



MN914 – Joint Project

Topic 2: Watermarking AI generated images and video

Hoang-Nhat TRAN

Thu-Hien LE

Imene BOUTIOUTA

Under the supervision of: **Matéo ZOUGHEBI**, **Carl De Sousa TRIAS**, and **Prof. Mihai MITREA**

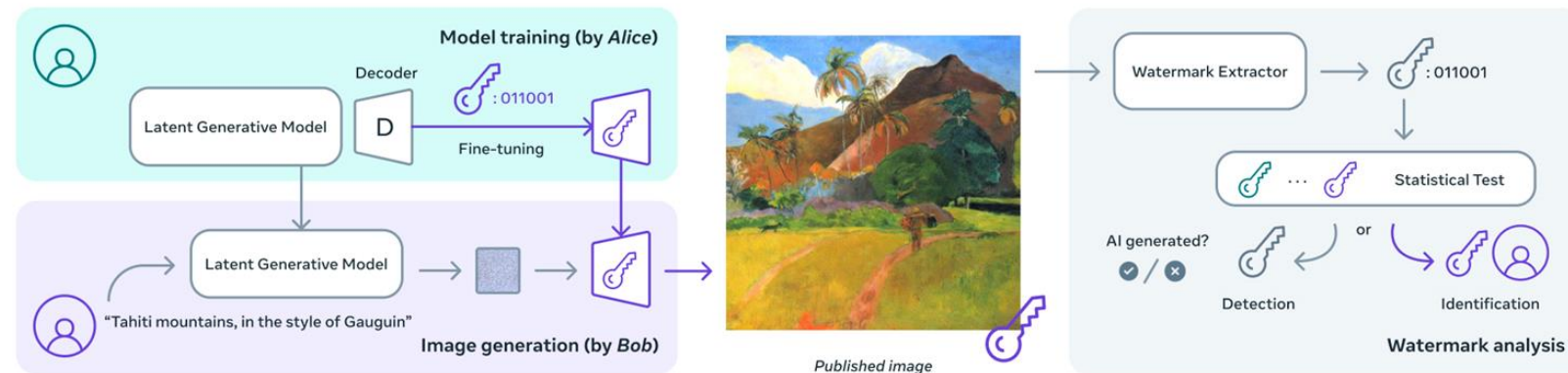
Content

- Introduction
- Related works
 - Stable Signature
 - HiDDeN Method
- Proposed methods
 - Stable Signature for GAN
 - U-Net-based watermark encoder
- Experiments
- Future Directions

Introduction

Problem Statement:

- Challenges of authenticity and copyright when distinguishing AI-generated content raises significant risks like deepfakes and misuses.
- Neural Network Watermarking (NNW) allow to insert a secret noise-tolerant metadata into a neural network, which can be extracted.
- Classical NNW methods require Bob to hand back the model to Alice to check if it was originally trained by her → **it is not the case in practice.**
- Solution: SOTA merge-in technique: **Stable Signature** [Fernandez2023] can be used to trace the model by only the outputs.



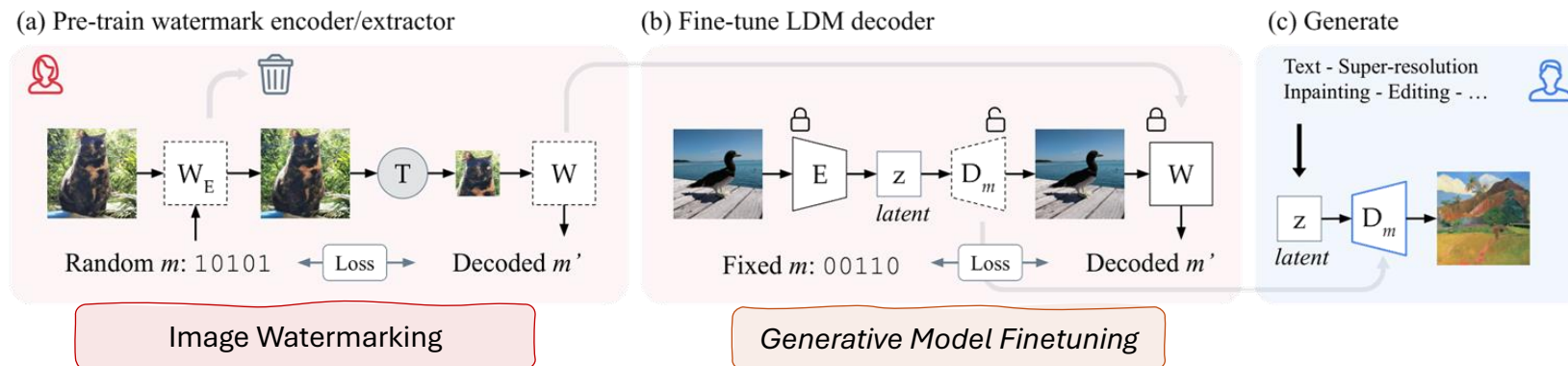
Introduction

Objective of the project:

- Explore how merge-in NNW techniques can be applied to AI-generated contents.
- Adapt Stable Signature for small generative models.
- Improve robustness or imperceptibility over the baseline.

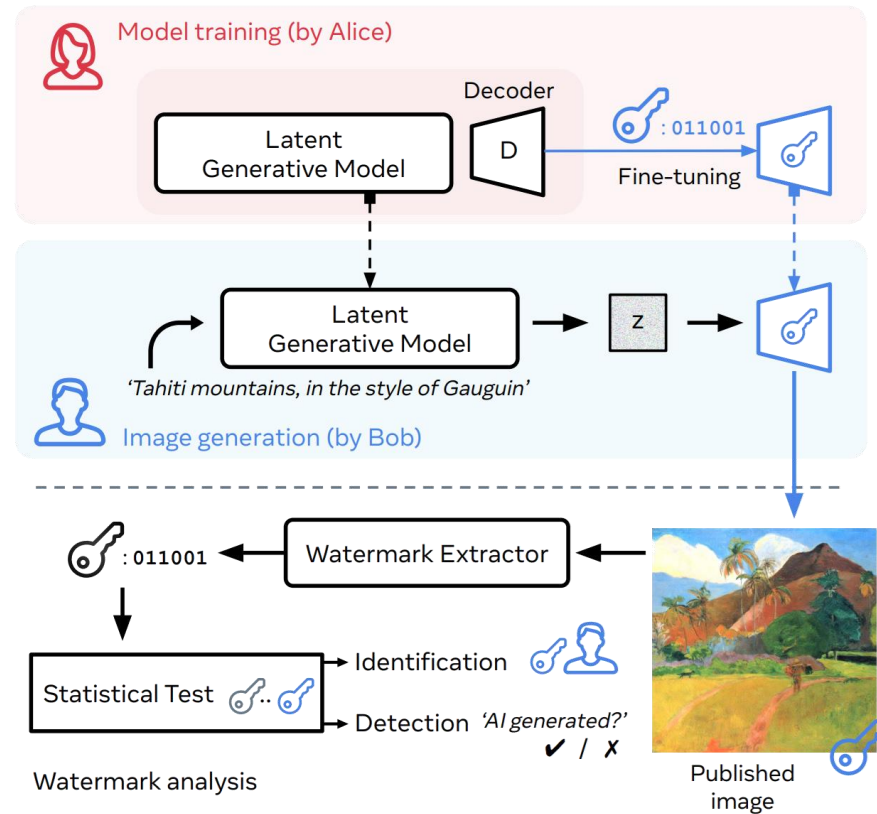
Stable Signature

- Merge-in NNW technique created to distinguish when an image is generated by a generative model.
- The watermark is invisible to human but can be detected by algorithms – even when the images are edited.
- There are **two** parts:
 - **Image watermarking**: Embed invisible and robust watermarks directly into images generated by Latent Diffusion Model (LDM).
 - **Generative model finetuning**: Finetune only the decoder of the generative model for detection and identification.



Stable Signature

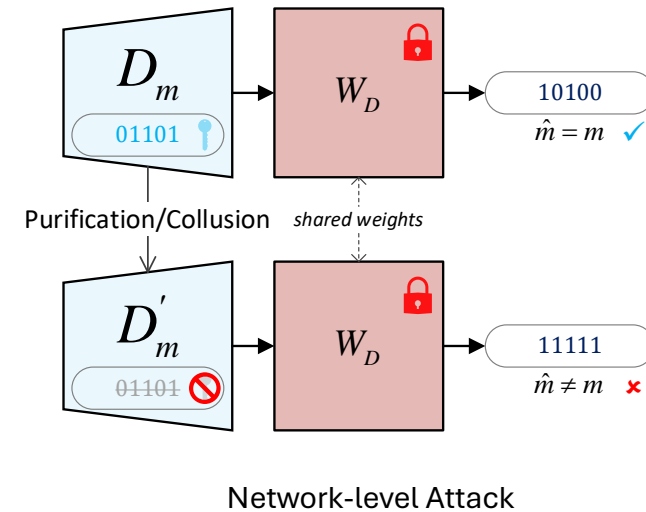
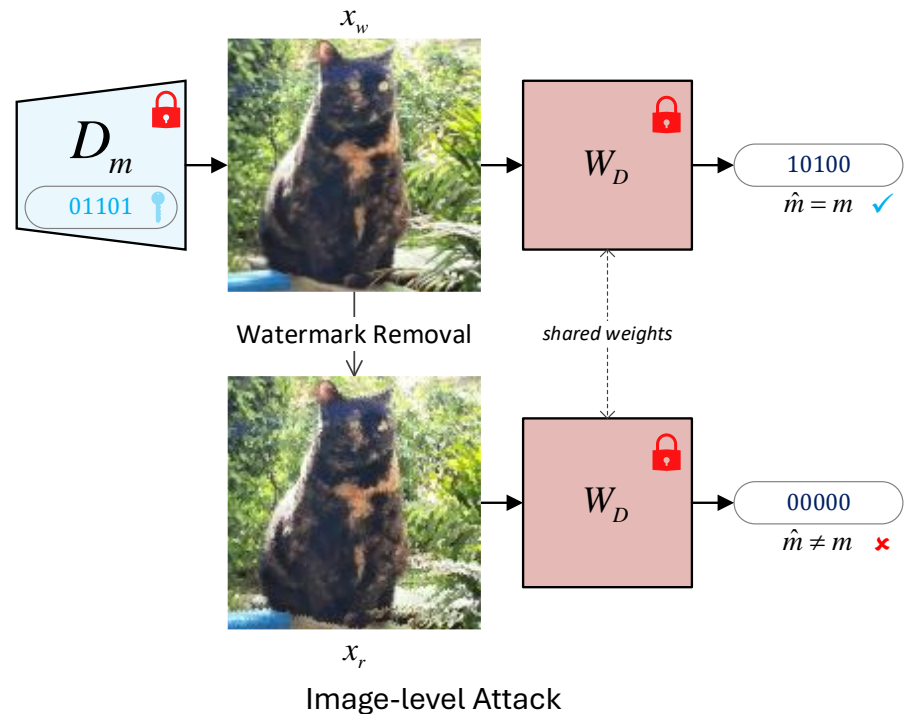
- The finetuned model can then be distributed and traced through a statistical test.



Stable Signature

Attacks:

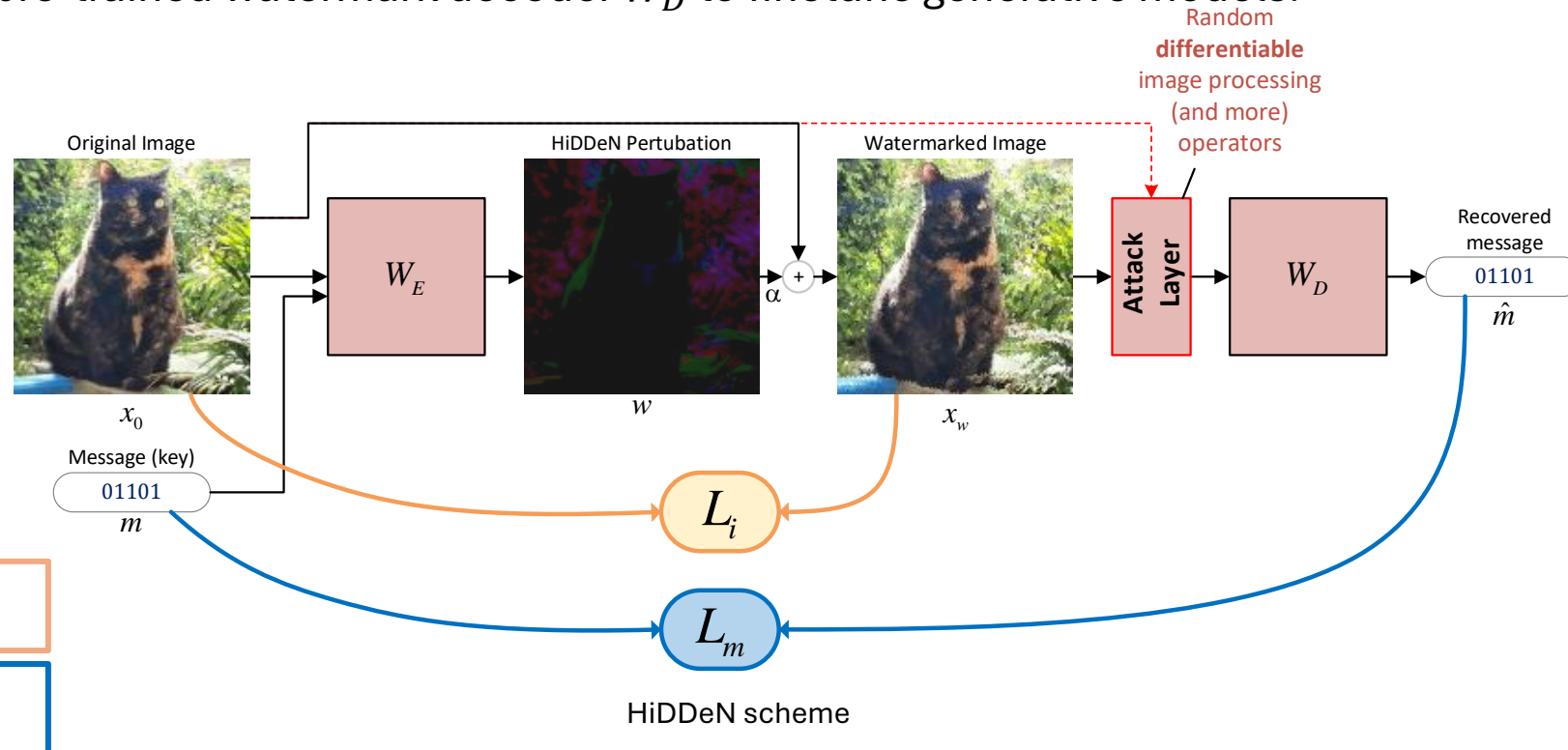
- Modifications that aim to remove the watermark, can be categorized into two families:
 - Image-targeted attack (*main focus of this project*)
 - Network-targeted attack



HiDDeN Method

Stable Signature relies heavily on an image watermarking method called HiDDeN [Zhu18]

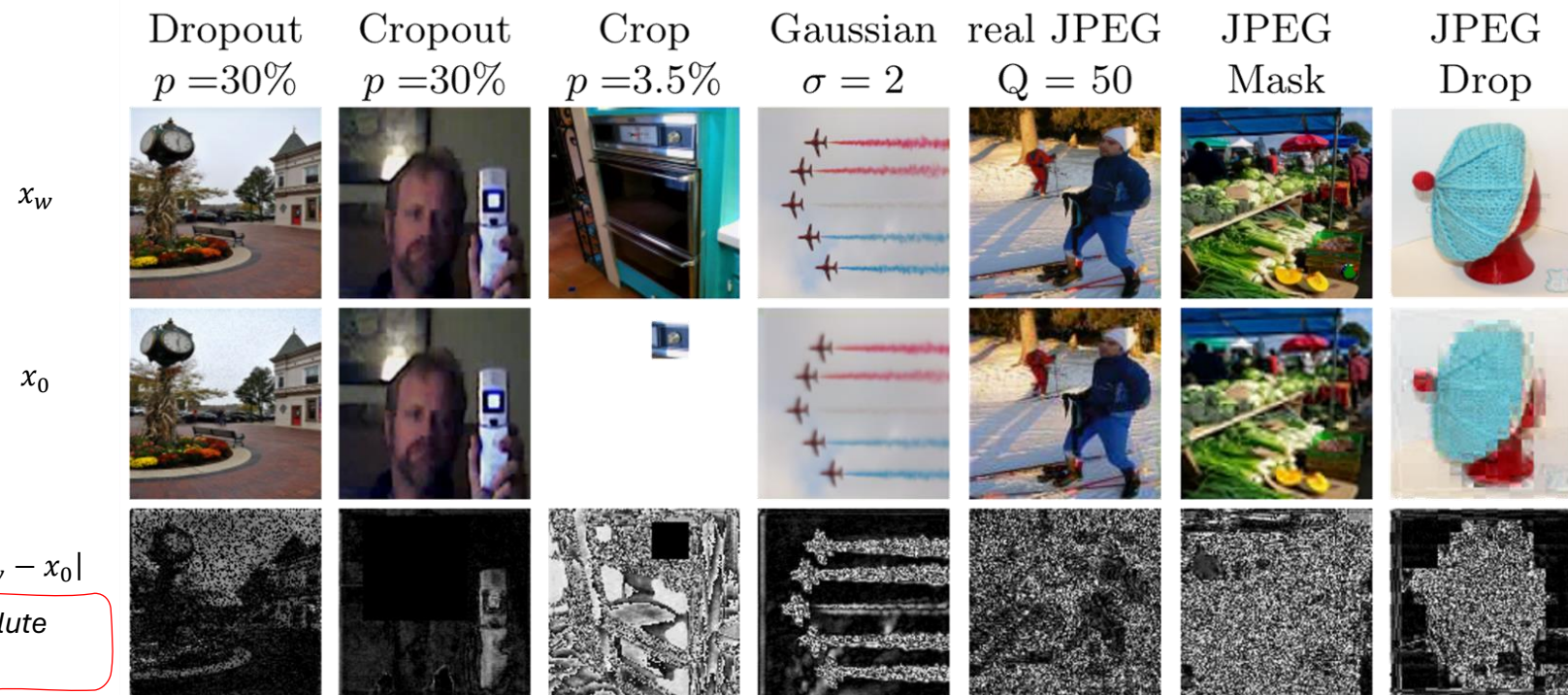
- **Main Idea:** learn a watermark encoder W_E that can hide messages in images while keeping them invisible. The message can be recovered by a watermark decoder W_D .
- Use the pre-trained watermark decoder W_D to finetune generative models.



HiDDeN Method

Attack Layer:

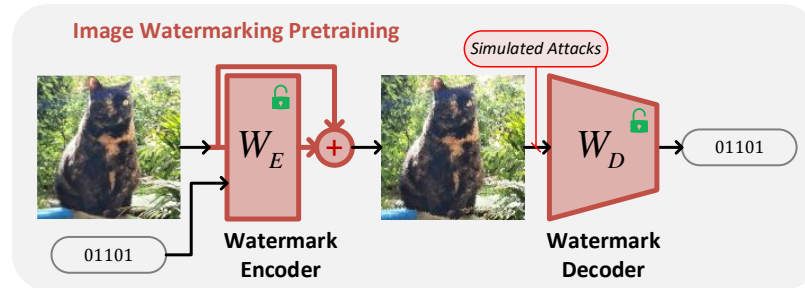
- Simulate differentiable image processing operations.



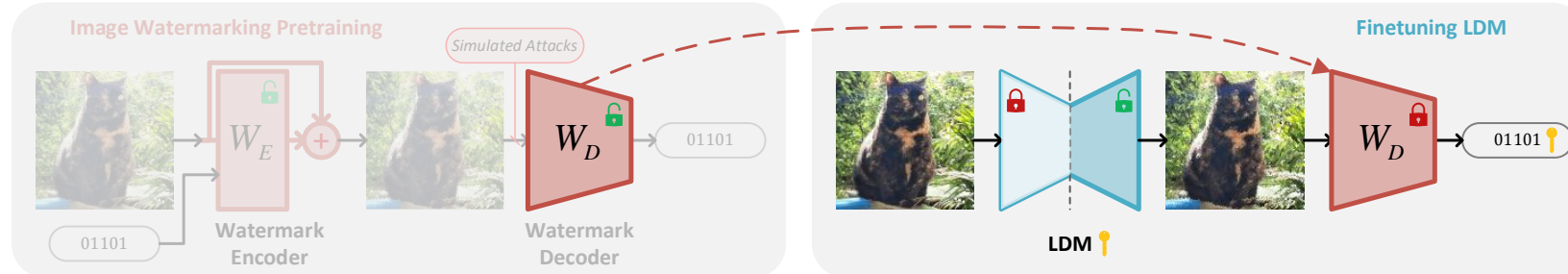
Differentiable Attacks used in the HiDDeN original paper

Stable Signature - Recap

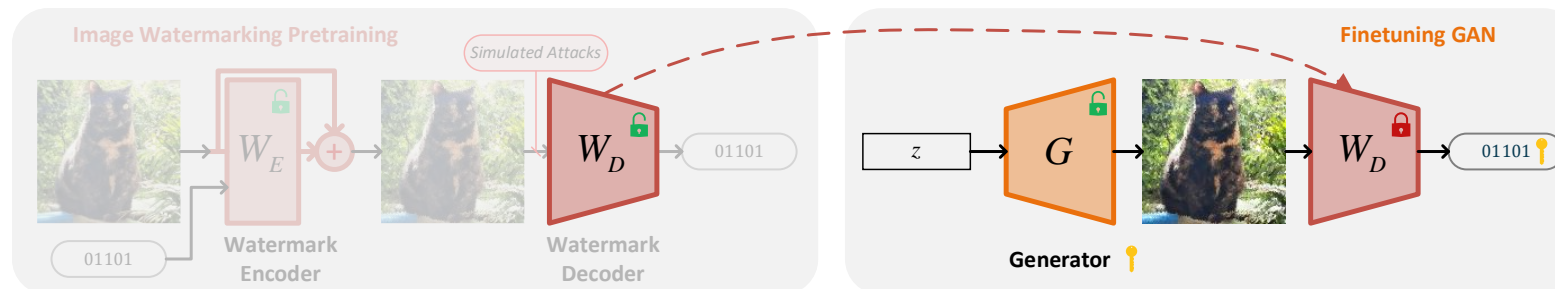
- **Stable Signature - Phase 1:** Pretrain HiDDeN



- **Stable Signature - Phase 2:** Finetune LDM [Rombach22]

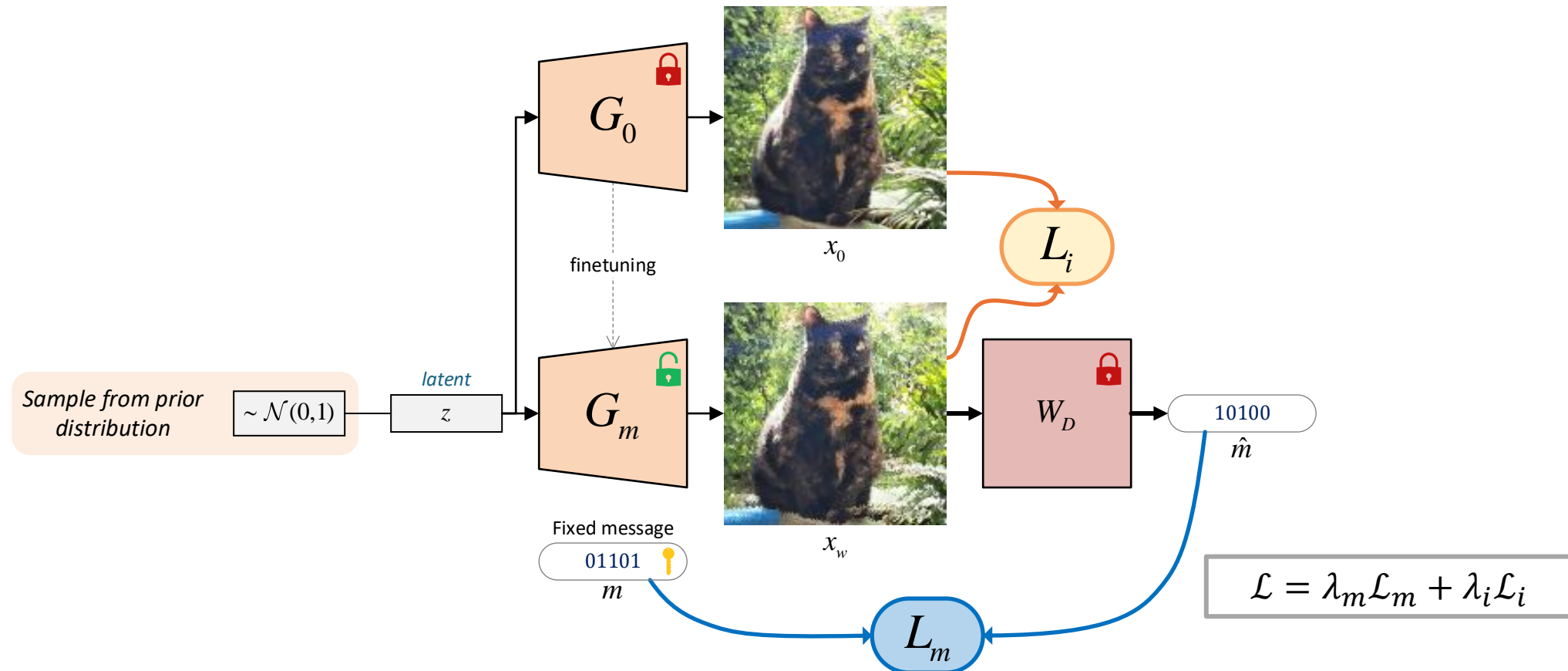


- **Our Phase 2:** Finetune a small GAN



Stable Signature for GAN

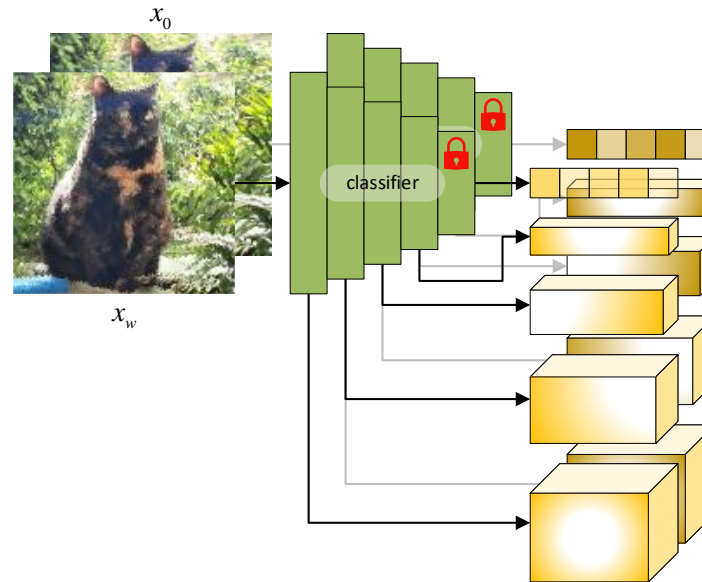
- Use a pretrained GAN model G and a Watermark Decoder W_D to finetune it:



Stable Signature for GAN

Loss Functions:

- **Message Loss:** Binary Cross Entropy (BCE)
 - $\mathcal{L}_m = \text{BCE}(m, \hat{m})$
- **Image Loss:** Perceptual Loss (Watson-VGG [Czolbe20])
 - Utilizes a pre-trained VGG network to extract meaningful feature representations.
 - Compares images in the feature space to ensure semantic similarity.

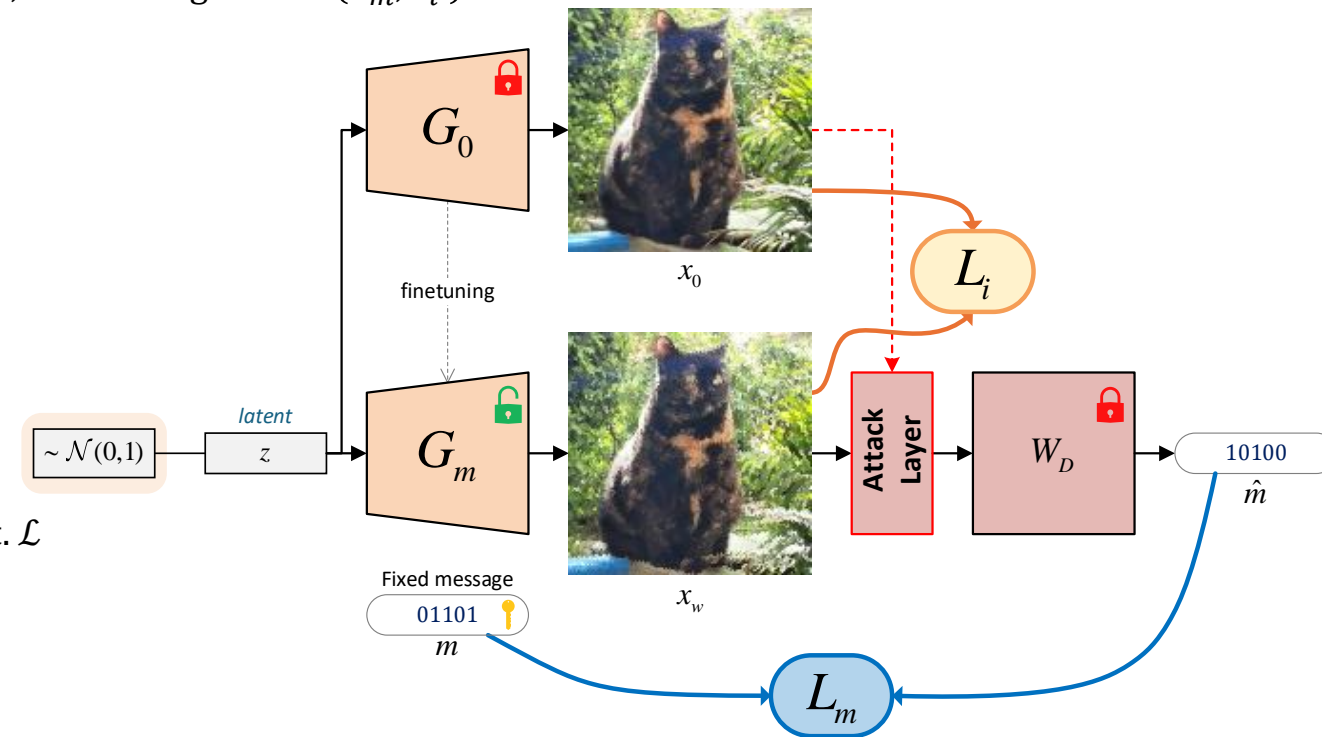


Watson-VGG compares multi-level features

Stable Signature for GAN

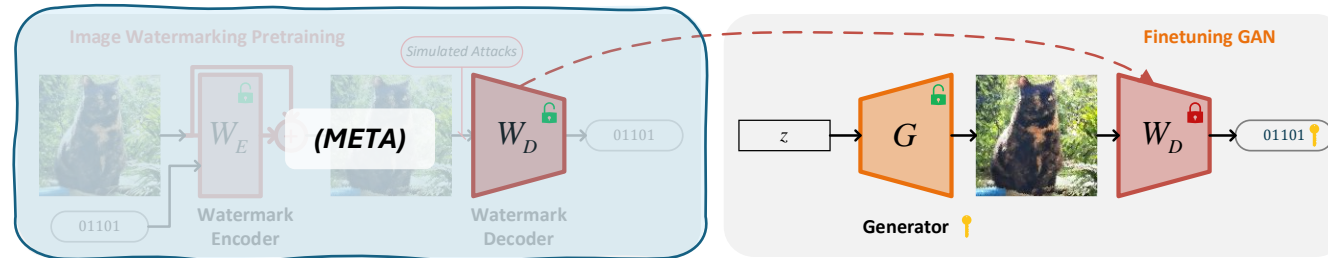
Finetuning Procedure for GAN (DCGAN [Radford15]):

- Input:
 - generator G_0 , watermark decoder W_D , message m , loss scaling factors (λ_m, λ_i)
 - attack_layer**
- $G_m = \text{clone}(G_0)$
- for i in range(train_iterations):
 - Sample $z \sim \mathcal{N}(0,1)$
 - $x_0 = G_0(z)$
 - $x_w = G_m(z)$
 - $x_r = \text{attack_layer}(x_w)$**
 - $\hat{m} = W_D(x_r)$
 - $\mathcal{L} = \lambda_m \mathcal{L}_m(\hat{m}, m) + \lambda_i \mathcal{L}_i(x_w, x_0)$
 - Compute gradient and update weights of G_m w.r.t. \mathcal{L}
- Return: G_m

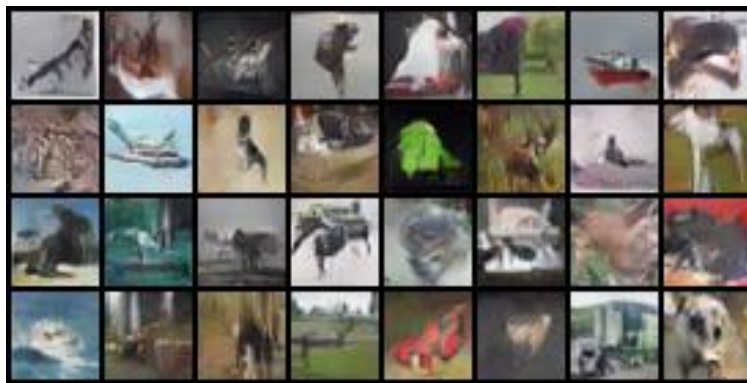


Stable Signature for GAN

- A straight forward implementation is to use the watermark decoder W_D published by the authors (META) to finetune the generator G .



- However, from our first results, we observed clear patterns of artifacts.



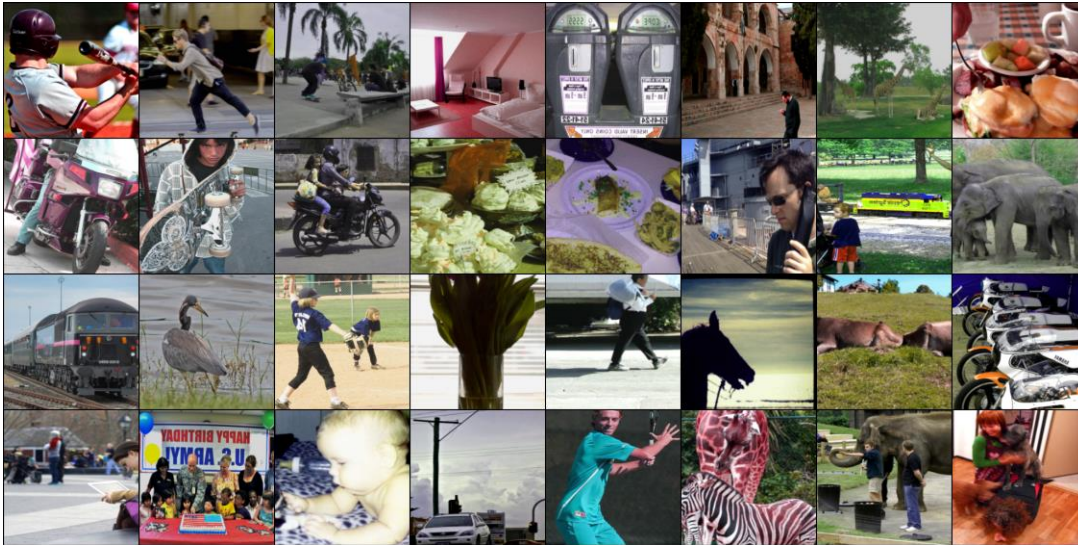
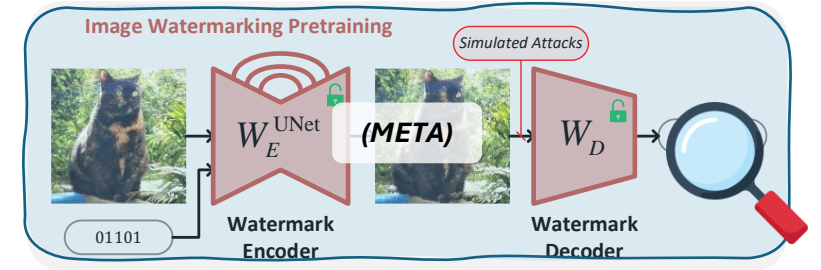
x_0



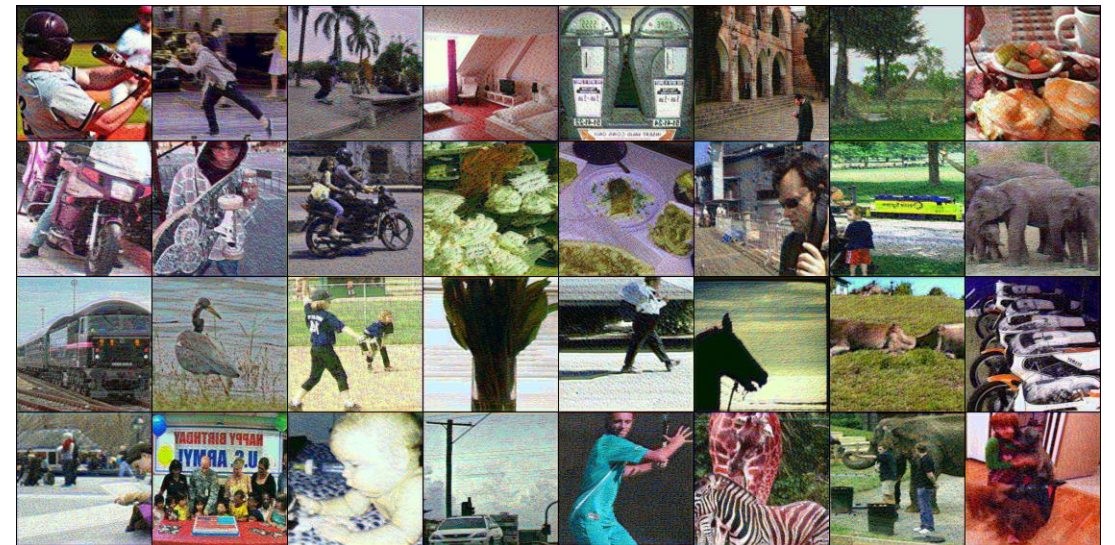
x_w

Problem Investigation: Potential pitfall of the HiDDeN-based Watermark

Multi-resolution Analysis of META's pretrained HiDDeN

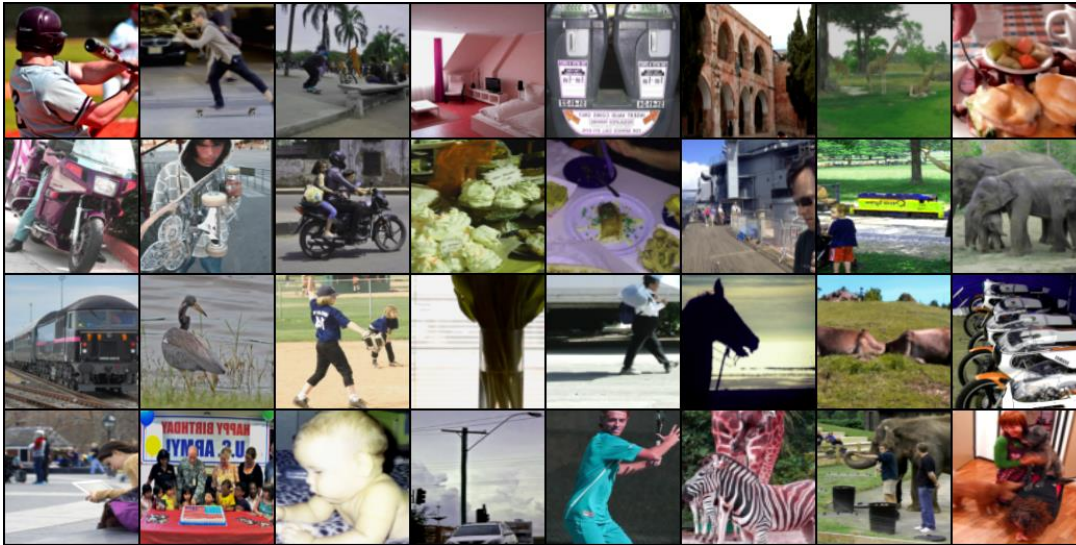
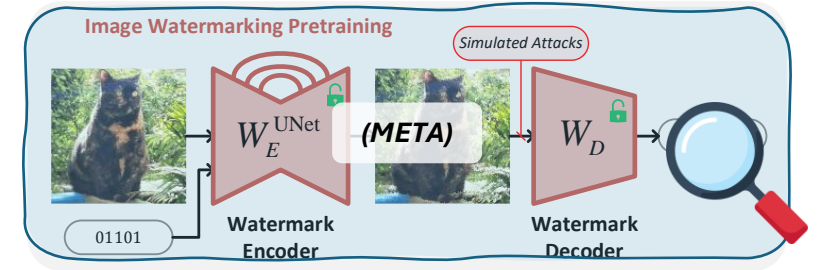


COCO images (256 × 256)



COCO images (256 × 256) watermarked by author's pre-trained HiDDeN
PSNR: 18.291

Multi-resolution Analysis of META's pretrained HiDDeN

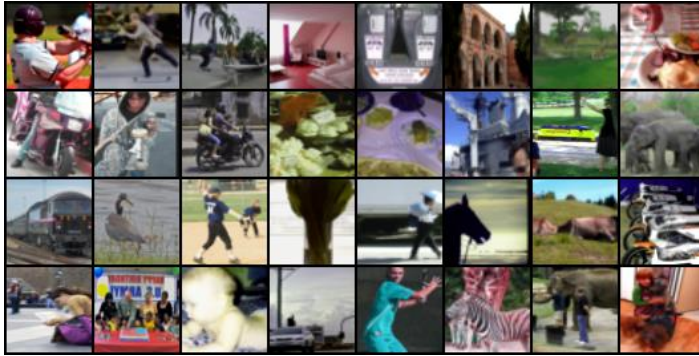
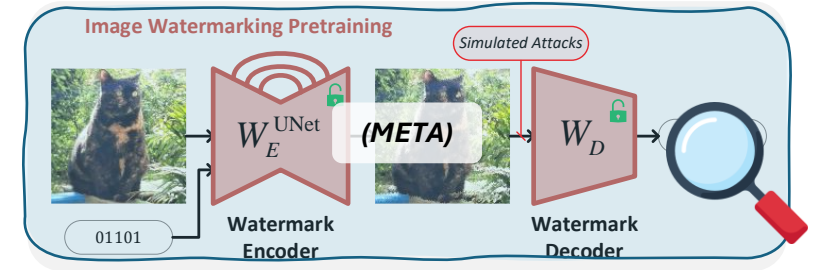


COCO images (128 × 128)



COCO images (128 × 128) watermarked by author's pre-trained HiDDeN
PSNR: 18.116

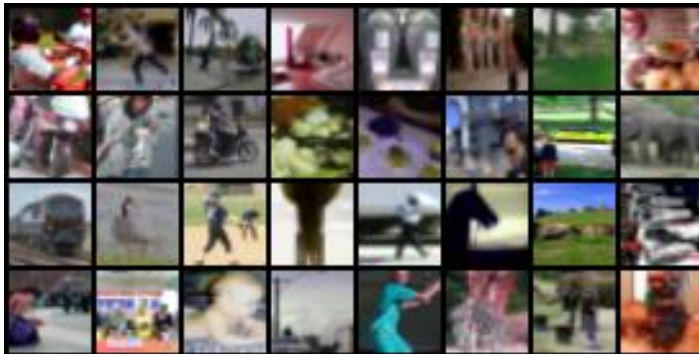
Multi-resolution Analysis of META's pretrained HiDDeN



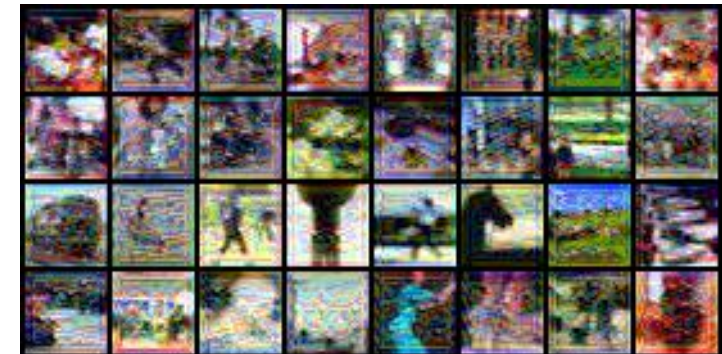
COCO image (64×64)



COCO image (64×64) watermarked by author's pre-trained HiDDeN
PSNR: 17.873



COCO image (32×32)



COCO image (32×32) watermarked by author's pre-trained HiDDeN
PSNR: 17.646

The smaller image size, the more visible the artifacts

If we use the META's Watermark Decoder (W_D) to guide the finetuning of GAN, it will likely mimic these artifacts

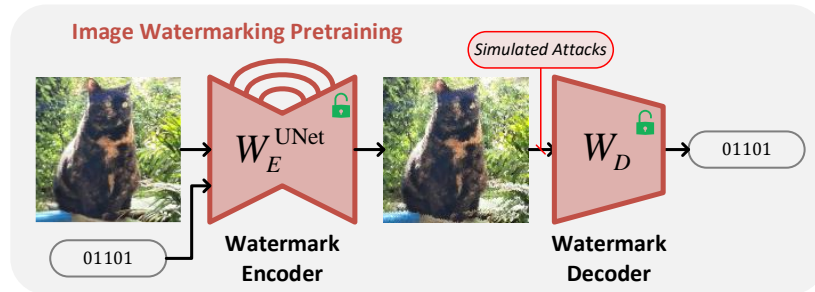
Observed Limitations

Observations:

- The HiDDeN model used in Stable Signature performs poorly on small-sized images.
- We hypothesize that W_D forces the GAN to mimic W_E .
- Autoencoder/U-Net-based architectures are widely used in deep steganography/image compression

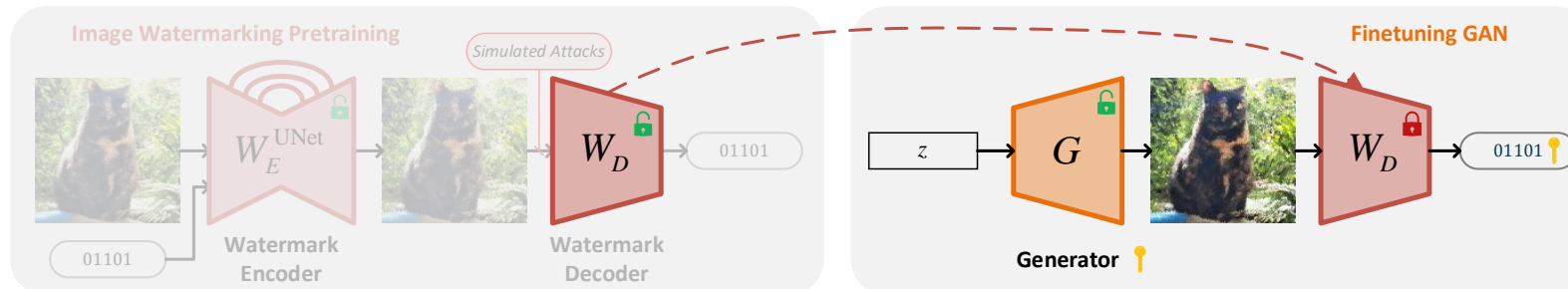
Our proposed scheme:

- **Our Phase 1:** Pretrain U-Net-based W_E



Note that W_D 's architecture is untouched!

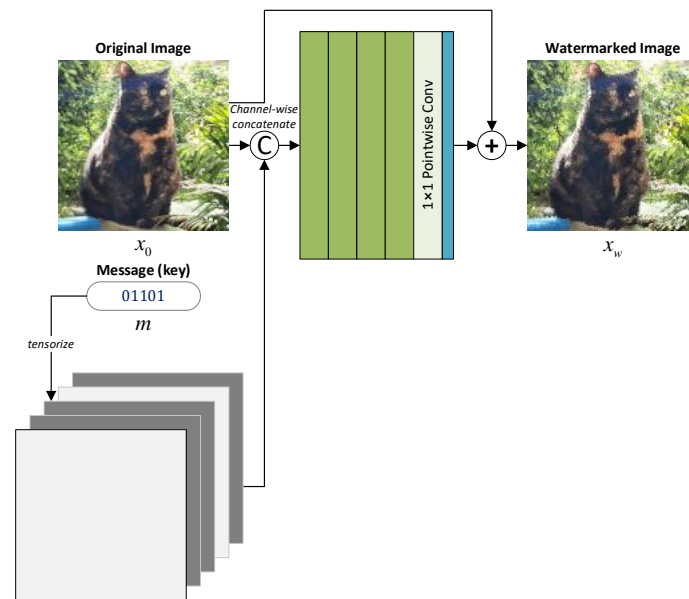
- **Our Phase 2:** Finetune G



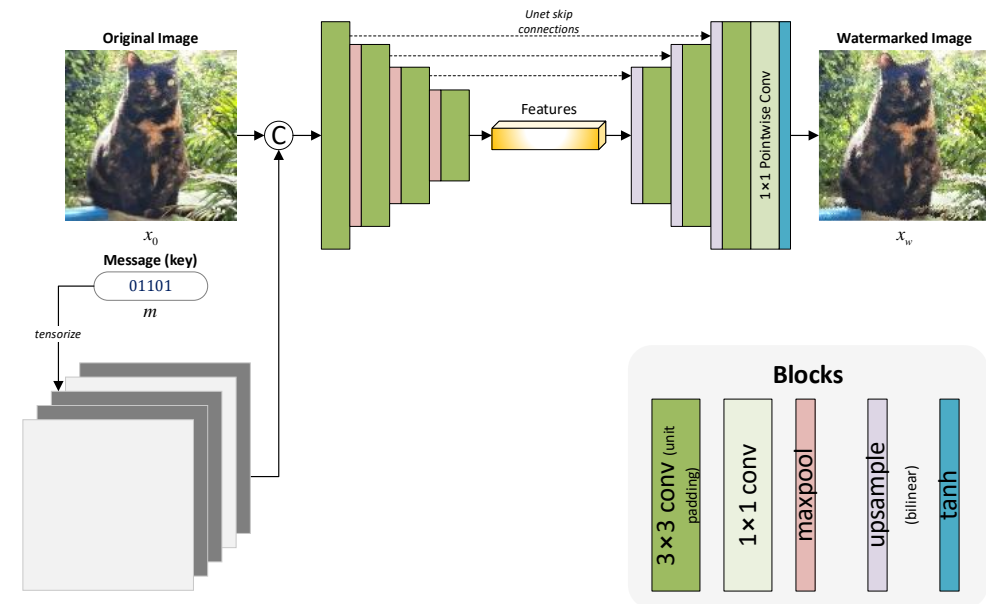
U-Net Architecture

Motivation: Instead of spatial-domain watermark, we wish to insert feature-level watermarks

- HiDDeN uses a const-features-size CNN watermark generator.
- Autoencoder-like architectures have been widely adopted in high-capacity image steganography [Baluja17] as they can capture high-level semantics.



HiDDeN's watermark encoder architecture



Our U-Net-based watermark encoder architecture

Experimental Details

Datasets:

- MNIST: image size of $1 \times 28 \times 28$, 32 bits msg
- CIFAR-10: image size of $3 \times 32 \times 32$, 48 bits msg
- 1 common key for all GAN models

Metrics:

- *Imperceptibility*:
 - PSNR \uparrow
 - SSIM \uparrow
 - LPIPS \downarrow [Zhang18] (for CIFAR-10 only)
- *Robustness*:
 - Bit accuracy \uparrow

Simulated Attacks:




































- These are differentiable attacks in the attack layer.
- Applied randomly one of these attacks.
- Random crop is not used for GAN finetuning because we always know the output resolution of the GAN.
- Differentiable JPEG is implemented as in [Zhang21].



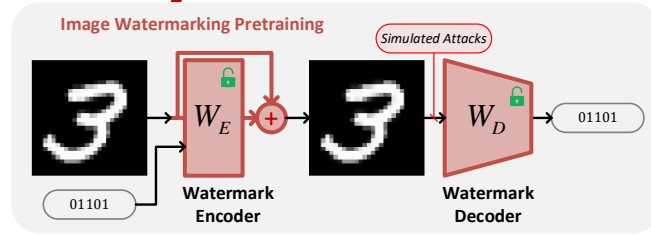
Experimental Details

Evaluation Attacks:

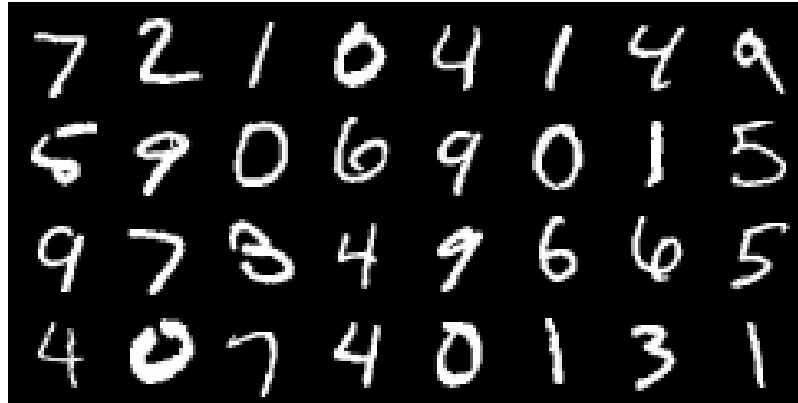
- Evaluate the whole test set against each attack, and report average bit accuracy.
- 10 geometrical attacks
- 15 image processing operators (*saturation and hue are for RGB only*)
- 6 compression attacks
- 4 mathematical morphology (*for grayscale only*)

identity	crop_08	crop_05	rotate left 25 degrees	rotate right 25 degrees	rotate left	rotate right	hflip	vflip	resize_08	resize_05
										
increase brightness	decrease brightness	increase contrast	decrease contrast	increase saturation	decrease saturation	increase hue	decrease hue	increase sharpness	decrease sharpness	blur
										
posterize 7	posterize 6	posterize 5	autocontrast	jpeg QF=80	jpeg QF=50	jpeg2000 QF=80	jpeg2000 QF=50	webp QF=80	webp QF=50	
										
erosion	dilation	opening	closing							
										

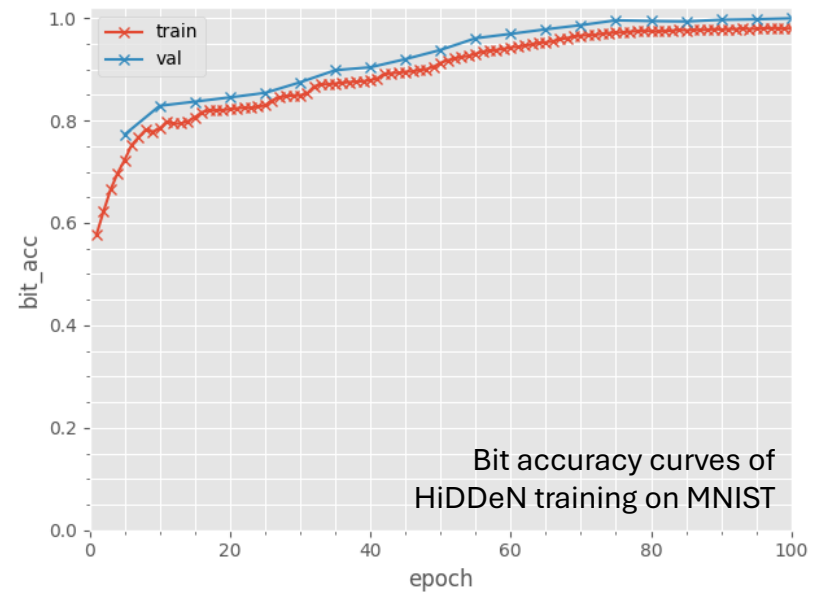
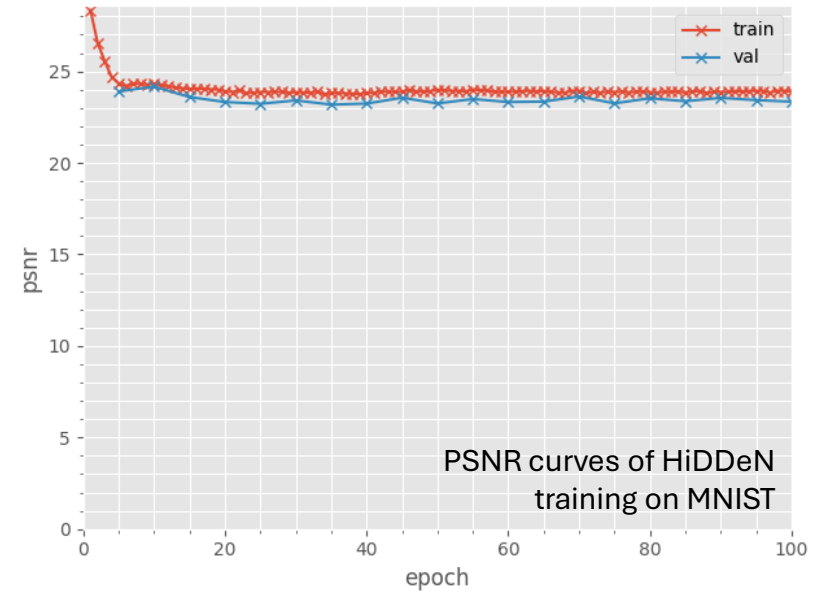
Results on MNIST (Phase 1)



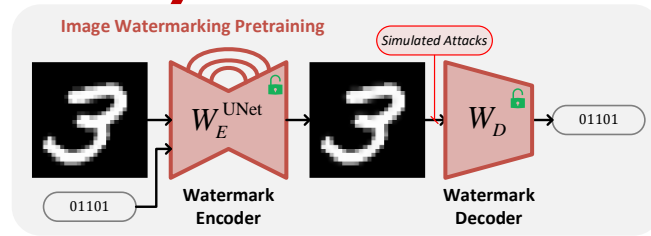
x_0



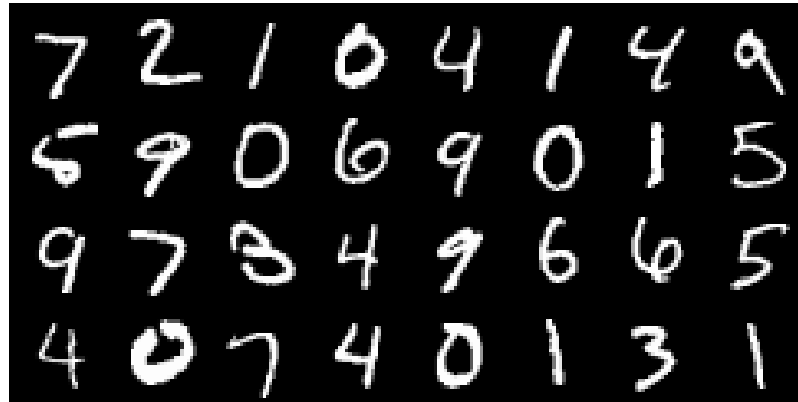
x_w



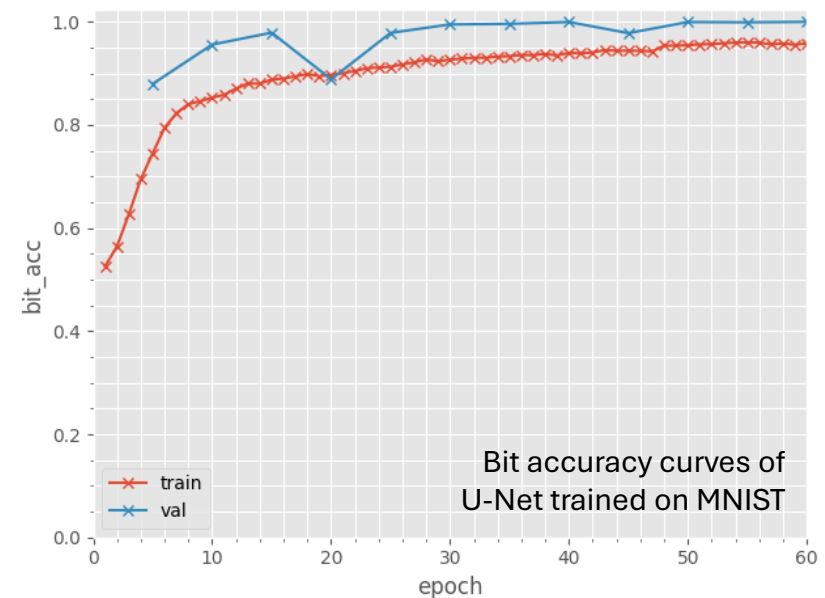
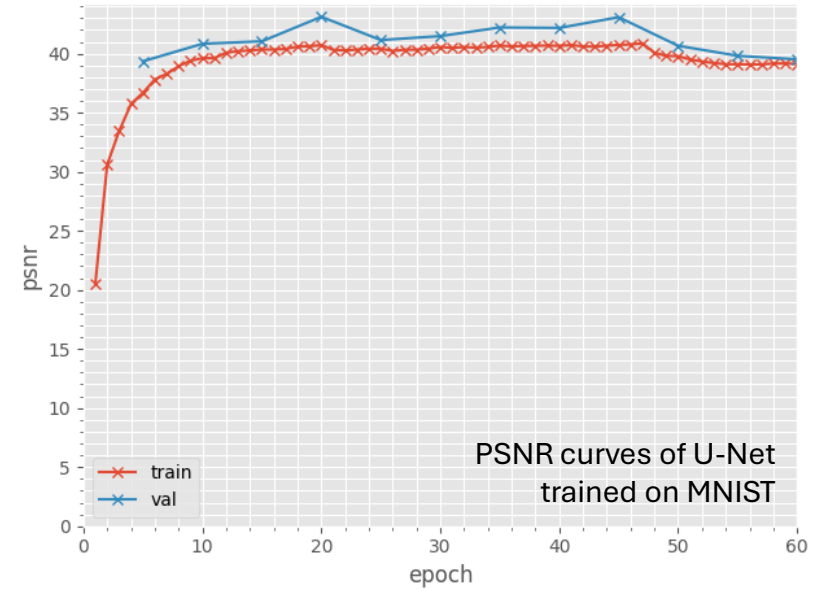
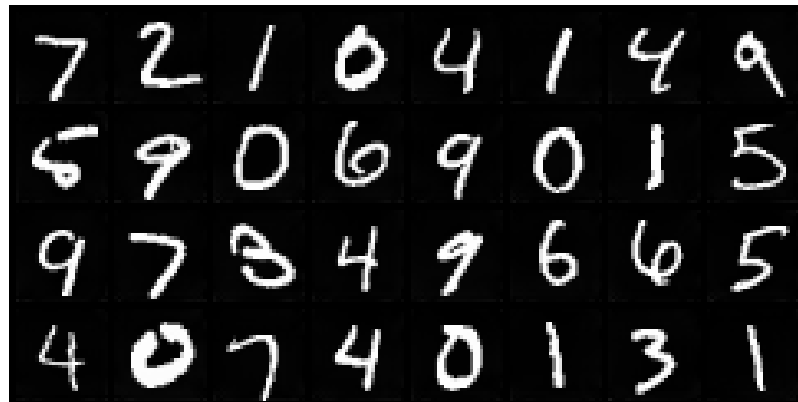
Results on MNIST (Phase 1)



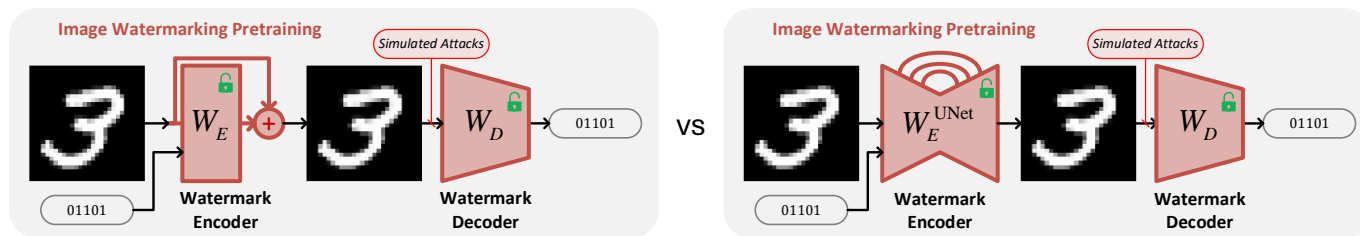
x_0



x_w



Results on MNIST (Phase 1)



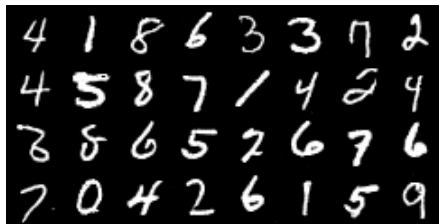
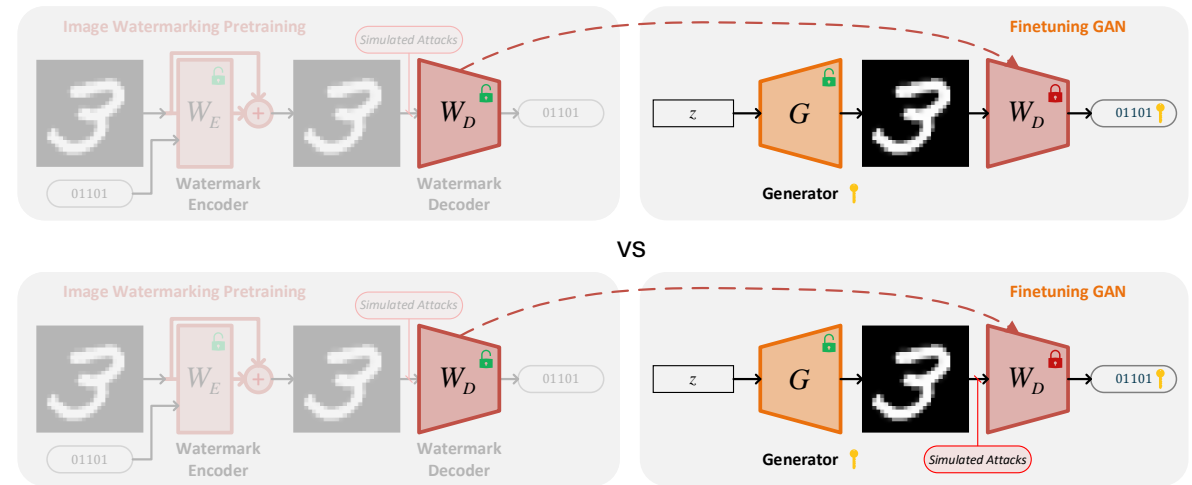
- Bit accuracy↑ per attacks:

Metrics	PSNR↑	SSIM↑
HiDDeN	23.462	0.819
U-Net	39.519	0.942

Attacks	identity	decrease brightness	increase brightness	decrease contrast	increase contrast	decrease sharpness	increase sharpness	blur	posterize 7	posterize 6	posterize 5	autocontrast	erosion	dilation	opening	closing
HiDDeN	0.9946	0.8494	0.8608	0.8623	0.8274	0.8998	0.8893	0.9769	0.5224	0.5203	0.5174	0.6787	0.5850	0.6401	0.6758	0.7409
U-Net	1.0000	0.9980	0.9999	0.7203	0.5736	0.9978	0.9947	0.9994	0.9466	0.7471	0.5891	1.0000	0.8259	0.8756	0.9213	0.9901
Attacks	crop 0.8	crop 0.5	resize 0.8	resize 0.5	rotate right 25 degrees	rotate left 25 degrees	rotate right	rotate left	horizontal flip	vertical flip	jpeg QF=80	jpeg QF=50	jpeg2000 QF=80	jpeg2000 QF=50	webp QF=80	webp QF=50
HiDDeN	0.7879	0.7079	0.7108	0.5380	0.8017	0.7958	0.6710	0.7732	0.9712	0.5936	0.5223	0.5204	0.5233	0.5233	0.5226	0.5211
U-Net	0.6241	0.5016	0.9080	0.5381	0.6333	0.6166	0.5684	0.5577	0.7923	0.5797	0.9071	0.7625	0.9995	0.9995	0.7545	0.6394

Results on MNIST (Phase 2)

- Phase 1: HiDDeN Image Watermarking
- Phase 2: DCGAN finetuning



x_0



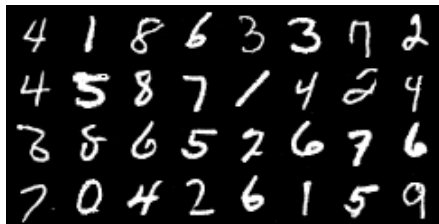
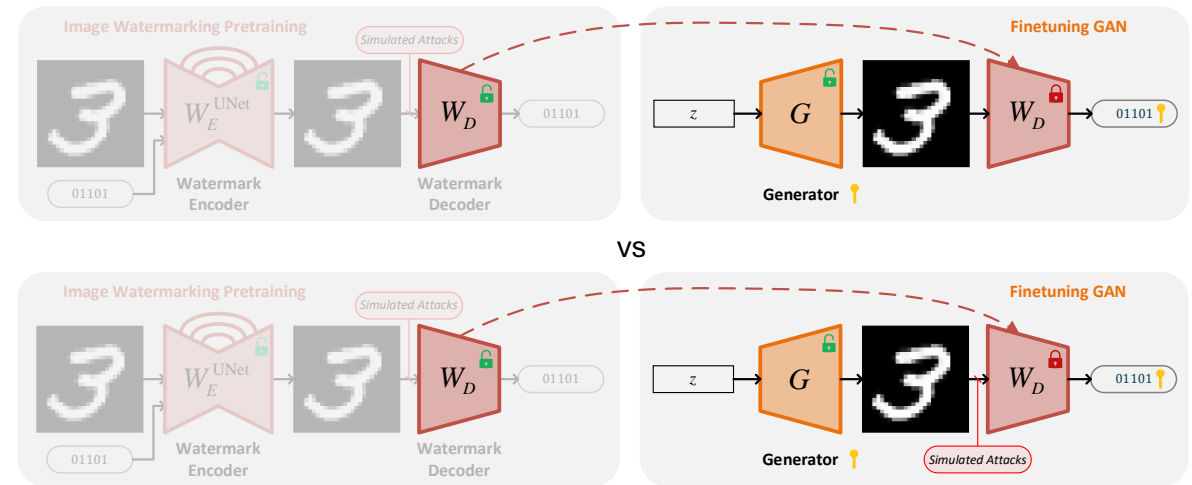
x_w generated by DCGAN



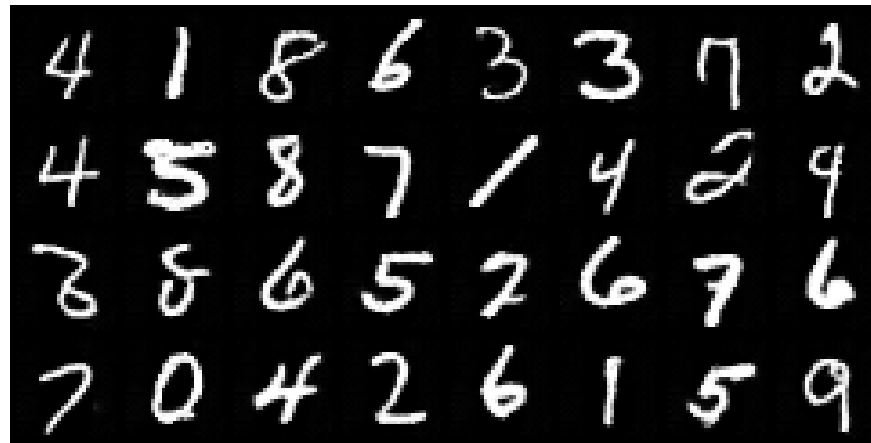
x_w generated by DCGAN + attack layer

Results on MNIST (Phase 2)

- Phase 1: U-Net Image Watermarking
- Phase 2: DCGAN finetuning



x_0



x_w generated by DCGAN



x_w generated by DCGAN + attack layer

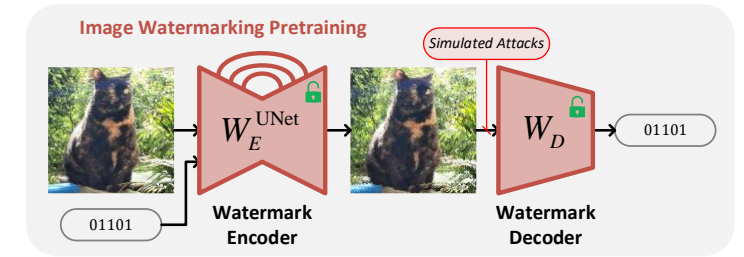
Results on MNIST

(Phase 2)

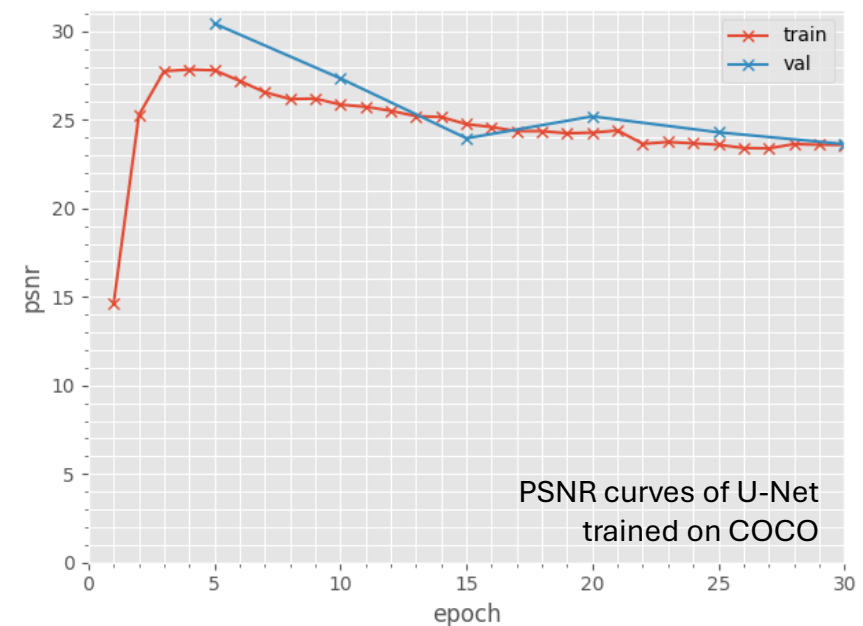
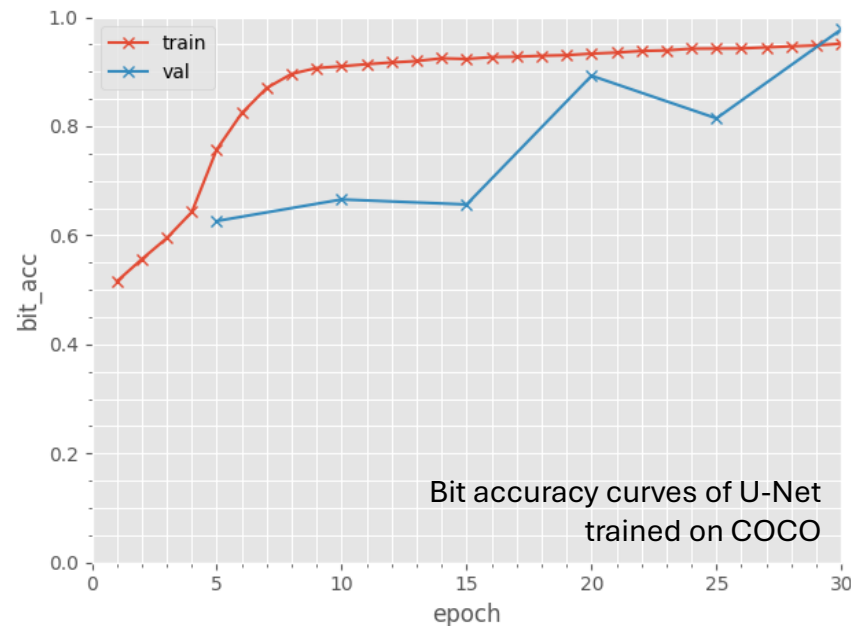
Metrics	PSNR↑	SSIM↑
HiDDeN Baseline	21.5415	0.8758
HiDDeN + Attack Layer	19.9810	0.8175
U-Net Baseline	27.1739	0.9566
U-Net + Attack Layer	23.6523	0.9106

Attacks	identity	decrease brightness	increase brightness	decrease contrast	increase contrast	decrease sharpness	increase sharpness	blur	posterize 7	posterize 6	posterize 5	autocontrast	erosion	dilation	opening	closing
HiDDen Baseline	0.9522	0.6723	0.8633	0.7077	0.6607	0.8014	0.9443	0.6851	0.8849	0.7868	0.7020	0.9522	0.4675	0.5798	0.5179	0.7080
HiDDen + Attacks	0.9429	0.7934	0.8766	0.7695	0.7803	0.9193	0.9398	0.8730	0.9360	0.9113	0.8326	0.9430	0.4630	0.5531	0.5627	0.7119
U-Net Baseline	0.9966	0.8705	0.9970	0.5836	0.4654	0.9365	0.9879	0.6877	0.5518	0.4742	0.4624	0.9966	0.4572	0.5740	0.4974	0.6576
U-Net + Attacks	0.9994	0.9933	0.9941	0.6724	0.6660	0.9946	0.9939	0.9969	0.9715	0.7722	0.6622	0.9994	0.6370	0.7274	0.9262	0.9220
Attacks	crop 0.8	crop 0.5	resize 0.8	resize 0.5	rotate right 25 degrees	rotate left 25 degrees	rotate right	rotate left	horizontal flip	vertical flip	jpeg QF=80	jpeg QF=50	jpeg2000 QF=80	jpeg2000 QF=50	webp QF=80	webp QF=50
HiDDen Baseline	0.6213	0.5380	0.5983	0.4304	0.5740	0.5520	0.5452	0.4390	0.6453	0.4716	0.6891	0.6123	0.9322	0.9322	0.7274	0.6378
HiDDen + Attacks	0.6495	0.5360	0.6478	0.4216	0.5886	0.5877	0.6430	0.5541	0.7841	0.5067	0.7907	0.6862	0.9415	0.9415	0.8363	0.7041
U-Net Baseline	0.4670	0.4741	0.4954	0.4065	0.5110	0.5672	0.5067	0.4887	0.5425	0.4789	0.4401	0.4738	0.7449	0.7449	0.4510	0.4475
U-Net + Attacks	0.5558	0.4758	0.9402	0.7594	0.6962	0.6963	0.9257	0.9228	0.9022	0.6075	0.9073	0.7183	0.9982	0.9982	0.7652	0.6607

Pretraining U-Net on COCO (Phase 1)

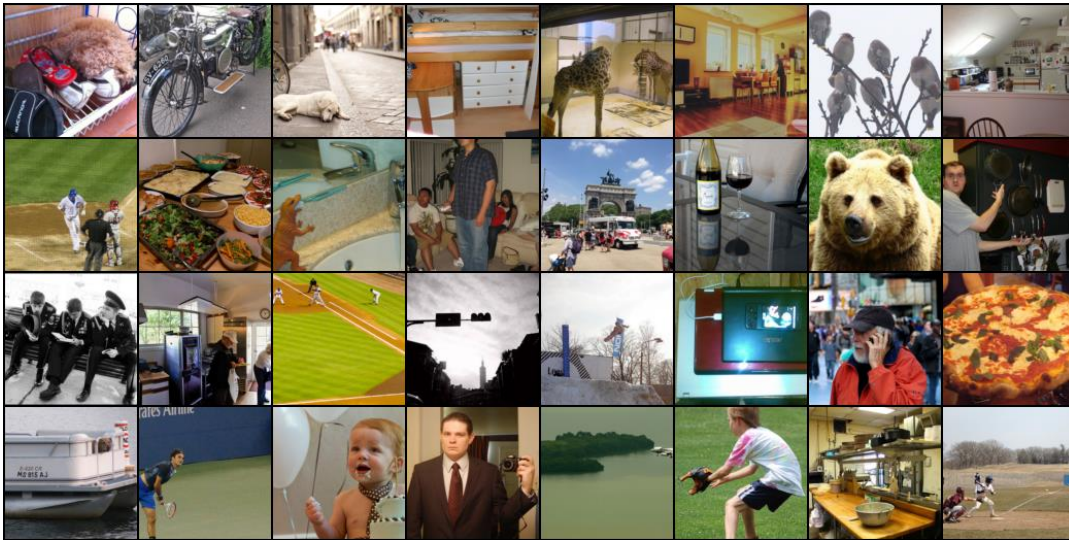
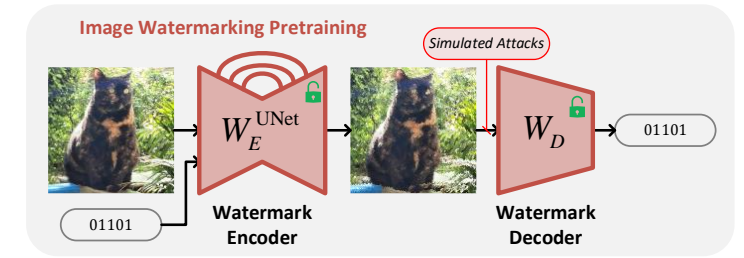


- From our experiments, training a U-Net watermarking network on low-resolution RGB datasets such as CIFAR-10 is extremely hard (bit accuracy does not improve).
- Instead, we opt to use a larger dataset, i.e. **COCO 2014**.
 - All COCO images are resized to 128×128
 - W_D is finetuned from META's published weights



Pretraining U-Net on COCO

(Phase 1)



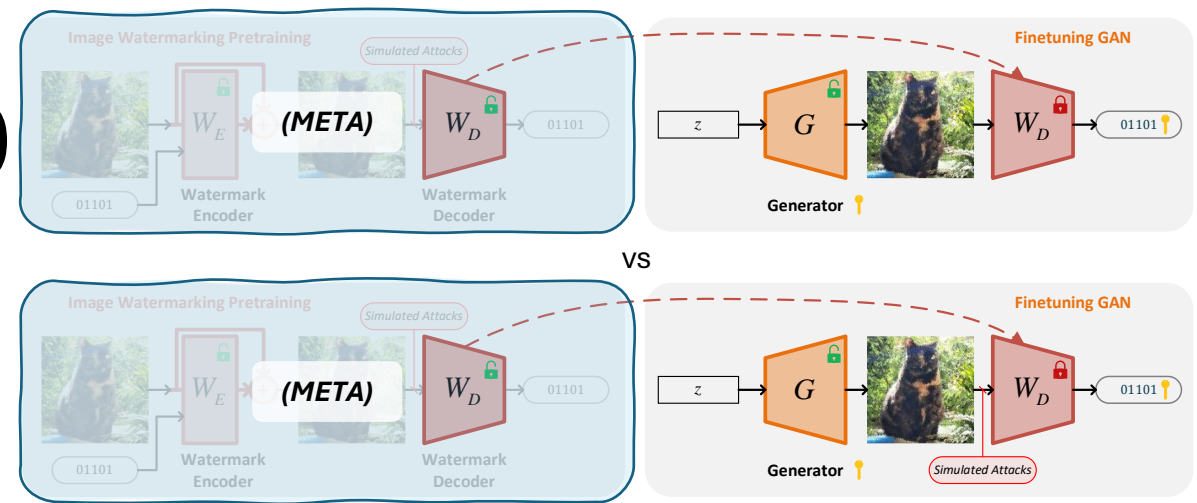
x_0



x_w

Results on CIFAR-10 (Phase 2)

- Phase 1: HiDDeN Image Watermarking (META)
- Phase 2: DCGAN finetuning



x_0



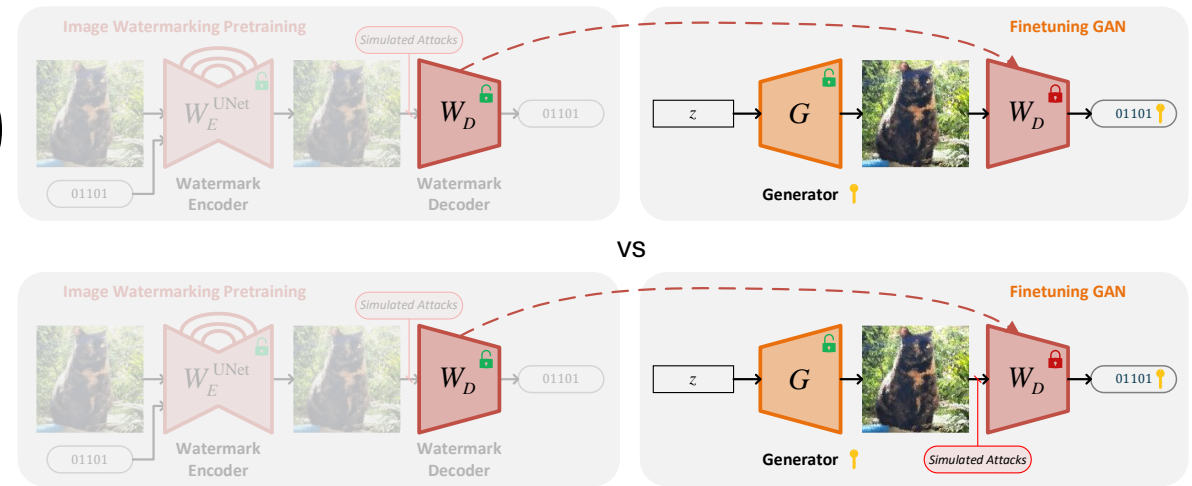
x_w generated by DCGAN



x_w generated by DCGAN + attack layer

Results on CIFAR-10 (Phase 2)

- Phase 1: U-Net Image Watermarking
- Phase 2: DCGAN finetuning



x_0



x_w generated by DCGAN



x_w generated by DCGAN + attack layer

Results on MNIST

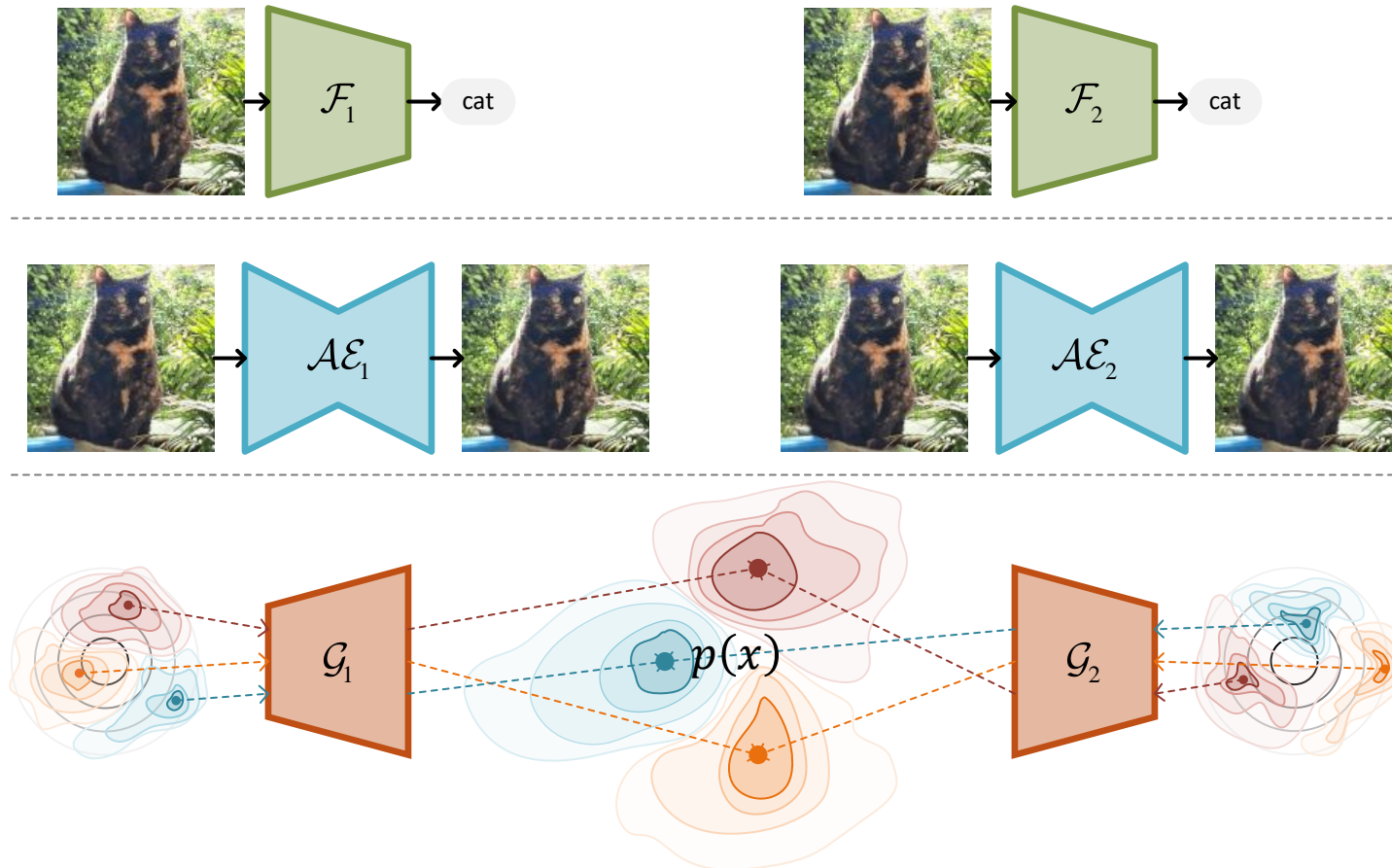
(Phase 2)

Metrics	PSNR↑	SSIM↑	LPIPS↓
HiDDeN Baseline	21.7667	0.6661	0.0090
HiDDeN + Attack Layer	13.8930	0.1958	0.1070
U-Net Baseline	23.0573	0.7436	0.0098
U-Net + Attack Layer	15.1636	0.3801	0.0327

Attacks	identity	decrease brightness	increase brightness	decrease contrast	increase contrast	decrease saturation	increase saturation	decrease hue	increase hue	decrease sharpness	increase sharpness	blur	posterize 7	posterize 6	posterize 5	autocontrast
HiDDen Baseline	0.9803	0.9276	0.9444	0.9242	0.9622	0.8842	0.9646	0.7047	0.6906	0.9416	0.9699	0.6963	0.9802	0.9797	0.9776	0.9578
HiDDen + Attacks	0.9977	0.9960	0.9939	0.9960	0.9948	0.9969	0.9901	0.8477	0.7941	0.9984	0.9950	0.9884	0.9977	0.9977	0.9976	0.9982
U-Net Baseline	0.9459	0.8990	0.8983	0.8971	0.9221	0.8892	0.9298	0.6515	0.6161	0.9194	0.9331	0.6985	0.9455	0.9443	0.9377	0.9030
U-Net + Attacks	0.8922	0.8354	0.8308	0.8196	0.8285	0.8403	0.8474	0.6691	0.6573	0.8487	0.8619	0.6549	0.8923	0.8920	0.8900	0.8798
Attacks	crop 0.8	crop 0.5	resize 0.8	resize 0.5	rotate right 25 degrees	rotate left 25 degrees	rotate right	rotate left	horizontal flip	vertical flip	jpeg QF=80	jpeg QF=50	jpeg2000 QF=80	jpeg2000 QF=50	webp QF=80	webp QF=50
HiDDen Baseline	0.6539	0.6145	0.5014	0.4774	0.5210	0.5194	0.5645	0.5594	0.6295	0.6564	0.4928	0.4912	0.9802	0.9802	0.4953	0.4925
HiDDen + Attacks	0.8927	0.8791	0.8859	0.5940	0.6505	0.6597	0.8815	0.9162	0.9127	0.5826	0.9608	0.8396	0.9977	0.9977	0.9525	0.9289
U-Net Baseline	0.5773	0.5944	0.5621	0.5365	0.5420	0.5089	0.5670	0.5841	0.6030	0.8115	0.5401	0.5372	0.9457	0.9457	0.5352	0.5288
U-Net + Attacks	0.6116	0.5472	0.5581	0.5072	0.5747	0.5980	0.6419	0.6269	0.7361	0.6258	0.6335	0.6211	0.8923	0.8923	0.6351	0.6236

Future Directions

The Adversarial scheme: Discussion on “*imperceptibility*” in Neural Network Watermarking for GANs

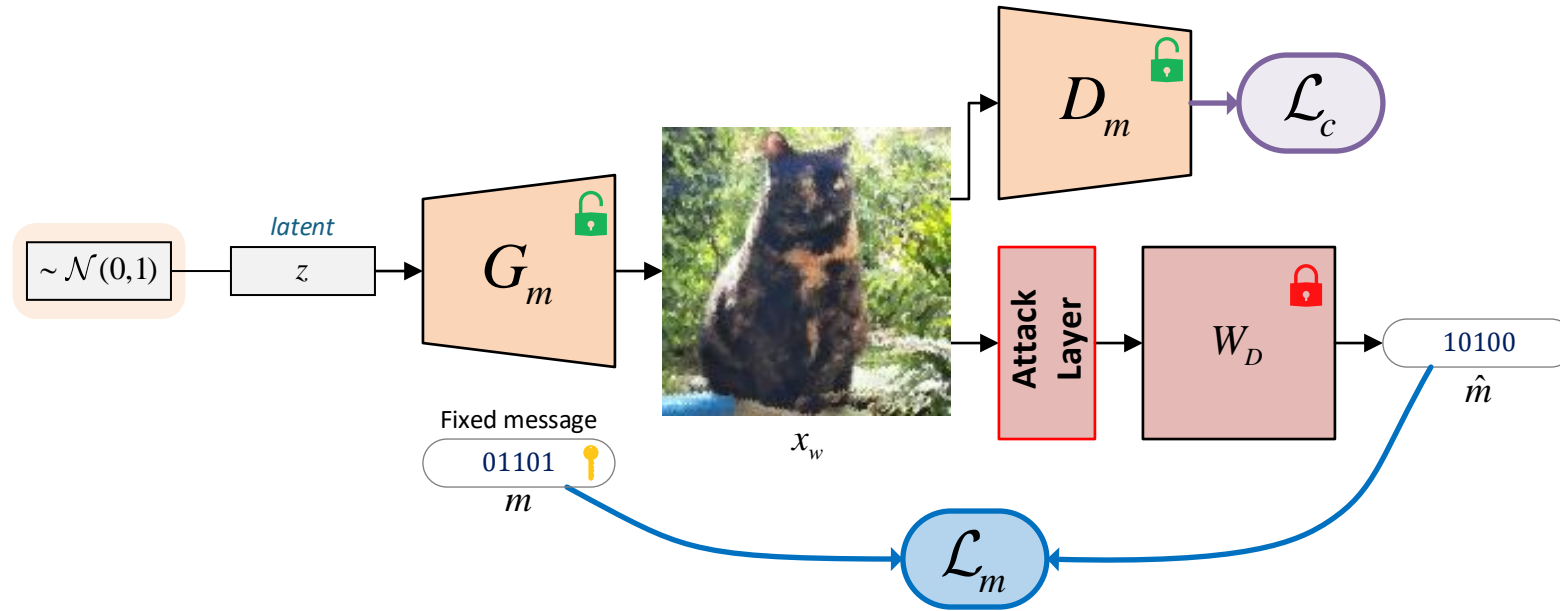


Since GAN tries to learn to the dataset distribution $p(x)$, if given the same normal distribution, two GAN models can both output $p(x)$, should they be considered to have equal performance?

Future Directions

The Adversarial scheme:

- If we loosen the constraint that the finetuned G_m must output images similar to G_0 given the same noise vector, we can define a new “adversarial scheme”:



Adversarial GAN finetuning scheme

Why is it potentially interesting?

- More flexible optimization.
- Allow finetuned models to diverge completely, which could mean it is harder to do collusion attack.

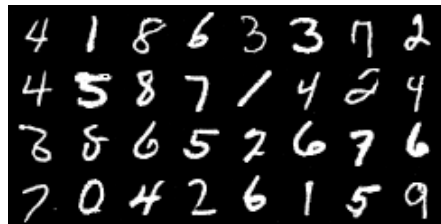
- where \mathcal{L}_c is the critic loss used to train both the generator and the discriminator.

Future Directions

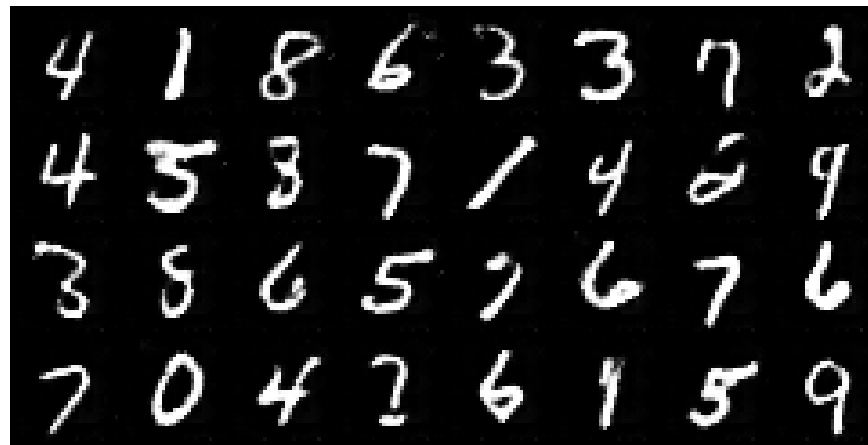
Results (Phase 2)

The Adversarial scheme:

- Demonstration of feasibility:
 - We trained two GAN models using the Adversarial scheme on MNIST



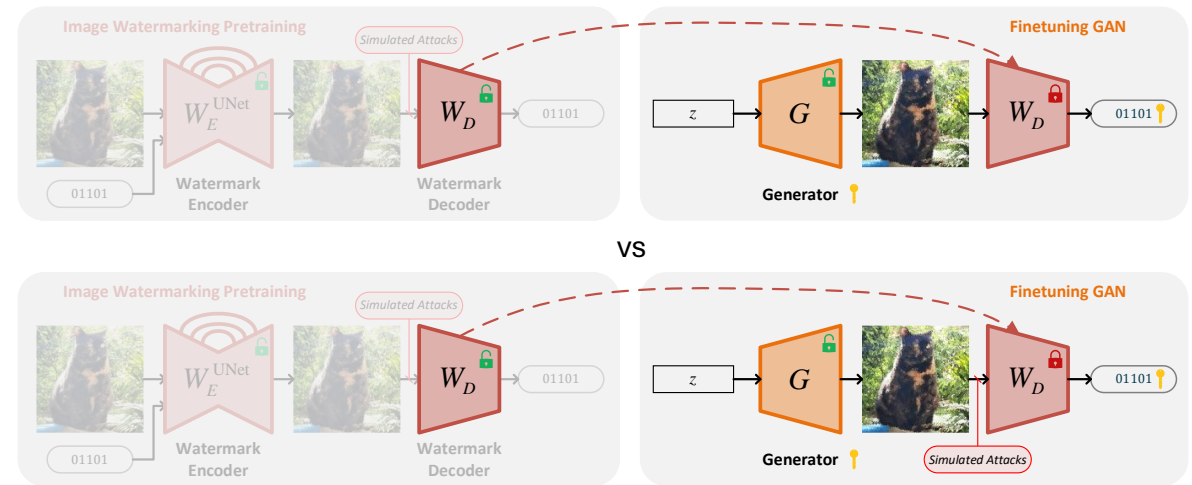
x_0



x_w generated by DCGAN

Metrics	PSNR↑	SSIM↑
U-Net Baseline	15.8509	0.6865
U-Net + Attack Layer	23.3677	0.9089

while we do report distortion metrics, note that they are **NOT** relevant in this scheme!



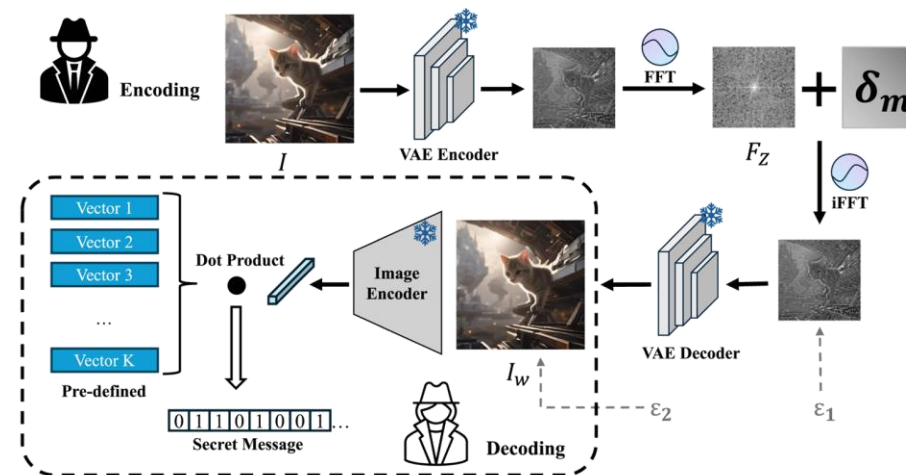
x_w generated by DCGAN + attack layer

Attacks	identity	image processing	geometric attacks	compression	mathematical morphology
U-Net Baseline	0.9748	0.6890	0.5027	0.5450	0.5830
U-Net + Attacks	0.9976	0.8685	0.7273	0.8433	0.7831

Future Directions

Other future works:

- Experiments:
 - Increase number of models and keys.
 - Subjective test.
 - Test network-level attacks.
- Study techniques to improve robustness against network-level attacks (Phase 2), e.g. [Hu24].
- Investigate more recent AE-based image watermarking methods (Phase 1), e.g. FreqMark [Guo24], and apply them for finetuning GAN.



FreqMark combines VAE architecture and FFT

Thank you for listening!