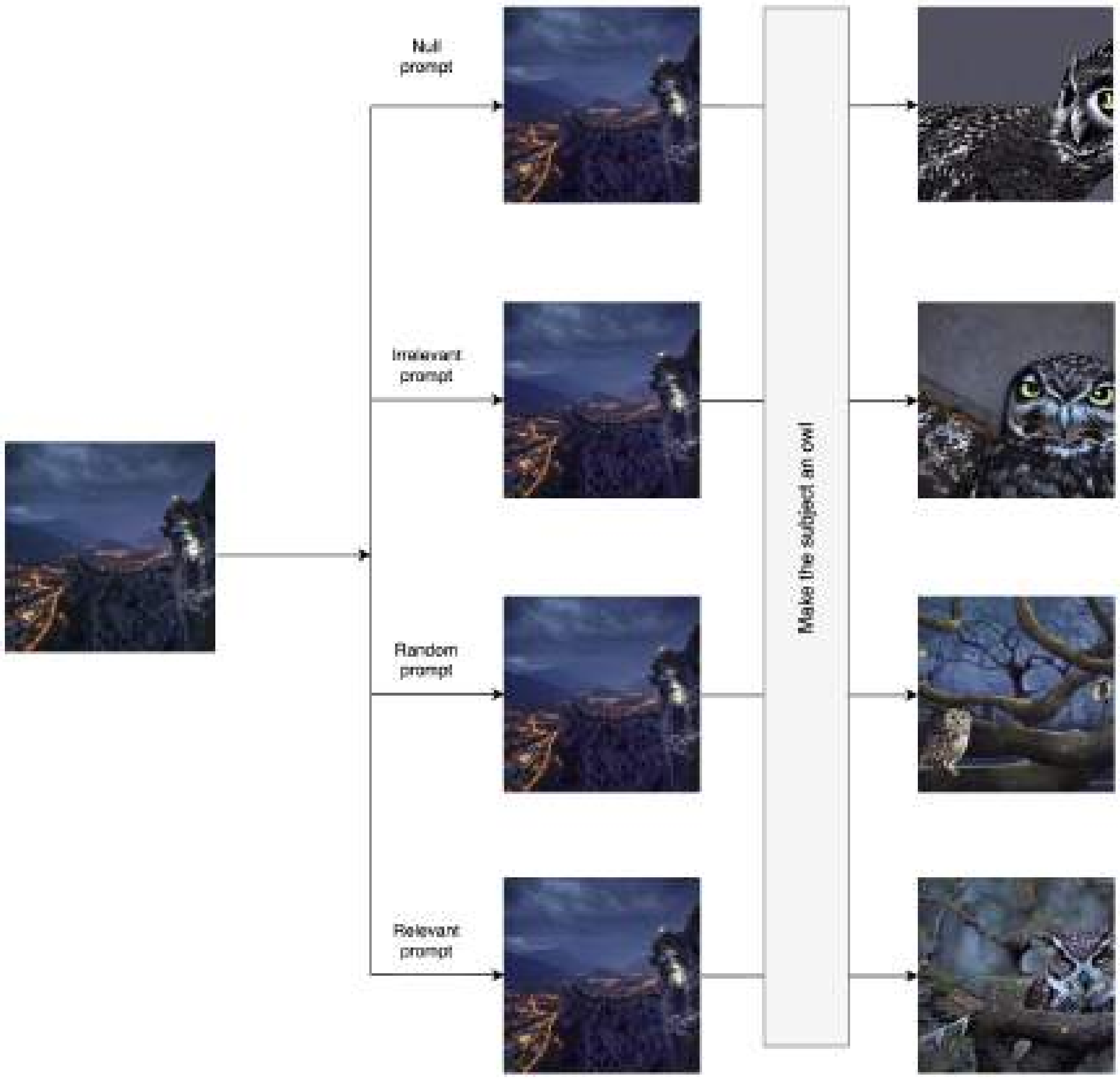


Figure A8: Influence of prompt type on adversarial image manipulation.

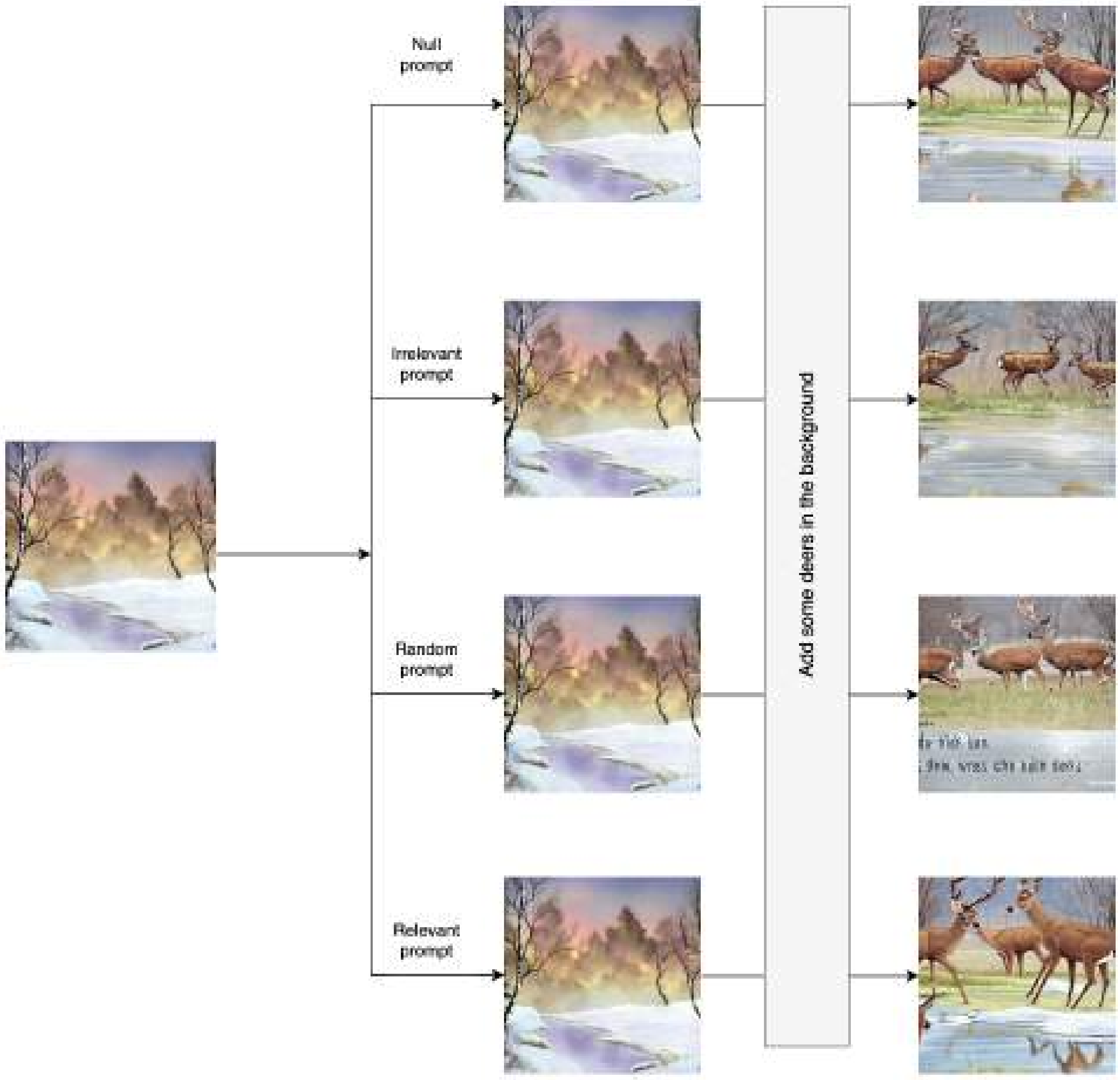


Figure A9: Influence of prompt type on adversarial image manipulation.