NEURAL INFORMATION PROCESSING SYSTEMS

浙江大学 区块链与数据安全全国重点实验室
STATE KEY LABORATORY OF BLOCKCHAIN AND DATA SECURITY
ZHEJIANG UNIVERSITY

浙江大学滨江研究院
BINJIANG INSTITUTE OF ZHEJIANG UNIVERSITY

WMCopier: Forging Invisible Watermarks on Arbitrary Images

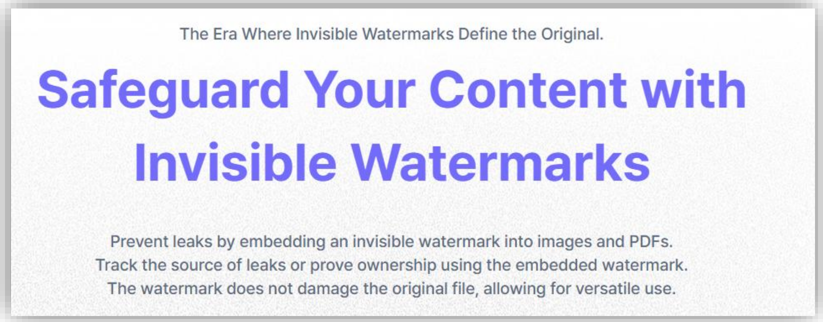
Ziping Dong[1], Chao Shuai[1] Zhongjie Ba[1,2] *, Peng Cheng[1,2],
Zhan Qin[1,2], Qinglong Wang[1,2], Kui Ren[1,2]

[1]The State Key Laboratory of Blockchain and Data Security, Zhejiang University
[2]Hangzhou High−Tech Zone (Binjiang) Institute of Blockchain and Data Security

NEURAL INFORMATION PROCESSING SYSTEMS

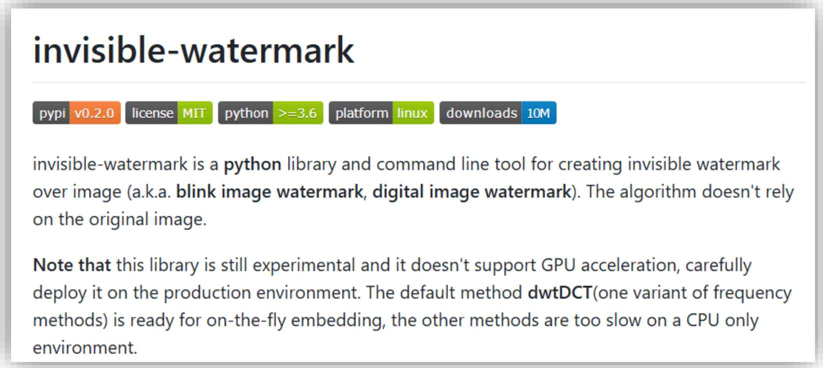
Background

Image Watermarking is becoming a key technique for copyright and AIGC Detection.


The Era Where Invisible Watermarks Define the Original.

Safeguard Your Content with Invisible Watermarks

Prevent leaks by embedding an invisible watermark into images and PDFs.
Track the source of leaks or prove ownership using the embedded watermark.
The watermark does not damage the original file, allowing for versatile use.

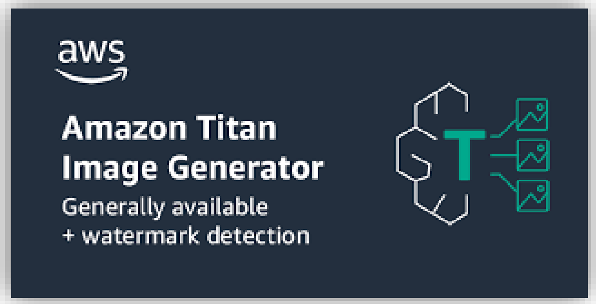

invisible-watermark

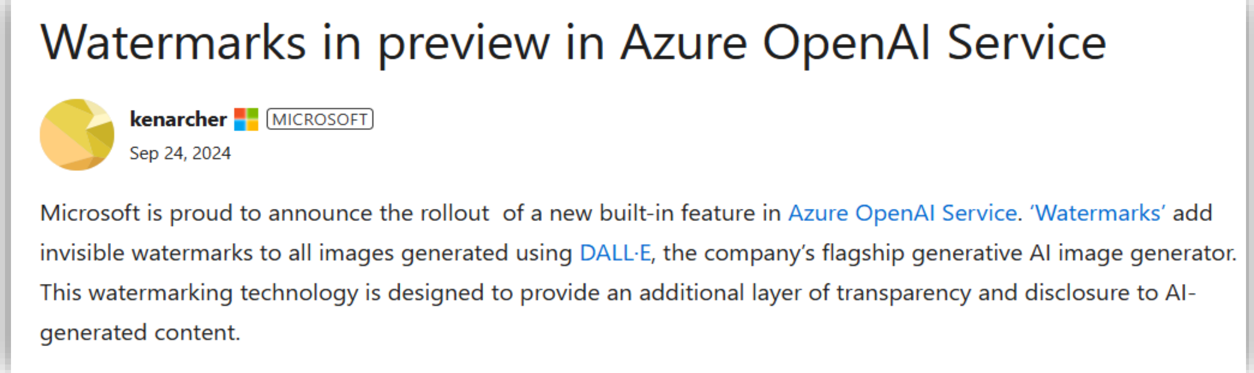
pypi v0.2.0 | license MIT | python >=3.6 | platform linux | downloads 10M

invisible-watermark is a **python** library and command line tool for creating invisible watermark over image (a.k.a. **blink image watermark**, **digital image watermark**). The algorithm doesn't rely on the original image.

Note that this library is still experimental and it doesn't support GPU acceleration, carefully deploy it on the production environment. The default method **dwtDCT**(one variant of frequency methods) is ready for on-the-fly embedding, the other methods are too slow on a CPU only environment.

Copyright Protection




Watermarks in preview in Azure OpenAI Service

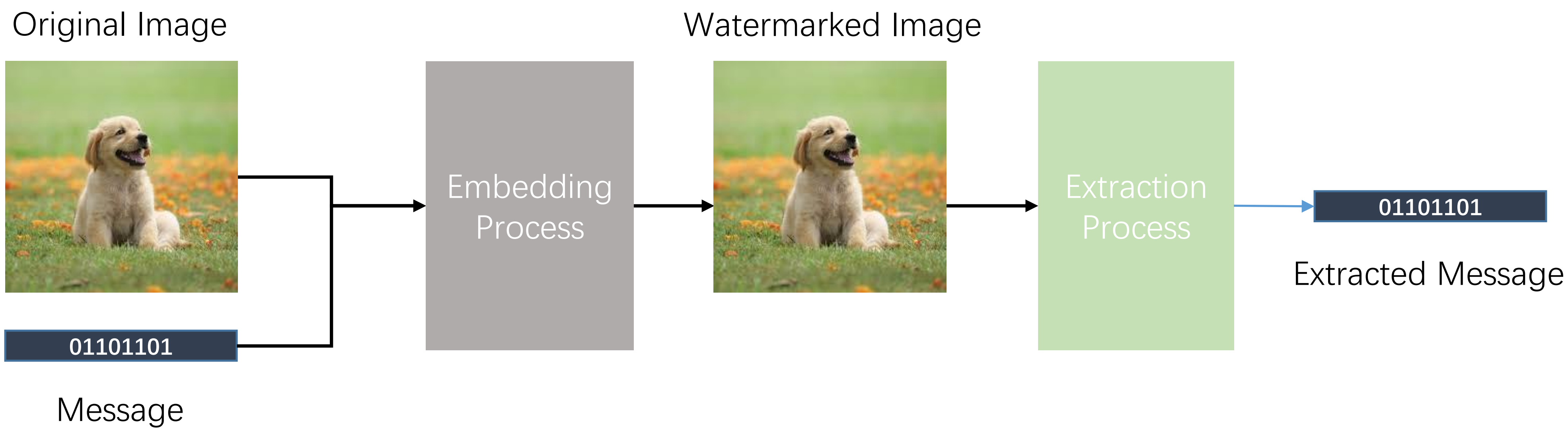
kenarcher MICROSOFT
Sep 24, 2024

Microsoft is proud to announce the rollout of a new built-in feature in Azure OpenAI Service. 'Watermarks' add invisible watermarks to all images generated using DALL·E, the company's flagship generative AI image generator. This watermarking technology is designed to provide an additional layer of transparency and disclosure to AI-generated content.

AIGC Detection

[1] https://github.com/ShieldMnt/invisible-watermark
[2] https://www.saforus.com
[3] https://deepmind.google/science/synthid

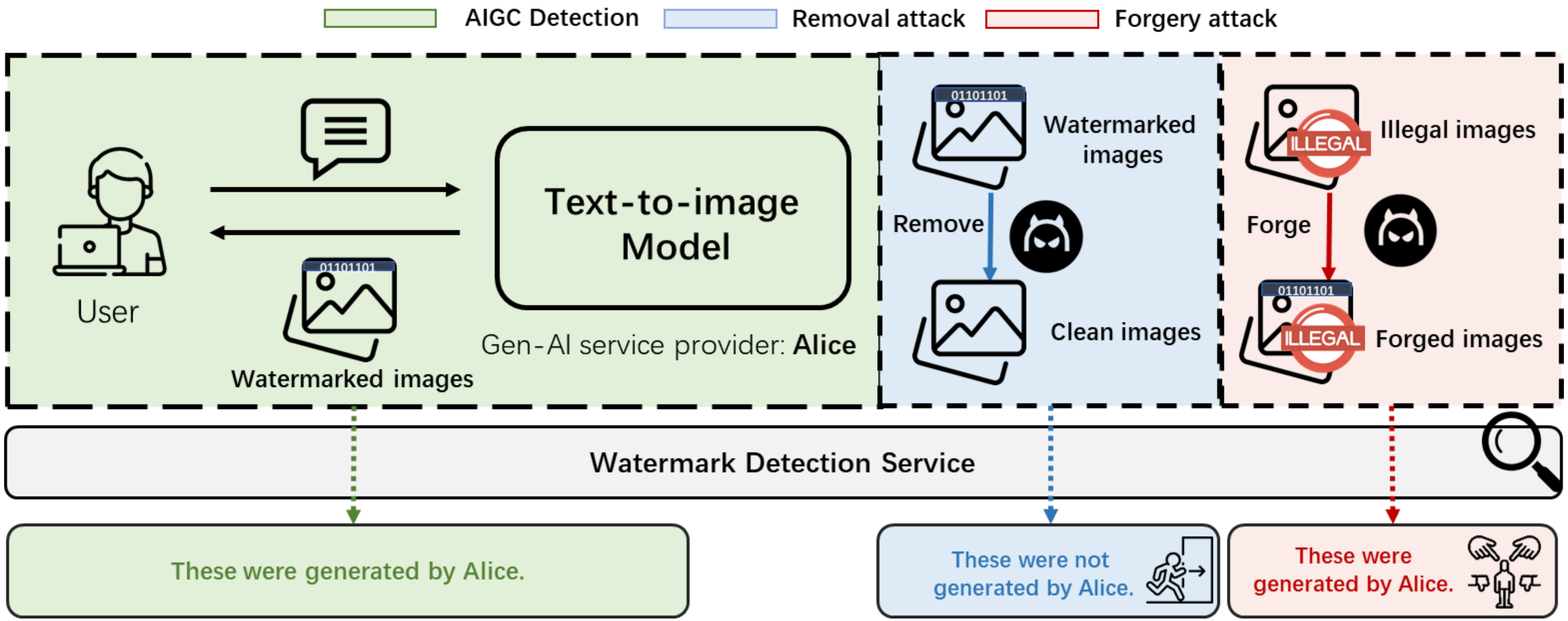
Background



Overview of image watermark embedding and extraction.

Background

Watermarking are vulnerable to adversarial attacks.



Threat Model

Attacker's Goal.

x : clean image (no watermark)

x_f : forged image created by the attacker

✔ **Visual Fidelity** $\mathcal{L}_{\text{visual}}(x_f, x) \leq \epsilon$

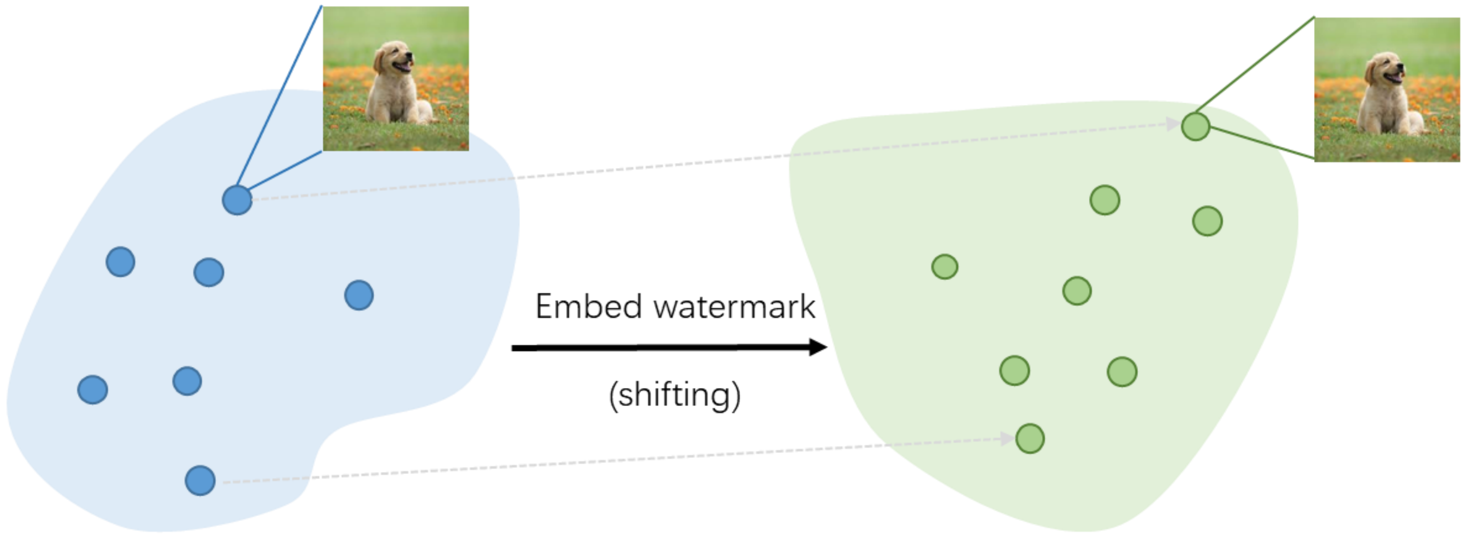
✔ **Forgery Efficiency** $\text{Extraction}(x_f) = m$

Attacker's Capability.

- Don't know any knowledge(parameters, message, scheme) of target watermarking scheme
- No access to embedding or detection(extraction) pipeline
- Can only collect watermarked images from the internet or by using GenAI service

NEURAL INFORMATION PROCESSING SYSTEMS

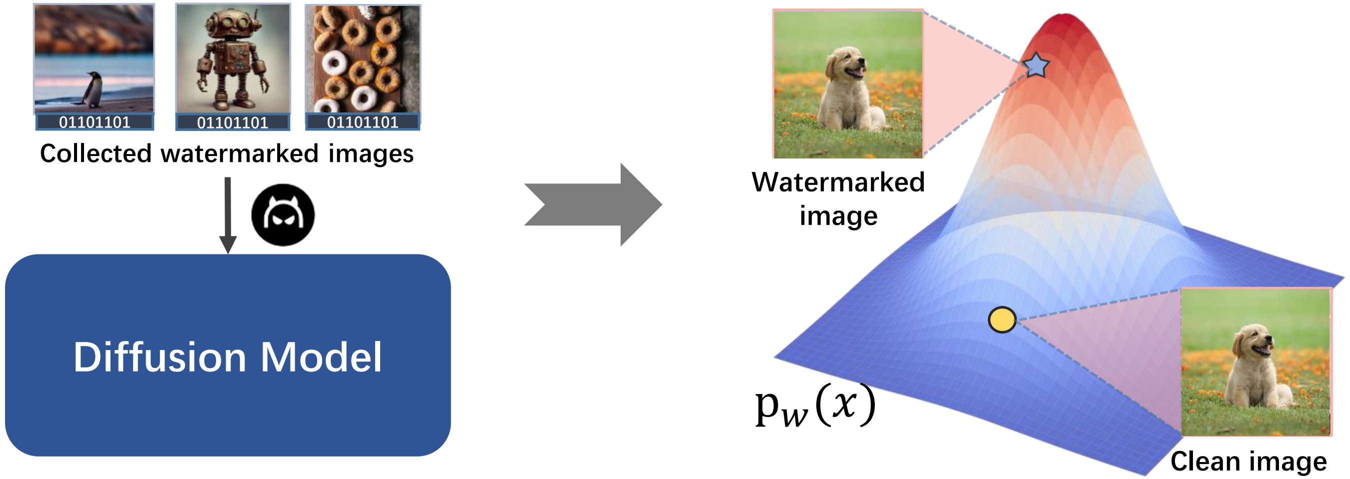
Method



Can we model the distribution of the watermarked images using a diffusion model?

Method

Generative models (e.g., GANs, Diffusion Models, Language Models) can **learn to generate watermarked content** by training on watermarked data[1−3] .



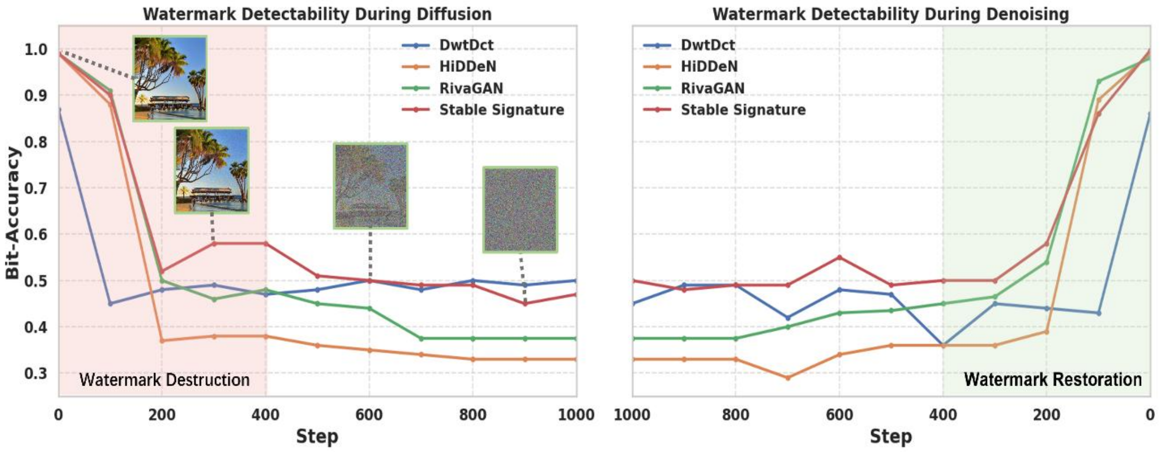
Watermark Estimation

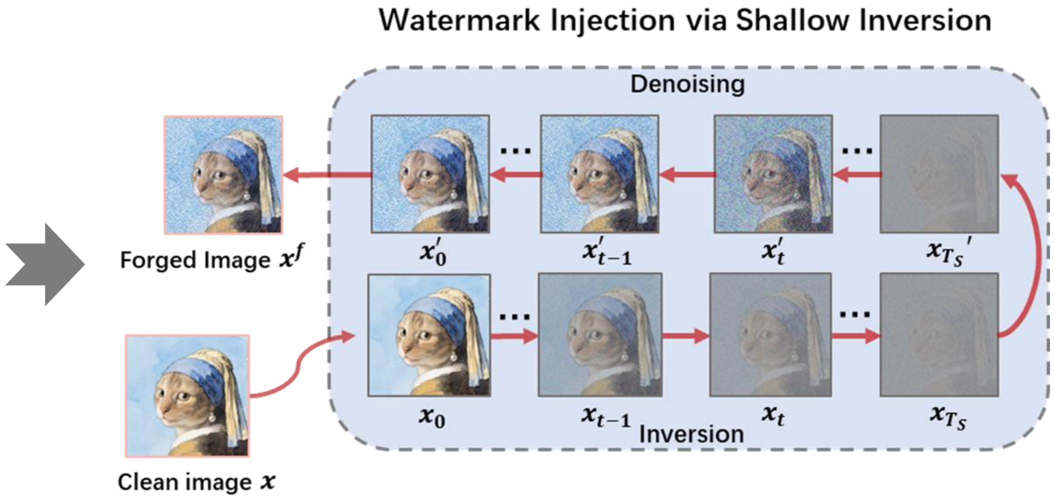
How to forge the watermark signal on a given clean image?

[1] On the Learnability of Watermarks for Language Models
[2] Artificial Fingerprinting for Generative Models: Rooting Deepfake Attribution in Training Data
[3] A Recipe for Watermarking Diffusion Models

Method

We find that watermark restoration occurs in the **shallow steps** of denoising, where semantic content is not destroyed.

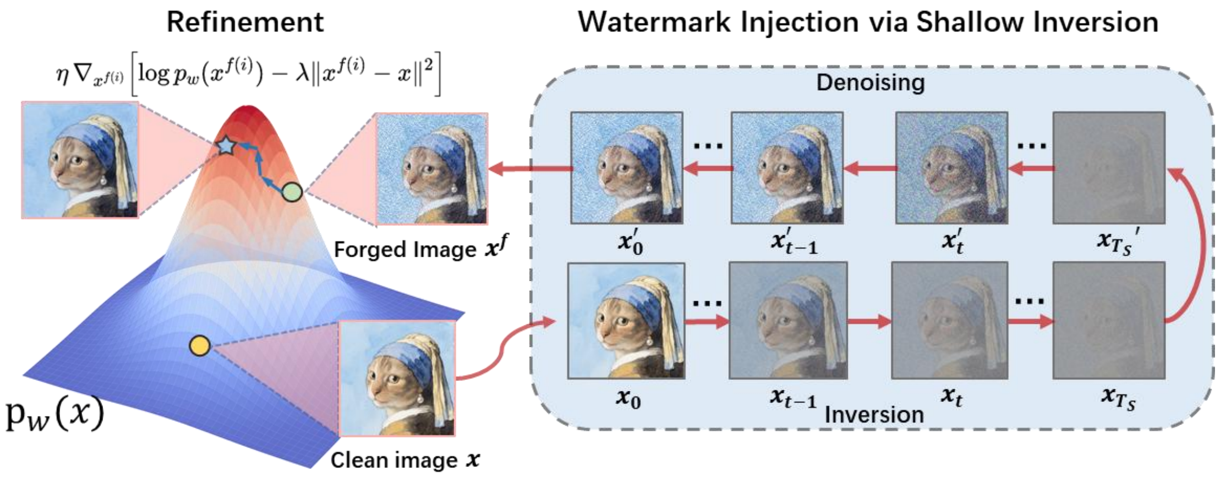




We perform a **shallow inversion** to map clean images to their **latent representations**, followed by a denoising process that injects the watermark signal utilizing the trained diffusion model.

Method

To further improve visual quality and forgery efficiency, we propose a **refinement procedure** that jointly optimizes image quality and alignment with the target watermark distribution.



$$x_{f(i+1)} = x_{f(i)} + \overbrace{\boxed{\eta \nabla_{x_{f(i)}} \log p_w(x_{f(i)})}}^{\text{Score}} - \overbrace{\boxed{\lambda \|x_{f(i)} - x\|_i^2}}^{\text{MSE}},$$

$$i \in \{0, 1, \ldots, L\}$$

For simplicity,

$$\nabla_{x^f} \log p_w(x^f) \approx \nabla_{x^f_{t_l}} \log p_w^{t_l}(x^f_{t_l}) \approx -\frac{1}{\sqrt{1-\alpha_{t_l}}} \epsilon_\theta(x^f_{t_l}, t_l).$$

The log-likelihood constrains the samples to lie in regions of high probability under $p_w(x)$, while the MSE term ensures that the refined image remains similar to the clean image x.

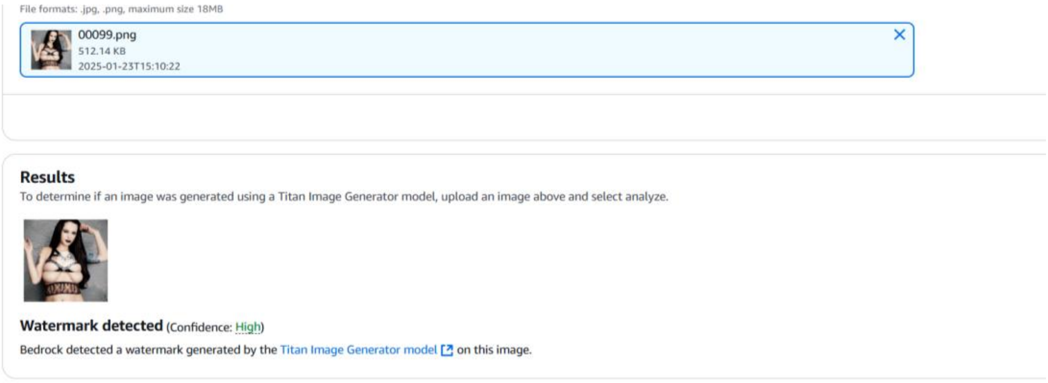
Experiments

Attack Performance

Attacks		Black Box Wang et al. [16]			No-Box Yang et al. [10]			No-Box Ours		
Watermark scheme	**Dataset**	**PSNR↑**	**Forged Bit-acc↑**	**FPR@10^{-6}↑**	**PSNR↑**	**Forged Bit-acc.↑**	**FPR@10^{-6}↑**	**PSNR↑**	**Forged Bit-acc.↑**	**FPR@10^{-6}↑**
DWT-DCT	MS-COCO	31.33	74.32%	57.20%	32.87	53.08%	0.50%	33.69	89.19%	60.20%
	CelebAHQ	32.19	81.29%	50.70%	32.90	53.68%	0.10%	35.29	89.46%	53.20%
	ImageNet	30.16	79.64%	55.10%	32.92	51.96%	0.20%	33.75	88.25%	55.80%
	Diffusiondb	31.87	78.22%	50.80%	32.90	51.59%	0.40%	33.84	85.17%	54.30%
HiddeN	MS-COCO	31.02	80.56%	77.60%	29.68	63.12%	0.00%	31.74	99.34%	95.90%
	CelebAHQ	31.57	82.28%	80.20%	29.79	61.52%	0.00%	33.12	98.08%	92.50%
	ImageNet	31.24	78.61%	83.90%	29.78	62.66%	0.00%	31.76	98.99%	94.30%
	Diffusiondb	30.74	79.99%	79.20%	29.68	63.36%	0.00%	31.46	98.83%	94.60%
RivaGAN	MS-COCO	32.94	93.26%	88.80%	29.12	50.80%	0.00%	34.07	95.74%	90.90%
	CelebAHQ	32.64	93.67%	93.80%	29.23	52.29%	0.00%	35.28	98.61%	96.00%
	ImageNet	33.11	90.94%	71.40%	29.22	50.92%	0.00%	33.87	93.83%	77.10%
	Diffusiondb	33.31	89.69%	80.60%	29.12	48.70%	0.00%	34.50	90.43%	84.80%
Stable Signature	MS-COCO	28.87	91.68%	88.90%	30.77	52.67%	0.00%	31.29	98.04%	94.60%
	CelebAHQ	32.33	79.90%	90.10%	30.51	51.73%	0.00%	30.54	96.04%	100.00%
	ImageNet	29.59	85.77%	85.90%	30.75	51.59%	0.00%	31.33	97.03%	98.60%
	Diffusiondb	31.11	89.24%	92.10%	30.65	52.69%	0.00%	31.59	96.24%	96.60%
Average		31.50	84.32%	76.64%	30.62	54.52%	0.08%	32.94	94.58%	83.71%

Table 1: Comparison of our WMCopier with two baselines on four open-source watermarking methods. The cells highlighted in indicate the highest values in each row for the corresponding metrics. Arrows indicate the desired direction of each metric (↑ for higher values being better).

Attack on open source watermark



Watermark Scheme	Attack	Yang et al. [10]		Ours	
	Dataset	**PSNR↑**	**SR↑/Con.↑**	**PSNR↑**	**SR↑/Con.↑**
Amazon WM	Diffusiondb	23.42	29.0%/2	**32.57**	**100.0%/2.94**
	MS-COCO	24.18	32.0%/2	**32.93**	**100.0%/2.97**
	CelebA-HQ	24.10	42.0%/2	**31.84**	**100.0%/2.98**
	ImageNet	23.95	28.0%/2	**32.88**	**99.0%/2.89**

Table 2: Performance comparison of baseline and WMCopier on Amazon Watermark.

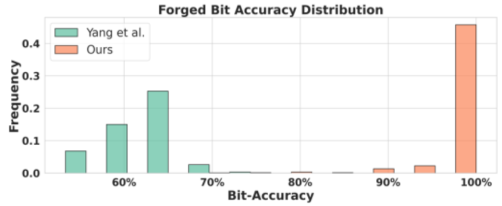


Figure 4: Comparison of forged bit accuracy distribution: Yang's method. vs. Ours.

Attack on close source watermark

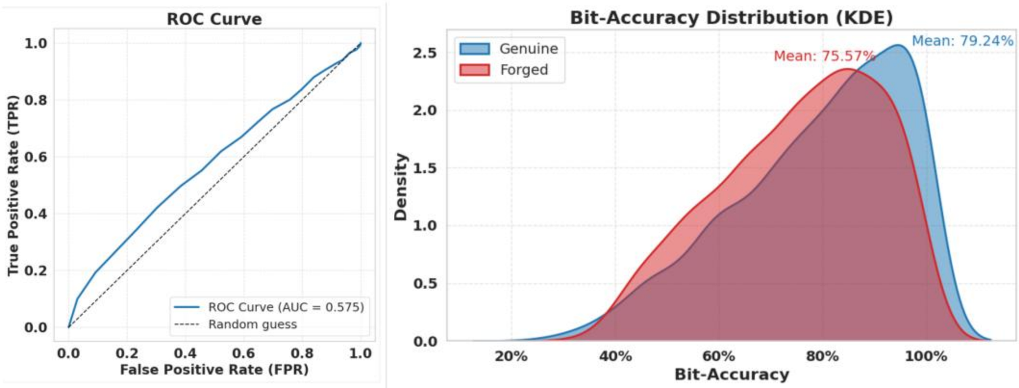
NEURAL INFORMATION PROCESSING SYSTEMS

Experiments

Potential Defenses

Watermark scheme	Distortion / Dataset	JPEG Genuine	JPEG Forged	Blur Genuine	Blur Forged	Gaussian Noise Genuine	Gaussian Noise Forged	Brightness Genuine	Brightness Forged
DWT-DCT	MS-COCO	56.44%	53.00%	59.84%	56.56%	67.86%	66.90%	54.66%	58.36%
	CelebAHQ	55.42%	53.14%	63.12%	58.26%	64.84%	66.49%	53.89%	57.73%
	ImageNet	56.08%	52.31%	59.37%	54.39%	68.27%	67.60%	54.08%	57.37%
	Diffusiondb	58.16%	53.23%	62.12%	55.74%	66.90%	64.43%	54.73%	56.83%
HiddeN	MS-COCO	58.68%	58.06%	78.50%	71.95%	54.13%	49.55%	82.40%	78.99%
	CelebAHQ	57.05%	55.07%	79.83%	69.07%	48.94%	46.02%	83.63%	73.21%
	ImageNet	58.86%	57.83%	78.20%	71.34%	54.10%	49.57%	80.95%	77.40%
	Diffusiondb	58.57%	57.61%	79.69%	72.89%	54.41%	50.19%	81.53%	77.66%
RivaGAN	MS-COCO	99.44%	93.32%	99.60%	94.99%	85.71%	75.00%	84.51%	78.81%
	CelebAHQ	99.92%	97.22%	99.97%	98.23%	85.93%	74.83%	84.60%	79.53%
	ImageNet	98.95%	92.00%	99.28%	93.89%	84.95%	74.74%	82.77%	77.25%
	Diffusiondb	96.56%	84.85%	97.27%	86.96%	77.33%	66.27%	79.14%	71.65%
StableSignature	MS-COCO	93.99%	89.48%	86.91%	68.34%	73.78%	67.14%	92.30%	88.63%
	CelebAHQ		86.73%		65.42%		65.33%		86.86%
	ImageNet		87.73%		64.88%		61.79%		91.41%
	Diffusiondb		85.69%		65.45%		61.60%		87.45%

Table 3: Bit Accuracy of the genuine watermark and the forged watermark under various image distortions. The distortion parameters are: Gaussian Noise ($\sigma = 0.05$), JPEG (quality=90), Blur (radius=1), and Brightness (factor=6). Cells with background indicate a degradation gap between 10% and 20%, and cells with background indicate a degradation gap greater than 20%.



Discrimination of Forged Watermarks by Robustness Gap

Although forged samples demonstrate poorer robustness against distortion, **it is still difficult to distinguish forged from genuine watermarked images using this**.

Experiments

Potential Defenses

We propose a **multi-message strategy** as a simple yet effective countermeasure. Instead of embedding a fixed watermark message, the system randomly selects one from a predefined message pool $m_1, m_2, m_3, \ldots, m_K$ for each image.

Table 4: Performance comparison across different K values.

Dataset	K=10			K=50			K=100		
	PSNR↑	Forged Bit-acc.↑	FPR@10^{-6}↑	PSNR↑	Forged Bit-acc.↑	FPR@10^{-6}↑	PSNR↑	Forged Bit-acc.↑	FPR@10^{-6}↑
MS-COCO	34.73	81.63%	34.00%	34.62	69.78%	0.00%	34.86	71.56%	0.00%
CelebAHQ	36.13	83.41%	44.00%	35.89	71.00%	0.00%	35.87	72.91%	0.00%
ImageNet	34.55	79.25%	25.00%	34.35	70.09%	0.00%	34.58	71.44%	0.00%
Diffusiondb	35.14	76.28%	17.00%	35.10	70.66%	0.00%	35.40	72.28%	0.00%

Table 5: Performance comparison across datasets with a larger size of D_{aux} for $K = 100$.

Dataset	5000			20000			50000		
	PSNR↑	Forged Bit-acc.↑	FPR@10^{-6}↑	PSNR↑	Forged Bit-acc.↑	FPR@10^{-6}↑	PSNR↑	Forged Bit-acc.↑	FPR@10^{-6}↑
MS-COCO	34.86	71.56%	0.00%	34.78	71.91%	0.00%	30.77	71.94%	0.00%
CelebA-HQ	35.87	72.91%	0.00%	34.15	72.97%	1.00%	27.99	72.72%	1.00%
ImageNet	34.58	71.44%	0.00%	34.57	72.56%	0.00%	30.47	72.19%	0.00%
DiffusionDB	35.40	72.28%	0.00%	34.99	72.34%	0.00%	31.15	72.06%	0.00%

Future work

The exploration of alternative architectures and training schemes to future work.

Create a UNet2DModel

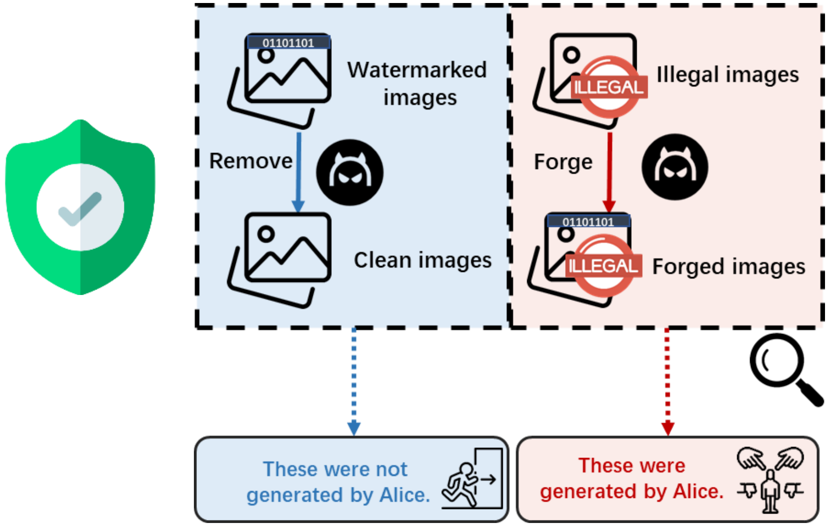
Pretrained models in 🧨 Diffusers are easily created from their model class with the parameters you want. For example, to create a UNet2DModel:

```
>>> from diffusers import UNet2DModel

>>> model = UNet2DModel(
...     sample_size=config.image_size,  # the target image resolution
...     in_channels=3,  # the number of input channels, 3 for RGB images
...     out_channels=3,  # the number of output channels
...     layers_per_block=2,  # how many ResNet layers to use per UNet block
...     block_out_channels=(128, 128, 256, 256, 512, 512),  # the number of output channels for each UNe
...     down_block_types=(
...         "DownBlock2D",  # a regular ResNet downsampling block
...         "DownBlock2D",
...         "DownBlock2D",
...         "DownBlock2D",
...         "AttnDownBlock2D",  # a ResNet downsampling block with spatial self-attention
...         "DownBlock2D",
...     ),
...     up_block_types=(
...         "UpBlock2D",  # a regular ResNet upsampling block
...         "AttnUpBlock2D",  # a ResNet upsampling block with spatial self-attention
...         "UpBlock2D",
...         "UpBlock2D",
...         "UpBlock2D",
...         "UpBlock2D",
...     ),
... )
```

Future watermark schemes and their deployment should consider robustness against both removal attacks and forgery attacks.

Watermark Schemes



[1] https://huggingface.co/docs/diffusers/en/tutorials/basic_training