Robust Invisible Watermarking in the Age of Generative AI

Jessie E S Smith (220597432)

May 2025

BSc. Computer Science

Supervisor: Deepayan Bhowmik

Word Count: *[14038]*

Abstract

As generative AI becomes increasingly capable of producing hyperrealistic images and edits, watermarking has emerged as a potential defence. This project focuses on developing a watermarking model that is both robust to manipulation and visually imperceptible. This project began by evaluating several state-of-the-art models, revealing a key vulnerability across all of the models. Extreme watermark degradation under contextual, object-level edits. To address this, the project introduces VINE-S. A novel extension of VINE-R watermarking model that incorporates semantic embedding. VINE-S utilizes a stability predictor; a neural network trained to identify regions of an image least likely to be edited. The stability predictors predicted stable regions are used to guide watermark embedding into the semantically safe areas of an image. Ensuring the watermark is preserved even after context-based manipulations. VINE-S was evaluated against the same three state-of-the-art watermarking models used in the initial evaluation. The VINE-S model demonstrated strong overall robustness against common manipulations. Although it underperformed for image fidelity, it achieved state-of-the-art performance under local, object-based generative attacks. Highlighting the potential of guided, stable embedding for resisting semantic AI-driven manipulations.

Acknowledgments

I would like to thank my supervisor, Dr. Deepayan Bhowmik, for his support and guidance throughout this project. I also thank Lamyaa Aljuaid for kindly allowing me to use her generative-AI manipulation pipeline, which played a key role in the evaluation of my watermarking model.

Declaration

I declare that this dissertation is my own original work unless stated otherwise.

Keywords: Watermarking, Robustness, Generative AI, Image Manipulation, Semantic Embedding, Deep Learning

Contents

1 Introduction

Rapid advancements in AI have led to the conception of sophisticated generative models, such as Stable Diffusion [1] and DALL-E 3 [27]. These models enable users to quickly and easily produce photorealistic images and manipulations. This poses a significant threat to content provenance, making it increasingly difficult to determine whether an image has been manipulated or generated. Currently, there is no standardized framework for detecting AI-generated or AI-edited content. Thus, leaving a critical need for a media authentication framework. This lack of regulation also introduces the significant risk of generative AI being used maliciously. The effect of disinformation being shared on the Internet could have a serious impact on political and social discourse.

Digital watermarking is the process of embedding imperceptible and recoverable signals within some digital medium. Digital watermarking has existed for decades and historically being used for copyright purposes [11]. It has recently returned to popularity as a potential defence against the increasing presence of generative AI [13]. When a watermark is embedded into original material, it allows for content traceability and image verification even if the original material undergoes manipulations. Alternatively, when integrated into a generative model, it allows AI content to be tagged for future recognition. Many governments worldwide have began enforcing media authentication measures, such as watermarking, to AI-generated content. For example, the US administration has directed its focus towards defining standards for verifying content authenticity, tracing provenance and labelling digital media through the use of watermarking [35].

Despite the potential watermarking holds for defending against generative AI, its practical effectiveness hinges on robustness. For a watermark to be a reliable indicator of authenticity, it must be able to be extracted even after an image has been altered within a media pipeline. However, a lot of the watermarking systems that are now in use are not made to resist the sophisticated manipulations that generative models introduce. While conventional methods are resilient to conventional operations (such as cropping, Gaussian noise, or JPEG compression), they frequently break down when subjected to generative alterations (such as style transfer or inpainting). Watermarking needs to be

resistant to both traditional distortions and AI-driven edits in order to establish itself as a standardized method for content provenance. I aim to develop a new watermarking algorithm that will contribute to the advancement of watermarking. I will achieve this by presenting a watermarking model with a state-of-the-art robustness against generative manipulations.

Aim: For this project, I aim to develop a watermarking method with an enhanced robustness to Generative AI-based image manipulations. The embedded watermark should remain detectable even after undergoing a set of tradition and generative manipulations independently. By designing a watermarking approach that withstands such attacks, this research seeks to advance digital watermarking as a reliable tool for media authentication and content provenance.

Objectives:

1. Analyse and review a set of state-of-the-art watermarking techniques and evaluate their robustness against a set of common and generative image manipulations.
    - This objective will involve conducting a detailed literature review on current watermarking methods. So as to understand current watermarking frameworks and identify gaps in their methods. As well as conducting an initial test on a set of selected watermarking models to identify the best framework for my approach.
2. Design and implement a novel watermarking technique with consistent robustness against traditional AI-based and traditional image manipulations.
    - This objective focuses on developing a fully functional, new watermarking prototype that ensures watermark imperceptibility. Meaning the unedited, watermarked image must look like the original image and the watermark must be reliably and accurately decodable.
3. Develop and test generative AI-based removal attack test suites to evaluate the effectiveness of the proposed watermarking method.
    - This will involve implementing a set of traditional and generative manipulations to assess how well the watermark algorithm holds up under different

generative manipulations. The watermark must remain detectable after a set of specific image manipulations are performed.

4. Evaluate the robustness and image fidelity of the proposed watermarking method using quantitative performance metrics.

    - Performance will be assessed using key evaluation metrics, such as Peak Signal-to-Noise Ratio (PSNR) and structural similarity index measure (SSIM) to measure image fidelity after embedding the watermark. As well as bit error rate (BER) to measure watermark extraction robustness. Testing will be conducted and results will be compared to existing state-of-the-art watermarking methods.

5. Analyse findings and propose recommendations for further improving my watermarking framework.

    - This objective involves extracting results from the study to provide insights into the effectiveness of different watermarking strategies and best practices for future watermarking approaches.

2 Background Research

2.1 History of Digital Watermarking

Digital watermarking rose to prominence during the 1990s, with the new widespread use of digital media. This introduced new problems of protecting intellectual property and enforcing copyrights. Initial digital watermarking methods worked by creating detectable markings within images, audio, or video. Providing a way of tracing content distribution.

In recent years, watermarking has again become a focal point of research. The emergence of generative AI has posed a new threat to content authenticity and digital provenance. Watermarking is again being examined, no longer just for purposes of copyright protection, but also as a potential tool for detecting and tracing AI-created content.

2.2 Watermarking Categories

Due to its wide range of applications and extensive history, digital watermarking has expanded into a number of different categories. Fragile watermarking schemes are extremely sensitive to any alterations, meaning altering the watermarked image makes it undecipherable. They are typically applied for tamper detection and authentication. An example of a fragile watermarking technique is an approach suggested by Yeung and Mintzer (1997) [40], which incorporates checksums within image blocks for integrity checking.

Robust watermarking methods are geared toward withstanding various image manipulations, which could include compression, noise, and geometrical distortions. Examples of such robust watermarking techniques include HiDDeN [44] and VINE [25], which embed watermarks within features that are tolerant to many transformations. Robust watermarking, while being perfectly suited for copyright protection and content provenance, tends to be less accurate when it comes to detecting finer tampering. The project is focused primarily on strong watermarking because the objective is to create a watermarking scheme that is strong against emerging generative edits. Fragile techniques, although useful for forensic purposes, are less applicable for this purpose.

With the onset of generation AI, there are also novel categories of watermarks. Most significantly, a differentiation has emerged between post-processing and in-processing watermarking. In-processing watermarking involves methods embedded within the generation process of AI models. For instance, the Tree-Ring model [38] incorporates structured signals into a diffusion model's initial noise vector. Stable Signature [12] is another example that incorporates watermarking into a text-to-image pipeline by altering the sampling path. They provide excellent fidelity and strong integration with content generation.

Post-processing watermarks, embed watermarks into content that has already been created. Meaning that it is not model specific and much more scalable. Considering this project's emphasis being on using watermarking for AI-generated and natural images without altering generative pipelines, post-processing watermarking approaches are mostly investigated within this dissertation. Mostly due to their flexibility, scalability,

and compatibility with current generative workflows.

2.3 Watermark Removal Techniques

Watermark removal is the process of making an embedded watermark undetectable. It is usually done by heavily distorting and altering an image. Watermarks are typically encoded in pixel intensities or latent details, thus any operation which distorts these elements has the possibility of degrading or removing the watermark signal.

Watermark robustness has traditionally been evaluated using the non-generative distortions available. These manipulations are used throughout academic work to test model robustness under normal circumstances. For example, the HiDDeN models robustness is evaluated using a synthetic noise layer which includes cropping, JPEG compression, and dropout to test recoverability of the watermark.

While many deep watermarking models such as HiDDeN, RivaGAN [41], and StegaStamp [34] are resilient to such common transformations, they are less effective when dealing with generative AI-based manipulations. Recent experiments have established that it is possible to bypass numerous state-of-the-art watermarking systems using generative models. Carlini et al. [9] illustrated how diffusion models are capable of completely erasing invisible watermarks by creating inpainted reconstructions indistinguishable from the distribution of the original image and effectively removing any embedded watermarked signal without any perceptible evidence - regeneration attacks.

Although targeted removal attacks, such as regeneration, on watermarks do reveal underlying weaknesses, they typically rely on tightly controlled reconstruction processes. Most real-world generative manipulations occur as part of standard media degradation pipelines. Inpainting to erase an object, generative up-scaling, or stylistic modification are some examples. These transformations are far from intending to erase watermarks, but instead just edit an image. However, they still have a tendency to cause partial or even full destruction of the watermarked signal embedded in most models. This highlights that fact that, although there is a place for explicit removal attacks, It is important to assess robustness against the generative manipulations that watermark images will be

commonly facing in real-world applications.

2.4 Classical Watermarking

Classical watermarking algorithms are typically classified as spatial and frequency domain methods. Spatial domain watermarking directly changes pixel values to insert information. The Least Significant Bit (LSB) [10] insertion technique changes the least significant bits of particular pixels to hide the watermark. The Patchwork [6] algorithm also changes pairs of pixel intensities using statistical features to conceal information. Although both methods are fast and visually unremarkable, they are very susceptible to simple operations on images, such as compression, cropping, or noise.

Frequency domain methods provide better robustness by embedding watermarked data in transformed images. Discrete Cosine Transform (DCT) [11] adjusts middle-frequency coefficients to achieve a trade-off in robustness and perceptual visibility, whereas Discrete Wavelet Transform (DWT) [5] uses multi-scale decomposition to diffuse the watermarked data into image frequencies. Singular Value Decomposition (SVD) [23] manipulates the singular values of the matrix of the image, which are less affected by compression and noise. Hybrid models like DCT-DWT-SVD [17] integrate a blend of above methods and are more resilient by using frequency, scale, and statistical stability. Nonetheless, frequency methods are still prone to suffering under many common manipulations, due to their lack of latent embedding.

2.5 Deep Learning-Based Watermarking Techniques

Watermarking based on deep learning, commonly called deep watermarking, trains neural networks to do the job of embedding and detecting watermarks. A shared architectural basis in deep models for watermarking is the encoder–decoder framework. The encoder takes as input an image and a secret message (typically a binary bit sequence), and outputs a watermarked image. The decoder then tries to reconstruct the original message from a possibly distorted version of the watermarked image.

HiDDeN, an earlier deep watermarking framework introduced an encoder–decoder neural network model based on CNNs, jointly trained with a differentiable noise layer. RivaGAN took this a step further, employing adversarial training so as to allow for watermarking at the level of individual video frames.

StegaStamp advanced the field further by putting in place a more sophisticated distortion simulation pipeline and achieving robustness against physical-world transformations like printing and re-photography. Still, this robustness came at a price: visual fidelity. The images produced after stamping often bore subtle artifacts that would be detectable by any sufficiently observant human.

RoSteALS [8] attempted to improve robustness by embedding the watermarks into the autoencoder's latent space. Despite these advancements, most deep watermarking models exclusively evaluate robustness using common distortions like JPEG compression, Gaussian noise, or blurring. Recent developments in generative manipulation techniques have exposed the limitations of these systems. As generative AI evolves, ensuring watermark robustness against these complex transformations remains an open and urgent challenge.

2.6 Robust Watermarking

New watermarking techniques have been proposed with explicit attention to generative threats. InvisMark [39] embeds imperceptible 256-bit watermarks into high-resolution images and rigorously benchmarks its robustness against multiple attack types, including two generative AI-based methods. These included regeneration attacks, where an image is passed through a generative model, such as DALL-E or Stable Diffusion to overwrite embedded signals. As well as adversarial removal attacks, where a model is trained to erase the watermark without altering visible content by slightly perturbing an image. These represent targeted attacks aimed at deliberately removing watermarks.

JigMark [29] proposes a black-box contrastive learning framework. It improves watermark robustness against diffusion models by training on pairs of watermarked and edited images, without needing access to the underlying generative model's internals.

Building upon JigMark's principles, VINE incorporates surrogate-based training and

frequency-aware embedding to create a watermarking system specifically designed for resilience against generative transformations. VINE [25] evaluates its robustness using a wide benchmark (W-Bench) applying generative pipelines such as UltraEdit [43], InstructPix2Pix [7], and ControlNet [42]. These pipelines simulate natural user edits, such as object replacement or stylistic adjustments. Meaning VINE offers a watermarking solution that is robust to the common generative manipulations.

2.7 Semantic Watermarking

An increasing number of studies now highlight semantic watermarking approaches as a key step in combating generative manipulations. Rather than focusing on latent or pixel attributes, these methods embed the watermark based on an image's semantic structure. This can offer two significant benefits: improved traceability or robustness. For traceability, the watermark acts like a semantic fingerprint, tied to the contextual understanding of the image. From a robustness standpoint, aligning the watermark with semantic features helps it embed selectively, due to the models awareness of the images structure.

The SEAL [3] model uses CLIP (Contrastive Language–Image Pretraining) [30] to extract semantic elements from an image. It then applies locality-sensitive hashing to generate a content-specific watermark. This is then embedded using noise perturbations in a diffusion model's latent space. SEAL performs well against non-destructive edits and can flag semantic forgeries, such as object insertions. However, SEAL's reliability depends on the image's semantic consistency. Meaning, if the context of the image changes, the watermark looses its robustness. This trade-off makes SEAL less suitable for cases where image content is expected to change, such as with generative edits.

More contemporary watermarking frameworks, such as Robust-Wide [15] and VINE, integrate the generative editing tool InstructPix2Pix, into their training processes. By exposing watermarking models to prompt-driven semantic alterations, these systems increase their resistance against intricate generative edits that alter the meaning of images globally.

InstructPix2Pix modifies images in response to text prompts, transforming the full image rather than just certain elements. Although this increases resilience to generative alterations, semantically stable regions (areas that are usually left unaltered in routine editing) are not considered. Because of this, existing models may overlook opportunities to place watermarks in these stable areas, making them more vulnerable to focused, localized manipulations.

3 Methodology

3.1 Existing State-of-the-Arts Models

3.1.1 Selected Models

A set of three state-of-the-art image watermarking models were chosen for evaluation; VINE-B, VINE-R, and InvisMark. I chose these models based on their performance in recent benchmarking studies, their use of generative priors, and their demonstrated robustness to modern image manipulations, particularly generative AI-based manipulations. VINE-B leverages a pretrained generative prior for high-quality, robust watermark embedding, VINE-R builds on this with fine-tuning against generative edits, and InvisMark takes a residual-based encoding approach designed for high-resolution images. These models will be robustness-tested against a unified distortion pipeline to determine the optimal foundation for a new watermarking method.

3.1.2 VINE-B

VINE-B utilises a one-step text-to-image generative model, SDXL-Turbo [31], as its watermark encoder. A Condition Adaptor first fuses the input image with a 100-bit watermark, which is then passed through the VAE and UNet of SDXL-Turbo. Then a ConvNeXt-B [24] decoder is trained to recover the embedded watermark from potentially manipulated images. VINE-B achieves a strong balance between imperceptibility and robustness by embedding watermarks in low-frequency image components, without explicit

training against generative edits, but instead incorporating surrogate blur attacks during training.

3.1.3 VINE-R

VINE-R builds on VINE-B by fine-tuning the model with InstructPix2Pix as a surrogate editing attack in the third stage of training. This improves robustness to generative instruction-driven edits.

3.1.4 InvisMark

InvisMark uses a MUNIT-based [16] encoder and a ConvNeXt-based decoder to embed and extract up to 256-bit watermarks. It focuses on high-resolution image fidelity and robustness by applying residual-based watermarking and robust optimization techniques. InvisMark consistently achieves state-of-the-art performance in imperceptibility and good robust decoding under diverse distortions, including geometric and colour-based transformations.

3.2 Initial Testing

3.2.1 Goal

The main goal of this phase is to assess the robustness and imperceptibility of three leading watermarking models: VINE-B, VINE-R, and InvisMark. Each model will be evaluated under controlled image degradation conditions using a range of common manipulation techniques. This benchmarking process is critical for determining which model offers the most reliable foundation for the proposed watermarking algorithm. The selected base will be extended and adapted with novel techniques introduced in later sections. If all baseline models perform poorly under realistic attack conditions, I will consider a more extreme architectural modification or hybrid approach. This stress test establishes

a performance baseline and informs the direction of my model design decisions.

3.2.2 Common Manipulations

To ensure robustness in real-world deployment, watermarking models must withstand a variety of image manipulations commonly applied either intentionally (to edit or obfuscate) or unintentionally (through compression and transmission pipelines). I have categorized these manipulations into four primary domains: colour alterations, blurring, noise injection, and compression.

- **colour alterations** simulate changes in appearance due to device settings, filters, or automated enhancement tools. These include modifications to brightness, contrast, saturation, hue, and exposure. Each of these operations perturbs global colour statistics and pixel distributions, which may interfere with embedded watermark signals.

- **Blurring techniques** includes motion blur, Gaussian blur and zoom blur. These manipulations techniques can degrade spatial detail and suppress high-frequency watermark patterns.

- **Noise-based distortion** includes Gaussian noise. This distortion simulates transmission interference or analog sensor degradation. These can overwhelm subtle watermark encodings if not properly defended against.

- **Compression** includes JPEG and WebP, as well as bit-depth reduction, downscaling, and up-sampling. These are frequently encountered in social media and web delivery pipelines. These processes discard information that may include or disrupt embedded watermark features.

To ensure comprehensive evaluation, each manipulation type was applied at multiple strength levels, randomly selected within a predefined range encompassing low to extreme distortion levels. This random variability aims to simulate real-world inconsistency in manipulation severity, from minor degradations to aggressive alterations.

3.2.3 Global vs Local Manipulations

The VINE watermarking paper introduces an important distinction between two types of manipulations [25], particularly in the context of watermark robustness testing. Global manipulations are transformations that can affect the entire image. Some Global edit examples include style changes, colour shifts, or modifications of major scene elements (without altering the key image features). In contrast, local manipulations are edits occurring in specific regions or objects within the image, leaving the broader image structure largely intact. An example of local editing is generative inpainting or object replacement.

This distinction is critical for watermark evaluation because global edits tend to perturb the overall image distribution, however do not disrupt the key image features. Meaning if the watermark is embedded into the latent features of the image, global shifts to the image shouldn't effect or degrade the embedded watermark critically. Local edits, however, primarily threaten the watermark's integrity in specific regions and features, requiring the watermarking system to ensure robustness even when parts of the image, therefore the watermark, are heavily modified or removed completely.

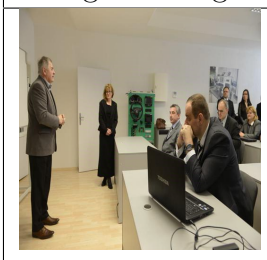
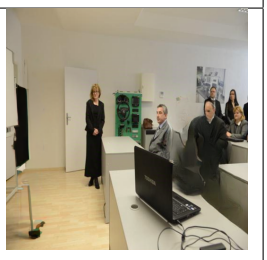
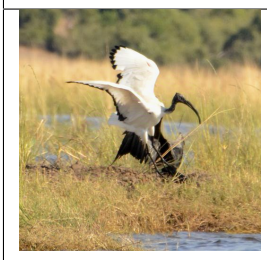
Original Image	Global Edit	Local Edit
		
No Edit	Van Gogh painting	Remove man
		
No Edit	Remake as vintage picture	Remove Bird

Table 1: Visual comparison of global and local image manipulations. The original image is shown alongside two modified versions: a global edit, which affects the entire image's appearance (e.g., style, colour, tone), and a local edit, which alters specific objects or regions (e.g., object removal or replacement)

To assess robustness against global generative edits, I used the InstructPix2Pix pipeline. InstructPix2Pix is particularly suited for this task because it applies high-level text-guided transformations throughout the image, such as changing the style or colour palette. Importantly, InstructPix2Pix introduces coherent, realistic alterations while preserving the underlying features within the image. Its ability to modify both content and style globally makes it an effective benchmark for stress-testing watermark resilience against global manipulations.

For testing robustness to local edits, a novel attack pipeline was used. This method combines Segment Anything Model (SAM) [19] for precise object segmentation, BLIP (Bootstrapping language-image pre-training) [21] for semantic understanding and captioning, GPT-5.3-Turbo [28] for various object manipulations and LaMa [33] for inpainting removed regions. By selectively targeting and replacing objects, it simulate realistic editing behaviours such as object removal, substitution, or creations. This localized editing pipeline stresses the watermark's ability to survive when specific semantic regions are altered, without significantly disrupting the rest of the image.

3.2.4 Local-Manipulation Pipeline

To test the robustness of watermarking models against local generative edits, a custom remove-and-replace pipeline [2] was used for my project. This pipeline simulates realistic multi-object level manipulations by automatically identifying, segmenting, and editing objects within an image without requiring any human intervention or pre-existing ground truth edit masks. Its implements multiple local manipulation types - object removal, object creation and object replacement.

The pipeline first employs YOLO (You Only Look Once) [36] model to detect meaningful objects within an image for manipulation. The pipeline uses SAM (Segment Anything Model) to produce accurate segmentation masks for detected objects within an input image. Then, once the masks and are generated, BLIP (Bootstrapped Language Image Pretraining), a vision-language model, is used to generate semantic captions describing the masked objects, describing both the masked object and the entire image.

After segmentation and captioning, the model applies several types of manipulations to the images, using the extracted information (object masks and image captions). GPT-3.5-Turbo, via OpenAI, is utilised to create detailed prompts, relevant to the image context and detected objects. This finalized prompt is then used by various generative AI models, such as DALL-E 2 [26] and LaMa (Large Mask Impainting Model). LaMa fills masked regions with visually plausible content by hallucinating background textures and scene elements, effectively simulating object removal. This pipeline enables several types of local manipulations, such as:

- **Object Removal:** The object is removed from the image, and the background is realistically reconstructed to look natural and unedited.

- **Object Creation:** A new object is realistically added into the original image.

- **Object Replacement:** Replace the masked object realistically with another within the image.

- **Combined Operations:** For initial robustness testing, multiple manipultation were applied simultaneously, challenging watermark integrity more severely, as a larger section of the image is locally edited.

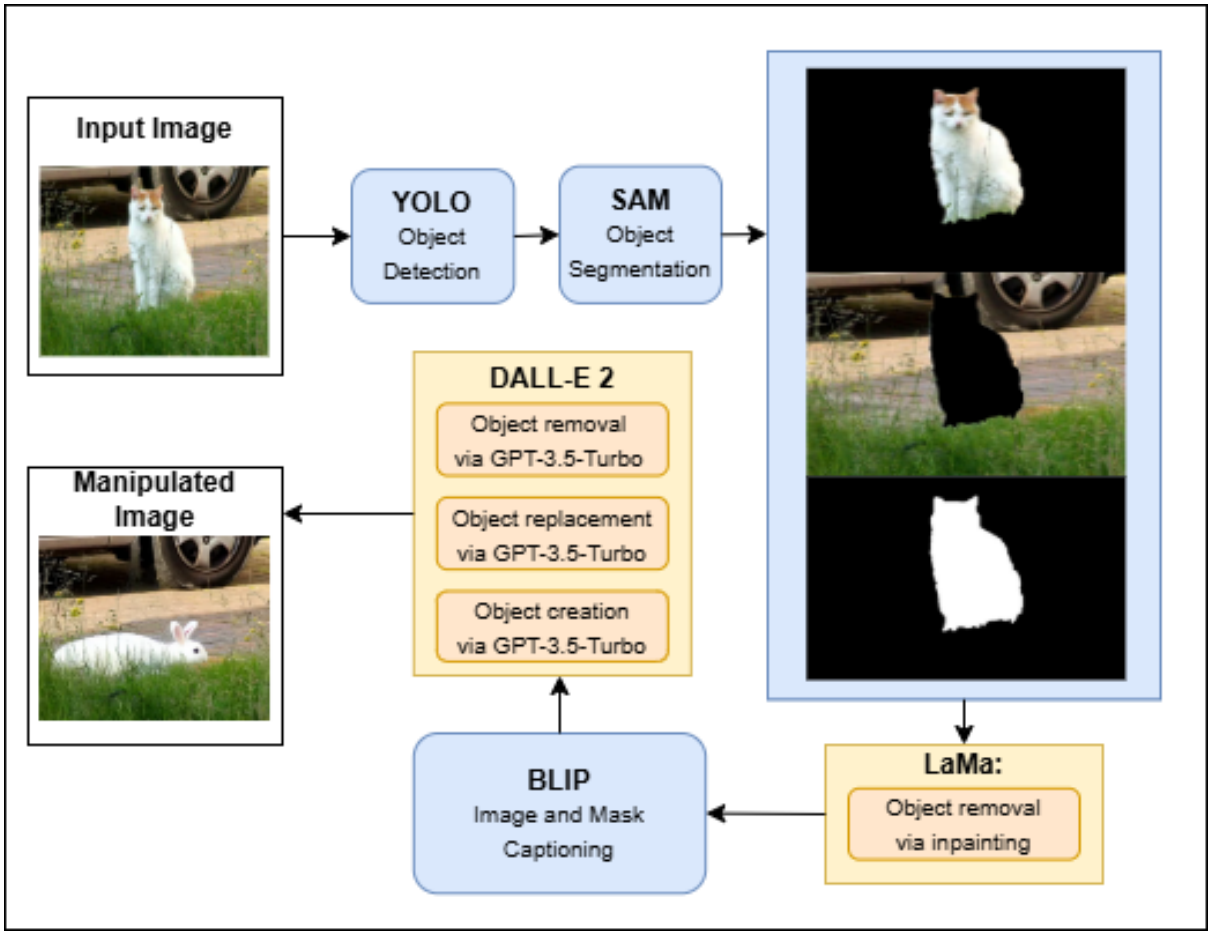


Figure 1: Overview of the Remove-and-Replace manipulation pipeline. The pipeline combines object detection, segmentation, captioning, prompt generation, and generative editing using LaMa inpainting and GPT-3.5-Turbo to simulate realistic local image manipulations.

A notable innovation of this local manipulation pipeline is that it automatically generates masks and prompts for each image. Other local editing pipelines, such as UltraEdit [43] and ControlNet-based inpainting [42], typically require annotated datasets where predefined masks and textual prompts correspond to specific images. These preset dependencies limit flexibility when it comes to training and testing. In contrast, the automatic generation approach allows the pipeline to be deployed on any arbitrary dataset without additional manual annotation, significantly enhancing experimental scalability.

This fully automated, multi-stage local manipulation pipeline introduces diverse, contextually plausible local edits at object-level. It enables a rigorous and realistic evaluation of

watermark robustness under localized generative attacks, a scenario increasingly relevant today.

3.2.5 Initial Testing Dataset Preparation

For the initial testing, a test set of 1,000 images was used from the publicly available COCO Image dataset [22]. The COCO dataset was chosen for testing due to its diverse set of real-world images featuring complex and varied scenes, making it well-suited for evaluating watermark robustness. This dataset was used to evaluate image fidelity as well as watermark robustness against all manipulation types, except for Global manipulations.

The second dataset used for testing is the InstructPix2Pix [7] dataset, where 1000 images and prompts where extracted for evaluating watermark robustness against global edits. This is because we are evaluating robustness against global edits using InstructPix2Pix. None of the other manipulation types being used in initial testing require predefined dataset annotations in the same way that InstructPix2Pix does - which requires corresponding image prompts. Therefore using InstructPix2Pix dataset offers practical convenience for evaluation of watermark robustness to global edits - as InstructPix2Pix manipulation pipeline will be used to asses this. Essentially this will streamlines the global edit testing pipeline. Therefore allowing consistent, automated application of edits across all test images, with the correct corresponding prompts being applied.

3.2.6 Initial Testing Results

In order to test with GPU acceleration using Google Colab, I re-implemented both InvisMark and VINE encoding/decoding pipelines to support the Colab environment. This meant altering their respective original repository scripts or environment compatibility and adapting file handling.

I evaluated the three selected watermarking models; InvisMark, VINE-B, and VINE-R. I use both image fidelity and watermark robustness metrics to comprehensively asses the models performance. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) were used to assess watermark imperceptibility. While Bit Error

Rate (BER) was used to quantify watermark robustness. PSNR and SSIM are widely used metrics in the field of watermarking, where SSIM evaluates perceptual degradation in terms of structure and luminance, whereas PSNR quantifies absolute pixel-level distortion. Together these imperceptibility metrics provide an accurate estimate of the visual quality of watermarked images. BER is a widely accepted measure of watermark robustness, as it calculates the percentage of incorrectly decoded bits when compared to the original ground truth watermark. The final robustness score for a given model within each manipulation category was computed as the mean BER across all manipulation types and strength levels within that category (compression, blur, noise, etc).

Model	**PSNR**	**SSIM**
VINE-R	44.0225	0.9928
VINE-B	44.9692	0.9954
InvisMark	**51.7421**	**0.9985**

Table 2: Comparison of image fidelity metrics (PSNR and SSIM) for three watermarking models: VINE-R, VINE-B, and InvisMark. Higher values indicate better preservation of visual quality after watermark embedding. InvisMark achieves the highest fidelity, outperforming both VINE variants.

These image fidelity results are expected given the InvisMark architecture heavily prioritized watermark imperceptibility. The model is optimized to minimize visual distortion through advanced perceptual losses and robust residual scaling. Its performance could be attributed to its use of a MUNIT-based encoder with skip connections. As well as the use of low-amplitude residuals that are blended into the original image to reduce visual impact, unlike VINE, which embeds watermarks via generative priors and may cause more noticeable artifacts. However, this is at the cost of some robustness under aggressive or generative edits.

VINE-R significantly outperformed the other models in watermark robustness, consis-

	Average BER for Each Distortion Category					
Model	**Noise**	**Blur**	**Colour**	**Compression**	**Global**	**Local**
VINE-R	**0.0045**	**0.0049**	**0.0137**	**0.0073**	**0.0214**	**0.0891**
VINE-B	0.0104	0.0158	0.0215	0.0231	0.0471	0.1964
InvisMark	0.0693	0.1816	0.0667	0.2277	0.2481	0.3092

Table 3: Compares the average Bit Error Rate (BER) of watermark recovery across different distortion categories for VINE-R, VINE-B, and InvisMark. Lower BER indicates greater robustness to the specific manipulation type. VINE-R performs best overall, while InvisMark exhibits the highest error rates across most distortion types.

tently achieving the lowest BER across all manipulations. Given this project's main goal of developing a robust watermarking model, VINE-R provides the best foundation to build my model from. I believe a compromise of slightly worse imperceptibility is justified by VINE-R's exemplary robustness to multiple types of aggressive editing.

However, VINE-R performance degraded noticeably on local manipulations. As seen in its lower BER score of 0.0836 for the removal and replacement pipeline. Although this is still a good robustness score, this drop in accuracy is significant, due to VINE's BER being constantly below 0.02 for all other manipulations. This is likely due to VINE's non-semantic embedding strategy. As without semantic awareness, the model cannot prioritize embedding in stable regions. Making local object edits more destructive, as seen in initial testing.

To address this, we propose integrating semantic awareness into the VINE-R framework. Through being able to identify and target semantically stable image regions, the watermark can be embedded into areas less likely to be affected by local edits, which could substantially enhance robustness against local manipulations.

3.3 Foundational VINE Model

3.3.1 Architecture Details

The VINE architecture consists of two variants: VINE-B (VINE-baseline) and VINE-R (VINE-robust). Both are built on the SDXL-Turbo one-step text-to-image diffusion model, which serves as a powerful generative prior for watermark embedding. Both variants incorporates a Condition Adaptor, VAE encoder/decoder, UNet, zero-convolution skip connections, and a ConvNeXt-B watermark decoder.

The encoder acts like a conditional generative model, accepting an input image and a 100-bit binary watermark. These are fused via the Condition Adaptor, which projects both modalities into a shared image space. This representation is passed through the VAE encoder to extract latent features, which are then de-noised by the UNet and reconstructed via the VAE decoder to create the final watermarked image. Skip connections and zero-convolution layers help preserve image details and visual fidelity.

For decoding, a ConvNeXt-B network learns to extract the watermark from potentially edited images. The model's output is processed by a fully connected head to recover the original bit sequence.

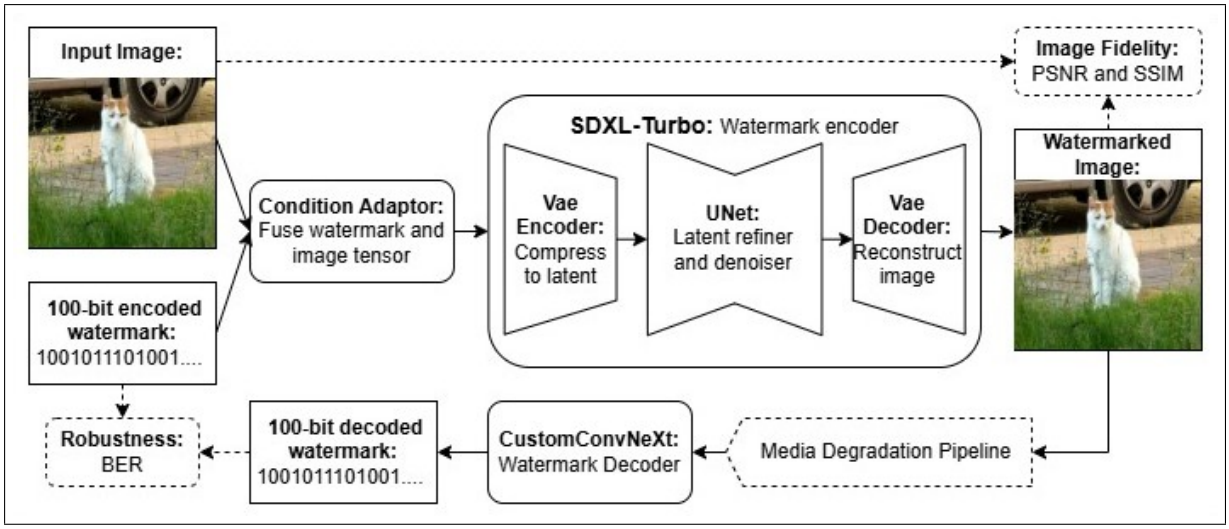


Figure 2: Contains a diagram outlining the original VINE architecture. The input image and 100-bit encoded watermark are fused by the Condition Adaptor, then processed by the SDXL-Turbo watermark encoder consisting of a VAE encoder, UNet, and VAE decoder to produce the watermarked image. A separate CustomConvNeXt decoder can extract the watermark, even after the image has passed through various manipulations. Robustness is evaluated via bit error rate (BER), while image fidelity is measured using PSNR and SSIM.

In VINE-B, robustness is enhanced during training by applying common and blur-based distortions. These include multiple blur distortions, including Gaussian blur, motion blur, zoom blur. As well as Common image distortions, including brightness, contrast and saturation shifts, pixelation, Gaussian noise and JPEG compression. These surrogate attacks, especially the blur attacks, target mid and high-frequency bands known to degrade under generative editing.

VINE-R expands on this by incorporating fine-tuning using InstructPix2Pix-based edits in a third additional training stage. This significantly improves the VINE-B robustness to global generative manipulations, as well as further optimising the models overall robustness to other non-generative manipulations.

3.3.2 Strengths

The VINE architecture introduces several innovations that contribute to its exemplary performance in both watermark imperceptibility and, in particular, robustness.

Firstly, its use of SDXL-Turbo model as a watermark encoder enables good perceptual alignment with the input image. By operating in the latent space of a powerful generative prior, VINE preserves image quality very effectively. However, it does not achieve the same level of imperceptibility as residual blending techniques, such as those employed in the Invismark model. Nonetheless, leveraging pretrained generative models like SDXL-Turbo accelerates training and provides a reliable foundation for decent watermark imperceptibility.

Second, the Condition Adaptor is an effective mechanism for fusing image and watermark information into a shared representation. Unlike residual merging, the Condition Adaptor could enable the network to learn contextual dependencies between watermark content and image features during training.

Third, VINE uses frequency-informed training. By analysing how different editing techniques affect the Fourier spectrum of images, the authors of VINE identify low-frequency regions as more stable under generative edits. Meaning embedding watermarks in these low-frequency bands improves resilience. The authors also observe that blur manipulations exhibit similar frequency distortions to generative edits. Therefore the incorporation of blur-based surrogate attacks during training allows VINE-B to simulate generative degradation efficiently and scalably, without having to apply computationally heavy generative manipulations throughout training.

Lastly, fine-tuning with real editing models, such as InstructPix2Pix in VINE-R, makes the architecture robust to complex, global generative edits.

3.3.3 Limitations

Despite its strong performance, the VINE architecture has notable limitations, particularly in relation to its lack of semantic awareness. One core limitation is that the

watermark is embedded in the latent representation of the image, without considering semantic stability or the context of the image. This distribution means that if certain parts of an image are removed or altered, such as in object-level and local edits, the watermark signal in those areas is lost.

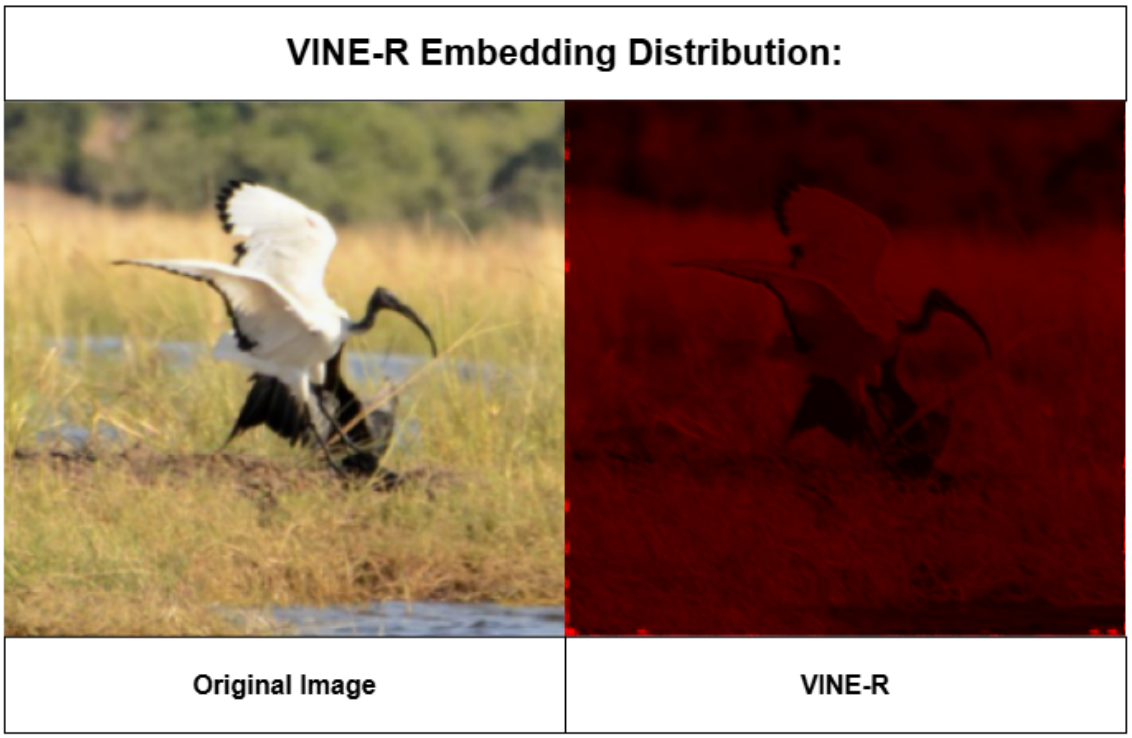


Figure 3: The left image in this figure is the input image. The heat-map (right) illustrates per-pixel differences between the original image and VINE-R's condition adaptor output, averaged across RGB channels. VINE-R displays a more uniform embedding across the image, guided only by latent image structure

As seen in figure 3, the watermark is embedding This becomes especially problematic under local editing attacks, such as those applied in the remove-and-replace pipeline. Because VINE does not selectively target semantically stable regions (areas likely to not be edited, given the image context), its robustness suffers when edits affect high-saliency or foreground areas where a significant portion of the watermark could be embedded.

Finally, the VINE pipeline, especially VINE-R, is computationally expensive due to the

use of a large generative prior and a fine-tuning procedure involving back-propagation through editing models. This increases training complexity and limits accessibility for computationally limited environments.

The absence of semantic embedding or stability prediction represents a significant opportunity for improvement in subsequent models.

3.4 Proposed Watermarking Model

3.4.1 Semantic Embedding

Semantic embedding refers to embedding information into semantically stable or meaningful regions of an image based on its content. In watermarking, this would involve identifying areas less likely to undergo manipulation, and concentrating watermark signals there. The VINE watermarking framework currently embeds watermarks uniformly across the latent representation of the input image. The watermark is fused with the watermark via the Condition Adaptor and processed through the VAE encoder, UNet, and decoder. However, this strategy lacks awareness of image semantics, meaning that localized edits, such as object removal or replacement, can destroy embedded information and reduce robustness.

Stable image region: The term I have used in this project to define the regions of an image less likely to undergo local manipulations - depending on image context and object/region saliency.

To integrate semantic embedding into VINE, a mask could be generated, highlighting stable regions of an image. These masks could guide the Condition Adaptor to embed watermark information more selectively. This would essentially turn the Condition Adaptor into a spatial controller, as it is determining where in the image the watermark should be embedded. By targeting robust regions, VINE's resistance to localized, semantic-driven attacks would improve, ideally without significantly compromising imperceptibility.

3.4.2 BLIP/SAM Approach

My initial approach to developing a semantic watermarking method involved constructing an image stability mask using attention maps derived from the BLIP model. Which is a vision-language transformer model that can generate image captions and align text-image embeddings.

The BLIP model uses a cross-attention mechanism to relate visual tokens to language tokens, producing attention maps that indicate which regions of an image are most salient with respect to a generated caption. These maps reflect regions that the model considers semantically important or contextually relevant during vision-language alignment.

The initial plan was to combine BLIP attention maps [32] with SAM. By intersecting SAM-generated masks with regions of high attention in the BLIP maps, I aimed to identify semantically important objects. Which, theoretically, should be likely targets for local edits in generative manipulation pipelines, given their importance to the image context. This method had theoretical advantages. It would produce dynamic, content-aware stability masks, allowing the watermark to be embedded away from regions that are highly salient or likely to be modified. Such a pipeline could potentially outperform VINE-R's non-semantic embedding embedding by focusing watermark signals on background regions or secondary objects that tend to remain untouched during common editing workflows. Additionally, the models being used are pre- trained and reliable, saving me the time of having to train a model myself to predict the stable regions of images.

However, this approach was ultimately not pursued for three key reasons. First, the computational overhead was high. Both SAM and BLIP are large transformer-based models that require substantial memory and processing power, making them difficult to run in tandem on limited hardware. Second, implementing this system required deep understanding of BLIP's internal attention extraction mechanisms and their alignment with token-level language outputs, which was outside my current technical scope. Thirdly, BLIP's attention maps indicate semantic significance, not edit likelihood. A region being important to the image caption does not necessarily mean it is likely to be edited, making

the resulting stability masks unreliable for robustness targeting.

Given these limitations, I shifted focus toward developing a lighter-weight, supervised stable region predictor specifically trained to identify regions most likely to be edited in local generative edits, rather than those that are just semantically important to an image caption.

3.4.3 CNN Approach

The final design of my proposed watermarking model involves integrating a stability prediction (stable region predictor) model into the watermark embedding pipeline. The stability predictor is a convolutional neural network (CNN) that combines a pre-trained ConvNeXt-Tiny encoder [24] with a lightweight decoder. The encoder extracts deep visual features from the input image, while the decoder up-samples these features to produce a 256×256 probability map. This indicates the likelihood that each pixel will remain unchanged during future edits. The model outputs a stability mask; a single-channel map where each pixel is assigned a continuous value between 0 and 1, representing its predicted probability of being edited. These masks can then used to bias watermark embedding toward the most stable areas of the image. This should theoretically increasing the chances of watermark survival under localized or semantic edits.

To train this model, I aim to use datasets consisting of of original unedited images, alongside corresponding binary manipulation masks. These masks highlight precisely which parts of the image were altered. By treating the mask as ground truth and training the model in a supervised manner, the predictor should learns to infer semantic and structural cues that correlate with commonly edited regions. These might include prominent foreground objects, people, or text elements.

Unlike approaches that rely on synthetic saliency estimations (such as BLIP attention maps), this method is grounded purely in predicting edit likelihood. Ideally, the manipulations in the dataset should be driven by human-authored prompts or actions, rather than automated and synthetic prompts. This ensures that the predictor captures authentic human intent and intuition. As this would yield more accurate estimates of edit

likelihood and improving watermark robustness for local manipulations, particularly if implemented within real-world applications and media.

3.5 Stability Predictor Training

3.5.1 Model Design

Stability predictor model is a CNN that is trained to generate pixel-level manipulation likelihood prediction masks. The model is based on a pre-trained ConvNeXt-Tiny encoder from the timm library due to its efficient performance on image recognition tasks with low computational costs. The encoder is able to extract rich hierarchical features from the image and produces a 768-channel high-level feature map from its last layer

The decoder is a light-weight custom head that maps this high-dimensional feature map to a single-channel stability prediction map. It has a 3-layer design: a 3×3 channel dimension-reducing convolution (768 → 256), ReLU activation, an upsample layer with scale factor 2, a second convolution from 256 → 64, a second ReLU, and a last 1×1 convolution mapping to a single channel. A Sigmoid activation outputs a probabilistic map in [0, 1]. The final output is then bi-linearly up-sampled to a 256×256 mask, which is the size of the ground truth mask.

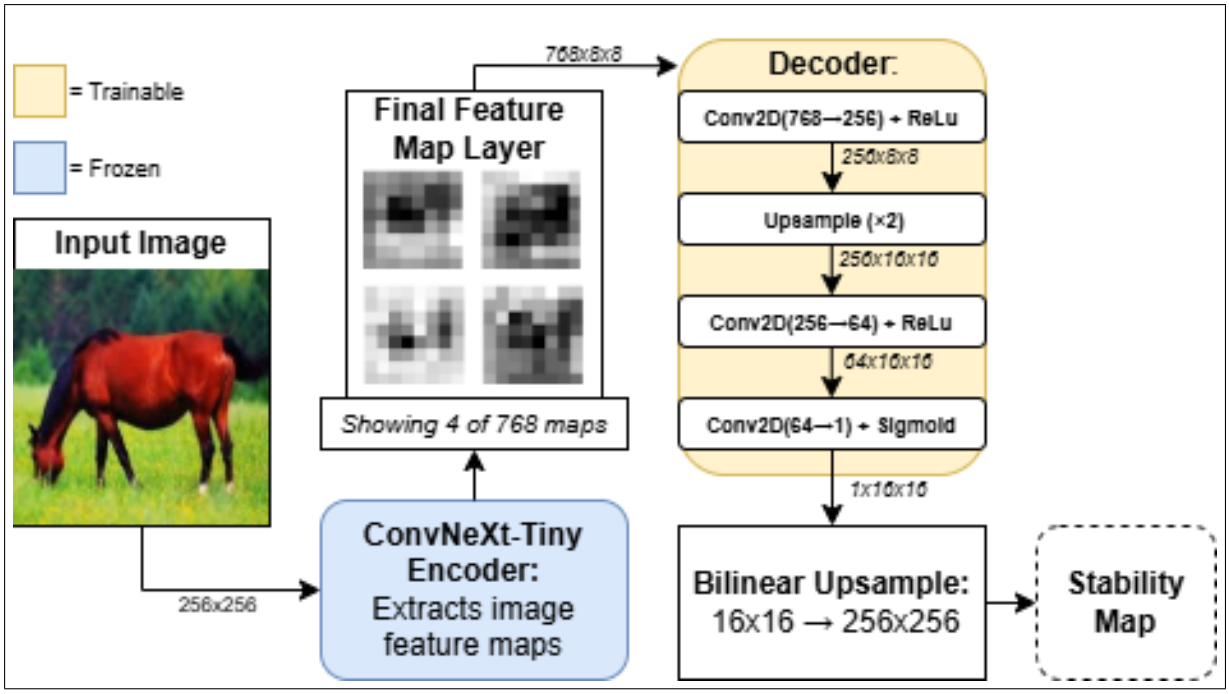


Figure 4: Outlines the Stability Predictor model's architecture. A frozen ConvNeXt-Tiny encoder extracts feature maps from a 256×256 input image. A trainable decoder generates a low-resolution stability map, which is upsampled to 256×256. The output highlights stable regions suitable for watermark embedding

This architecture offers a strong balance between semantic understanding and spatial resolution. The ConvNeXt backbone captures high-level contextual features, while the lightweight decoder reconstructs fine spatial details, enabling effective prediction of stable versus editable regions and efficient usage within watermarking pipelines.

3.5.2 Dataset Preparation

For the purpose of training the stability predictor, I compiled a dataset that comprised two reliable sources: HumanEdit [4] and PIPE (Paint-by-Inpaint) [37]. Together, I had around 10,000 image-mask pairs that made for a rich and varied dataset for training.

HumanEdit provided 5,700 samples that each comprise a real image from the world, a human-authored edit command, and a corresponding binary mask that indicates what regions have been manipulated. Importantly, the edit prompts for HumanEdit are not

synthetically created but instead mimic human-intent-driven local manipulations. Manipulations applied in HumanEdit cover a large range of world scenarios and include Action, Add, Counting, Relation, Remove, and Replace. Images provided to HumanEdit are collected from a wide variety of sources with a combination of low and high resolutions, making it fitting for high-fidelity, photorealistic edits. This makes HumanEdit particularly well-suited for training my model on where edits are most likely to take place in real-world, human-driven scenarios.

The PIPE dataset supplied 4,300 image-mask pairs. PIPE employs a mask-based object removal strategy using high-performance inpainting models (Stable Diffusion [1]) and then constructs an object addition dataset by reversing the removal process. The dataset sources images utilize COCO [22] and Open Images datasets [20], with segmentation annotations being carried out using LVIS [14]. PIPE's images and ground truth edit masks will help the stability predictor model learn manipulation patterns in semantically coherent image regions. However, unlike HumanEdit, PIPE uses synthetically generated prompt, making it potentially less consistent with human behaviour when it comes local, object-based manipulations.

Together, the datasets provide a well-balanced mix of human-driven and AI-driven object-level modifications. Their diversity, spanning image domains, edit types, and manipulation semantics, should ensure good generalization and robustness in stability predictor model.

3.5.3 Training Parameters

The final training configuration used a ConvNeXt-Tiny encoder with a custom decoder, optimized using the Adam [18] optimizer with a learning rate of 1e-4 and a batch size of 4. The binary masks were predicted using a BCELoss for the stability model predictions and ground truth binary masks from the dataset.

It was trained for 30 epochs on images resized to 256×256 resolution. During the initial training run, the model very quickly reached a low loss (0.04). However, it was found to be over-predicting stability on the whole image. This was because the majority of

image regions on HumanEdit and PIPE binary masks have not been edited, and thus the network reduced the loss by predicting most image pixels with a high stability value, without truly detecting the localized edits.

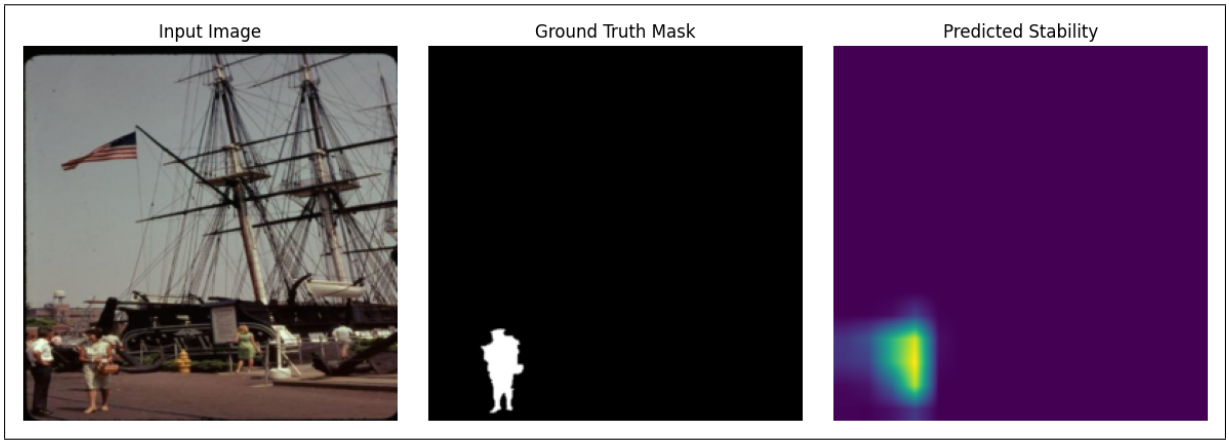


Figure 5: The left-most image is the input, with the binary mask (centre) showing the edited region of the input image. The right-most image is a heat-map showing the stability predictors output. Where bright regions mean unstable and darker regions mean stable. The stable region in this figures heat map is too small, due to the model under-generalising when detecting instability.

To address this, the model was retrained with a higher priority on accurately detecting unstable (edited) areas. This involved optimizing decoder weights to ensure the model was mostly focused on identifying the edited regions. This should result in a higher loss value, but a larger, more general unstable region mask.

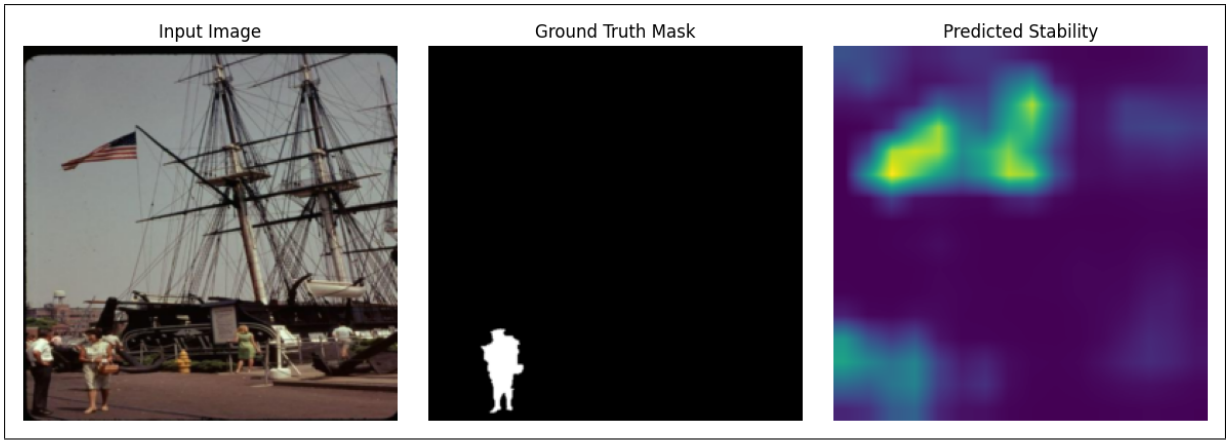


Figure 6: The left-most image is the input, with the binary mask (centre) showing the edited region of the input image. The right-most image is a heat-map showing the stability predictors output. Where bright regions mean unstable and darker regions mean stable.

The new run of training resulted in a balanced loss of 0.28, which converged out much later on subsequent epochs than the initial training run. I believed 0.28 to be a sensible loss metric, considering the intrinsically sparse and ambiguous task of identifying stable and unstable regions of an image. The heat-map from Figure 6 demonstrates the larger, more generalised stability predictor, compared to the one seen in Figure 5.

3.5.4 Finalized Model

Once training of the stability predictor, the final challenge was ensuring both the predicted stable and unstable regions in the stability map covered a large enough image area. As, if these regions are too small, visual artifacts may arise due to over-concentration. Conversely, if unstable regions are too small, the predictor fails to capture likely areas for local manipulation. This would result in normal embedding - identical to VINE-R's [25] embedding process that is not resilient to local edits.

To address this, I implemented a balanced thresholding strategy: once the model outputs a continuous-valued stability map (with each pixel scored from 0 to 1 based on predicted edit likelihood), I flag half of the image pixels with the highest instability scores. A new binary mask is created where these most unstable pixels are marked as 1 (edited), and

the remaining half are set to 0 (stable).

Stable Region Algorithm:

1. `prediction` ← output of stability predictor model (shape: `[256, 256]`)
2. `flattened` ← flatten `prediction` into a 1D array
3. `K` ← total number of pixels divided by 2 (keep top 50%)
4. `threshold` ← the `K`-th highest value in `flattened`
5. `binary_mask` ← if `pred[x, y]` $\geq$ `threshold`, then 1 (stable), else 0 (unstable)
6. reshape `binary_mask` back to `[256, 256]`

This method ensures that both stable and unstable regions occupy sufficient spatial regions of an image. It enables the watermark to be embedded across a large, confident stable region, minimizing perceptual distortion. While simultaneously maintaining a well-defined unstable zone to guide robustness evaluation and model training. This 50% threshold achieves a practical trade-off between visual quality and semantic targeting. To evaluate the effectiveness of the binary stability masks produced by this algorithm, we conducted a validation study on 752 images from the PIPE test dataset. The models predicted unstable region correctly encompassed the true manipulated region fully in 83.27% of the samples. This result provides empirical support for the reliability of the stability maps in guiding semantically-aware watermark embedding.

3.5.5 Implementation Details

The stability predictor was implemented and trained in a Google Colab Pro environment using an NVIDIA L4 GPU, which offered sufficient GPU RAM for training at a resolution of 256×256. The architecture was built using PyTorch, with key dependencies including torchvision, tqdm, PIL, and matplotlib for preprocessing, visualization, and evaluation.

Model checkpoints were logged to drive, enabling repeatable experiments. All data loading and augmentation was handled via PyTorch's Dataset and DataLoader interfaces, ensuring efficient GPU utilization during training.

3.6 Semantic Integration

3.6.1 VINE Architecture Changes

In the original VINE pipeline [25], the Condition Adaptor only accepted two inputs: the image and the secret watermark. This fusion was performed without knowing the semantic structure of the image.

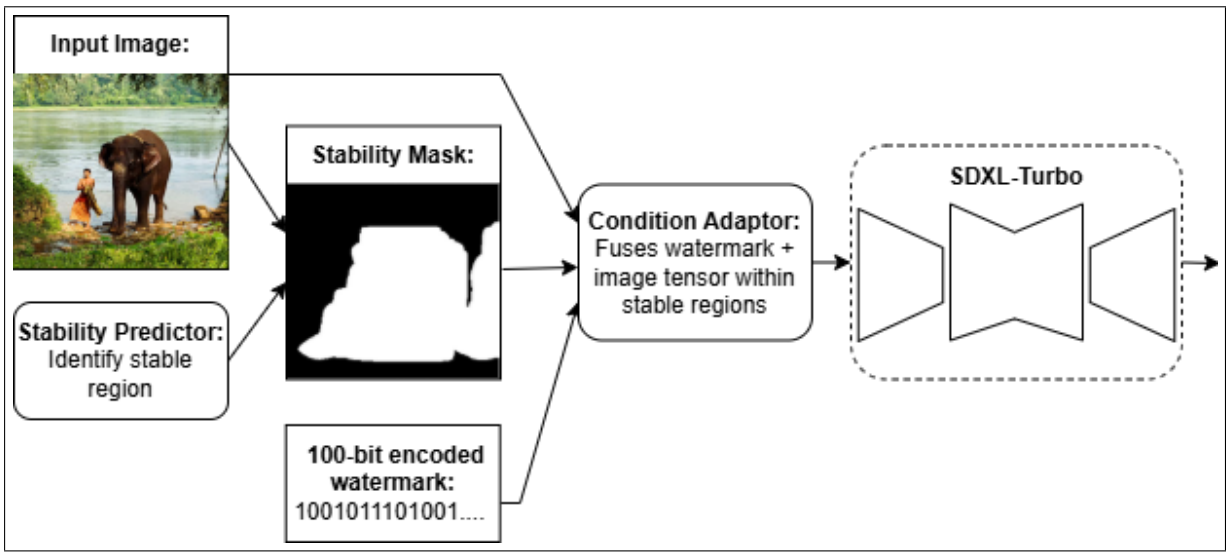


Figure 7: Describes the new encoding process - where the Condition Adaptor received 3 inputs, watermark, image and stability mask. This is so it can learn, through training, to spatially guide the model to embed into the stable regions regions of an image. The VINE pipeline proceeding Condition Adaptor remains the same architecturally.

In my new implementation, a third input (the stability mask produced by the predictor) is passed into the Condition Adaptor. This mask is used to bias watermark encoding

away from unstable regions. Specifically, the updated Condition Adaptor now accepts secret, image and stability mask during the forward pass. The binary mask is processed in the forward pass, and is then concatenated with the secret and image tensor. This increases the input dimensionality from VINE-R's 6 channels to 9 channels. The fused output of the Condition Adaptor is then used as input to the conv1 layer (update to accept 9 channels now).

In `VINE_Turbo`, the `forward` method was modified to include a call to `self.stability_predictor(x)` before encoding, ensuring that the mask reflects each input. This tensor is then passed to `self.sec_encoder` (the Condition Adaptor), guiding watermark fusion at a feature level conditioned on semantic stability.

The VAE encoder/decoder, UNet2DConditionModel, and CustomConvNeXt decoder remained architecturally unchanged. They were not modified because the models Condition Adaptor is what determines where the watermark is encoded within the image.

3.6.2 Embed Mask Loss

To encourage watermark embedding into semantically stable image regions, the model must be retrained to avoid encoding watermark signal in areas likely to be edited. For this purpose, an auxiliary objective - embed mask loss - was incorporated into training. This loss uses the binary stability mask generated by the stability predictor to indicate unstable regions. The embed mask loss is applied at the output of the Condition Adaptor, where the image and watermark are first fused. It computes the pixel-wise residual between the Condition Adaptor output and the original image and penalizes distortion only in unstable regions. Effectively pushing watermark signal into stable zones.

$$\mathcal{L}_{\text{embed}} = \frac{1}{N} \sum_{i=1}^{N} \left| x_{\text{sec}}^{(i)} - x_{\text{img}}^{(i)} \right| \cdot M^{(i)}$$

where:

- x_{sec} is the watermarked image output from the encoder
- x_{img} is the original input image
- M is the binary stability mask, where $M^{(i)} = 1$ for semantically stable pixels and 0 otherwise
- N is the total number of pixels in the image
- $|\cdot|$ denotes element-wise absolute difference
- $\cdot$ is element-wise product

3.6.3 Pretrained Model Approach

My initial plan for integrating semantic awareness into the VINE watermarking pipeline was to reuse and fine-tune the pretrained weights from VINE-R. This was in order to preserve its frequency-aware embedding and strong robustness to multiple manipulation types.By leveraging these pretrained weights, I hoped to maintain its fidelity and robustness advantages while introducing semantic selectivity via a stability predictor.

However, during implementation, it became clear that the modified architecture was incompatible with VINE-R's pre-trained Condition Adaptor parameter. In the original model, the Condition Adaptor accepted a 6-channel input -concatenated image and watermark tensors. In the updated pipeline, I introduced a third input, the stability mask. Thus raising the input dimensionality to 9 channels. As a result, the first convolutional layer (conv1) in the updated Condition Adaptor no longer matched the pretrained weight shape from VINE-R. This made parameter transfer infeasible without retraining from scratch.

I attempted to freeze the pretrained VAE, UNet, and decoder modules, while training only the modified Condition Adaptor from scratch. However, this caused the model to under perform and fail to converge. The pretrained components were trained to support unguided watermark embedding, and expected latent features aligned with that behavior. Whereas the new Condition Adaptor attempted to shift watermark distribution based on

semantic masks. This new embedding clashed with the expectations of the downstream models. Resulting in unstable learning and ineffective encoding.

Moreover, this approach created an imbalance in training initialization. The Condition Adaptor was randomly initialized, while all other components were frozen or pretrained. I believe the lack of synchronized adaptation meant the models in the watermarking pipeline were not able to train together cohesively.

In retrospect, this was a misguided attempt stemming from an underestimation of the architectural entanglement within VINE. Fully supporting semantic embedding requires joint training of all modules with the new Condition Adaptor architecture.

3.6.4 Complete Retraining Approach

My final approach to implement semantic integration into the VINE watermarking pipeline, I decided to follow a complete retraining strategy. This was structured around the same three-stage training framework used in the original VINE-R training process.

Like in the original VINE training process, I also incorporated a noise layer throughout training. This layer applies randomized image distortions, such as Gaussian noise, JPEG compression, brightness/contrast shifts, blurring and pixelation during training. With its main purpose being to simulate a variety of different manipulations, enhancing the model's ability to embed watermarks resiliently in low-frequency image regions where edits are less destructive.

I chose to try and follow the general VINE-R training strategy to achieve its strong robustness against the surrogate attacks, extending it to support semantic embedding in contextually stable image regions. While this training strategy reflects VINE-R's overall training structure, certain deviations in my training implementation were necessary due to resource constraints.

3.6.5 Training Dataset Preparation

For the first two stages training, I utilized 10,000 images from the PIPE dataset [37], for the same reasons it was utilized for stability predictor training (detailed in section 4.5.2). Whereas for the final third training stage, I utilized 10,000 images from the InstructPix2Pix [7] dataset, an image collection specifically designed for instruction-guided image editing tasks. This dataset includes a diverse range of hyper-realistic image-editing pairs. Where each original image is accompanied by an edited version and a corresponding textual instruction. The edits can encompass a wide range of transformations, including object addition or removal, style alterations, and colour modifications, reflecting realistic and semantically meaningful changes.

Changing the dataset for the third stage of training was necessary, as this training stage focuses on integrating InstructPix2Pix semantic edits into the noise layer. By using images already associated with specific editing instructions, I could seamlessly apply the InstructPix2Pix pipeline during training. I ensured that none of the images included in the Stage three training dataset overlapped with the initial testing dataset, which also utilised the InstructPix2Pix dataset.

The training was conducted in Google Colab Pro using a single NVIDIA A100 GPU, a powerful but limited environment compared to the original VINE-R setup that employed 8×A100-80GB GPUs. Consequently, training hyperparameters were tuned to remain within feasible memory and runtime constraints. For example, batch sizes were reduced to 4 to accommodate high-resolution inputs at 512×512, and mixed precision (AMP) training was used to optimize memory usage. I used the Adam [18] optimizer with a learning rate of 1e-4 during the initial stages (Stage one) and reduced it to 1e-5 in later fine-tuning (Stage two and three). All images were resized to 256×256 for embedding, then re-projected back to 512×512 to match the input resolution during residual-based reconstruction.

3.6.6 Stage One Training

$$\mathcal{L}_{\text{total}} = 10 \cdot \mathcal{L}_{\text{BCE}} + 0.01 \cdot \mathcal{L}_{\text{MSE}} + 0.01 \cdot \mathcal{L}_{\text{LPIPS}} + 0.01 \cdot \mathcal{L}_{\text{EMB}}$$

Here, $\mathcal{L}_{\mathcal{BCE}}$ is the binary cross-entropy loss between the decoded and ground-truth watermark bits, serving as the primary supervision signal. $\mathcal{L}_{\mathcal{MSE}}$ is the mean squared error between the original and reconstructed images, while $\mathcal{L}_{\mathcal{LPIPS}}$ is the perceptual loss capturing high-level visual similarity. $\mathcal{L}_{\mathcal{EMB}}$, the embed mask loss, encourages watermark embedding in semantically stable regions identified by the stability predictor. The main goal of Stage one training is to optimize the models watermark extraction accuracy. Therefore, setting $\mathcal{L}_{\mathcal{BCE}}$ weight to 10 and all the other loss weights to 0.01 encourages the model to prioritize that objective.

During Stage one, I trained the Condition Adaptor and watermark Decoder modules from scratch while fine-tuning the pretrained SDXL-Turbo VAE encoder. In line with the VINE-R approach, the VAE decoder and UNet remained frozen to preserve the generative prior.

The key deviation from the original VINE-R protocol was my decision to proceed to Stage two only after achieving 0.95 bit accuracy, rather than the 0.85 threshold used in the original VINE paper. An earlier attempt at transitioning at 0.85 led to unstable convergence in Stage two. This was likely due to the added complexity introduced by the semantic embedding, which restricts the embedding space and increases the difficulty of learning a robust watermark decoding function early in training. Therefore, I believed it was necessary to further optimize the models watermark extraction accuracy in Stage one, to ensure a more robust foundation in the proceeding training stages.

3.6.7 Stage Two Training

$$\mathcal{L}_{\text{total}} = 1.5 \cdot \mathcal{L}_{\text{BCE}} + 2.0 \cdot \mathcal{L}_{\text{MSE}} + 2.0 \cdot \mathcal{L}_{\text{LPIPS}} + 1.5 \cdot \mathcal{L}_{\text{EMB}}$$

Here, $\mathcal{L}_{\mathcal{BCE}}$ have been reduced to 1.5, while $\mathcal{L}_{\mathcal{LPIPS}}$ and $\mathcal{L}_{\mathcal{MSE}}$ weights have been in-

creased to 2. This was done to shift this training stages focus to improving perceptual and pixel-level image fidelity. $\mathcal{L}_{\mathcal{EMB}}$ weight has been set to 1.5 to further reinforce watermark localization within the stable regions predicted by the stability mask. In the original VINE Stage two training the $\mathcal{L}_{\mathcal{LPIPS}}$ weights were set 1.5, however I chose to raise both the weights of $\mathcal{L}_{\mathcal{LPIPS}}$ and $\mathcal{L}_{\mathcal{MSE}}$ to 2.0 to compensate for the absence of a GAN discriminator. I intentionally omitted GAN loss from training to avoid the added computational burden, especially in resource-limited environments like Google Colab Pro.

Throughout Stage two, I observed that $\mathcal{L}_{\mathcal{LPIPS}}$ and $\mathcal{L}_{\mathcal{MSE}}$ plateaued at 0.0345 and 0.0057, respectively. Meanwhile, the secret loss stabilized around 0.0402, and bit accuracy peaked at 0.9884. At the time, I misinterpreted these signs as indicators of convergence and acceptable performance. In hindsight, these values were suboptimal for Stage two completion, and would have likely benefited from further training. My over reliance on bit accuracy is a key misjudgment in my Stage two training process.

The embed mask loss reached 0.0353 by the end of Stage two, suggesting reasonably successful semantic targeting of watermark embedding.

3.6.8 Stage Three Training

In the final stage of training, I extended the noise layer by integrating the InstructPix2Pix generative editing pipeline. This allowed each image to be edited according to its associated prompt. simulating realistic generative modifications such as style and colour shifts, or environmental changes

In this third stage, following the original VINE-R training process I ensured all parameters remained unfrozen. Additionally I maintained all the same loss weights as Stage two.

However, due to limited computational resources and the addition of InstructPix2Pix into the noise layer, each epoch took approximately 20 minutes to complete. This significantly restricted the opportunity for extensive experimentation and refinement during this training stage, given the limited time available. I ran Stage three training over 15 epochs, before the loss values began to converge.

Adding InstructPix2Pix was conceptually effective for simulating semantic-level attacks. As it should encourage the model to learn watermark patterns that survive real-world edits. However, for my implementation of Stage three training, the performance did not improve as much as I hoped. $\mathcal{L}_{\mathcal{BCE}}$ initially spiked to 0.0693, likely due to the abrupt shift in the noise distribution introduced by the edits, before gradually declining to 0.0374.

This is a positive outcome, as it shows the watermarking model has improved its robustness to the generative manipulation added to the noise layer over the 15 training epochs. Similarly, $\mathcal{L}_{\mathcal{LPIPS}}$ and $\mathcal{L}_{\mathcal{MSE}}$ plateaued at 0.0292 and 0.0042, respectively. Although this result better than Stage two, it is not by a significant margin. Additionally, $\mathcal{L}_{\mathcal{EMB}}$ unexpectedly increased to 0.0563 at the start of training, possibly due to instability introduced by the generative noise layer, but later declined to 0.0362, showing no substantial gain over Stage two.

Reflecting on these results, I believe that mediocre convergence in Stage two likely limited the effectiveness of Stage three training.

3.6.9 VINE-S

For the remainder of this dissertation, I will refer to my final trained model as ***VINE-S***, which stands for VINE-Semantic. This is due to the model being a direct extension of the VINE architecture, but with semantic-aware embedding integrated.

3.7 Implementation Details

VINE-S was implemented and trained in PyTorch and trained in Google Colab using GPU acceleration. The environment included timm, diffusers, transformers, and accelerate, with my project repository cloned from GitHub. Google Drive was used for checkpoint management. I have used Accelerate for device handling and reproducibility.

3.7.1 Training GPU Limitations

All training was conducted on Google Colab, using a single NVIDIA A100 GPU (40GB). While the A100 provides considerable compute and memory bandwidth, this setup is limited compared to the original VINE-R environment, which used 8×A100-80GB GPUs in parallel. Consequently, training parameters had to be adjusted to fit within resource constraints. For example, batch sizes were limited to 4, and automatic mixed precision (AMP) was enabled to manage GPU memory more efficiently.

4 Results and Evaluation

4.1 Testing Setup

4.1.1 Testing Dataset Preparation

For final testing, I have uses the same images used for initial testing (section 4.2.5). This includes 1000 images from the COCO [22] image dataset, for evaluating image fidelity and robustness against all image manipulations besides global edits. As well as 1000 images and prompts from the InstructPix2Pix [7] dataset, for evaluating watermark robustness again global generative edits (using the InstructPix2Pix edit pipeline).

4.1.2 Encoder-Decoder Script

The original Colab-compatible VINE encoder-decoder script, used for initial testing, was adapted for VINE-S. While the encoding script remained unchanged, the decoding script required modification. Mixed-precision training using autocast was used when training VINE-S. This was essential for managing limited GPU memory and improving training efficiency. Due to known instability when using BCELoss under autocast, BCEWithLogitsLoss was used instead, as it is more stable with half-precision tensors. As a result, the final sigmoid activation was removed from the CustomConvNeXt decoder. Therefore in the VINE-S decoding script, the decoder's output logits are explicitly passed through a

sigmoid function to obtain probabilities.

4.1.3 Manipulations Applied

To evaluate watermark robustness under real-world editing conditions, I applied the same manipulations used during initial testing, as outlined in the Methodology section. These include a set of classical image distortions: colour-based adjustments (brightness, contrast, saturation, exposure), blur operations (Gaussian, motion, and zoom blur) and noise-based perturbations (Gaussian noise). As well as compression manipulations such as JPEG quality degradation, WebP conversion, and resolution downsampling were applied to simulate internet-based image degradation pipelines. The local manipulation pipeline [2] (Section 3.2.4) was used to evaluate robustness to local generative edits. Whereas the InstructPix2Pix edit pipeline was used to evaluate watermark robustness against global generative Edits.

4.1.4 Benchmarking

For benchmarking, my proposed semantic watermarking model, VINE-S, was evaluated alongside VINE-B [25] and VINE-R [25] the two primary variants of the VINE architecture. This comparison was selected deliberately: my method is a direct architectural and conceptual extension of VINE-R, and thus must be tested against its baseline and robust versions to fully evaluate its improvements. For consistency with initial testing, InvisMark [39] will also be included in this evaluation.

Although the original VINE-R paper evaluates performance across a broader set of generative edits, this full benchmark was not replicated for my final testing. Many of the VINE benchmark manipulations rely on computationally expensive models and edit pipelines, and the resource demands for their deployment exceeded my available resources. Therefore, the selected manipulations for the final benchmarking represent a practical subset that still provides a strong test of watermark robustness under both global and localized distortions.

4.1.5 Testing Environment and Parameters

All models were tested in Google Colab Pro, using an NVIDIA T4 GPU with 16GB VRAM. The evaluation pipeline was implemented in Python 3.10 using PyTorch 2.0.1, with supporting libraries including torchvision, numpy, and PIL.

4.2 Evaluation Metrics

4.2.1 Testing Metrics

For consistency with initial testing, I continue to evaluate image fidelity using PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index). These metrics assess the pixel-wise and structural integrity of the watermarked image relative to the original.

For robustness evaluation, I used BER (Bit Error Rate), the same robustness metric used in initial testing.

4.2.2 Semantic Embedding Quality Assessment

To assess the effectiveness of semantic embedding in VINE-S, I have evaluated how well the watermark signal is concentrated within stable regions of the image. In this model, the Condition Adaptor is responsible for determining where the watermark is embedded. Its input now includes a stability mask, which guides the fusion of the image and watermark toward semantically stable regions. To visualize and validate correct embedding, I have computed a residual map defined as the absolute difference between the Condition Adaptor's output and the original input image. This residual highlights the strength and spatial location of watermark perturbations. By overlaying this map with the binary stability mask, a heat-map can be generated that shows whether watermark signal is concentrated in the stable regions. This serves as a qualitative assessment of the embedding behaviour.

4.3 Results

4.3.1 Image Fidelity Results

Model	PSNR	SSIM
VINE-S	31.4973	0.8932
VINE-R	44.0225	0.9928
VINE-B	44.9692	0.9954
InvisMark	**51.7421**	**0.9985**

Table 4: Evaluating VINE-S using fidelity metrics PSNR and SSIM against three state-of-the-arts watermarking models: VINE-R, VINE-B, and InvisMark. Higher values indicate better preservation of visual quality after watermark embedding. InvisMark achieves the highest fidelity, outperforming all VINE variants. VINE-S yeilds the worst performance, with the lowest PSNR and SSIM

Table 4 demonstrates that VINE-B has the best imperceptibility performance, achieving the highest score across all three metrics. However, VINE-S achieved the lowest scores, implying a fundamental issue with how the model was implemented and trained.

Lower PSNR suggests that VINE-S watermarked images exhibit higher pixel-wise deviation from the original images. A lower SSIM value for VINE-S implies perceptible alterations in texture or spatial consistency.

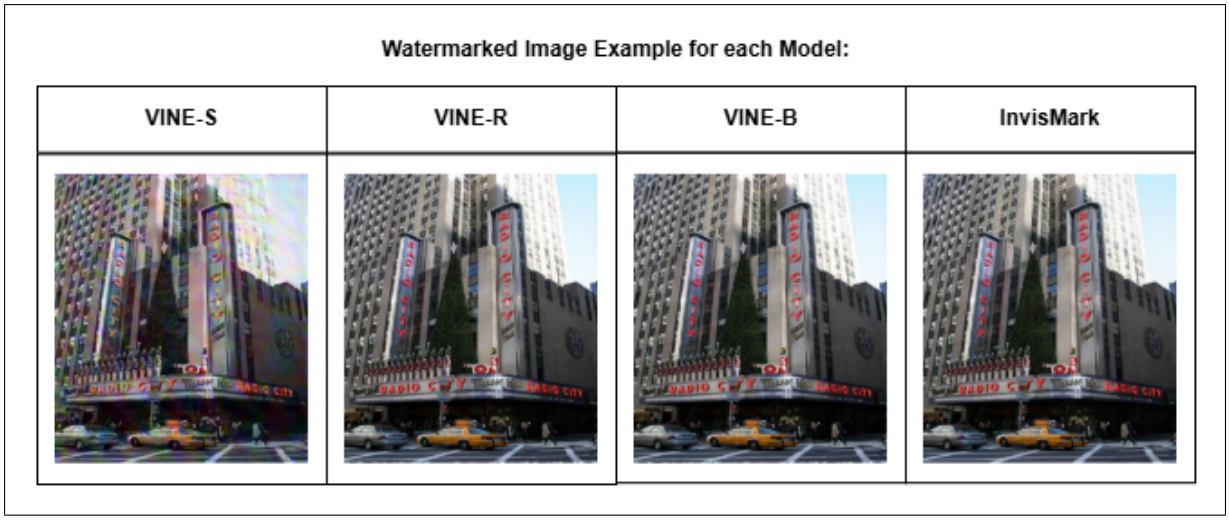


Figure 8: Contains four instances of the same image, each watermarked by a different model; VINE-S(left), VINE-R(centre-left), VINE-B(centre-right) and InvisMark(right). Its purpose is to visually demonstrate the fidelity performance of each model.

Figure 8 visualizes watermarked images from each model. VINE-S outputs exhibit uniform grid patterns, subtle hue shifts, and "rainbow" artifacts throughout the image. Since the stability mask enforces watermark embedding only within specific spatial regions, it introduces localized pressure on the encoder to compress watermark information into constrained areas, which can result in visual artifacts. Additionally, the misalignment of my training loss weights likely resulted in my image fidelity losses not converging correctly, resulting in the high watermark perceptibility seen in the VINE-S model. While the visual performance of VINE-S under-performs relative to VINE-R, this does not completely invalidate the method's potential. The training procedure for VINE-S definitely requires architectural modifications and reconfiguration. With further optimization, such as re-balancing loss weights, visual fidelity could be significantly improved without compromising semantic robustness.

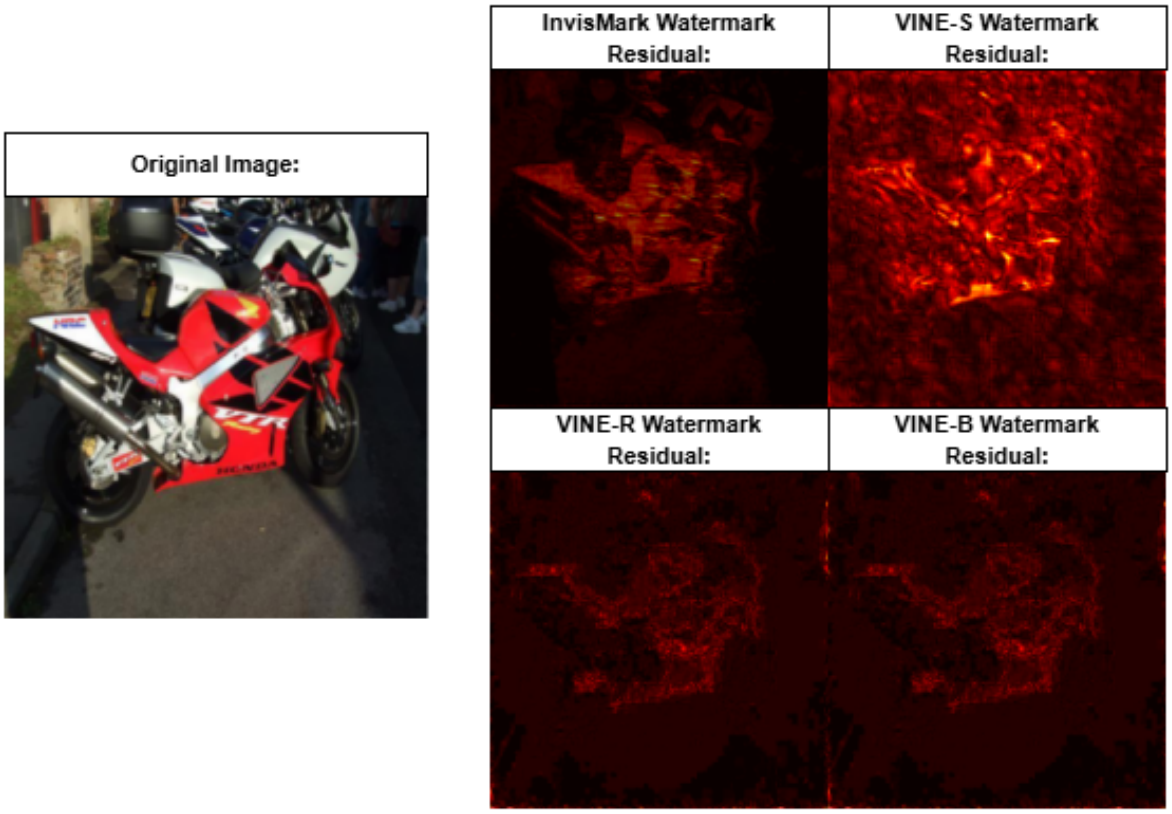


Figure 9: The left-most image is the input image. The grid of heat-maps illustrates average per-pixel differences between the original image and the watermarked image outputs from InvisMark (top-left), VINE-S (top-right), VINE-R (bottom-left) and VINE-B (bottom-right). This is to visualise how well each model reconstructs its watermarked images by inspecting the residual difference from the original image.

The residual heat-maps in Figure 9 visualise the image distortions and misalignments present within the final watermarked images for each model. The VINE-S stands out, as its residual heat-map contains much brighter regions, when compared to the other models residual heat-maps. Showing how the VINE-S's watermarked image contained intense distortions, compared to the other models more imperceptible embedding. Conveying how VINE-S underachieved when it comes to watermark imperceptibility performance.

4.3.2 Robustness Results

	Average BER for Each Distortion Category					
Model	**Noise**	**Blur**	**Colour**	**Compression**	**Global**	**Local**
VINE-S	0.0286	0.0953	0.0672	0.0864	0.1853	0.0156
VINE-R	**0.0045**	**0.0049**	**0.0137**	**0.0073**	**0.0214**	**0.0891**
VINE-B	0.0104	0.0158	0.0215	0.0231	0.0471	0.1964
InvisMark	0.0693	0.1816	0.0667	0.2277	0.2481	0.3092

Table 5: Compares Average Bit Error Rate (BER) of watermark recovery across different distortion categories for VINE-S, VINE-R, VINE-B, and InvisMark. Lower BER indicates greater robustness to the specific manipulation type. VINE-R performs best overall, while InvisMark exhibits the highest error rates across most distortion types. My Model, VINE-S, performs best for local edits.

Table 5 shows that VINE-R is overall the most robust algorithm, across the majority of manipulations used in testing. However, compared to VINE-R and VINE-B, the VINE-S model generally underperformed in terms of watermark extraction accuracy for global and traditional distortions. Across most manipulations, bit accuracy for VINE-S was lower. This likely stems from the semantically constrained embedding strategy. While VINE-R and VINE-B distributes watermark information mostly equally across the image, VINE-S encodes primarily within semantically stable regions. This narrows the spatial redundancy and increases the models sensitivity to edits that disrupt the "stable" regions of the image.

However, in local generative edits (tested using the local-manipulation pipeline from Section 3.2.4) VINE-S showed encouraging improvements over VINE-R. This is likely because VINE-S explicitly avoids encoding in semantically unstable regions. Meaning it retains the embedded watermark more effectively than VINE-R, during local edits.

Although VINE-S robustness against most edit categories failed to reach or outmatch the initial VINE-R architecture, increased resistance to local manipulations demonstrates the potential of semantically-aware embedding methods.

The relatively poorer performance on global edits, though, indicates an important limitation. The semantic embedding method focuses watermark signal in an image's smaller region, lowering redundancy while enhancing fragility under global transformation. There is scope for further refinement of the model, reformulation and retraining of its implementation to achieve an optimal trade-off between robustness and imperceptibility. These results show that although VINE-S compromises certain overall robustness, it achieves significant progress toward edit-aware watermarking.

4.3.3 Semantic Embedding Results

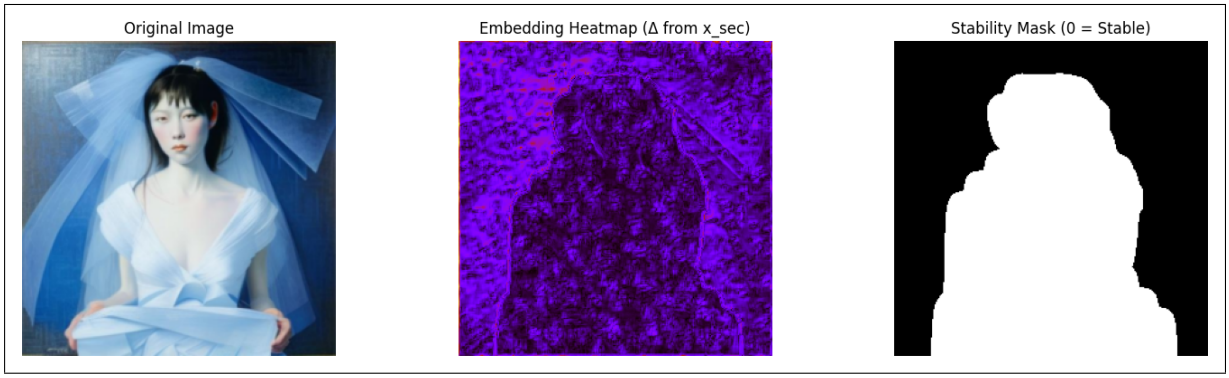


Figure 10: The left-most image in this figure is the input image. The heatmap (centre) illustrates per-pixel differences between the original image and VINE-S's condition adaptor output. The binary mask (right) represents the stability map, predicted by the stability predictor model.

The heat-map seen in Figure 10 represents the absolute pixel-wise difference between the input image and the intermediate encoding output of the Condition Adaptor. A representative heat-map clearly shows that the strongest residual signal, corresponding to where the watermark is embedded, is concentrated in semantically stable regions as predicted by the stability predictor. Whereas the darker region of the heat map, indicating a less

intense watermark signal, occurs within the unstable regions predicted by the stability predictor. This confirms that the model has correctly learned to use the stability mask to inform watermark placement, demonstrating successful semantic integration.

Along side VINE-S's high robustness performance for local generative manipulations, this result directly evidences the successful implementation of semantic encoding within my model. Supporting the theory that semantic embedding would likely improve the VINE-R robustness performance against contextual image edits.

4.4 Results Interpretation and Discussion

4.4.1 Image Fidelity Results Analysis

The image fidelity of my proposed VINE-S model was significantly lower than that of both VINE-B and VINE-R. This result is likely due to a miscalibration during the training process, specifically, in the handling of fidelity losses during the early stages of training. In Stage one, I continued training until bit accuracy reached 0.95 before advancing to Stage two.

However, the original VINE-R paper suggests transitioning at 0.85 bit accuracy. By waiting too long, I gave the binary cross-entropy loss excessive influence. Allowing the model to over-optimize for decoding accuracy while neglecting perceptual quality. Meaning, the image fidelity losses (MSE and LPIPS) were not sufficiently prioritized and therefore failed to properly converge during Stage two and Stage three.

Additionally, the decision not to include a GAN-based discriminator loss in the training pipeline likely contributed to the VINE-S's degraded visual quality. The original VINE models are trained using a perceptual realism loss via a GAN discriminator to further prevent visual artifacts. I excluded this component due to computational constraints. However, this simplification appears to have impacted the model's ability to reconstruct high-quality images under the constraints of semantic embedding.

The VINE-S model introduces fundamental changes to the VINE-R architecture. Namely, semantic-guided embedding, which diverges from the original VINE-R embedding approach. This added complexity means that the original VINE training strategy should have not been copied verbatim. Instead, a reconfiguration of loss weights, loss thresholds is necessary to better support the semantic dynamics introduced by the stability mask. These adjustments were not fully realized due to time, resource and knowledge limitations at the time of training. Retraining with higher LPIPS and MSE loss weighting, earlier Stage two transition, and an additional GAN loss could significantly improve image quality while retaining the semantic robustness that defines VINE-S.

4.4.2 Robustness Results Analysis

The robustness performance of VINE-S yielded inconsistent results in different categories of manipulation. Although the model performed much better under the local generative manipulation, it underperformed with respect to VINE-R in the cases of global and common distortions. This disparity may be attributable to the tentative implementation of the noise layer used during robustness training.

The original VINE paper does not release training code or explicitly detail how this noise layer is implemented. Key uncertainties remain: the framework used for these distortions, the number of manipulations applied per image, the distribution of manipulation strengths, and whether these distortions were applied sequentially or independently. Without these details, I approximated it by randomly selecting one manipulation for each image from the manipulation pool noted earlier (blur, noise, compression, or colour shift), with each manipulation selecting an arbitrarily chosen strength level. These strength values were selected subjectively, with an effort to reproduce distortions from imperceptible to extreme.

While this implementation helped simulate realistic degradations, the lack of precise replication likely hindered the VINE-S models chances of achieving the same level of robustness as VINE-R. The absence of multi-manipulation combinations per image and a more systematic strength scaling may be what caused a weaker watermark signal during VINE-S training, explaining the lower bit accuracy under most distortions.

In contrast, the VINE-S model exhibited high performance in robustness to localized semantic edits. This further supports the theory that because original VINE architecture embeds watermarks uniformly across the image's latent space, it lacks mechanisms to protect against spatially concentrated, local alterations. In my model, the integration of semantic embedding strategy enabled the watermark to be focused in regions less likely to be edited. As a result, VINE-S achieved notably higher bit accuracy under localized attacks, validating the effectiveness of the stability predictor based embedding.

This result underscores the promise of semantically-informed watermarking. Particularly for defending against increasingly common object-level generative edits. However, to achieve competitive robustness across both global, local and common distortions, a more accurate reproduction of the original VINE noise layer and training schedule is likely required.

4.4.3 Semantic Embedding Results Analysis

The residual heat-map observed Figure 11, show significantly higher watermark signal intensity in areas marked stable by the stability mask. This indicates that the model has effectively learned to avoid unstable regions of the image.

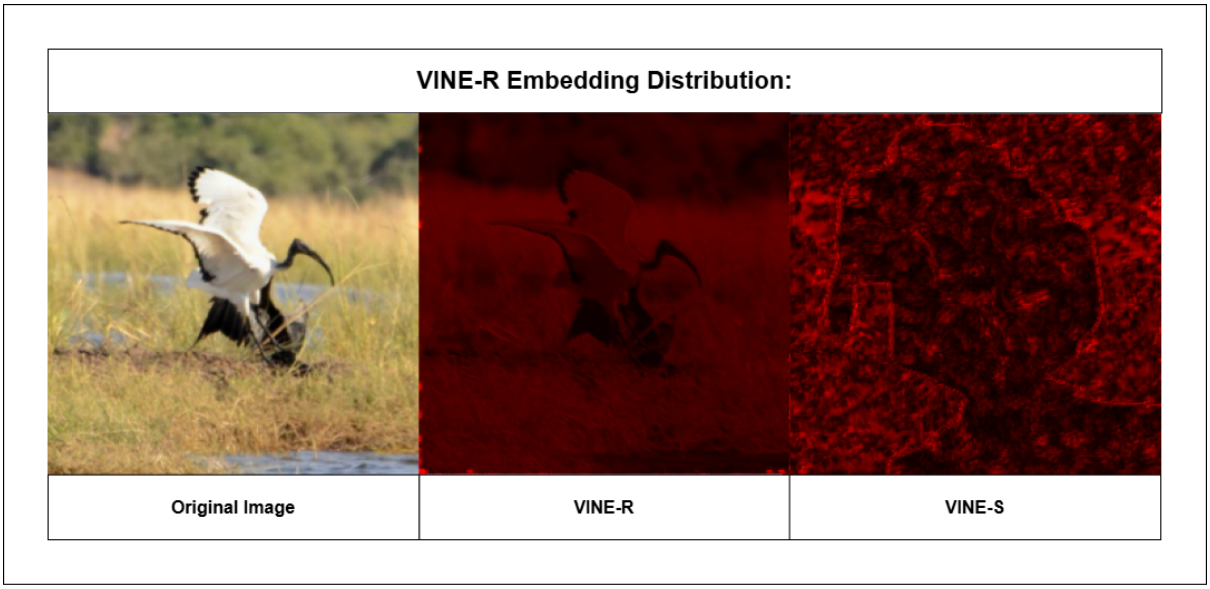


Figure 11: The left-most image is the input image. The heat-maps illustrate per-pixel differences between the original image and the condition adaptor outputs from VINE-R (middle) and VINE-S (right), averaged across RGB channels. VINE-R displays a more uniform embedding across the image, guided only by latent structure. In contrast, VINE-S exhibits more selective embedding. With minimal modification in semantically important area, such as the centre where the bird is positioned. Thus demonstrating effective use of the stability mask to guide watermark placement.

The embedding behaviour of VINE-S in Figure 11 directly supports the improved robustness observed in local edit evaluations using the remove-and-replace pipeline. Unlike the original VINE-R, which embeds watermarks non-semantically across the latent space, VINE-S intentionally unstable regions, making the watermark more resilient to object-level edits.

This method still utilizes a strict thresholding scheme: binary stability mask assigns 50% of image pixels to being unstable, and only half of the image can be utilized for embedding. This threshold is selected for robust semantic targeting, but limits the spatial flexibility of the encoder. If models in the future can decrease the unstable prediction coverage to the range of 30–35% with the same current high prediction accuracy, they would provide an increased and more diverse embedding space, further enhancing fidelity as well as robustness.

The residual heat-map indicates that though the watermark embedding is predominantly localized in the image's regions of semantic stability, there is still a small amount of embedding present in regions of instability. This is not strictly negative, but indicates that the distribution of the watermark still has an element of spatial generality and is not strictly localized to areas that are classified as stable. The higher embedding intensity in stable regions aligns with the objective of enhancing robustness against content-altering manipulations. Meanwhile, the persistent presence of watermark signal in unstable regions may provide some protection in cases where the stability predictor incorrectly estimates the likelihood of region-specific edits

4.4.4 Results Visualisation

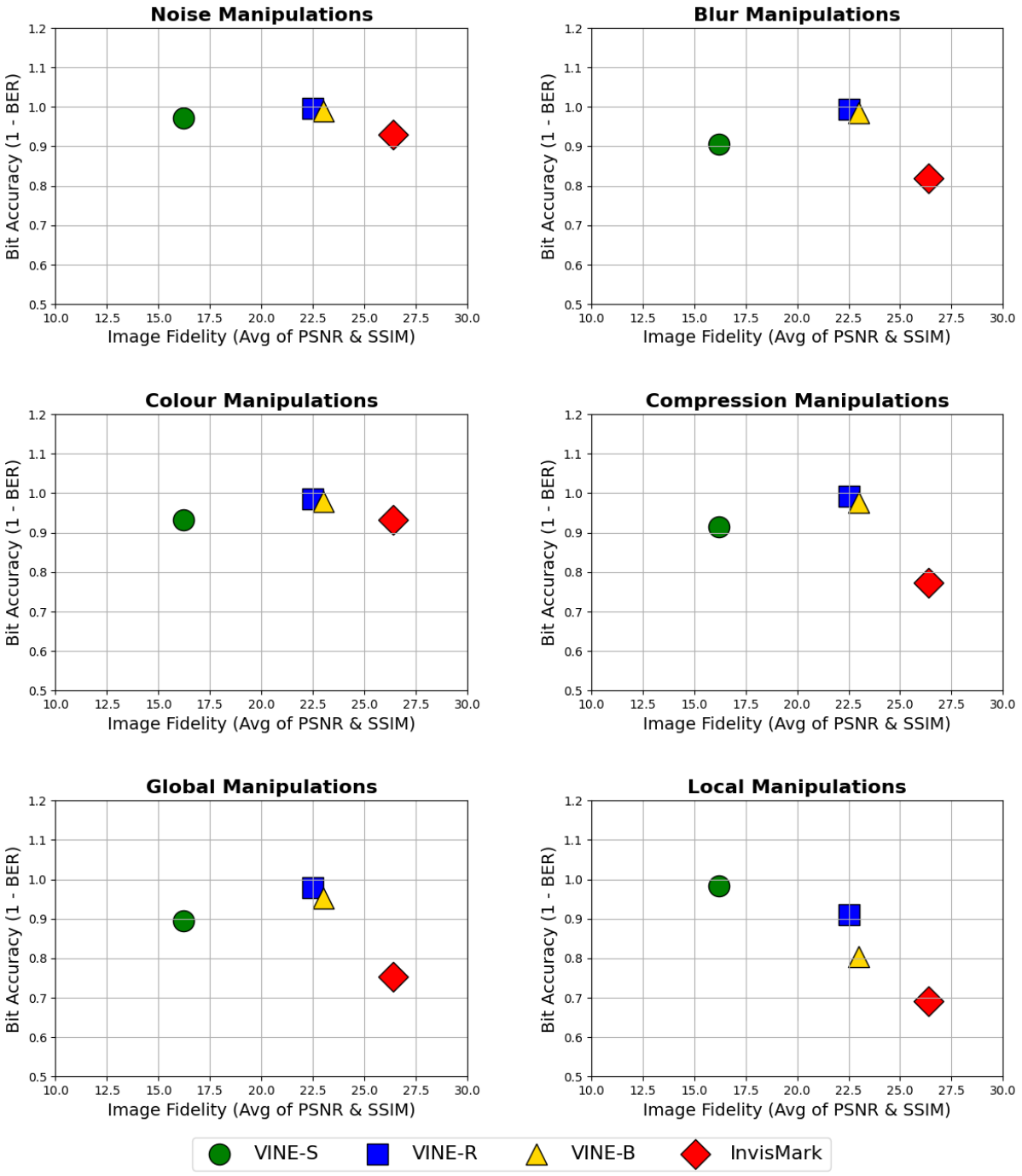


Figure 12: Scatter plots comparing bit accuracy (1-BER) against image fidelity (average of PSNR and SSIM) across six distortion categories. Each point represents a watermarking model, differentiated by color and marker shape: VINE-S (green), VINE-R (blue), VINE-B (yellow), InvisMark (red).

In Figure 12, the three models; VINE-R, VINE-B, and InvisMark, illustrate the common trade-off between watermark robustness and image imperceptibility.

InvisMark demonstrates the highest image fidelity but suffers from the worst robustness. Particularly under compression, global, and local manipulations. Whereas, VINE-R achieves the highest bit accuracy across most distortion categories but does so at the cost of image quality, yielding the lowest fidelity of the group. This visualisation shows how most watermarking models, due to architectural or training configurations, often prioritize robustness or fidelity, usually at the cost of the other.

VINE-S, however, stands out by achieving only moderate results in both dimensions, displaying neither exceptional robustness nor high image fidelity. This outcome may indicate a deeper issue with the model's training process or architectural configuration. Future work should focus on refining the VINE-S architecture and potentially reassessing how semantic information is integrated into the model during training.

4.4.5 Robustness and Fidelity Trade-off

One main challenge of image watermarking is finding the trade-off point between robustness and image fidelity. Maximizing robustness is often achieved by inserting more robust or more redundant signals into an image, which generates visual distortion and impairs perceptual quality. On the other hand, optimization towards imperceptibility often compromises the resistance of the watermark to manipulations. This trade-off is evident in the VINE variants: while VINE-R achieves significantly improved robustness over VINE-B due to its Stage three training against generative edits, it does so at the cost of slightly reduced image fidelity.

In my model, this trade of was evidenced clearly. As, despite closely following the three-stage training protocol outlined for VINE-R, my model underperformed in terms of visual quality. This is likely due to both the inherent difficulty of semantic embedding and the specific training parameters I selected. Additionally, VINE-S's semantic watermarking framework introduces a fundamentally different embedding mechanism, guided by a stability predictor to prioritize stable regions, unlike the unguided embedding in VINE-R.

As a result, my model's training could not be treated as a simple extension of VINE-R's. It likely requires a rebalanced loss configuration and perhaps an entirely new training regimen.

Furthermore, due to time and computational constraints, I could not experiment with different loss weightings or retrain to optimize both robustness and fidelity simultaneously. This restriction probably accounted for suboptimal performance, especially in the fidelity scores.

Ultimately, while my model brought promising improvements in robustness to local edits through semantic embedding, the fidelity deficits show the essential need to re-balance training objectives when architectural alterations are introduced. Finding the optimal trade-off is still an open problem and would be one of the priorities in future versions.

4.4.6 Weaknesses of VINE-S

The most notable weakness of VINE-S is its poor image fidelity. Visual artifacts, such as grid-like distortions, and local colour inconsistencies, appear prominently in watermarked outputs. These are likely due to the over-concentration of the watermark signal into specific regions, a result of prioritizing embedding into semantically stable regions. In contrast to VINE-R, where the watermark signal is spread equally over the image, VINE-S tried to compress it into half the image area. This presents two problems:

1. **Vulnerability to Stability Prediction error:** In the event of a local edit inside the predicted stable area of an image, large areas of the condensed watermark may get erased. This is unlike distributed embedding in VINE-R, which ensures improved spatial spread of the watermark.
2. **Localized Visual Artifacts:** Watermarking in local areas leads to higher perceptual distortion since the watermark signal becomes denser and more prominent in localized areas.

These limitations could be reduced by refining the stability predictor to create a more precise stability mask. For example, one that identifies only 30–40% of pixels as un-

stable, allowing a larger embedding region. Alternatively, somehow integrating BLIP attention maps may improve the mask's precision, although this would require increased computational resources and model complexity.

Another key weakness stems from suboptimal training configurations. The training process was adapted from the original VINE-R pipeline, however, I did this without adjusting for the architectural changes introduced by semantic integration. Meaning the loss weights and transition criteria between stages was not ideal. For instance, running Stage one until a bit accuracy of 0.95 likely prevented the image fidelity loss components from converging in the later training stages. Additionally, GAN loss was omitted from this project due to resource constraints, potentially limiting the visual realism achievable by the generator. Both of these factor likely contributed to the poor image fidelity achieved by VINE-S.

4.4.7 Strengths of VINE-S

In spite of these weakness, VINE-S excels in one area: robustness against local edits. By inserting the majority of the watermark into regions that are expected to be semantically stable, the model exhibits lesser watermark degradation under the remove-and-replace pipeline. This represents a meaningful improvement over both VINE-B and VINE-R, which do not account for semantic stability and thus suffer more under local edits.

Furthermore, while overall robustness is slightly lower than VINE-R under global manipulations and classical distortions, VINE-S still achieves a decoding bit accuracy of ¿0.90 in most scenarios. This suggests that the model has still managed to achieve a decent robustness through following the VINE-R training stages. Despite not achieving this project goal of matching or even surpassing the level of robustness achieved by VINE-R.

In summary, VINE-S's primary strength lies in its innovative approach of semantic-embedding. Demonstrating a promising pathway for integrating semantic awareness into watermarking models. Future iterations could improve both fidelity and robustness by fine-tuning the embedding strategy, optimizing training transitions, and introducing more suitable loss weighting.

4.5 Project Limitations

4.5.1 Project Drawbacks and Training Constraints

Developing a semantic-aware watermarking model based on the VINE architecture posed significant computational and architectural challenges. The proposed model required the integration of a stability predictor and multi-stage training of the models core components to successfully incorporate watermark embedding in the semantically stable regions of an image.

The original VINE-R model was trained on 8 $\times$ A100-80GB GPUs in parallel, providing the authors with the flexibility to experiment with multiple loss configurations, different training regimen and parameter configurations. This parallel environment allowed the developers to refine their training regimen and optimize for both robustness and fidelity through iterative experimentation. In contrast, my training environment was constrained to a single NVIDIA A100 GPU (40GB) in a Google Colab environment. This setup significantly limited my batch size (restricted to 4), as well as the number of feasible experiments per day.

Training the full VINE-S architecture with semantic embedding took nearly 24 hours per full 3-stage training cycle. This left me with limited room for iterative tuning or testing multiple configurations. Therefore, many training parameters, such as learning rates, noise schedules, and stage transitions, were set based on the VINE paper's descriptions rather than empirical tuning due to time and hardware limitations.

Further complicating development was the absence of official training code for VINE-R. The VINE repository did not include training scripts, forcing me to interpret architectural diagrams and textual descriptions from the paper, and attempt to reconstruct the entire training pipeline. This added uncertainty to many implementation decisions, especially the structure of the noise layer.

Despite these constraints, I successfully implemented a decent semantic watermarking model that demonstrates clear improvements in robustness to local manipulations. However, with more computational resources and guidance, and further architectural tuning,

this project could likely have resulted in significantly improved performance for VINE-S across both fidelity and robustness metrics.

5 Conclusion

5.1 Aims and Objectives

The Aim of the project was creating a watermarking technique that would be removal to removal through multiple traditional or generative AI image manipulation. This goal was partially met. Although VINE-S did not surpass the robustness of VINE-R toward all forms of manipulation, it significantly improved local edit robustness because of its semantic embedding strategy. This directly supports the primary aim, although full parity with VINE-R's global manipulation robustness was not achieved, likely due to architectural and resource constraints.

The first objective, analysing existing watermarking methods, was thoroughly addressed through an extensive literature review and initial testing of VINE-B, VINE-R, and InvisMark. These evaluations informed the selection of VINE-R as the base architecture. As well as justified the need for semantic enhancements to improve robustness against local manipulations. The second objective, designing a novel, robust watermarking method, was partially met through the development of VINE-S. This model integrated a custom-trained stability predictor to guide watermark placement into semantically stable image regions, innovating on VINE's core embedding mechanism. Whilst the watermark was constantly extractable from the plain watermark image, the model did not perform well when it came to image fidelity.

For the third objective, a generative removal attack test suite was implemented and applied in a structured evaluation framework. VINE-S was tested under a range of generative and classical distortions using a dedicated benchmarking pipeline used for both initial and final testing.

The fourth objective, evaluating robustness and image fidelity using quantitative metrics,

was achieved. Robustness was assessed using Bit Error Rate and decoding accuracy, while fidelity was measured via PSNR and LPIPS. The results were critically analysed, and the trade-offs between robustness and image quality were thoroughly discussed.

The final objective, involving analysis and future recommendations, was also addressed. The project contains many detailed reflections on model weaknesses. Such as fidelity degradation and sensitivity to stability mask thresholds, for example. I have outlined areas of improvement such as fine-tuning mask precision and training configurations were outlined for future work.

In conclusion, the aim and all five objectives were met to a meaningful extent. With clear innovation, robust evaluation, and insightful analysis underpinning the project's contribution to digital watermarking under generative edits.

5.2 Takeaways

I acquired a broad set of skills during this project, spanning deep learning, digital watermarking, and general academic skills. I developed a strong foundation in deep watermarking architectures, particularly encoder-decoder frameworks. I also gained experience training, creating and adapting deep learning models. These included VINE-S and the stability predictor model. I acquired in-depth knowledge of classical and deep watermarking paradigms. Alongside advanced semantic embedding strategies that consider image structure for improved watermark resilience.

The implementation of a novel watermarking model demanded proficiency in model benchmarking, neural network design, and supervised learning techniques. This included working with labelled datasets and applying loss functions to balance robustness and fidelity. From an engineering perspective, I refined my skills in adapting open-source repositories to constrained environments like Google Colab, while managing model optimization via AMP, mixed precision, and staged training.

Beyond the technical domain, I strengthened my academic research capabilities. This was through carrying out experimental design, critical result evaluation, and structured technical writing. This project also improved my project management skills, especially

when managing limited computational resources and research scope.

5.3 Improvements

- **Improved Time Management:** More effective time allocation during the initial development phase may have notably improved the final outcome of this project. A key issue was the limited training time available, due both to constrained computational resources and the overall project timeline. A significant portion of time was invested in exploring the BLIP-SAM approach to semantic embedding, which ultimately proved impractical given its complexity and hardware demands. In hindsight, an earlier decision to pivot away from this method would have allowed more time to fine-tune the final model, optimize training procedures, and achieve a better balance between robustness and fidelity. Recognizing when to move on from a technically ambitious but unfeasible idea is a valuable lesson, particularly in the context of time-bound research. This experience reinforced the importance of adaptive planning and prioritization in technical project work.

- **Training Re-configuration:** The decision to adopt a training configuration closely aligned with the original VINE-R model was, in retrospect, less than ideal. While the intention was replicate a robust framework, this approach didn't account for the fundamental architectural differences introduced in VINE-S. These modifications significantly altered how the watermark is embedded. Meaning that many of the assumptions determining VINE-R's training strategy were no longer applicable. A rigorous assessment of VINE-S's architectural implications would have better informed a new custom training strategy. This could have lead to improved convergence and overall performance.

- **Pre-trained Approach:** Attempting to reuse VINE-R's pretrained weights, after modifying the Condition Adaptor to embed semantically, delayed progress. Analysis of the VINE-R model's architectural dependencies, such as unguided embedding, could have prevented wasted time attempting training VINE-S using the VINE-R pre-trained weights.

- **Greater Evaluation Diversity:** Would have significantly strengthened the gen-

eralizability and validity of the benchmarking process of this project. While the current evaluation included both local and global manipulations, incorporating additional generative manipulation pipelines would have better reflected overall robustness performance to generative edits. The evaluation would have been notably enhanced by integrating the full set of manipulations used in the original VINE benchmarking framework (W-Bench). Doing so would have enabled a more direct, meaningful and generalized comparison between VINE-R and VINE-S.

5.4 Future Work

The VINE-S model demonstrates promising robustness, however its performance is only exemplary against local edit scenarios. Meaning it doesn't fully matched VINE-R's overall robustness for other manipulation types. To move forward, the model must be reconfigured to improve the current balance between watermark imperceptibility and robustness.This could be achieved through retraining using parameters and loss weights better catered to VINE-S's semantic embedding feature. As well as re-implementing the noise layer to truly replicate the pipeline used in the VINE-R model. If through these improvements, the VINE-S model can achieve a level of robustness that exceeds or matches VINE-R for common and global edits, then I would proceed with the following developments.

One future direction involves improving the precision of the stability predictor. Currently, the stability maps designates half the image pixels as stable regions. Causing watermark signals to become condensed, resulting in poor fidelity and robustness if the stability predictor is incorrect. By enhancing the stability predictor, possibly by revisiting BLIP and SAM integration for refining the existing CNN model, it could learn to predict unstable regions with the same current degree of accuracy, but at a much smaller scale. Enhancing the the stability predictors precision in this way would mean that the watermark has more stable region to embed its signal within the image. Likely leading to less visual artifacts, less condensed and vulnerability watermark signals, and possibly an improved semantic accuracy.

If these refinements can elevate general robustness, VINE-S could be further extended to

support broader research on semantically conditioned watermarking. Additionally, the stability predictor component could be modularly integrated into other watermarking frameworks to enhance their robustness against localized, context-driven edits.

Another key area for future development would be retraining the stability predictor exclusively on human-edited images. In this project, the limited size of the HumanEdit dataset (5,700 images) lead to the addition of PIPE. Which uses synthetic prompts to manipulate images. These synthetic manipulations do not accurately capture the patterns of human editing. A future extension of this work should involve curating or sourcing a larger dataset composed entirely of human edited image–mask pairs. This would enable the stability predictor to better learn real-world editing tendencies. Better preparing it for practical, real-word deployment scenarios.

In industrial contexts, a robust and semantically-aware watermarking system has strong potential applications. In journalism, where visual media plays a critical role in conveying truth, images are particularly susceptible to tampering. This is usually done to mislead or manipulate narratives. These edits are often localized, making conventional robust watermarking methods vulnerable. However, VINE-S offers a valuable solution by embedding watermarks into semantically stable regions of an image. This makes it highly applicable in journalistic contexts, where ensuring the authenticity and provenance of imagery is essential for maintaining public trust and combating misinformation in an era of AI-generated content.F

6 References

6.1 References

[1] Stability AI. Stable Diffusion. https://stability.ai/blog/stable-diffusion-announcement, 2022. Accessed: 2025-04-10.

[2] L Aljuiad and D Bhowmik. Fusion++: A method to detect generative ai manipulated images. *Newcaslte University*, 2024.

[3] Kasra Arabi, R. Teal Witter, Chinmay Hegde, and Niv Cohen. Seal: Semantic aware image watermarking, 2025. URL https://arxiv.org/abs/2503.12172.

[4] Jinbin Bai, Wei Chow, Ling Yang, Xiangtai Li, Juncheng Li, Hanwang Zhang, and Shuicheng Yan. Humanedit: A high-quality human-rewarded dataset for instruction-based image editing. *arXiv preprint arXiv:2412.04280*, 2024.

[5] Mauro Barni, Franco Bartolini, Vito Cappellini, Alessandro Lippi, and Alessandro Piva. Dwt-based technique for spatio-frequency masking of digital signatures. In *Security and Watermarking of Multimedia Contents*, volume 3657, pages 31–39. SPIE, 1999.

[6] Walter Bender, Daniel Gruhl, Norishige Morimoto, and Anthony Lu. Techniques for data hiding. *IBM systems journal*, 35(3.4):313–336, 1996.

[7] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions, 2023. URL https://arxiv.org/abs/2211.09800.

[8] Tu Bui, Shruti Agarwal, Ning Yu, and John Collomosse. Rosteals: Robust steganography using autoencoder latent space, 2023. URL https://arxiv.org/abs/2304.03400.

[9] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Kannan Lee, Zhiwei Steven Gong, Andy Zou, Colin Raffel, et al. The fundamental insecurity of watermarking for large language models. *arXiv preprint arXiv:2310.11367*, 2023.

[10] Chi-Kwong Chan and Lee-Ming Cheng. Hiding data in images by simple lsb substitution. *Pattern recognition*, 37(3):469–474, 2004.

[11] Ingemar J Cox, Joe Kilian, F Thomson Leighton, and Talal Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE transactions on image processing*, 6 (12):1673–1687, 1997.

[12] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22466–22477, 2023.

[13] Google DeepMind. SynthID: Identifying AI-generated content. `https://deepmind.google/technologies/synthid/`, 2023. Accessed: 2025-04-10.

[14] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.

[15] Runyi Hu, Jie Zhang, Ting Xu, Jiwei Li, and Tianwei Zhang. Robust-wide: Robust watermarking against instruction-driven image editing, 2024. URL `https://arxiv.org/abs/2402.12688`.

[16] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 172–189, 2018.

[17] Manie Kansal, Gursharanjeet Singh, and BV Kranthi. Dwt, dct and svd based digital image watermarking. In *2012 International Conference on Computing Sciences*, pages 77–81. IEEE, 2012.

[18] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. URL `https://arxiv.org/abs/2304.02643`.

[20] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision*, 128(7):1956–1981, 2020.

[21] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014.

[23] Ruizhen Liu and Tieniu Tan. An svd-based watermarking scheme for protecting rightful ownership. *IEEE transactions on multimedia*, 4(1):121–128, 2002.

[24] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[25] Shilin Lu, Zihan Zhou, Jiayou Lu, Yuanzhi Zhu, and Adams Wai-Kin Kong. Robust watermarking using generative priors against image editing: From benchmarking to advances, 2025. URL `https://arxiv.org/abs/2410.18775`.

[26] OpenAI. Dall·e 2: Creating images from text. `https://openai.com/dall-e-2`, 2022. Accessed: 2025-05-14.

[27] OpenAI. DALL·E 3. `https://openai.com/dall-e`, 2023. Accessed: 2025-05-14.

[28] OpenAI. Gpt-3.5 technical overview. `https://platform.openai.com/docs/models/gpt-3-5`, 2023. Accessed: 2025-05-14.

[29] Minzhou Pan, Yi Zeng, Xue Lin, Ning Yu, Cho-Jui Hsieh, Peter Henderson, and Ruoxi Jia. Jigmark: A black-box approach for enhancing image watermarks against diffusion model edits. *arXiv preprint arXiv:2406.03720*, 2024.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. URL `https://arxiv.org/abs/2103.00020`.

[31] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pages 87–103. Springer, 2024.

[32] Yinan Sun, Xiongkuo Min, Huiyu Duan, and Guangtao Zhai. How is visual attention influenced by text guidance? database and model. *IEEE Transactions on Image Processing*, 2024.

[33] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159, 2022.

[34] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2117–2126, 2020.

[35] The White House. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence. `https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artifi` 2023. Accessed: 2025-02-20.

[36] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, et al. Yolov10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems*, 37:107984–108011, 2024.

[37] Navve Wasserman, Noam Rotstein, Roy Ganz, and Ron Kimmel. Paint by inpaint: Learning to add image objects by removing them first. *arXiv preprint arXiv:2404.18212*, 2024.

[38] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*, 2023.

[39] Rui Xu, Mengya Hu, Deren Lei, Yaxi Li, David Lowe, Alex Gorevski, Mingyu Wang, Emily Ching, and Alex Deng. Invismark: Invisible and robust watermarking for ai-generated image provenance, 2024. URL `https://arxiv.org/abs/2411.07795`.

[40] Minerva M Yeung and Fred Mintzer. An invisible watermarking technique for image verification. In *Proceedings of international conference on image processing*, volume 2, pages 680–683. IEEE, 1997.

[41] Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285*, 2019.

[42] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. URL `https://arxiv.org/abs/2302.05543`.

[43] Haozhe Zhao, Xiaojian Ma, Liang Chen, Shuzheng Si, Rujie Wu, Kaikai An, Peiyu Yu, Minjia Zhang, Qing Li, and Baobao Chang. Ultraedit: Instruction-based fine-grained image editing at scale, 2024. URL `https://arxiv.org/abs/2407.05282`.

[44] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672, 2018.