Watermarking in Diffusion Model: Gaussian Shading with Exact Diffusion Inversion via Coupled Transformations (EDICT)

Krishna Panthi
Clemson University, SC, USA

1. Introduction

AI-generated images have become increasingly prevalent. Just a few years ago, it was very expensive to generate realistic-looking images using computers. But now, in 2024, they have become the norm. It's becoming harder and harder to tell whether an image is real or AI-generated. Images have the capability to deceive people who view them. When a person sees an image, it leaves a long-lasting impression on their mind. Unlike text, understanding an image does not require much time. Consequently, it has become far easier to spread misinformation.

For example, an AI-generated image was used to report fake news about an explosion outside the Pentagon on the social media platform X. This caused stock prices to drop temporarily. This highlights the importance of identifying whether a given image was generated by AI or not. Furthermore, being able to identify who generated the image could help deter people from creating such misleading images.

Diffusion models are the current state-of-the-art approach to generating images. Most modern advanced systems, such as MidJourney, DALL-E, and Stable Diffusion, all use some form of diffusion models to generate images. Diffusion models generate images by iteratively denoising input Gaussian noise in the direction of the prompt. If we can embed some form of secret watermark into the image generation process, we could later recover it to determine whether the image was generated by a specific model or even trace its creator. Gaussian Shading [7] is one such technique that embeds a watermark into the noise itself, which is used to generate the image at runtime. The same model can then be used to recover the noise and, ultimately, the watermark for detection and traceability. This method works with diffusion models that use Denoising Diffusion Implicit Models (DDIMs)[4], which employ non-Markovian methods of image generation. The challenge, however, is that the initial noise must be recoverable from the image for this method to be successful.

Even though DDIMs are non-Markovian models, it is still not possible to fully recover the initial noise. DDIM inversion for images is unstable because it relies on local linearization assumptions. These assumptions result in the propagation of errors, leading to incorrect image reconstruction and loss of content. This also affects the embedded watermark, making it harder to recover. This is especially true when images are manipulated after being generated.

Exact Diffusion Inversion via Coupled Transformations (EDICT)[6] is an inversion method inspired by affine coupling layers. EDICT enables mathematically exact inversion of real and model-generated images by maintaining two coupled noise vectors that invert each other in an alternating fashion. In this paper, we implement EDICT alongside the Gaussian Shading watermarking technique and conduct experiments to determine if it improves performance.

Both EDICT and Gaussian Shading do not require model training or fine-tuning. We observed a slight increase in the performance of Gaussian Shading when EDICT is implemented.

Project Website: https://krishnapanthi.com/projects/gaussian-shading-with-edict/

2. Related Work

This work builds on two key contributions. The first is Gaussian Shading [7], a watermarking technique that embeds a unique watermark directly into latent noise. It works with DDIM based latent diffusion models. A model owner can integrate such a watermark, the resulting images carry a signature that proves it's authenticity as well as it can be traced back to user who generated it. If malicious users generate harmful or misleading content, the traceable watermark can identify the user, helping to hold the user accountable and discouraging misuse.

Let us consider c x h x w is the dimension of the latent. In such a scenario, the watermark capacity becomes l x c x h x w. To enhance robustness, the watermark is represented by $\frac{1}{f_{hw}}$ of the height and width, $\frac{1}{f_c}$ of the channel. The watermark is then expanded to fill the latent by $f_{hw}^2 \cdot f_c$. The watermark has capacity $l \times \frac{c}{f_c} \times \frac{h}{f_{hw}} \times \frac{w}{f_{hw}}$. Let s^d be the diffused watermark. The diffused watermark is encrypted with a key K. The resulting message m is a randomized binary string and has a uniform distribution, which is a property of a good encryption algorithm. l refers to the number of bits represented by each dimension which results in an integer y in the range [0, 1,...,2^{l-1}].

To embed y, the standard normal distribution $N(0, 1)$ is divided into 2^l equal cumulative probability portions. For implementation with l=1, this means for each bit value of 0, we sample from a truncated standard normal distribution $N(0, 1)$ in the range $[-\infty, 0)$, and for each bit value of 1, we sample from a truncated standard normal distribution $N(0, 1)$ in the range $[0, \infty)$.

The second work is Exact Diffusion Inversion via Coupled Transformations (EDICT) [6]. EDICT is inspired by affine coupling layers (ACL) from invertible neural networks [1, 2]. This process ensures a mathematically exact inversion of diffusion processes by utilizing two coupled latents that invert each other in an alternating manner. Unlike standard DDIM inversion, which is prone to errors and leads to imperfect reconstruction, EDICT guarantees the recovery of the original noise latent without approximation.

The way it works is in a normal diffusion one latent is used for denoising and noising in the forward and reverse process, however in EDICT in each time step, two latents are used, also called coupled latents where one latent is used to noise or denoise other in an alternative way. After that they are mixed together. And the same process is repeater for all time steps. Given x_t and y_t are the latents at time t, and $0 \leq p \leq 1$ is mixing factor, we have the next latents x_{t-1}, y_{t-1} calculated in denoising process as

$$\begin{aligned} x_t^{inter} &= \text{denoise}(x_t, t, y_t) \\ y_t^{inter} &= \text{denoise}(y_t, t, x_t^{inter}) \\ x_{t-1} &= p \cdot x_t^{inter} + (1-p) \cdot y_t^{inter} \\ y_{t-1} &= p \cdot y_t^{inter} + (1-p) \cdot x_{t-1} \end{aligned}$$

and the deterministic noising inversion process is

$$\begin{aligned} y_{t+1}^{inter} &= (y_t - (1-p) \cdot x_t)/p \\ x_{t+1}^{inter} &= (x_t - (1-p) \cdot y_{t+1}^{inter})/p \\ y_{t+1} &= \text{addnoise}(y_{t+1}^{inter}, t+1, x_{t+1}^{inter}) \\ x_{t+1} &= \text{addnoise}(x_{t+1}^{inter}, t+1, y_{t+1}) \end{aligned}$$

$denoise(x, t, y)$, given a latent variable x at time step t and another latent y, it applies the model's reverse diffusion step. It typically employs a neural network conditioned on y and time t to estimate the denoised version of x, reducing the noise from the current latent representation.

$addnoise(x, t, y)$, given a latent variable x at time step t and another latent y, it applies the forward diffusion step. It adds a controlled amount of noise to x, according to a predefined noise schedule, optionally conditioned on y and the current time step t. This operation effectively inverts the denoising process, enabling transitions between different time steps.

In practice, the order in which the x and y series are calculated are alternated at each time step in order to symmetrize the process with respect to both sequences.

Combining these approaches is novel because it enhances the watermarking method's reliability under a wide range of transformations and image manipulations. While Gaussian Shading can embed a watermark, and DDIM-based methods can partially invert the process, EDICT's exact inversion makes it possible to precisely extract the embedded watermark from generated images. Our extension focuses on leveraging EDICT to maintain higher fidelity in watermark recovery, something previous work did not fully address.

3. Approach

Our approach involves implementing EDICT with Gaussian Shading. In Gaussian Shading, a single noise latent is iteratively denoised to obtain the denoised latent, and single image latent is iteratively noised to obtain the noisy latent. In EDICT, the latent is duplicated to obtain two latents. The pseudo code of the implementation is as follows.

```
Input:
$W$ : Watermark to be Embedded
$Decoder(\cdot)$: Image generation model
$Encoder(\cdot)$: Latent space encoding model
$EDICT\_denoise(\cdot, \cdot)$: EDICT based bidirectional denoising process.
$EDICT\_reverse(\cdot, \cdot)$: Reverse EDICT process.
$T$: Total timesteps in the denoising process.

Output:
Recovered watermark: $\hat{W}$.
Generated Image: $I$.

Algorithm:

Forward Process (Denoising)
1. $x_t \leftarrow EmbedWatermark(W)$
2. $y_t \leftarrow Duplicate(x_1)$
3. $(x_0, y_0) \leftarrow EDICT\_denoise(x_t, t, y_t)$
4. $I \leftarrow Decoder(x_0)$

Reverse Process (Adding Noise)
1. $x_0 \leftarrow Encoder(I)$
2. $y_0 \leftarrow Duplicate(x_0)$
3. $(x_t, y_t) \leftarrow EDICT\_reverse(x_0, t, y_0)$
4. $\hat{W} \leftarrow RecoverWatermark(x_t)$
5. $Compare(W, \hat{W})$
```

In essence, we use two coupled latents (x and y) that guide each other's denoising in the forward pass, and then use the reverse operation to reintroduce noise step-by-step.

4. Experiments and Results

We evaluated our approach by comparing Gaussian Shading's watermark recovery performance with and without EDICT. The experiment was conducted with Stable Diffusion 2.1 [3] provided by huggingface. The size of the generated images is 512×512, and the latent space dimension is $4 \times 64 \times 64$. During inference we emply the prompt from Stable-Diffusion-Prompt with a guidance scale of 7.5 similar to the Gaussian Shading paper. We sample 50 steps using DDIMSolver [5]. 50 steps of DDIM inversion was performed. For the experiment the settings of Gaussian Shading were $f_c = 1$, $f_{hw} = 8$, $l = 1$, resulting in an actual capacity of 256 bits. For robustness, we subjected generated images to nine different noise as shown in Figure 1.

All experiments are conducted using the PyTorch 2.5.1 framework, running on a single A100 GPU.

Evaluation metrics: For detection, we calculate the true positive rate (TPR) corresponding to a fixed false positive rate (FPR). For traceability, bit accuracy is calculated.

4.1. Performance

For detection, we consider Gaussian Shading a single bit watermark, with a fixed watermark. We approximate a fixed FPR of 10^{-6}, calculate the corresponding threshold τ which comes out to be approximately equal to 78% of bits being correct and test the TPR on 1000 images. As seen in the results table, when using EDICT the TPR reduces for ColorJitter and SPNoise, increases for GauNoise and remains same for all others.

For traceability, Gaussian Shading serves as a multi-bit watermark. In the experiment, we consider that 1,000 users generate 10 images each with a watermark, resulting to 10,000 watermarked images. During testing we calculate the threshold τ to control the FPR at 10^{-6} which approximately equals 88% of bits being correct. As seen in the results table, when using EDICT, the traceability improves for GauNoise, Random Drop and SPNoise but reduces for ColorJitter, remains same for all others.

We also calculated the bit accuracy for both EDICT implementation and baseline to compare the results. As seen in the table, EDICT improves bit accuracy for 7 out of 9 noise addition methods. The accuracy goes slightly down for Brightness and S & P Noise.

5. Conclusions

We integrated EDICT's exact inversion capability with Gaussian Shading's watermarking method to improve the fidelity of watermark recovery in diffusion-generated images. Our experiments show that while the gains are not always dramatic, EDICT does provide a more stable inversion process, reducing errors in watermark recovery, particularly in challenging transformations. The method does not require model retraining or finetuning, which makes it easily adaptable.

Limitations include the complexity of implementing exact inversion and the computational overhead introduced by EDICT's coupled transforms. This method is 2 x slower than the baseline. In the future we will try to improve the performance of EDICT. We will also explore more faster exact inversion techniques. Similarly, we will try to extend these concepts beyond DDIM-based diffusion models to other generative frameworks.

References

[1] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation, 2015. 2

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2017. 2

[3] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 3

[4] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, 2020. 1

[5] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 3

[6] Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. *arXiv preprint arXiv:2211.12446*, 2022. 1, 2

[7] Zijin Yang, Kai Zeng, Kejiang Chen, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian shading: Provable performance-lossless image watermarking for diffusion models. *arXiv preprint arXiv:2404.04956*, 2024. 1

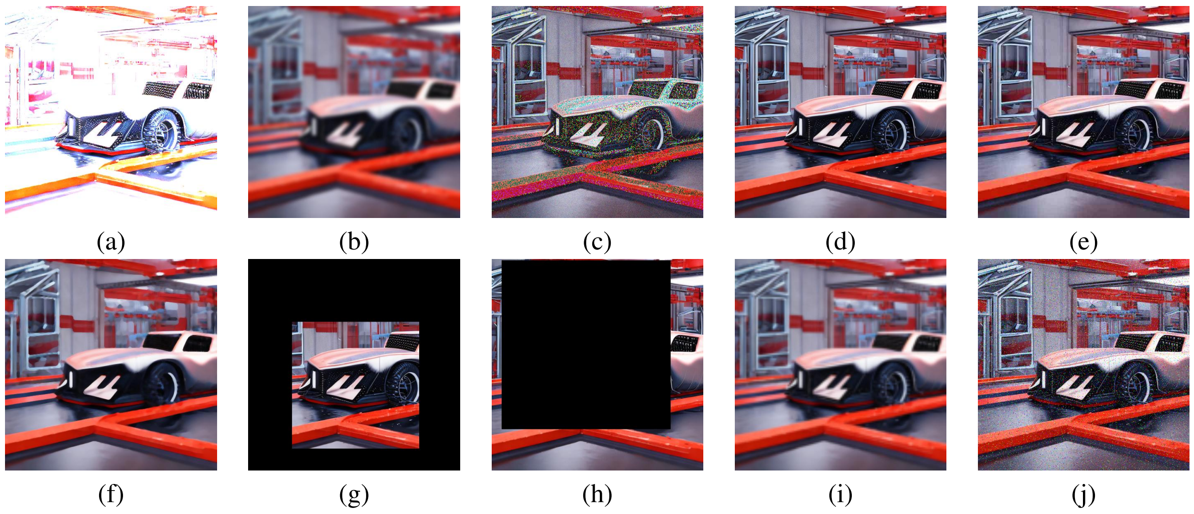

Figure 1. Example of images manipulated. (a) Brightness, factor = 6 (Color Jitter), (b) Gaussian Blur, r=4 (GauBlur), (c) Gaussian Noise, $\mu = 0, \sigma = 0.05$ (GauNoise), (d) Identity, (e) JPEG, QF=25, (f) Median Filter, k=7 (MedBlur), (g) 60% area random crop, (h) 80% area random drop, (i) 25% Resize and restore (Resize), (j) Salt and Pepper Noise, p = 0.05 (S&PNoise)

	TPR_detection ↑		**TPR_traceability ↑**		**mean_accuracy ↑**		**σ_accuracy ↓**	
	Default	EDICT	Default	EDICT	Default	EDICT	Default	EDICT
ColorJitter	**0.979**	0.959	**0.957**	0.934	**0.952**	0.939	**0.092**	0.107
GauBlur	1	1	1	1	0.985	**0.988**	0.020	**0.015**
GauNoise	0.995	**0.998**	0.986	**0.995**	0.954	**0.971**	0.070	**0.053**
Identity	1	1	1	1	1.0	1	0	0
Jpeg	1	1	1	1	0.987	0.987	**0.031**	0.032
MedBlur	1	1	1	1	0.999	**1.0**	0.005	**0.002**
RandomCrop	1	1	1	1	0.975	**0.976**	0.017	**0.013**
RandomDrop	1	1	0.998	**1**	0.966	**0.969**	0.029	**0.020**
Resize	1	1	1	1	0.997	**0.999**	0.009	**0.003**
SPNoise	**1**	0.999	0.985	**0.99**	**0.935**	0.934	0.071	**0.067**

Figure 2. The table shows the results obtained by testing our method against the baseline. It demonstrates that when EDICT is used, performance improves or remains consistent across all image manipulation methods, except when brightness is increased (ColorJitter) and when Salt and Pepper noise is added.