HIGH PAYLOAD ROBUST WATERMARKING OF GENERATIVE MODELS WITH MULTIPLE TRIGGERS AND CHANNEL CODING

Jianwei Fei
Faculty of Science and Technology
University of Macau
Macau, China
fjw826244895@163.com

Benedetta Tondi & Mauro Barni
Department of Information Engineering and Mathematics
University of Siena
Siena, Italy
{benedetta.tondi, mauro.barni}@unisi.it

ABSTRACT

We present a robust and high-payload black-box multi-bit watermarking scheme for generative models. In order to embed a high payload message while retaining robustness against modifications of the watermarked network, we rely on the use of channel codes with strong error correction capacity (polar codes). This, in turn, increases the number of (coded) bits to be embedded within the network, thus challenging the embedding capabilities of the watermarking scheme. For this reason, we split the watermark bits into several chunks, each of which is associated with a different watermark triggering input. Through extensive experiments on the StyleGAN family of generative models, we show that the proposed method has excellent payload and robustness performance, allowing great flexibility to trade off between payload and robustness. Noticeably, our method demonstrates the capability of embedding over 100,000 coded bits for a net payload of up to 8192 bits while maintaining high image quality, with a PSNR exceeding 37 dB. Experiments demonstrate that the proposed high-payload strategy effectively improves the robustness of messages via high-performance channel codes, against white-box model attacks such as fine-tuning and pruning. Codes at: https://github.com/jumpycat/CCMark

1 INTRODUCTION

In recent years, numerous watermarking methods have been proposed to protect the Intellectual Property Rights (IPR) of generative models, like Generative Adversarial Networks (GANs) and Diffusion Models (DM). Watermarking methods can be categorized into white-box, black-box, and box-free methods Li et al. (2021); Barni et al. (2021); Xue et al. (2021); Hua & Teoh (2023). White-box methods require full access to the network weights hence their use is limited to specific application scenarios wherein such access can be granted Uchida et al. (2017). Black-box methods embed the watermark into the input-output behavior of the model. With such methods the watermark is retrieved by feeding the network with specific watermark triggering inputs (a.k.a. triggers), enabling ownership verification without accessing the internal parameters of the network Adi et al. (2018); Namba & Sakuma (2019); Szyller et al. (2021). Black-box watermarking methods are typically zero-bit schemes, making it possible only to verify the presence within the network of a given watermark. Box-free methods, on the other hand, embed the watermark into the generated content (e.g., images), without requiring interaction with the model. They allow the retrieval of the watermark from every content generated by the network Fei et al. (2022; 2024); Lin et al. (2024). Both black-box and box-free methods may be successfully used for IPR protection, however, they differ in capacity, robustness, and accessibility.

In this paper, we focus on black-box watermarking of image generative models. Despite intense research activity, existing methods belonging to this category have some unresolved limitations: (1) Payload limitation. Black-box methods are typically zero-bit watermarking, therefore they allow only to verify whether a known watermark is present in the queried model. This limits the flexibility and usability of black-box methods in practical applications; (2) Robustness limitation. Several works have proposed methods to enhance the robustness of the watermark against attacks Fei et al. (2023; 2024); Wang et al. (2024). However, the design of a robust, multi-bit, black-box watermarking scheme with a large payload is a challenging problem that has not been addressed properly so far. The possibility to trade off payload and robustness is also missing in the current state-of-the-art methods as these methods do not face these two requirements by adopting a unified perspective and therefore lack scalability. Note that typical applications of black-box watermarking do not require that the watermark is robust to image-level modifications since the images containing the watermark are generated on the fly when the watermark is extracted and require the knowledge of the triggers, hence, requiring that the watermark be robust only against model-level modifications. In this paper, we propose a new approach to design a robust and high-payload generative model watermarking method addressing the above limitations.

Our starting point is to treat multi-bit watermarking as a communication problem and rely on channel coding to improve its robustness against modifications of the network weights playing the role of the message carrier [1] However, channel coding requires the introduction of redundancy bits, which increases the number of (coded) bits that must be embedded within the network. This, in turn, has a negative impact either on the robustness of the (coded) watermark bits or the visibility of the watermark. To cope with this problem, we use a multiple triggers strategy, with the images corresponding to different triggers carrying a portion of the watermark bits. In this way, it is possible to use low-rate codes with a net positive effect on robustness. Of course, the interdependency between the code rate, the payload, and the number of triggers, and their impact on the robustness and visibility of the watermark, must be carefully studied, which is one of the goals of this work.

We evaluated the effectiveness, robustness, and achievable payload of our method on the StyleGAN family of generative models, including StyleGAN2 Karras et al. (2020) and StyleGAN3 Karras et al. (2021). We chose the StyleGAN family because it is the most popular architecture among GANs and is widely applied for various tasks such as image generation, editing, or processing. Even if diffusion models have recently shown more advanced performance in image generation, StyleGAN architectures are still very popular and sometimes are even used to improve the performance of DM models Trinh & Hamagami (2024).

With the above ideas in mind, the contributions of this work can be summarized as follows:

- We propose a scalable multi-bit black-box watermarking scheme, resorting to the use of multiple triggers, capable of embedding a large number of bits within the network.
- We propose the use of channel coding to improve the robustness and get a scalable method that allows a trade-off between payload and robustness. The joint use of channel coding and multiple triggers ensures a high payload and good robustness with nearly perfect accuracy, even in the presence of model-level attacks.

2 METHOD

The notation used throughout the paper is summarized in Table 1. The overall watermarking pipeline is illustrated in Figure 1. Our method aims to fine-tune a target pre-trained watermark-free image generator $\mathcal{G}_t$ embedding within it the desired watermark. As a preliminary step, we train an encoder/decoder pair to learn how to embed a generic string of bits into a generative model and retrieve it from the generated images. At watermarking time, the d_m-bit-long message $\boldsymbol{m}$ that we want to hide within the model is coded to produce a sequence of coded bits $\boldsymbol{w}$ of length $d_w > d_m$. The watermarked generator $\mathcal{G}_s$ is trained in a teacher-student modality by using $\mathcal{G}_t$ as a teacher and by

[1]Even if the watermark is read from the images produced when the model is fed with the triggers, ultimately the watermark is embedded in the network weights.

Table 1: Definitions and notation

Notation	Definition	Notation	Definition
$\boldsymbol{m}$	Message bit string	`WMEnc`	Channel encoding algorithm
d_m	Number of bits in $\boldsymbol{m}$	`WMDec`	Channel decoding algorithm
$\boldsymbol{w}$	Watermark bit string	$\mathcal{G}_t$	Pre-trained generator (teacher)
d_w	Number of bits in $\boldsymbol{w}$	$\mathcal{G}_s$	Watermarked generator (student)
M	Number of watermark blocks	$\mathcal{E}$	Watermark encoder
$\boldsymbol{w}^{(i)}$	Block i of watermark bit string $\boldsymbol{w}$	$\mathcal{D}$	Watermark decoder
$\boldsymbol{t}$	Trigger input	$\boldsymbol{z}$	Normal noise inputs
L	Number of bits per watermark block	$\mathcal{N}(0,1)$	Standard Normal Distribution

requiring that the application of the watermark decoder $\mathcal{D}$ to the images produced in correspondence to the trigger inputs coincides with $\boldsymbol{w}$. The watermark bits are split between different triggers and permuted to avoid the presence of error bursts that would not be corrected by the channel code.

In the following, the details of each of the above steps are given.

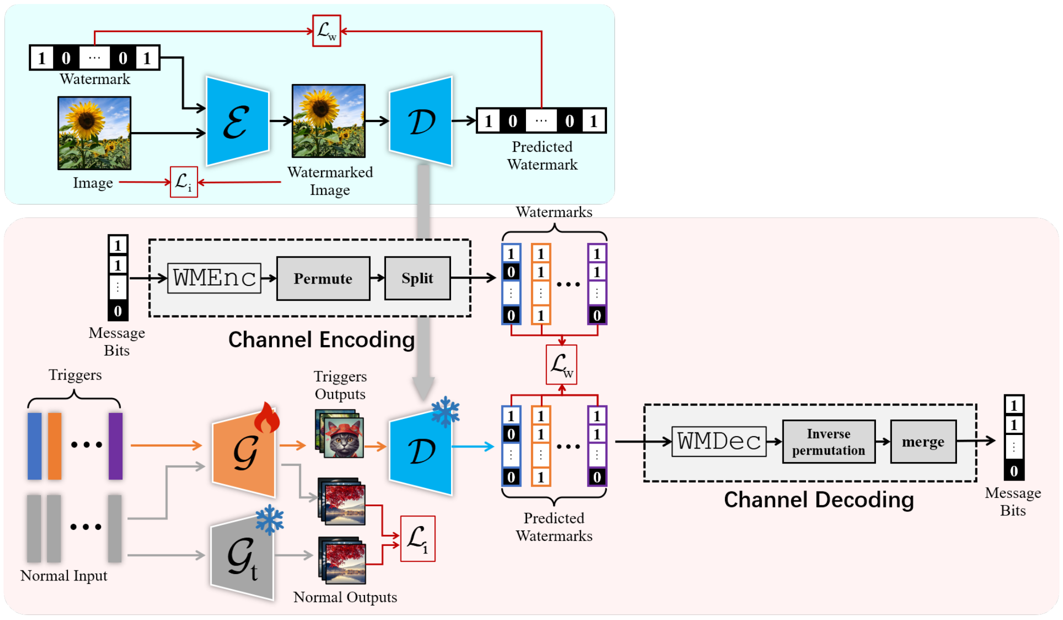

Figure 1: Overall pipeline of the proposed method.

2.1 Channel Coding

Current model watermarking methods either use model weights as direct carriers (white-box methods) or consider model weights as indirect carriers, where the watermark is embedded into model behaviors or outputs, which serve as the direct carriers (black-box and box-free methods). Such a strategy causes a decreased watermark extraction accuracy when the model is subject to modifications like quantization, compression, or fine-tuning during its lifecycle. In this work, we exploit channel coding to improve the robustness of the message bits against model modifications.

The sequence $\boldsymbol{m}$ is encoded by the channel encoder `WMEnc`, to generate the coded watermark $\boldsymbol{w} \in \{0,1\}^{d_w}$, thereby d_m/d_w is the rate of the code. Subsequently, we apply a random permutation to $\boldsymbol{w}$, to spread possible watermark extraction errors evenly across the sequence (for the sake of simplicity we still indicate with $\boldsymbol{w}$ the permuted sequence). $\boldsymbol{w}$ is then divided into M blocks each consisting of $L = d_w/M$ bits. In this work, each watermark blocks $\boldsymbol{w}^{(i)}$ is embedded within a single image generated by using a specific trigger. Following previous research in box-free watermarking Yu et al. (2021b), we selected a payload of $L = 128$ bits per trigger (hence the number of trigger images is

$M = d_w/128$). For channel coding, we used polar codes Arikan (2009). By associating different groups of bits to different triggers, we can embed a large number of bits within the model hence allowing for a large number of redundancy bits (low code rates), ensuring a reliable decoding of the message bits in $\boldsymbol{m}$.

2.2 Embedding procedure

As a preliminary step, we train an image watermarking scheme consisting of an encoder $\mathcal{E} : (\boldsymbol{x}, \boldsymbol{w}) \rightarrow \boldsymbol{x}_w$, embedding a watermark $\boldsymbol{w}$ of L bits into an image $\boldsymbol{x}$, producing a watermarked image $\boldsymbol{x}_w$, and a decoder $\mathcal{D} : \boldsymbol{x}_w \rightarrow \boldsymbol{w}$ that extract $\boldsymbol{w}$ from $\boldsymbol{x}_w$. Specifically, we adopted the architecture of StegaStamp Tancik et al. (2020) and its loss functions. The encoder-decoder pair is trained by a proper combination of two loss terms, an image loss $\mathcal{L}_i$ and a watermarking loss $\mathcal{L}_w$, implemented by Mean Squared Error (MSE) and Binary Cross Entropy (BCE), respectively. We stress that this training step is carried out only once to get $\mathcal{D}$ and does not need to be repeated when embedding different messages or watermarking different generators.

Given a pre-trained clean image generator model $\mathcal{G}_t$, our goal is to fine-tune $\mathcal{G}_t$ to create the watermarked generator $\mathcal{G}_s$. To do this, we first establish a set of trigger inputs $\{\boldsymbol{t}^{(i)} \sim \mathcal{N}(0,1)\}_{i=1}^{M}$, and associate each $\boldsymbol{t}^{(i)}$ with a watermark block, ensuring that

$$\mathcal{D}(\mathcal{G}_s(\boldsymbol{t}^{(i)})) = \boldsymbol{w}^{(i)}. \tag{1}$$

To achieve this, the generator $\mathcal{G}_s$ is fine-tuned by using two loss terms: an image loss $\mathcal{L}_i$ and a watermark loss $\mathcal{L}_w$. The former ensures consistency in the outputs of $\mathcal{G}_t$ and $\mathcal{G}_s$, while the latter guarantees the watermark extraction accuracy. For the first objective, we used the Learned Perceptual Image Patch Similarity (LPIPS Zhang et al. (2018)) loss between $\mathcal{G}_t$ and $\mathcal{G}_s$: $\mathcal{L}_i = \text{LPIPS}(\mathcal{G}_t, \mathcal{G}_s)$, to ensure that $\mathcal{G}_s$ performs as well as $\mathcal{G}_t$, in terms of generation capabilities.

For the second objective, we used the frozen pre-trained decoder $\mathcal{D}$ to supervise the embedding (see Fei et al. (2022; 2024); Fernandez et al. (2023)). Since we aim to embed the watermarks in a black-box manner, only the outputs corresponding to the trigger inputs are required to carry the watermarks. Moreover, for images generated with different triggers, we use different bits. Hence we define

$$\mathcal{L}_w = \frac{1}{M \cdot L} \sum_{i=1}^{M} \sum_{j=1}^{L} \text{BCE}(\mathcal{D}(\mathcal{G}_s(\boldsymbol{t}^{(i)}))_j, \boldsymbol{w}_j^{(i)}). \tag{2}$$

Then the final loss is:

$$\mathcal{L}_{tot} = \lambda_1 \mathcal{L}_i + \lambda_2 \mathcal{L}_m \tag{3}$$

Notably, the parameters of $\mathcal{G}_s$ are initialized from $\mathcal{G}_t$ and then fine-tuned using both loss functions.

3 Implementation

We used the advanced style-based GAN, StyleGAN2 Karras et al. (2020) and StyleGAN3 Karras et al. (2021) as target models. With regard to the training datasets, we used FFHQ (Flickr-Faces-High-Quality) Karras et al. (2019) and LHQ (Landscapes-High-Quality) Skorokhodov et al. (2021), all images were resized to 256×256 pixels. For the watermarking network, we used the network proposed in Stegastamp Tancik et al. (2020) and trained it on FFHQ.

We fine-tuned $\mathcal{G}_s$ for 20k steps (batches) for watermark embedding. λ_1 and λ_2 were set to 1.0 and 0.1. We used Adm optimizer with a learning rate $3e^{-4}$ and batch size 32. The polar code implementation is based on the Sionna library developed by Nvidia Hoydis et al. (2022). For encoding, the codeword length is set to 1024, and the information bits vary according to the code rate. For the decoder, we used the Successive Cancellation List (SCL). and set the list size to 1024.

For the message and watermark, we first generated texts of different lengths (letters) manually and converted them into message bits using UTF-8 encoding. These message bits are then encoded into the watermark bits by `WMEnc`. As to the trigger inputs, we randomly drawn them from an i.i.d. normal distribution, $\boldsymbol{t}^{(i)} \in \mathcal{N}(0,1), i = 1, ..., M$, which is the same procedure used for normal inputs. That is to say, we did not use out-of-distribution triggers and only modified a very small portion of the samples in the training distribution, which minimizes the impact on the performances of the image generator.

3.1 Metrics

The two main metrics we used to evaluate the effectiveness of our scheme are fidelity and accuracy Fidelity refers to the quality of the images generated by the watermarked model $\mathcal{G}_s$. We evaluated it by using the Peak Signal-to-Noise-Ratio (PSNR) between the images generated by $\mathcal{G}_s$ and $\mathcal{G}_t$. In the section, we report the mean value of PSNR computed on 3,200 images.

We regard to accuracy, we computed it at both the watermark bits level and the message level. The former is defined as the bit-wise matching accuracy between the extracted watermark $\mathcal{D}(\mathcal{G}_s(\boldsymbol{t}^{(i)})))$ and the coded watermark $\boldsymbol{w}^{(i)}$: $p_w = \frac{1}{M \cdot L} \sum_{i=1}^{M} \sum_{j=1}^{L} \mathbb{1}\left(\mathcal{D}(\mathcal{G}_s(\boldsymbol{t}^{(i)}))_j = \boldsymbol{w}_j^{(i)}\right)$, where $\mathbb{1}(\cdot)$ is the indicator function, which returns 1 if the condition is true, and 0 if it is false. In the same way, we let p_m denote the Bit Acc of the decoded message. For robustness evaluation, we used p_w and p_m in the presence of model modification attacks. We used a fixed random seed and conducted a single experiment for each configuration (capacity and code rate). The reported accuracy corresponds to the results over the full sequence of watermark and message bits.

We conducted comparative studies between the proposed method, which leverages polar codes, and a baseline method based on simple repetition coding. For repetition codes, we first repeated the bits in $\boldsymbol{m}$ d_w/d_m times and randomly shuffled them before embedding them bny using the algorithm described proposed in Sec. 2.2. In the following, we denote repetition coding by **Rep** and polar codes by **Polar**.

4 Experimental Results

4.1 Fidelity analysis

The fidelity, measured in terms of PSNR, evaluates the impact of the watermark on the visual quality of the images generated by the watermarked model. As shown in Table 2 and Table 3, the results of our experiments indicate that the embedding process maintains high fidelity when a small to medium payload is used. Specifically, PSNR values remain above 48 dB for message sizes of 128, 256, and 512 bits regardless of the encoding method (Rep or Polar) or the code rate. As the payload size increases, a gradual decline of the PSNR is observed, indicating a trade-off between embedding payload and fidelity. For a message length of 512 bits and a code rate of 1/16, we observe a drop of the PSNR, which, however, remains around 45dBs. For larger payloads, such as 4096 and 8192 message bits, the PSNR decreases significantly expecially when the code rate decreases. For instance, when the payload reaches 8192 bits with a code rate of 1/32 (262,144 watermark bits), the PSNR drops to approximately 37 dB. The overall results suggest that the proposed method can embed a substantial amount of information while maintaining high fidelity.

4.2 Accuracy Analysis: Small to Medium Payload

In this section, we evaluate the accuracy of our watermarking scheme. Our method allows a very flexible setup, which includes several key hyper-parameters including, the number of message bits (d_m), the number of coded bits (d_w), the number of triggers M, and the code rate (d_w/d_m). With a fixed number of message bits, increasing the number of coded bits implies a lower code rate, resulting in a better error correction capability. However, increasing the number of coded bits tends to render the watermark harder to embed and less robust. Thus, for a given number of message bits, the code rate is a key factor in determining the performance of the watermark.

We begin by analyzing scenarios with small to medium payloads. In Table 2, we report the performance of our method on StyleGAN2 and StyleGAN3 with a low/medium payload, comparing variants with repetition codes and polar codes. The **Msg bits** column represents the number of bits of the original message (d_m), while the watermark Bits indicate the number of bits in the channel-coded watermark. We also report the code rate. We compare the performance of two models across varying payload and code rates, reporting both watermark accuracy and message accuracy.

By analyzing the table, we observe that: i) Both Rep and polar codes achieve perfect watermark accuracy ($p_w = 1.00$) and message decoding accuracy ($p_m = 1.00$) across all tested payload sizes, models, and code rates. This indicates that the watermark can be reliably detected regardless of the encoding strategy. ii) Both p_w and p_m remain at 1.00 across all payload sizes. This indicates that for small to medium payloads, increasing the payload does not compromise the effectiveness of the watermark or the message. While in this case,e the positive effect of channel coding, its utility will become evident in the presence of model modifications. iii) Increasing the payload size slightly reduces fidelity but does not affect watermark or message accuracy.

Table 2: Small to medium payload results for StyleGAN2 and StyleGAN3.

Model	Msg bits	Watermark Bits = Msg Bits / rate												
			rate=1/2			rate=1/4			rate=1/8			rate=1/16		
		RE	PSNR	p_w	p_m	PSNR	p_w	p_m	PSNR	p_w	p_m	PSNR	p_w	p_m
StyleGAN2	128	Rep	50.27	1.00	1.00	49.80	1.00	1.00	48.74	1.00	1.00	47.67	1.00	1.00
		Polar	50.29	1.00	1.00	49.91	1.00	1.00	49.15	1.00	1.00	48.02	1.00	1.00
	256	Rep	49.68	1.00	1.00	49.23	1.00	1.00	47.82	1.00	1.00	46.59	1.00	1.00
		Polar	49.80	1.00	1.00	49.05	1.00	1.00	47.63	1.00	1.00	46.63	1.00	1.00
	512	Rep	48.90	1.00	1.00	47.67	1.00	1.00	46.92	1.00	1.00	44.65	1.00	1.00
		Polar	48.88	1.00	1.00	47.76	1.00	1.00	46.81	1.00	1.00	44.78	1.00	1.00
StyleGAN3	128	Rep	50.82	1.00	1.00	49.90	1.00	1.00	48.36	1.00	1.00	48.27	1.00	1.00
		Polar	50.77	1.00	1.00	50.79	1.00	1.00	49.56	1.00	1.00	47.60	1.00	1.00
	256	Rep	50.93	1.00	1.00	49.45	1.00	1.00	48.30	1.00	1.00	46.49	1.00	1.00
		Polar	50.86	1.00	1.00	49.37	1.00	1.00	47.90	1.00	1.00	46.54	1.00	1.00
	512	Rep	48.64	1.00	1.00	48.37	1.00	1.00	46.47	1.00	1.00	44.55	1.00	1.00
		Polar	49.44	1.00	1.00	48.02	1.00	1.00	46.36	1.00	1.00	44.57	1.00	1.00

4.3 Accuracy Analysis: Large Payload

We further evaluated our method with increased message payload and lower code rates. Specifically, for these tests, we used d_m equal to 4096 and 8192, with code rates equal to 1/2, 1/4, 1/8, 1/16, and 1/32.

The results we got are shown in Table 3, from which we can observe that: i) Increasing the number of watermark bits further degrades fidelity, with PSNR dropping to around 37dB when the model carries $8192 \times 32 = 262,144$ bits. ii) When the model carries $8192 \times 16 = 131,072$ bits or fewer, the watermark accuracy p_w reaches 0.99, with p_m always equal to 1.00. These results indicate that even without channel coding, our method can embed 131,072 bits under conditions where the PSNR exceeds 37db. iii) For larger payloads (8192 message bits), p_w begins to degrade more noticeably at lower rates. For instance, at rate=1/32, p_w drops to 0.90 (Rep) and 0.89 (Polar) for StyleGAN2 and StyleGAN3. iv) polar codes consistently outperform Rep codes at lower code rates and larger payloads. For example, at rate=1/32 with 8192 message bits, p_m for polar codes remains at 1.00 for StyleGAN2 and only drops slightly to 0.99 for StyleGAN3, while Rep codes drop to 0.90 for both StyleGAN2 and StyleGAN3.

Table 3: Large payload results for StyleGAN2 and StyleGAN3.

Model	Msg Bits	Type	Watermark Bits = Msg Bits / rate														
			rate=1/2			rate=1/4			rate=1/8			rate=1/16			rate=1/32		
			PSNR	p_w	p_m	PSNR	p_w	p_m	PSNR	p_w	p_m	PSNR	p_w	p_m	PSNR	p_w	p_m
StyleGAN2	4096	Rep	44.80	1.00	1.00	42.90	1.00	1.00	41.11	0.99	1.00	39.15	0.99	1.00	37.36	0.99	0.99
		Polar	44.95	1.00	1.00	43.01	1.00	1.00	41.14	0.99	1.00	39.29	0.99	1.00	37.54	0.99	1.00
	8192	Rep	42.96	1.00	1.00	41.10	1.00	1.00	39.22	1.00	1.00	37.43	0.99	1.00	37.44	0.90	0.90
		Polar	43.09	1.00	1.00	41.23	0.99	0.99	39.36	0.99	0.99	37.54	0.99	1.00	37.74	0.89	1.00
StyleGAN3	4096	Rep	44.94	1.00	1.00	43.04	1.00	1.00	41.13	0.99	1.00	39.20	0.99	1.00	37.49	0.99	0.99
		Polar	44.95	1.00	1.00	43.01	1.00	1.00	41.14	0.99	1.00	39.29	0.99	1.00	37.54	0.99	1.00
	8192	Rep	43.11	1.00	1.00	41.19	0.99	1.00	39.35	0.99	0.99	37.49	0.99	1.00	37.79	0.89	0.90
		Polar	43.09	1.00	1.00	41.23	0.99	0.99	39.36	0.99	0.99	37.54	0.99	1.00	37.74	0.89	1.00

4.4 Robustness Evaluation

In this section, we evaluate the robustness of the proposed method against white-box model attacks, including fine-tuning and pruning attacks.

For fine-tuning, we used LPIPS only to fine-tune the watermarked $\mathcal{G}_s$ for 4,000 steps, to remove the watermark. The results for fine-tuning attacks are summarised in Table 4. We observe that:

i) The robustness of the watermark under attacks shows clear trade-offs between payload size, code rate, and encoding algorithms. Smaller payloads (4096 message bits) show significantly better performance in terms of message accuracy (p_m) compared to larger payloads (8192 message bits). For smaller payloads, p_m remains high across most conditions, even at lower redundancy. For example, for StyleGAN2 with rate=1/4, Polar encoding achieves perfect p_m, while Rep can only ensure $p_m = 0.71$. Similarly, for StyleGAN3, Polar coding consistently outperforms Rep, achieving perfect message accuracy under the same conditions. In contrast, for larger payloads, p_m drops more noticeably at lower redundancy levels. For instance, in StyleGAN2 with rate=1/16 and 8192 message bits, p_m for Polar coding is 0.84, slightly larger than Rep's 0.75. These results indicate that smaller payloads are inherently more robust to attacks, while larger payloads are more sensitive, requiring careful tuning of the code rate and encoding method.

ii) The code rate, which determines the redundancy of the watermark is critical for robustness. At larger code rates (e.g., 1/2 and 1/4), p_m is generally larger, as the watermark introduces less complexity and distortion. For example, for StyleGAN3 with rate=1/4 and 4096 message bits, polar codes achieve perfect p_m of 1.00, while for Rep p_m is only 0.79. However, as the code rate decreases (e.g., 1/16 1/32), p_m begins to degrade due to the increased redundancy and embedding complexity, particularly for longer watermarks. For instance, for StyleGAN2, with rate=1/32 and 8192 message bits, p_m drops to 0.43 for polar codes and 0.64 for Rep codes. While lower code rates generally provide better error correction capability, when the error rate exceeds the correction capability of the code the accuracy irremediably drops. These observations highlight the need to carefully balance redundancy and payload based on the application requirements.

iii) Polar codes outperform Rep codes in the great majority of the cases, especially at medium redundancy levels (1/8 and 1/16) and small payloads. For StyleGAN2 with rate=1/16 and 4096 message bits, Polar achieves a p_m of 0.99, compared to Rep's 0.83. This advantage stems from Polar Codes's superior error correction capabilities, which become particularly effective when redundancy is moderate. However, as the redundancy increases further (e.g., rate=1/32), the performance gap between Polar and Rep narrows, especially for larger payloads. In StyleGAN3 with rate=1/32 and 8192 message bits, p_m drops to 0.43 for Polar encoding, compared to 0.64 for Rep. This indicates that while Polar encoding is generally more robust, its advantage diminishes when the coded message becomes too long, since in this case the error rate on coded bits exceeds the correction capability of the code.

Table 4: Robustness of StyleGAN2 and StyleGAN3 watermarking against fine-tuning attacks (4k steps). Values of p_m exceeding 0.80 are highlighted in bold.

Model	Msg Bits	Type	Watermark Bits = Msg Bits / rate									
			rate=1/2		rate=1/4		rate=1/8		rate=1/16		rate=1/32	
			p_w	p_m	p_w	p_m	p_w	p_m	p_w	p_m	p_w	p_m
StyleGAN2	4096	Rep	0.83	0.77	0.78	0.71	0.74	**0.84**	0.69	**0.83**	0.65	**0.81**
		Polar	0.84	0.51	0.80	**1.00**	0.75	**0.97**	0.70	**0.99**	0.66	**0.97**
	8192	Rep	0.78	0.71	0.74	0.76	0.69	0.76	0.65	0.75	0.59	0.63
		Polar	0.79	0.50	0.75	0.53	0.70	0.68	0.66	**0.84**	0.61	0.45
StyleGAN3	4096	Rep	0.85	0.80	0.83	0.79	0.78	**0.83**	0.72	0.79	0.66	**0.83**
		Polar	0.86	0.54	0.84	**1.00**	0.79	**1.00**	0.73	**1.00**	0.66	0.45
	8192	Rep	0.82	0.71	0.79	0.75	0.71	0.80	0.65	0.75	0.59	0.64
		Polar	0.83	0.50	0.80	0.77	0.72	**0.94**	0.66	**0.84**	0.59	0.43

We also evaluated the robustness against pruning attacks. Specifically, for the 4096-payload scenario, we pruned a certain proportion of the weights with the smallest absolute values to zero and measured both watermark and message accuracy. As shown in Fig. 2, **w Acc** and **m Acc** denote the watermark accuracy and message accuracy respectively. The pruning rates vary from 0.02 to 0.1. We can observe that:

i) As the pruning rate increases, both p_w and p_m degrade, but the rate of degradation of p_m varies significantly between encoding methods. For Rep codes, p_m drops sharply as pruning exceeds 6-8%, particularly at higher redundancy levels (e.g., rate=1/16 and rate=1/32). In contrast, polar codes maintain near-perfect p_m across all pruning rates, especially for higher redundancy configurations. This indicates that polar codes are significantly more robust to pruning attacks, ensuring reliable message retrieval even when the model undergoes substantial pruning.

ii) Higher redundancy (lower code rates) generally improves robustness, but the effectiveness of redundancy varies according to the encoding method. For Rep code, redundancy helps initially, but its benefits diminish as pruning increases, with p_m dropping below 0.9 at pruning rates of 8-10% for rate = 1/16 and 1/32. However, for the polar code, p_m remains nearly perfect across all pruning rates, even at the highest redundancy levels. This confirms that, as expected, polar codes leverage redundancy more effectively than Rep code.

iii) Polar codes outperform Rep coding across all code rates and pruning levels. While both encoding methods exhibit a decline of p_w as pruning increases, polar codes maintain stable and near-perfect p_m, regardless of the pruning rate. This is particularly evident for rate=1/16 and rate=1/32.

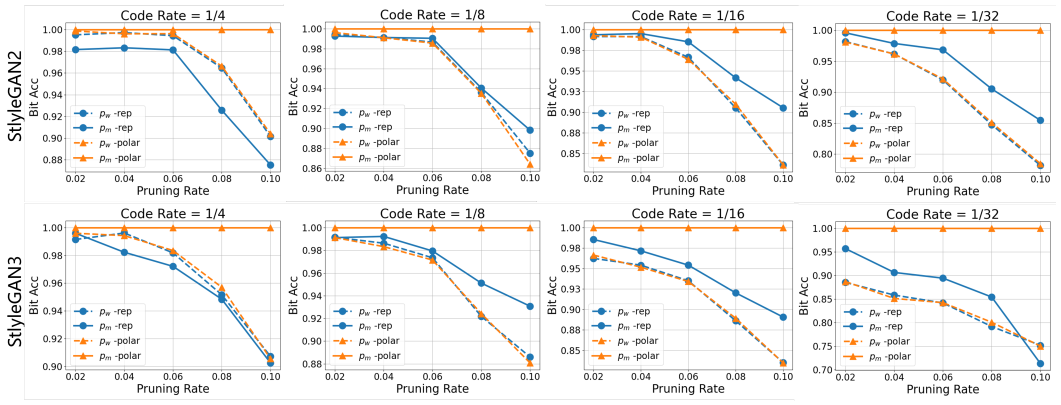

Figure 2: Results of pruning attacks with different strengths (the payload is set to 4096).

5 CONCLUSION

In this paper, we have presented a robust and high-capacity black-box multi-bit generative model watermarking method that addresses the limitations of existing techniques in terms of payload and robustness against white-box attacks (fine-tuning and pruning). We introduced the use of channel coding, specifically polar codes, to increase the robustness of the embedded watermarks.

Extensive experiments on the StyleGAN family of models demonstrate the effectiveness of our method, achieving a net payload capacity of 8192 bits while maintaining a PSNR exceeding 37 dB. The results show that our polar coding outperforms simple repetition coding in terms of both payload and robustness in most of the cases, providing a flexible solution for balancing these requirements. Our work not only advances generative model watermarking by introducing novel techniques inspired by communication technology but also provides a practical and scalable solution for protecting the intellectual property rights of generative models in real-world applications. Future work could explore the application of our method to other types of generative models, e.g., diffusion models, and further enhance its robustness against a broader range of attacks.

REFERENCES

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX security symposium (USENIX Security 18)*, pp. 1615–1631, 2018.

Erdal Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on information Theory*, 55(7): 3051–3073, 2009.

Mauro Barni, Fernando Pérez-González, and Benedetta Tondi. Dnn watermarking: Four challenges and a funeral. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pp. 189–196, 2021.

Jianwei Fei, Zhihua Xia, Benedetta Tondi, and Mauro Barni. Supervised gan watermarking for intellectual property protection. In *2022 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6. IEEE, 2022.

Jianwei Fei, Zhihua Xia, Benedetta Tondi, and Mauro Barni. Robust retraining-free gan fingerprinting via personalized normalization. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6. IEEE, 2023.

Jianwei Fei, Zhihua Xia, Benedetta Tondi, and Mauro Barni. Wide flat minimum watermarking for robust ownership verification of gans. *IEEE Transactions on Information Forensics and Security*, 2024.

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*, pp. 22466–22477, 2023.

Jakob Hoydis, Sebastian Cammerer, Fayçal Ait Aoudia, Avinash Vem, Nikolaus Binder, Guillermo Marcus, and Alexander Keller. Sionna: An open-source library for next-generation physical layer research. *arXiv preprint arXiv:2203.11854*, 2022.

Guang Hua and Andrew Beng Jin Teoh. Deep fidelity in dnn watermarking: A study of backdoor watermarking for classification models. *Pattern Recognition*, 144:109844, 2023.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in neural information processing systems*, 34:852–863, 2021.

Changhoon Kim, Kyle Min, Maitreya Patel, Sheng Cheng, and Yezhou Yang. Wouaf: Weight modulation for user attribution and fingerprinting in text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8974–8983, 2024.

Yue Li, Hongxia Wang, and Mauro Barni. A survey of deep neural network watermarking techniques. *Neurocomputing*, 461:171–193, 2021.

Dongdong Lin, Benedetta Tondi, Bin Li, and Mauro Barni. A cyclegan watermarking method for ownership verification. *IEEE Transactions on Dependable and Secure Computing*, 2024.

Dongdong Lin, Yue Li, Bin Li, and Jiwu Huang. Exploiting robust model watermarking against the model fine-tuning attack via flat minima aware optimizers. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pp. 228–240, 2019.

Ding Sheng Ong, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. Protecting intellectual property of generative adversarial networks from ambiguity attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3630–3639, 2021.

Tong Qiao, Yuyan Ma, Ning Zheng, Hanzhou Wu, Yanli Chen, Ming Xu, and Xiangyang Luo. A novel model watermarking for protecting generative adversarial network. *Computers & Security*, 127:103102, 2023.

Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 14144–14153, 2021.

Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 4417–4425, 2021.

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*, pp. 2117–2126, 2020.

Luan Thanh Trinh and Tomoki Hamagami. Latent denoising diffusion gan: Faster sampling, higher image quality. *IEEE Access*, 2024.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pp. 269–277, 2017.

Zilan Wang, Junfeng Guo, Jiacheng Zhu, Yiming Li, Heng Huang, Muhao Chen, and Zhengzhong Tu. Sleepermark: Towards robust watermark against fine-tuning text-to-image diffusion models. *arXiv preprint arXiv:2412.04852*, 2024.

Mingfu Xue, Jian Wang, and Weiqiang Liu. Dnn intellectual property protection: Taxonomy, attacks and evaluations. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pp. 455–460, 2021.

Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proceedings of the IEEE/CVF International conference on computer vision*, pp. 14448–14457, 2021a.

Ning Yu, Vladislav Skripniuk, Dingfan Chen, Larry S Davis, and Mario Fritz. Responsible disclosure of generative models using scalable fingerprinting. In *International Conference on Learning Representations*, 2021b.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

6 Appendix

6.1 Related Work

In this section, we review current works on generative model watermarking with a focus on black-box and box-free methods, as these are most relevant to the proposed work.

6.1.1 Black-box Generative Model Watermarking

Black-box generative watermarking methods embed ownership information inside the model, in such a way that the it can be recovered from model outputs to specific trigger inputs, enabling verification through queries. These kinds of methods assume that during the verification phase, the verifier can not access the internal parameters and his capability is limited to querying the model.

The watermark embedding process involves the optimization of an additional loss term that drives the predictions on trigger samples toward the target outputs, while guaranteeing the image generation capability on normal inputs. Current research focuses on the design of triggers and the embedding strategy. Ong et al. (2021) proposed the first black-box GAN watermarking by training the generator to produce a distinct watermark (e.g., a logo) when triggered by a modified input, such as a masked latent vector for DCGAN or a noise-embedded image for SRGAN and CycleGAN. It also introduced a reconstructive regularization term based on SSIM to ensure the fidelity without compromising the generation capability. Differently, Qiao et al. (2023) built the trigger set by combining a verification image with a unique binary watermark. The watermark is embedded by ensuring the generator produces the verification image when triggered by the watermark. For verification, the owner inputs the watermark, and the model's output is compared to the verification image using similarity metrics.

These methods are primarily zero-bit watermarking techniques, designed for ownership verification in GAN IPR protection. In contrast, our method is a multi-bit watermarking one that supports a capacity of over 8k bits, which enables the effective embedding of a large amount of information.

6.1.2 Box-free Generative Model Watermarking

Unlike black-box methods, in box-free methods methods the watermark information is embedded by the model in any generated image. The watermark can then be extracted from any output by a decoder. These methods avoid the need for trigger queries. However, since the watermark is carried by a single image, the capacity of these methods is often limited.

Yu et al. (2021a) proposed the first box-free, multi-bit GAN watermarking method by directly embedding a watermark into the GAN training data. The authors demonstrate that the resulting GAN inherently learns to produce watermarked outputs, allowing for detection and attribution through watermark matching. Subsequently, Fei et al. (2022) proposed a supervised embedding method by introducing a novel regularization term derived from a pre-trained watermark decoder and fine-tuning the target GAN to ensure that every generated image contains a desired watermark. Later studies mainly focused on improving the robustness of this embedding strategy Fei et al. (2024); Lin et al. (2025) and incorporating bidirectional supervision for image translation networks Lin et al. (2024). Similarly, Fernandez et al. (2023) fine-tuned the decoder within the latent diffusion model using a pre-trained watermark decoder, and applied a whitening method on the watermark logits to reduce bias across bits.

To improve embedding efficiency, several fingerprinting methods have been proposed to enable scalable generation of model instances. Yu et al. (2021b) applies binary fingerprints to modulate the convolutional weights of GANs and enforces the extraction of corresponding fingerprints from generated images, thereby allowing efficient creation of fingerprinted model copies. Similarly, Kim et al. (2024) modulated the decoder of diffusion models using fingerprints, which results in the scalable fingerprint embedding as well. Fei et al. (2023) introduced a GAN fingerprinting method by embedding binary fingerprints into the parameters of the proposed personalized normalization layers.

Compared to the above methods, our method significantly increases the watermark capacity, providing sufficient room for redundant information, thus enabling the adoption of channel coding, which leads to improved robustness under model-level attacks.

6.2 Threat Model

We define the threat model which consists of two roles: the owner and the attacker. The owner aims to embed watermarks into the target model without impairing its quality, whereas the attacker tries to remove these watermark from the model. The capabilities and knowledge of owner and attacker are described in the following.

Owner. We outline the actions and the knowledge available to the owner in different phases:

(1) (Embedding Phase) The owner, or the model developer, generates the message $\boldsymbol{m}$ to be embedded and encodes it into the watermark $\boldsymbol{w}$ using a private coding strategy `WMEnc`. Then the owner embeds $\boldsymbol{w}$ into the target model using watermarking strategy `Embed`, which involves the triggers $\boldsymbol{t}_i, i = 1, \ldots, M$, a watermarking decoder $\mathcal{D}$ and the optimization term $\mathcal{L}_w$, and finally generate the watermarked model G_s. The owner has direct control over all variables and model components and will only release G_s while keeping everything else private.

(2) (Verification Phase) Upon identifying a suspicious model G', the owner can use the trigger $\boldsymbol{t}$ to query the model to obtain predicted watermark $\boldsymbol{w}'$, and employ the corresponding decoding strategy `WMDec` to obtain the message $\boldsymbol{m}'$. The decoded message $\boldsymbol{m}'$ is then compared with the original embedded message $\boldsymbol{m}$ to determine ownership of G'. If the difference between $\boldsymbol{m}'$ and $\boldsymbol{m}$ is smaller than a predefined threshold τ, it indicates that G' contains the watermark embedded by the model owner, thereby serving as evidence that G' is an unauthorized copy of the original model. Conversely, it suggests that the watermark is absent or unrecognizable in G', and ownership cannot be established.

Attacker. The attacker has access to the architecture and weights, namely, white-box access of G_s, and tries to remove the watermark by various attacks `Remove`. The attacker does not have any specific knowledge on $\boldsymbol{m}$, $\boldsymbol{w}$, $\boldsymbol{t}$, $\mathcal{D}$ and `WMEnc`. We also make the assumption that the model owner provides an API of the clean model. Under this assumption, the attacker can query the API to obtain clean images, and then fine-tune their model accordingly to remove the watermark.

6.3 Additional Experiments

We further evaluated our method on a diffusion-based image generation model, Stable Diffusion v2.1 (SD2.1), under large payload settings. Specifically, we fine-tune the decoder while keeping other components in SD2.1 fixed. We randomly sample real images from the COCO dataset and encode them into latent representations via the encoder. These latents serve as triggers, and the decoder is optimized to reconstruct the latents and introduce the watermarks in reconstructed images. The fine-tuning process is similar to that of a GAN, with the main difference being that, in GANs, the input random noise, whereas in diffusion models, it is the latent representation of real images encoded by the encoder.

As shown in Table 5, the results are consistent with those observed for StyleGAN2/3. As the number of message bits increases and the code rate decreases, PSNR drops gradually, reaching around 36.5dB with 262,144 (8192×32) watermarked bits. Besides, the watermark accuracy p_w remains over 0.99, and message accuracy p_m stays at 1.00 for nearly all cases. At the lowest code rate (1/32), we can observe a slight drop in p_w (down to 0.94), but p_m remains constantly robust with our Polar codes strategy. These results further confirm the scalability and reliability of our method for different kinds of generative models, and Polar codes always offer better robustness at low code rates.

We also show some samples in Fig. 4, where images generated by watermark-free models are compared with those from watermarked models across different architectures. The watermarked models carry either 4096 or 8192 message bits at a 1/2 rate, we can observe the outputs remain visually indistinguishable from the original images, and the corresponding difference images reveal only minor pixel-level changes. This demonstrates the high imperceptibility of our watermarking method.

The robustness results for SD2.1 under fine-tuning attacks (Table 6) are also consistent with those observed for StyleGAN2 and StyleGAN3 (Table 4). Despite applying a stronger attack setting (10k fine-tuning steps vs. 4k), watermark message accuracy p_m remains high for small payloads (4096 bits), especially when using our Polar codes, achieving $p_m \geq 0.90$ across most code rates. Our Rep codes show lower robustness, though they still perform reasonably well at higher code rates. For

Table 5: Large payload results for Stable Diffusion v2.1.

Msg Bits	Type	Watermark Bits = Msg Bits / rate														
		rate=1/2			rate=1/4			rate=1/8			rate=1/16			rate=1/32		
		PSNR	p_w	p_m	PSNR	p_w	p_m	PSNR	p_w	p_m	PSNR	p_w	p_m	PSNR	p_w	p_m
4096	Rep	42.26	1.00	1.00	41.51	1.00	1.00	41.05	1.00	1.00	40.51	0.99	1.00	38.37	0.99	0.99
	Polar	42.51	1.00	1.00	41.24	1.00	1.00	40.97	1.00	1.00	40.15	1.00	1.00	39.61	0.99	1.00
8192	Rep	41.60	1.00	1.00	41.04	1.00	1.00	40.29	1.00	1.00	38.94	1.00	1.00	36.52	0.94	0.98
	Polar	41.51	1.00	1.00	40.85	0.99	1.00	39.86	0.99	1.00	39.67	0.99	1.00	37.18	0.94	1.00

larger payloads (8192 bits), message accuracy p_m degrades more noticeably, particularly at lower code rates. However, Polar codes demonstrate superior performance under these settings, achieving perfect $p_m = 1.00$ at rate=1/2 and maintaining $p_m \geq 0.80$ down to rate=1/16. These results indicate that even under more aggressive fine-tuning attacks, our method remains robust for large payloads, and Polar codes continue to offer the advantage in robustness.

Table 6: Robustness of watermarking against fine-tuning attacks (10k steps) for Stable Diffusion v2.1. Values of p_m exceeding 0.80 are highlighted in bold.

Msg Bits	Type	Watermark Bits = Msg Bits / rate									
		rate=1/2		rate=1/4		rate=1/8		rate=1/16		rate=1/32	
		p_w	p_m	p_w	p_m	p_w	p_m	p_w	p_m	p_w	p_m
4096	Rep	0.89	0.84	0.81	0.72	0.78	**0.82**	0.72	**0.82**	0.66	**0.80**
	Polar	0.89	**0.95**	0.82	**0.96**	0.80	**0.93**	0.72	**0.90**	0.67	0.62
8192	Rep	0.82	0.74	0.79	0.77	0.72	0.78	0.69	0.72	0.60	0.63
	Polar	0.83	**1.00**	0.80	0.79	0.75	**0.88**	0.69	**0.80**	0.63	0.53

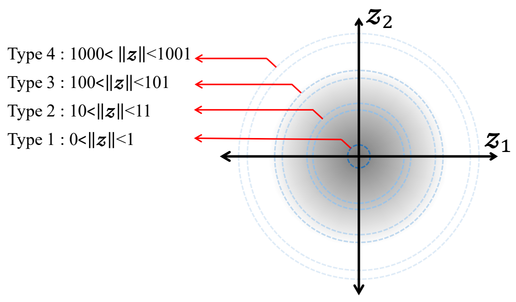

Figure 3: Different types of triggers with different probability density.

6.3.1 Trigger Selection

To analyze the impact of different types of triggers on fidelity, Bit Acc, and watermark robustness, we study how the choice of the trigger input — whether it is drawn from the same domain as the original input $\boldsymbol{z}$ or from a distinct one — affects the performance of the generator.

A concern is the overlap between trigger $\boldsymbol{t}$ and the normal input $\boldsymbol{z}$. Specifically, when $\boldsymbol{t}$ is drawn from a domain close to that of $\boldsymbol{z}$, does this result in a degradation of image quality due to interference in the input space? Conversely, if $\boldsymbol{t}$ lies far from the domain of $\boldsymbol{z}$, does this reduce watermark robustness due to the trigger being an outlier? To this end, we evaluate the effects of varying the distribution of triggers $\boldsymbol{t}$.

The default trigger is drawn from $\mathcal{N}(0, 1)$. To explore more diverse cases, we construct four additional trigger types. The construction of these triggers is as follows: (1) Random Unit Direction: we generate random unit vectors $\hat{\boldsymbol{t}}$ with length of d_m. (2) Controlled Scaling: we then scale $\hat{\boldsymbol{t}}$ by a random factor drawn from the predefined interval to adjust its magnitude. Each interval defines a specific range of scaling factors that adjust the norm of $\hat{\boldsymbol{t}}$: (0,1], [10,11], [100,101], and [1000,1001]. These operations ensure that all triggers lie along a random direction but vary in magnitude, thus falling into different probability regions of the Gaussian distribution.

We show a toy example of trigger selection in Fig. 3, where $\boldsymbol{z}$ has 2 dimensions. The gray radial region denotes the probability density of $\mathcal{N}(0, 1)$. As the trigger magnitude increases (from Type 1 to 4), the trigger moves further away from $\mathcal{N}(0, 1)$.

Table 7: Comparisons of different types of trigger. Results are based on 8192 message bits with 1/16 bit rate. Robustness stands for p_m after fine-tuning attacks for 4k steps.

Models	Metrics	Clean	Randn	Type 1	Type 2	Type 3	Type 4
StyleGAN2	Bit Acc	-	1.00	1.00	1.00	1.00	1.00
	FID	4.21	4.38	4.32	4.61	4.49	4.31
	Robustness	-	0.84	0.85	0.82	0.83	0.83
StyleGAN3	Bit Acc	-	0.99	0.98	0.98	0.99	0.98
	FID	3.55	3.88	3.94	3.61	3.94	3.72
	Robustness	-	0.84	0.83	0.82	0.82	0.82

The results are reported in Table 7. We tested the default triggers (denoted by Randn), as well as 4 other levels of triggers. We can observe that different kinds of triggers can achieve high Bit Acc, which remains above 0.98 for StyleGAN3 and 1.00 in most of the remaining cases. For fidelity, the difference between different types of triggers is irrelevant. In terms of robustness, the results indicate that while all trigger types perform similarly, the Randn trigger and the Type 1 achieve slightly higher robustness across both models.

The above results show that whether the triggers lie in the input space or not does not have a relevant effect. Therefore, we sample triggers from $\mathcal{N}(0, 1)$ in this work.

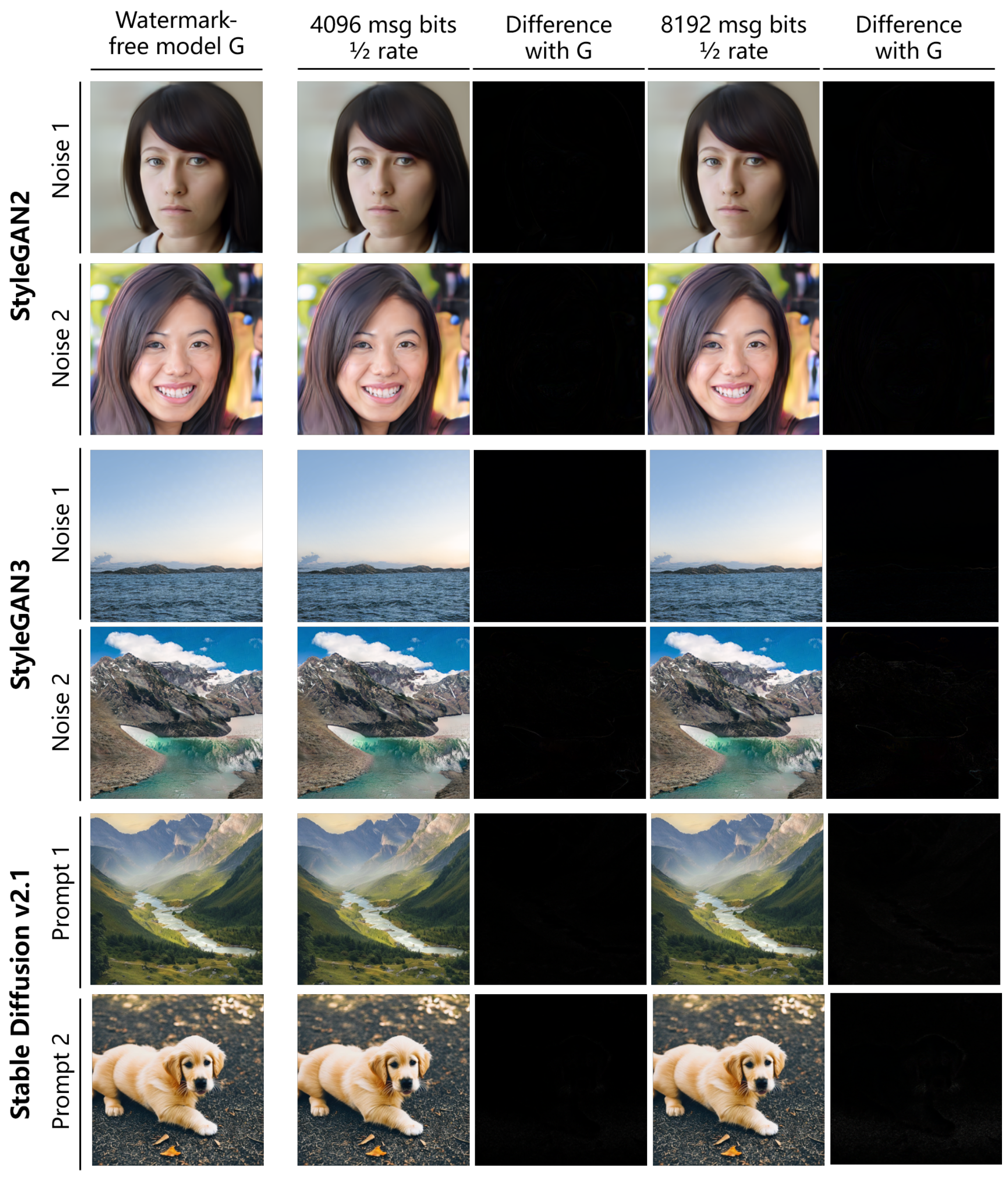

Figure 4: Samples produced by different models and differences with the watermark-free model.