

# Watermarking Generated Images

Steven Seiden, Hung Anh Vu, Zini Yang

# Overview

- **Deep dive:** Towards Universal Fake Image Detectors that Generalize Across Generative Models
  - Detecting images generated from different model family
- **Deep dive:** HiDDeN: Hiding Data With Deep Networks
  - Create robust image embedding
- **Deep dive:** Watermark-based Attribution of AI-Generated Content
  - Detecting where the images came from

# Towards Universal Fake Image Detectors that Generalize Across Generative Models

Utkarsh Ojha, Yuheng Li, Yong Jae Lee  
CVPR 2023

# Motivation

- Generated images come from multiple sources

# Motivation

- Generated images come from multiple sources
  - GANs
  - Diffusion models

# Motivation

- Generated images come from multiple sources
  - GANs
  - Diffusion models
  
- The goal is to develop a general purpose detection method that can distinguish between real and fake images from any source

# Previous work

- Train a convolutional network to classify between real vs fake images
  - Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot...for now. In *CVPR*, 2020.

# Previous work

- Train a convolutional network to classify between real vs fake images
  - Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot...for now. In *CVPR*, 2020.
  - The idea is to have a binary task of distinguishing between real (0) vs fake (1)
  - Trained Pro-GAN on 20 different object categories from LSUN and generated 18k fake images per category
  - 360k real vs 360k fake to train a classifier (Resnet-50)

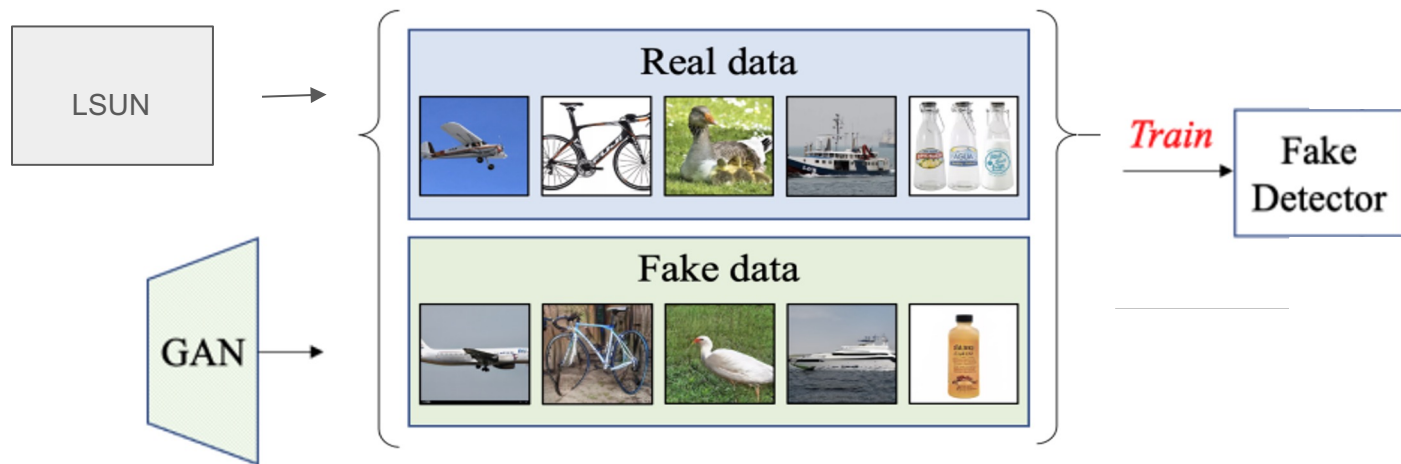


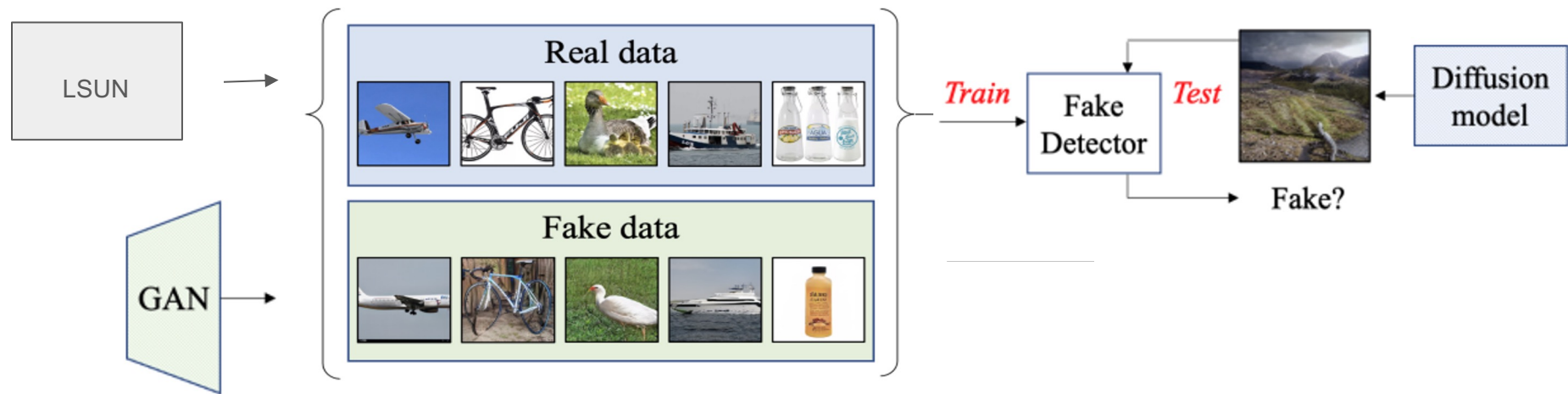
LSUN



GAN







- The problem with this approach is that it could not generalize to different families of models

	CycleGAN	GauGAN	LDM	Guided	DALL-E
Real acc.	98.64	99.4	99.61	99.14	99.61
Fake acc.	62.91	59.1	3.05	4.67	4.9
Average	80.77	79.25	51.33	51.9	52.26
Chance performance	50.00	50.00	50.00	50.00	50.00

Table 1. Accuracy of a real-vs-fake classifier [50] trained on ProGAN images in detecting real and fake images from different types of generative models. LDM, Guided, and DALL-E represent the breeds of image generation algorithms not seen during training.<sup>1</sup>

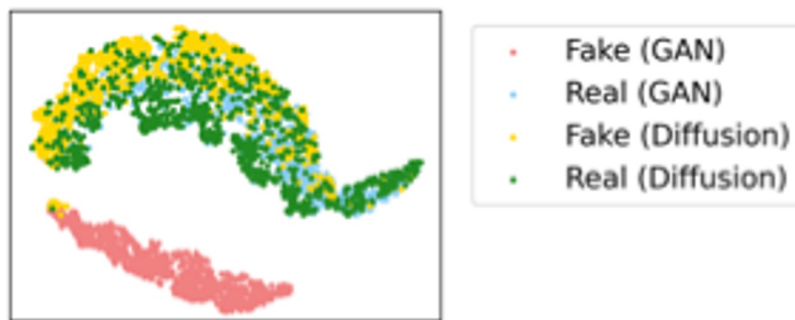


Figure 2. t-SNE visualization of real and fake images associated with two types of generative models. The feature space used is of a classifier trained to distinguish Fake (GAN) from Real (GAN).

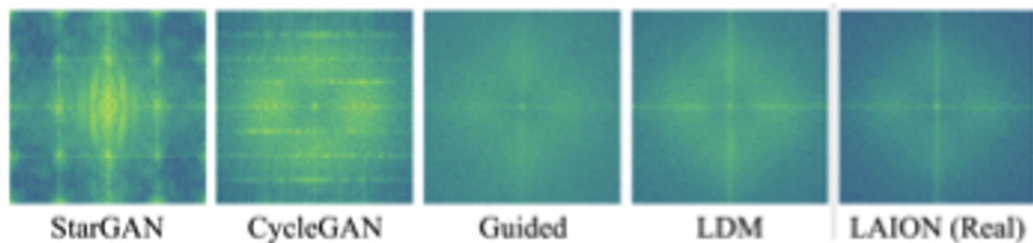


Figure 3. Average frequency spectra of each domain. The first four correspond to fake images from GANs and diffusion models. The last one represents real images from LAION [48] dataset.

ing  $F_{Diffusion}$ ? In what way are fake images from diffusion models different than images from GANs? We investigate this by visualizing the frequency spectra of different image distributions, inspired by [8, 9, 50, 53]. For each distribution (e.g.,  $F_{BigGAN}$ ), we start by performing a high pass filtering for each image by subtracting from it its median blurred image. We then take the average of the resulting high frequency component across 2000 images, and compute the Fourier transform. Fig. 3 shows this average frequency spectra for four fake domains and one real domain. Similar to [50], we see a distinct and repeated pattern in

# Approach

- Feature space (SHOULD NOT BE LEARNED)
  - This feature space should be exposed to large number of images so that i can generalize well to a variety of images
  - However, it should also be able to capture low-level details of an image (because the difference between F and R is usually at low level)
  - -> so authors chose to work with ViT-L/14

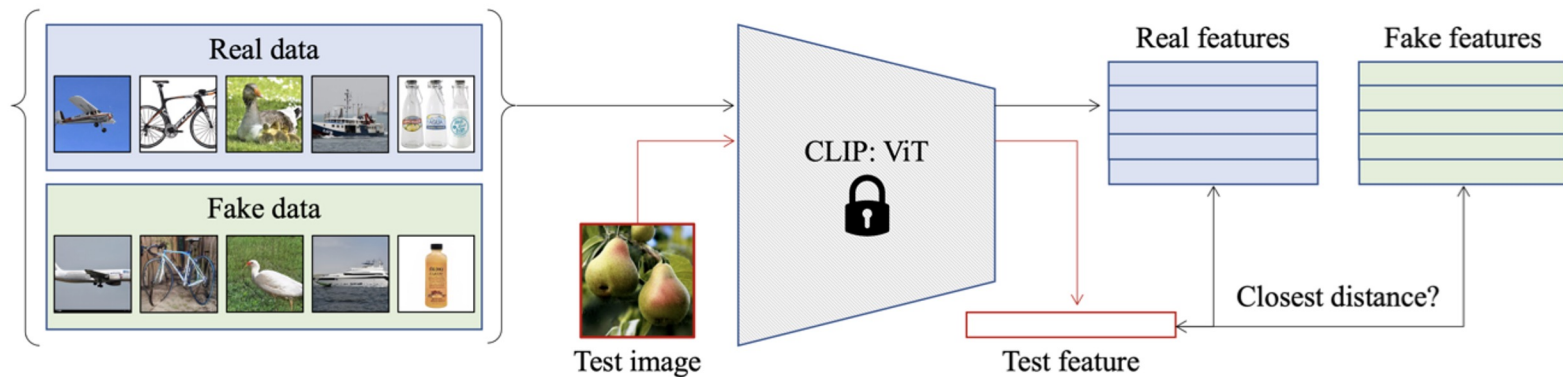


Figure 4. **Nearest neighbors for real-vs-fake classification.** We first map the real and fake images to their corresponding feature representations using a pre-trained CLIP:ViT network *not trained for this task*. A test image is mapped into the same feature space, and cosine distance is used to find the closest member in the feature bank. The label of that member is the predicted class.

## NEAREST NEIGHBOR

$$\text{pred}(x) = \begin{cases} 1, & \text{if } \min_i (d(\phi_x, \phi_{f_i})) < \min_i (d(\phi_x, \phi_{r_i})) \\ 0, & \text{otherwise.} \end{cases}$$

## LINEAR CLASSIFICATION

$$\mathcal{L} = - \sum_{f_i \in \mathcal{F}} \log(\psi(\phi_{f_i})) - \sum_{r_i \in \mathcal{R}} \log(1 - \psi(\phi_{r_i})).$$



Detection method	Variant	Generative Adversarial Networks						Deep fakes	Low level vision		Perceptual loss		Guided	LDM			Glide			DALL-E	Total
		Pro-GAN	Cycle-GAN	Big-GAN	Style-GAN	Gau-GAN	Star-GAN		SITD	SAN	CRN	IMLE		200 steps	200 w/ CFG	100 steps	100 27	50 27	100 10		Avg. acc
Trained deep network [50]	Blur+JPEG (0.1)	99.99	85.20	70.20	85.7	78.95	91.7	53.47	48.69	86.31	86.26	60.07	54.03	54.96	54.14	60.78	63.8	65.66	55.58	69.58	
	Blur+JPEG (0.5)	<b>100.0</b>	80.77	58.98	69.24	79.25	80.94	51.06	56.94	47.73	87.58	94.07	51.90	51.33	51.93	51.28	54.43	55.97	54.36	52.26	64.73
	Oracle* (B+J 0.5)	<b>100.0</b>	90.88	82.40	<b>93.11</b>	93.52	87.27	62.48	<b>76.67</b>	57.04	<b>95.28</b>	<b>96.93</b>	65.20	63.15	62.39	61.50	65.36	69.52	66.18	60.10	76.26
	ViT:CLIP (B+J 0.5)	98.94	78.80	60.62	60.56	66.82	62.31	52.28	65.28	47.97	64.09	79.54	50.66	50.74	51.04	50.76	52.15	53.07	52.06	53.18	60.57
Patch classifier [10]	ResNet50-Layer1	94.38	67.38	64.62	82.26	57.19	80.29	55.32	64.59	51.24	54.29	55.11	65.14	79.09	<b>76.17</b>	79.36	67.06	68.55	68.04	69.44	68.39
	Xception-Block2	75.03	68.97	68.47	79.16	64.23	63.94	75.54	75.14	<b>75.28</b>	72.33	55.3	67.41	76.5	76.1	75.77	74.81	73.28	68.52	67.91	71.24
Co-occurrence [35]	-	97.70	63.15	53.75	92.50	51.1	54.7	57.1	63.06	55.85	65.65	65.80	60.50	70.7	70.55	71.00	70.25	69.60	69.90	67.55	66.86
Freq-spec [53]	CycleGAN	49.90	<b>99.90</b>	50.50	49.90	50.30	<b>99.70</b>	50.10	50.00	48.00	50.60	50.10	50.90	50.40	50.40	50.30	51.70	51.40	50.40	50.00	55.45
<b>Ours</b>	NN, $k = 1$	99.58	94.70	86.95	80.24	96.67	98.84	80.9	71.0	56.0	66.3	76.5	68.76	89.56	68.99	89.51	86.44	88.02	87.27	77.52	82.30
	NN, $k = 3$	99.58	95.04	87.63	80.55	96.94	98.77	83.05	71.5	59.5	66.69	76.87	70.02	90.37	70.17	90.57	87.84	89.34	88.78	79.29	83.28
	NN, $k = 5$	99.60	94.32	88.23	80.60	97.00	98.90	83.85	71.5	60.0	67.04	78.02	70.55	90.89	70.97	91.01	88.42	90.07	89.60	80.19	83.72
	NN, $k = 9$	99.54	93.49	88.63	80.75	97.11	98.97	<b>84.5</b>	71.5	61.0	69.27	79.21	<b>71.06</b>	91.29	72.02	91.29	<b>89.05</b>	<b>90.67</b>	<b>90.08</b>	81.47	<b>84.25</b>
	LC	<b>100.0</b>	98.50	<b>94.50</b>	82.00	<b>99.50</b>	97.00	66.60	63.00	57.50	59.5	72.00	70.03	<b>94.19</b>	73.76	<b>94.36</b>	79.07	79.85	78.14	<b>86.78</b>	81.38

Table 3. **Generalization results.** Analogous result of Table 2, where we use classification accuracy (averaged over real and fake images) to compare the methods. Oracle with \* indicates that the method uses the test set to calibrate the confidence threshold. The fixed feature backbone (Ours NN/LC) has a significant gain in accuracy (+25-30% over the baselines) when testing on unseen generative model families.

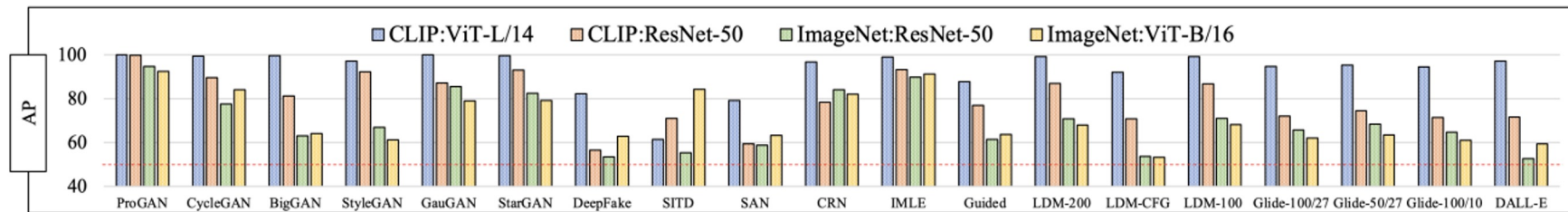


Figure 5. **Ablation on the network architecture and pre-training dataset.** A network trained on the task of CLIP is better equipped at separating fake images from real, compared to networks trained on ImageNet classification. The red dotted line depicts chance performance.

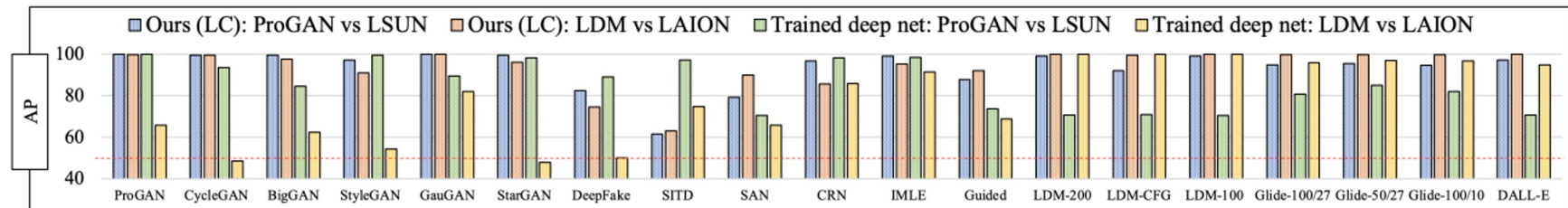


Figure 6. **Average precision of methods with respect to training data.** Both our linear classifier on CLIP:ViT’s features and the baseline trained deep network [50] are given access to two different types of training data: (i)  $\mathcal{R} = \text{LSUN}$  [51] and  $\mathcal{F} = \text{ProGAN}$  [28], (ii)  $\mathcal{R} = \text{LAION}$  [48] and  $\mathcal{F} = \text{LDM}$  [46]. Irrespective of the training data source, our linear classifier preserves its ability to generalize well on images from other unseen generative model families, which is not the case for the baseline trained deep network.

# HiDDeN: Hiding Data With Deep Networks

Jiren Zhu, Russell Kaplan, Justin Johnson and Li Fei-Fei

Computer Science Department, Stanford University

# Motivation

- Online, many want to have proof they are the original owner of an image
  - Artists creating artwork
  - Photographers uploading pictures
  - Generative models marking images as synthetic
- Visually present watermarks are ugly, text in the bottom of images can be cropped, and metadata can be easily altered

# Motivation

- Solution: Robustly embedding hidden data into images in a visually undetectable manner

# Motivation

- Solution: Robustly embedding hidden data into images in a visually undetectable manner
- Needs to persist through image perturbations
  - Cropping, blurring, compression, etc.
- Use cases: Watermarking, creator attribution

# Prior Work

- Previously, visually undetectable image perturbation has been used to cause misclassification in deep learning image classification models<sup>1,2</sup>

[1] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: ICLR. (2014)

[2] Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. In: ICLR Workshop. (2017)

# Prior Work

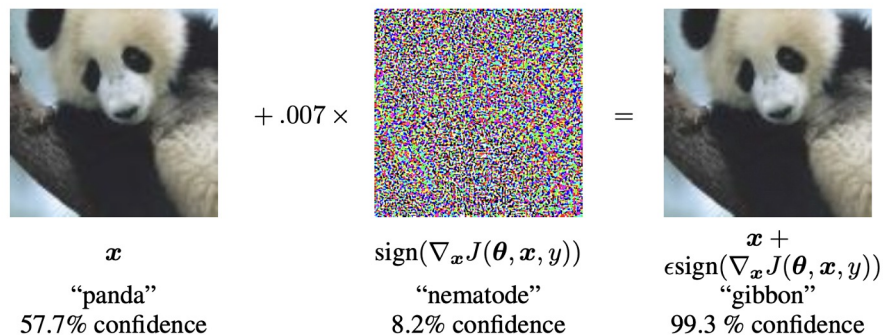


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet’s classification of the image. Here our  $\epsilon$  of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet’s conversion to real numbers.



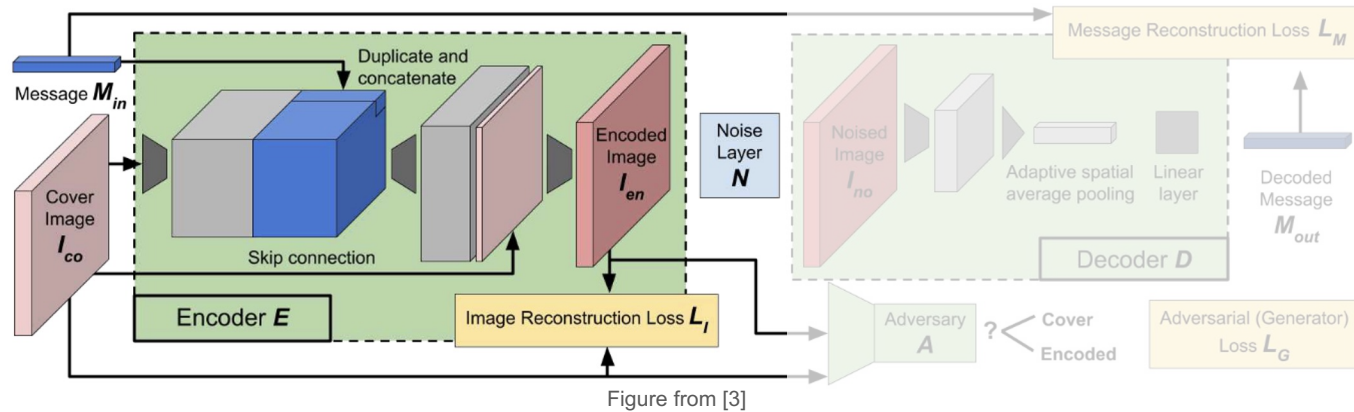
# Prior Work

- Previously, visually undetectable image perturbation has been used to cause misclassification in deep learning image classification models<sup>1,2</sup>
- Idea: Why not leverage this idea for creating watermarks?
- Similar to the previous work, this work repurpose a previous methodology.

[1] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: ICLR. (2014)

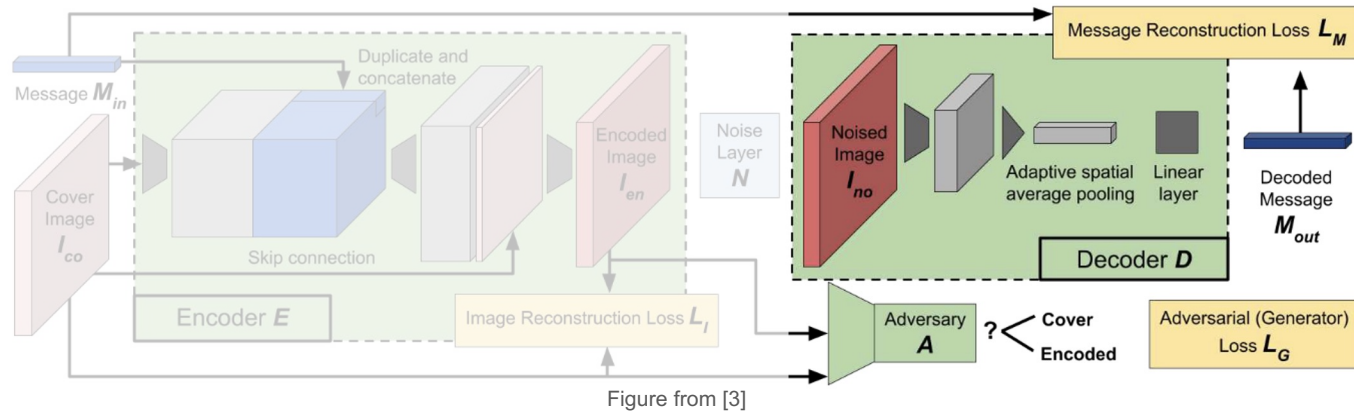
[2] Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. In: ICLR Workshop. (2017)

# Methodology



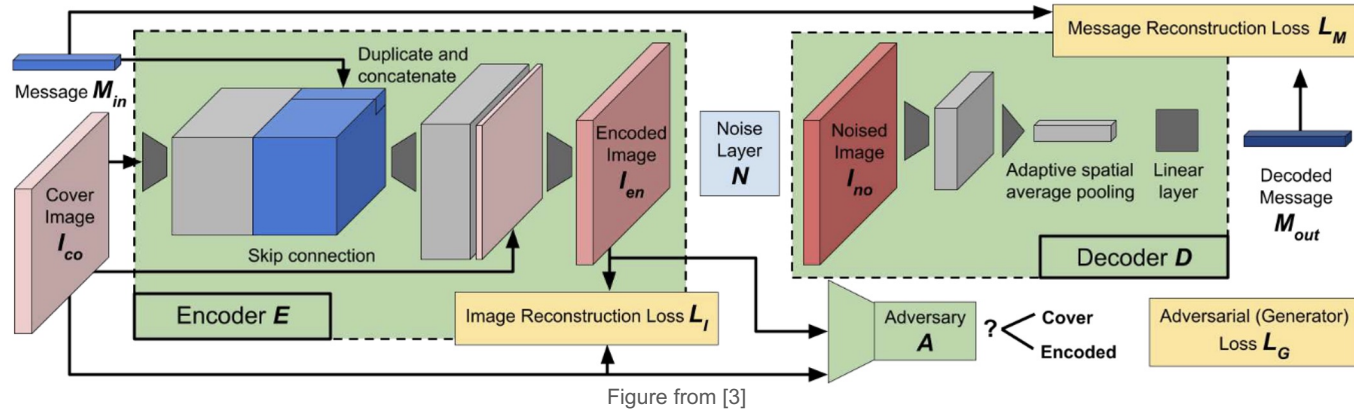
The input image  $I_{co}$  is convoluted by encoder  $E$  to incorporate a binary message  $M_{in}$ . This adds a noise layer  $N$ . The result is a noisy image  $I_{no}$ .

# Methodology



To decode the message,  $I_{no}$  is fed into the decoder, a deep neural network, that uses the image to predict the message. The loss of data from the original message  $L_m$  is calculated.

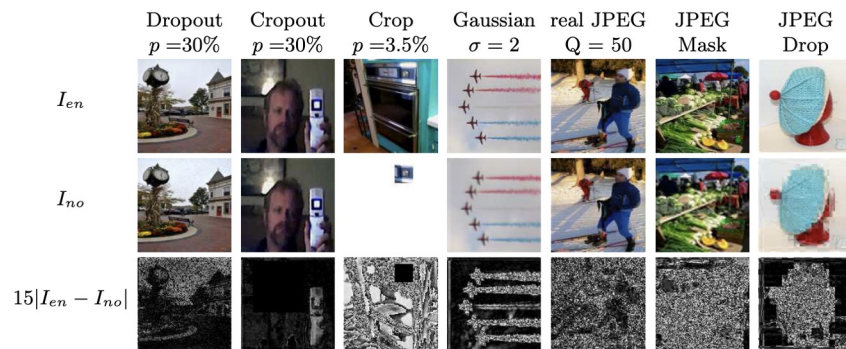
# Methodology



At the same time, a discriminator calculates the distance between  $I_{co}$  and  $I_{no}$ , ensuring the resulting image  $I_{no}$  looks visually similar to the original image  $I_{co}$ . This is to ensure the impact of the noise layer is minimal.

The ultimate goal is to find a configuration optimal for creating watermarks yet avoiding steganography detection.

# Methodology



**Fig. 3.** Illustration of non-identity noise layers. JPEG-Mask and JPEG-Drop are differentiable approximations of the JPEG compression (see Figure 4). **Top:** Encoded image  $I_{en}$ . **Middle:** Noised image  $I_{no}$ . **Bottom:** Magnified difference  $|I_{en} - I_{no}|$ . Even under heavy distortion, such as a Crop layer which retains only 3.5% of the original image, our model still learns to recover the watermark with high accuracy (see Section 4).

Figure from [3]

To test the robustness of this methodology, the authors distort the images in various ways, such as cropping, blurring, and compressing the images.

Above: The difference before and after different image manipulations is demonstrated.

# Experiments: Steganography

- Compared against prior message hiding algorithms HUGO, WOW, and S-UNIWARD
- Found benefit over these: encoding can mutate to avoid predictability

Method	Bits per pixel	Bit error	Detection rate (ATS [30]) (%)
HUGO [4]	0.200	-	70
WOW [10]	0.200	-	68
S-UNIWARD [11]	0.200	-	68
HiDDeN (model weights known)	0.203	$< 10^{-5}$	98
HiDDeN (model weights unknown)	0.203	$< 10^{-5}$	<b>50</b>

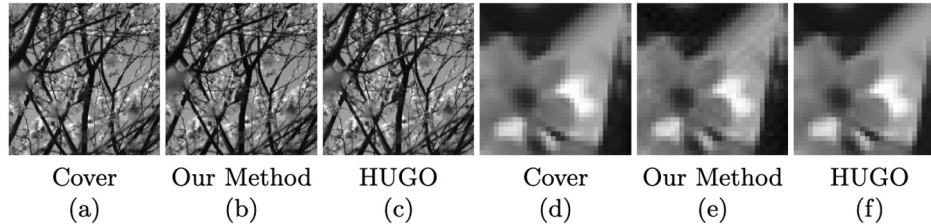


Figure from [3]

# Experiments: Steganography

- Compared against prior message hiding algorithms HUGO, WOW, and S-UNIWARD
- Found benefit over these: encoding can mutate to avoid predictability
  - 98% accuracy in detection when HiDDeN model weights are known
  - **However**, only a 50% accuracy when weights are unknown
  - Compare to 70% for HUGO and 68% for WOW and S-UNIWARD

# Experiments: Watermarking

Need to mitigate types of image distortions:

- Dropout and cropout: The original and encoded image are merged, undoing some of the noise layer
  - Dropout: random pixels throughout the image
  - Cropout: random cluster of pixels
- Gaussian: Blurs the images' pixels
- Crop: Removes all but a square subset of pixels
- JPEG: Applies JPEG compression to the image









# Experiments: Watermarking

- Model trained on 128 x 128 images with messages of 30 bytes
- Experimented with training on images distorted in various ways

	Digimarc	Identity Dropout	Cropout	Crop	Gaussian	JPEG-mask	JPEG-drop	Combined	
PSNR(Y)	62.12	44.63	42.52	47.24	35.20	40.55	30.09	28.79	33.55
PSNR(U)	38.23	45.44	38.52	40.97	33.31	41.96	35.33	32.51	38.92
PSNR(V)	52.06	46.90	41.05	41.88	35.86	42.88	36.27	33.42	39.38

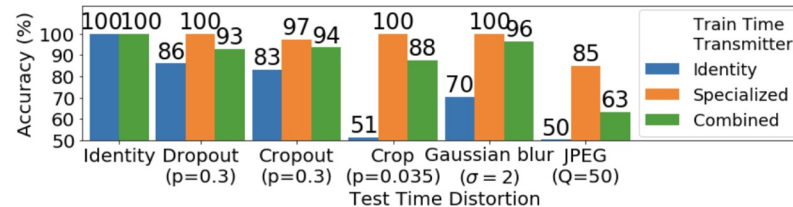
Cover	Digimarc	Trained with Adversary			No Adversary
		Crop	Gaussian	Combined	Combined
					

**Fig. 8.** Image distortions for watermarking algorithms. **Top:** Mean PSNR between cover and encoded images for Digimarc and our model trained with different noise layers. **Bottom:** A cover image and encoded images from both Digimarc and our model trained with Crop, Gaussian, and Combined noise layers. **Bottom Right:** An encoded image from a model trained with combined noise but without an adversary. Adversarial training significantly improves the visual quality of the encoded images.

Figure from [3]

# Experiments: Watermarking

- Different distortions impact the accuracy of the recovered watermark in different ways:



**Fig. 9.** Robustness of our models against different test time distortions. Each cluster uses a different test time distortion. Identity (blue) is trained with no image distortion; Specialized (orange) is trained on the same type of distortion used during testing; Combined (green) is trained on all types of distortions.

Figure from [3]

# Experiments: Watermarking

- Comparing to a baseline watermarking system, Digimarc:
  - Intuitively, Digimarc performs worse under all distortions except JPEG

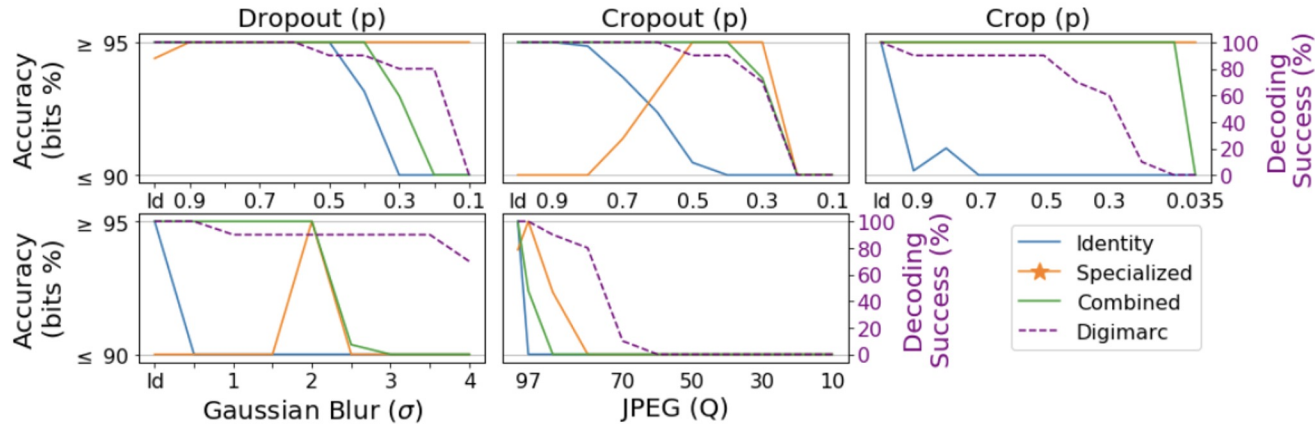


Figure from [3]

# Conclusion

HiDDeN is:

- More flexible than traditional message hiding techniques
- Resistant to many types of distortions

# Watermark-based Attribution of AI-Generated Content

Zhengyuan Jiang, Moyang Guo, Yuepeng Hu, Neil Zhenqiang Gong

# Motivation:

## We have talked about:

- “Towards Universal Fake Image Detectors that Generalize Across Generative Models” aims to detect AI-generated content.
- “HiDDeN: Hiding Data With Deep Networks” aims to develop watermarks that are robust against post-processing.

## New stuff to talk about — “Attribution”:

- Attribution seeks to trace **the origin of a piece of content** detected as AI-generated, specifically identifying the user of the GenAI service who created it.

# Idea

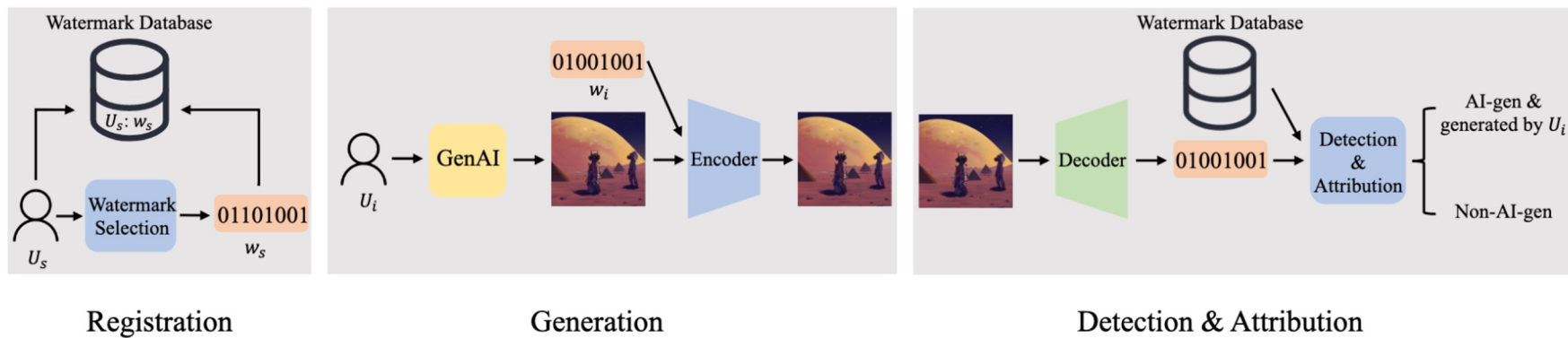


Figure 1: *Registration, generation, and detection & attribution* phases of watermark-based detection and attribution.

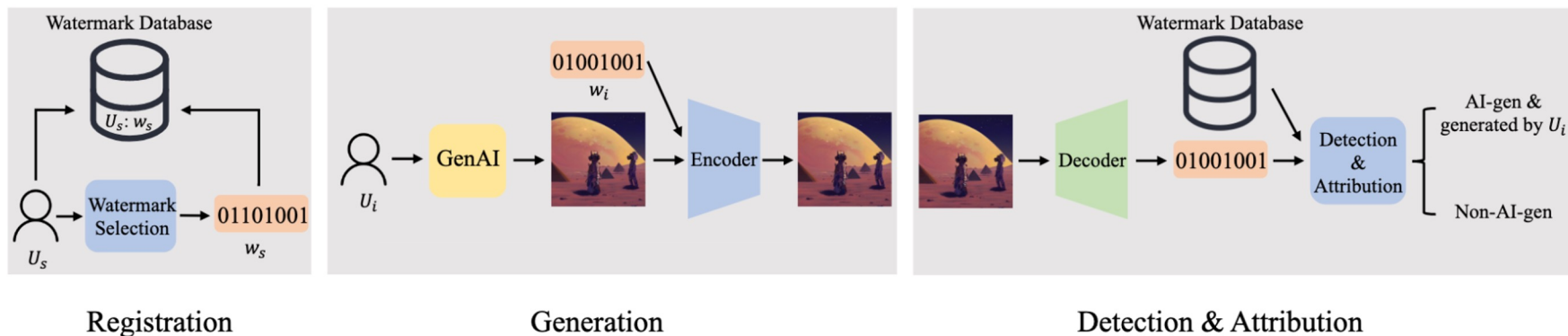


Figure 1: *Registration, generation, and detection & attribution* phases of watermark-based detection and attribution.

**AI-generated images models:** Stable Diffusion, Midjourney, and DALL-E

**Watermarking:** HiDDeN (bitstring based, learning based)

**The key problem:** How to choose the watermarks for users to maximize detection and attribution performance?



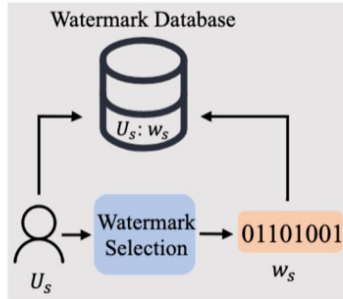
# Watermark Selection Problem

## Two-step method:

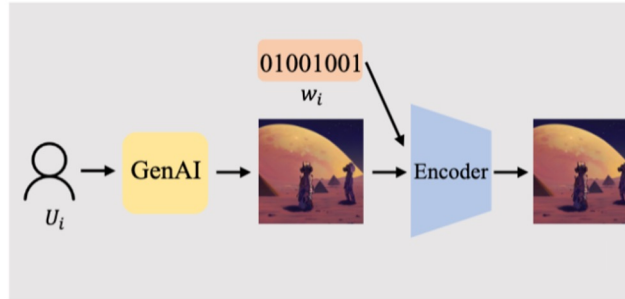
**Step1:** Theoretically evaluate the detection and attribution performance of watermarks. Define metrics for performance evaluation.

**Step2:** Select watermarks based on the performance metrics

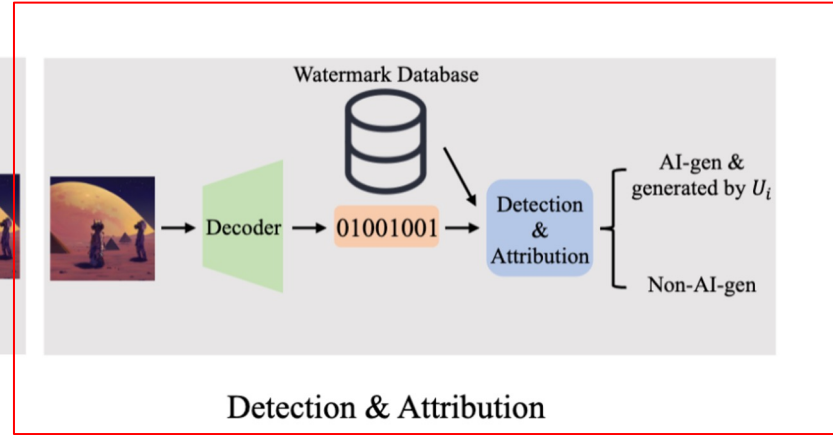
# Watermark Selection Problem



Registration



Generation



Detection & Attribution

# Detection and Attribution

**Bitwise Accuracy:** To measure the similarity of two watermarks.

$$BA(w, w') = \frac{1}{n} \sum_{k=1}^n \mathbb{I}(w[k] = w'[k])$$

**AI-generated detection:** content  $C$  is detected as AI-generated if and only if the following satisfies:

where  $\tau > 0.5$  is the detection threshold.

$$\max_{i \in \{1, 2, \dots, s\}} BA(D(C), w_i) \geq \tau$$

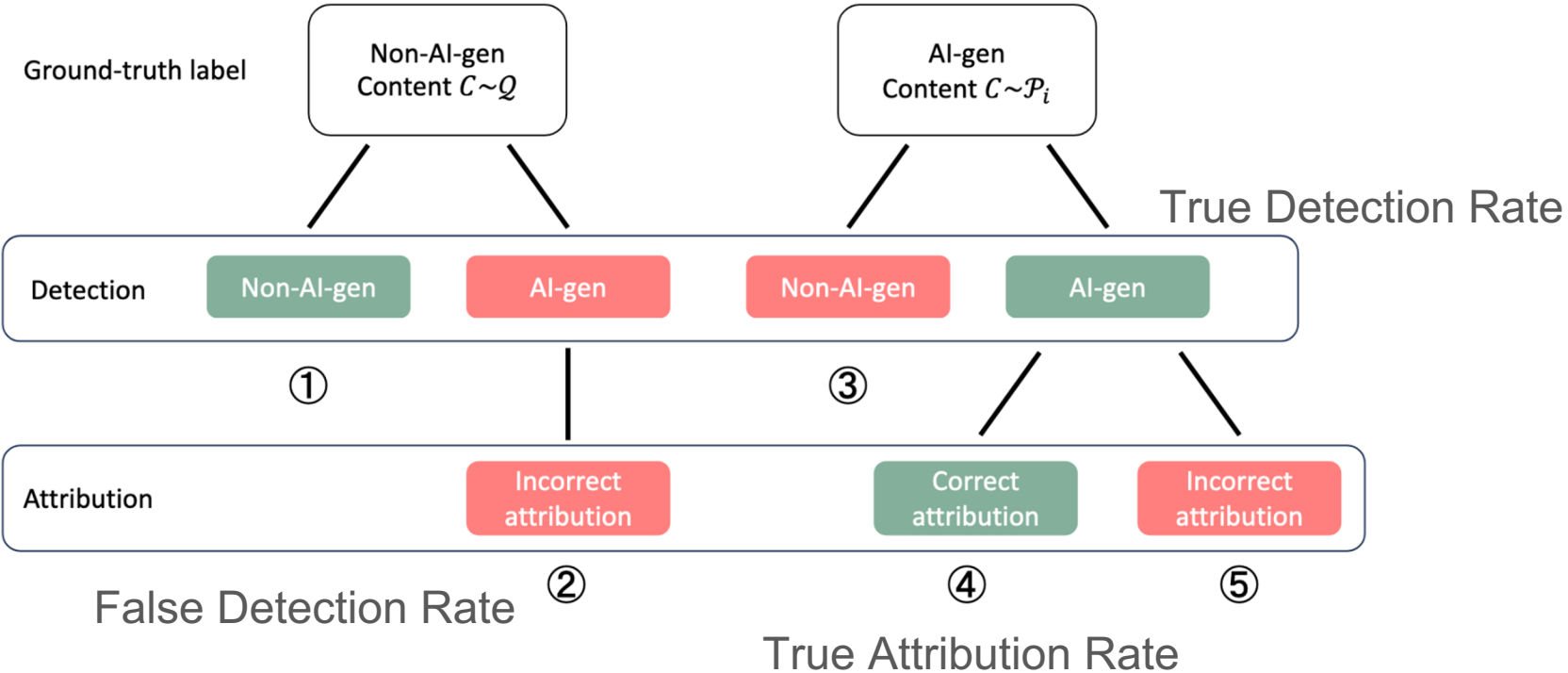
# Detection and Attribution

**Attribution:** We attribute the content to the user whose watermark is the most similar to the decoded watermark  $D(C)$ .

$$i^* = \arg \max_{i \in \{1, 2, \dots, s\}} BA(D(C), w_i)$$

---

# Detection and Attribution Performance



# Detection and Attribution Performance

True Detection Rate:

$$TDR_i = \Pr\left(\max_{j \in \{1, 2, \dots, s\}} BA(D(C), w_j) \geq \tau\right),$$

False Detection Rate:

$$FDR = \Pr\left(\max_{j \in \{1, 2, \dots, s\}} BA(D(C), w_j) \geq \tau\right),$$

True Attribution Rate:

$$TAR_i = \Pr\left(\max_{j \in \{1, 2, \dots, s\}} BA(D(C), w_j) \geq \tau \wedge BA(D(C), w_i) > \max_{j \in \{1, 2, \dots, s\} / \{i\}} BA(D(C), w_j)\right),$$

# Detection and Attribution Performance

**Theorem 4** (Lower bound of  $TAR_i$ ). *Suppose we are given  $s$  users with any  $s$  watermarks  $W = \{w_1, w_2, \dots, w_s\}$ . When the watermarking method is  $\beta_i$ -accurate for user  $U_i$ 's AI-generated content, we have a lower bound of  $TAR_i$  as follows:*

$$TAR_i \geq Pr(n_i \geq \max\{\lfloor \frac{1 + \bar{\alpha}_i}{2} n \rfloor + 1, \tau n\}), \quad (10)$$

where  $n_i$  follows a binomial distribution with parameters  $n$  and  $\beta_i$ , i.e.,  $n_i \sim B(n, \beta_i)$ ,  $\bar{\alpha}_i = \max_{j \in \{1, 2, \dots, s\} / \{i\}} BA(w_i, w_j)$ ,  $n$  is the watermark length, and  $\tau$  is the detection threshold.

The lower bound is larger when  $\bar{\alpha}_i$  is smaller because it is easier to distinguish between users.

# Watermark Selection Problem

Two-step method:

Step2:

Maximize the **lower bounds TAR**, the optimization simplifies to “**Selecting the most dissimilar watermarks for users**”.

This can be formulated as:

$$\min_{w_s} \max_{i \in \{1, 2, \dots, s-1\}} BA(w_i, w_s).$$



# Approach

How to solve the **Farthest string problem?**

Random, BSTA, NRG, while they finally adopt **A-BSTA**.

Method	Advantages	Disadvantages
Random	<ul style="list-style-type: none"><li>- Extremely simple to implement</li><li>- No computational overhead</li></ul>	<ul style="list-style-type: none"><li>- High chance of picking a watermark too similar to existing ones</li><li>- Can result in poor accuracy</li></ul>
BSTA	<ul style="list-style-type: none"><li>- Finds the exact “best” watermark solution (farthest string)</li></ul>	<ul style="list-style-type: none"><li>- Exponential time complexity (NP-hard)</li><li>- Not scalable for large inputs</li></ul>
NRG	<ul style="list-style-type: none"><li>- More efficient than BSTA</li><li>- Less likely to produce a redundant watermark than purely Random</li></ul>	<ul style="list-style-type: none"><li>- Not guaranteed to find the true best solution</li><li>- Still be expensive for large inputs</li></ul>
A-BSTA (Approx)	<ul style="list-style-type: none"><li>- Adapts BSTA for efficiency by limiting recursion depth</li><li>- Further benefits from random initialization</li></ul>	<ul style="list-style-type: none"><li>- Still approximate; can fail to find the globally optimal watermark</li><li>- Performance depends on chosen recursion limit</li></ul>

# Approach

---

## Algorithm 1 BSTA( $w_s, d, m$ )

---

**Input:** Initial watermark  $w_s$ , recursion depth  $d$ , and  $m$ .

**Output:**  $w_s$  or *NotExist*.

- 1: **if**  $d < 0$  **then**
  - 2:     return *NotExist*
  - 3:  $i^* \leftarrow \arg \max_{i \in \{1, 2, \dots, s-1\}} BA(w_i, w_s)$
  - 4: **if**  $BA(w_{i^*}, w_s) > (m + d)/n$  **then**
  - 5:     return *NotExist*
  - 6: **else if**  $BA(w_{i^*}, w_s) \leq m/n$  **then**
  - 7:     return  $w_s$
  - 8:  $B \leftarrow \{k | w_s[k] = w_{i^*}[k], k = 1, 2, \dots, n\}$
  - 9: Choose any  $B' \subset B$  with  $|B'| = m + 1$
  - 10: **for all**  $k \in B'$  **do**
  - 11:      $w'_s \leftarrow w_s$
  - 12:      $w'_s[k] \leftarrow \neg w'_s[k]$
  - 13:      $w'_s \leftarrow \text{BSTA}(w'_s, d - 1, m)$
  - 14:     **if**  $w'_s$  is not *NotExist* **then**
  - 15:         return  $w'_s$
  - 16: return *NotExist*
- 

---

## Algorithm 2 NRG( $w_s, m$ )

---

**Input:** Initial watermark  $w_s$  and  $m$ .

**Output:**  $w_s$  or *NotExist*.

- 1:  $F \leftarrow \emptyset$
  - 2:  $d \leftarrow m$
  - 3: **while**  $d > 0$  **do**
  - 4:      $i^* \leftarrow \arg \max_{i \in \{1, 2, \dots, s-1\}} BA(w_i, w_s)$
  - 5:     **if**  $BA(w_{i^*}, w_s) > 2m/n$  **then**
  - 6:         return *NotExist*
  - 7:     **else if**  $BA(w_{i^*}, w_s) \leq m/n$  **then**
  - 8:         return  $w_s$
  - 9:      $B \leftarrow \{k | w_s[k] = w_{i^*}[k] \wedge k \notin F, k = 1, 2, \dots, n\}$
  - 10:      $l \leftarrow n \cdot BA(w_{i^*}, w_s) - m$
  - 11:     Sample  $B' \subset B$  with  $|B'| = l$  uniformly at random
  - 12:     **for all**  $k \in B'$  **do**
  - 13:          $w_s[k] \leftarrow \neg w_s[k]$
  - 14:      $d \leftarrow d - l$
  - 15:      $F \leftarrow F \cup B'$
  - 16: return *NotExist*
-

# Approach

---

---

## Algorithm 3 Solving our watermark selection problem

---

**Input:** Existing  $s - 1$  watermarks  $w_1, w_2, \dots, w_{s-1}$ .

**Output:** Watermark  $w_s$ .

- 1:  $m \leftarrow \max_{i \in \{1, 2, \dots, s-2\}} n \cdot BA(w_i, w_{s-1})$
  - 2: **while**  $w_s$  is *NotExist* **do**
  - 3:   **if** BSTA **then**
  - 4:      $w_s \leftarrow \neg w_1$
  - 5:      $w_s \leftarrow BSTA(w_s, m, m)$
  - 6:   **if** NRG **then**
  - 7:      $w_s \leftarrow \neg w_1$
  - 8:      $w_s \leftarrow NRG(w_s, m)$
  - 9:   **if** A-BSTA **then**
  - 10:      $w_s \leftarrow$  sampled uniformly at random
  - 11:      $w_s \leftarrow BSTA(w_s, d, m)$
  - 12:   **if**  $w_s$  is *NotExist* **then**
  - 13:      $m \leftarrow m + 1$
  - 14: **return**  $w_s$
-

# Experiment

AI-generated: 10,000 images for training, 1,000 images for testing.

Non-AI-generated: 1,000 images

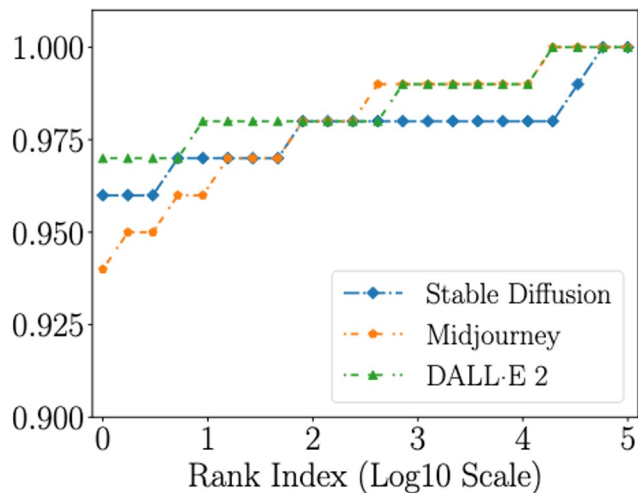
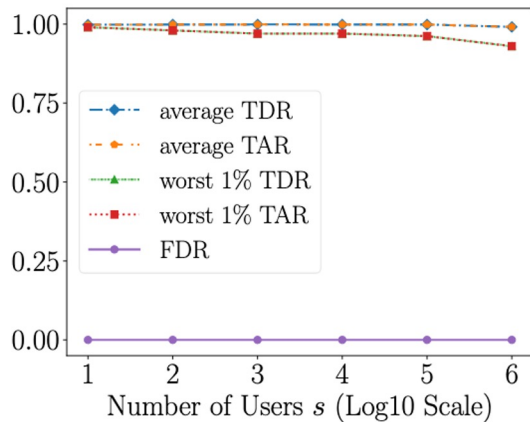
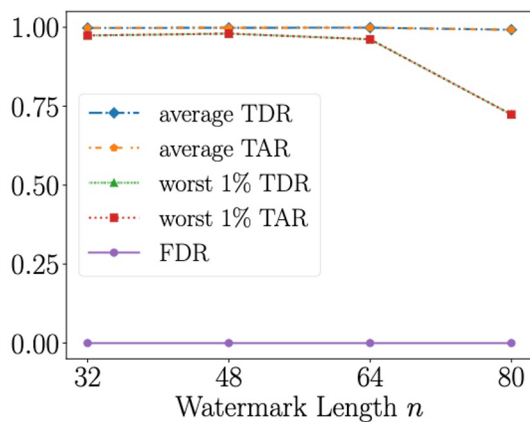


Figure 4: Ranked TARs of the 100,000 users.

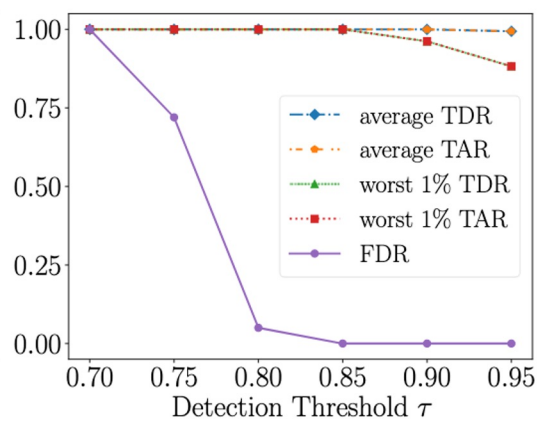
# Experiment Result



(a) Impact of  $s$



(b) Impact of  $n$

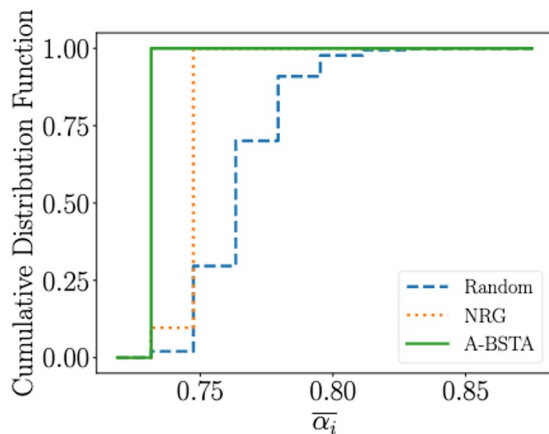


(c) Impact of  $\tau$

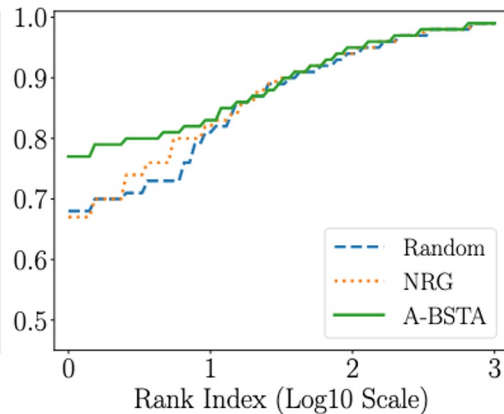
# Experiment Result: Different Watermarks Approach

Table 3: The average running time for different watermark selection methods to generate a watermark.

	Random	NRG	A-BSTA
Time (ms)	0.01	2.11	24.00

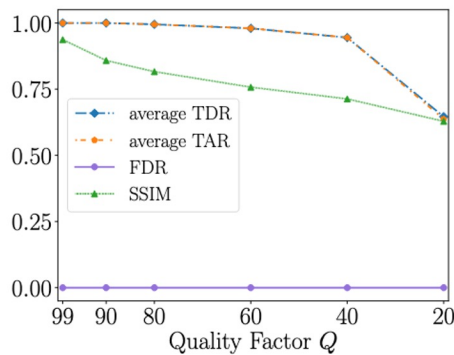


(a) CDF of  $\alpha_i$

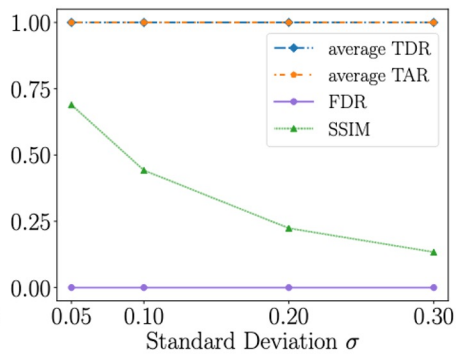


(b) Ranked TARs

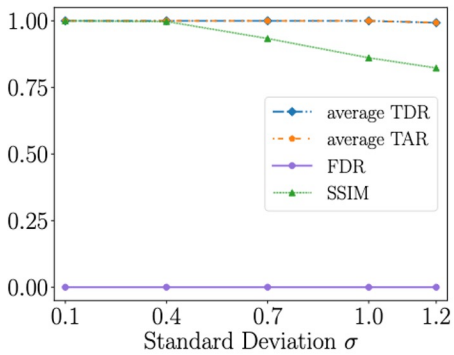
# Experiment Result: Robust Check



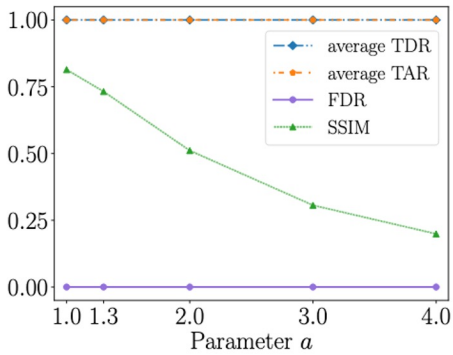
(a) JPEG



(b) Gaussian noise



(c) Gaussian blur



(d) Brightness/Contrast