Robustness of AI-Generated Image Detectors

Adam Kosinski & Anika Mitra

February 12, 2025

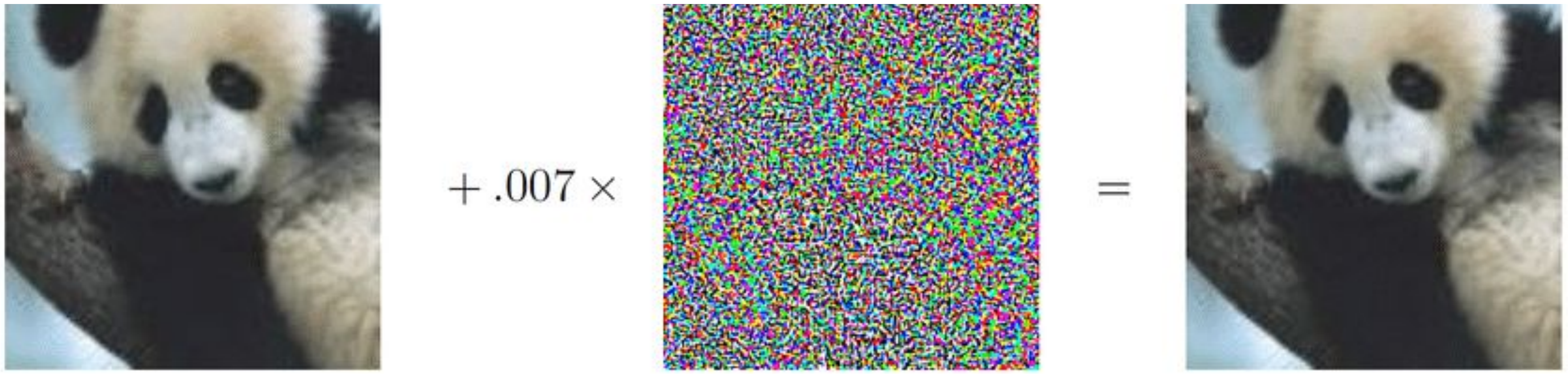
Outline

Motivation/Context

- Classifier neural networks are being used in security-critical systems that should be robust
- Adversaries try to fool the network with carefully chosen input (very small changes to the input)
- Robust classifiers need to protect against adversaries

How can we train deep neural networks that are robust to adversarial input?

Adversarial Attack: Fast Gradient Sign Method vs. GoogLeNet



$$\boldsymbol{x} + .007 \times \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)) = \boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$$

$\boldsymbol{x}$
"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence

$\boldsymbol{x} +$
$\epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

"Explaining and Harnessing Adversarial Examples" - Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy

Adversarial Attacks

- Random perturbation (e.g. Gaussian Noise)
- Image compression (e.g. JPEG, resizing/blurring)
- Fast Gradient Sign Method
- Projected Gradient Descent

Usually we assume perturbations should have a limited magnitude

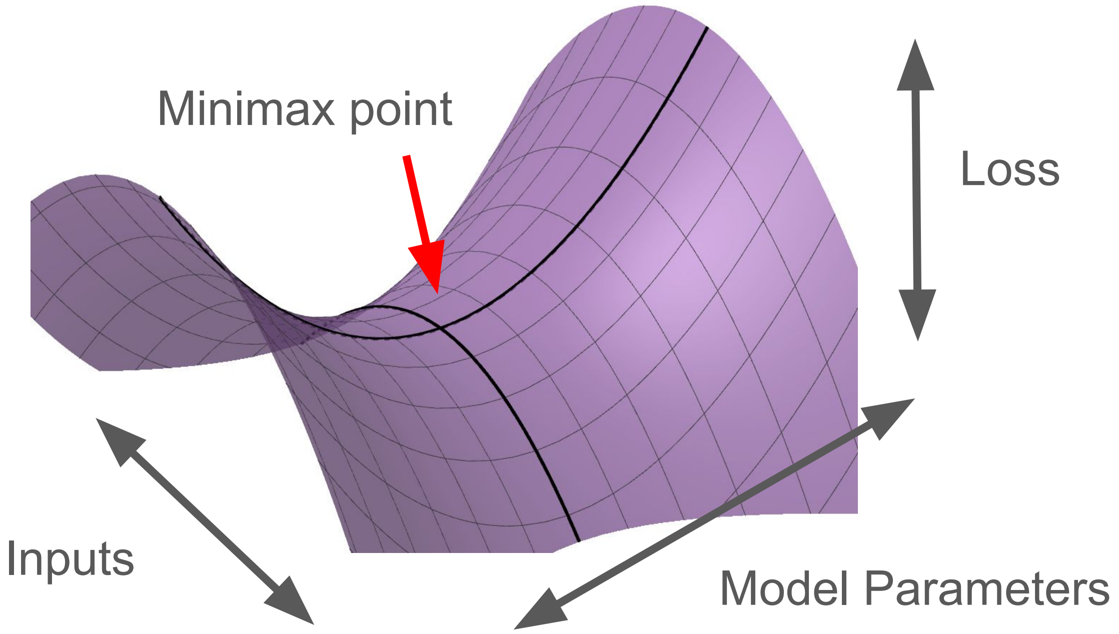
- L2 norm = square root of sum of squares of vector components
- $L\infty$ norm = max vector component value

“Towards Deep Learning Models Resistant to Adversarial Attacks”

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu

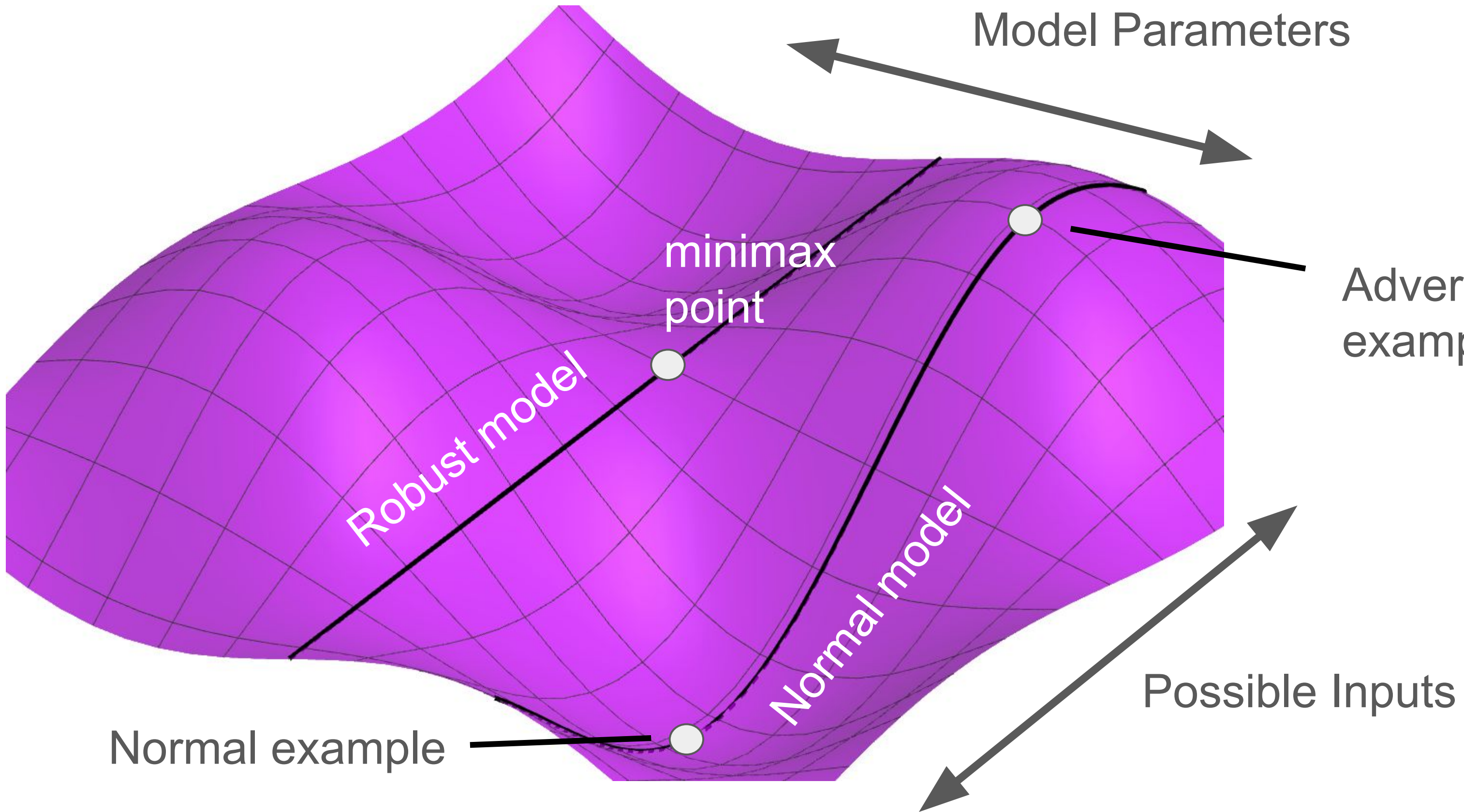
Adversarial Training: Saddle Point Problem

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y) \right]$$



- Outer maximization: model minimizes its expected loss by adjusting its parameters θ
- Inner maximization: adversarial attack maximizes loss by choosing a perturbation δ to apply to x

Adversarial Training - Saddle Point Problem



Fast Gradient Sign Method

$$\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{\varepsilon}\, \mathbf{sgn}\left(\nabla_{x} \mathrm{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})\right)$$

- x: input image
- ε: perturbation strength
- ∇_x L(θ, x, y): gradient of the loss function with respect to x
- Perturbation δ is calculated by taking the sign of the gradient and multiplying by the perturbation strength
- Simple but weaker than PGD

Projected Gradient Descent

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha \, \mathrm{sgn}(\nabla_x L(\theta, x, y)))$$

- S: set of allowed perturbations (defined either by L2 or L∞ norms from input x)
- α: learning rate
- Π: projection operator ensuring perturbed input stays within bounds
- ∇_x L(θ, x, y): gradient of the loss function with respect to x
- Multi-step variant of FGSM
- Stop iterating at some stop condition
  - Number of steps
  - Plateauing loss value
- Increased computational cost, but stronger than FGSM

Adversarial Input Examples

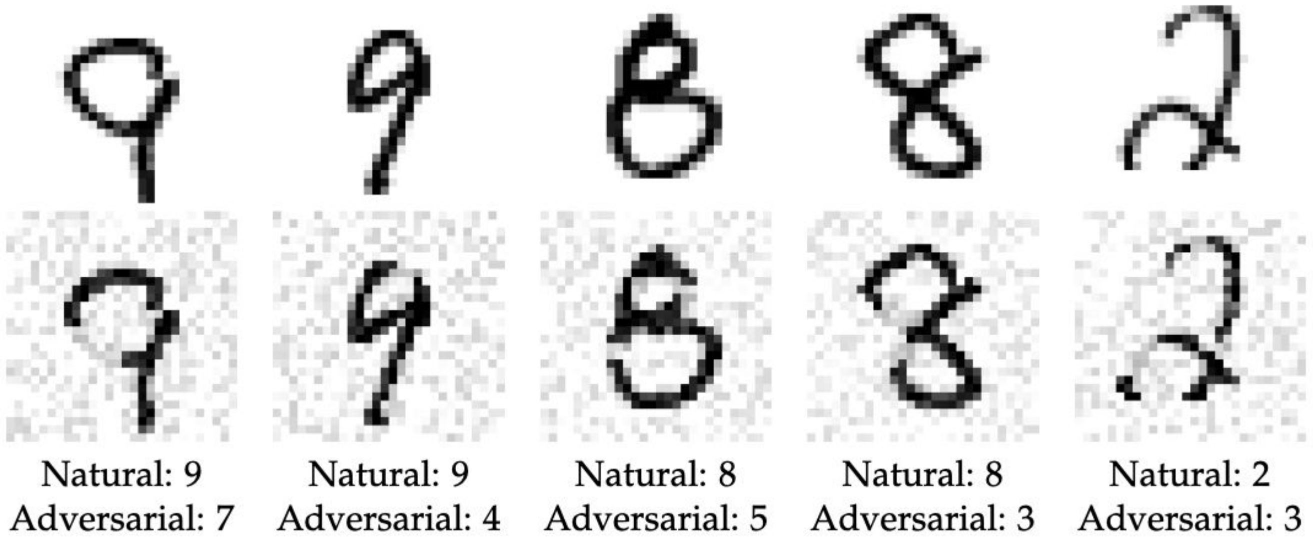


Figure 12: Sample adversarial examples with ℓ_2 norm bounded by 4. The perturbations are significant enough to cause misclassification by humans too.

PGD Local Maxima are Similar

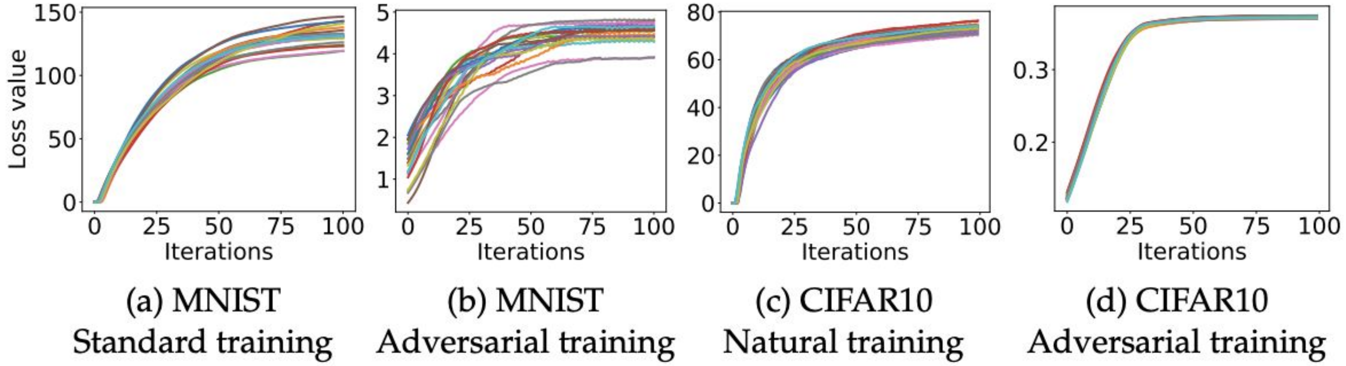


Figure 1: Cross-entropy loss values while creating an adversarial example from the MNIST and CIFAR10 evaluation datasets. The plots show how the loss evolves during 20 runs of projected gradient descent (PGD). Each run starts at a uniformly random point in the ℓ_∞-ball around the same natural example (additional plots for different examples appear in Figure 11). The adversarial loss plateaus after a small number of iterations. The optimization trajectories and final loss values are also fairly clustered, especially on CIFAR10. Moreover, the final loss values on adversarially trained networks are significantly smaller than on their standard counterparts.

Decision Boundaries

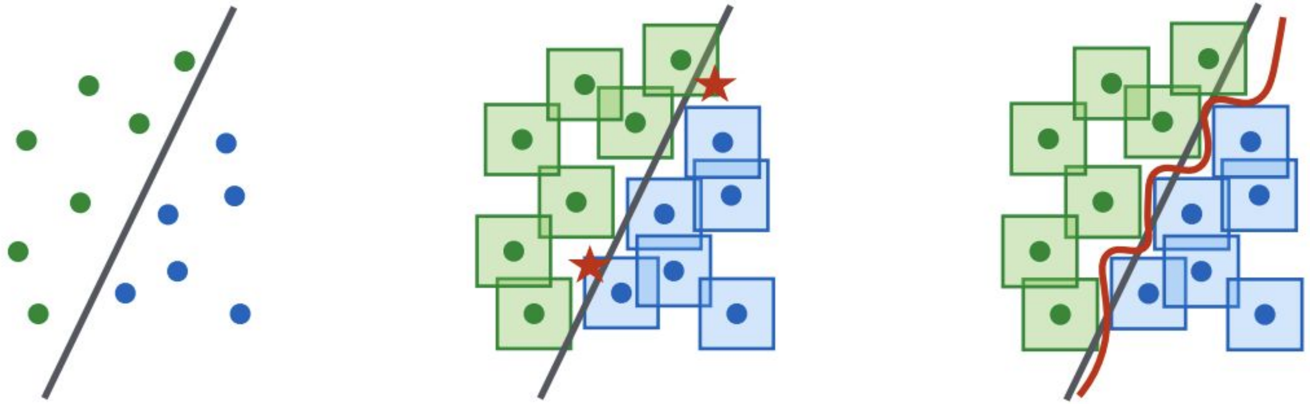
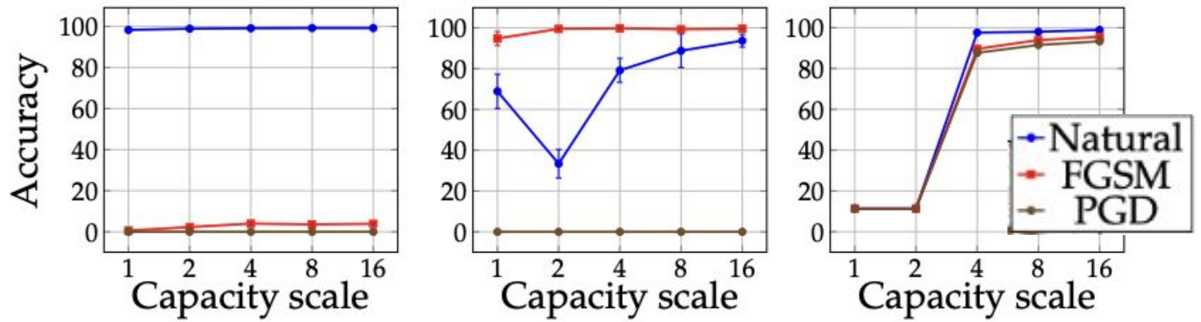


Figure 3: A conceptual illustration of standard vs. adversarial decision boundaries. Left: A set of points that can be easily separated with a simple (in this case, linear) decision boundary. Middle: The simple decision boundary does not separate the ℓ_∞-balls (here, squares) around the data points. Hence there are adversarial examples (the red stars) that will be misclassified. Right: Separating the ℓ_∞-balls requires a significantly more complicated decision boundary. The resulting classifier is robust to adversarial examples with bounded ℓ_∞-norm perturbations.

Model Capacity Helps with Robustness

MNIST



CIFAR10

	Simple	Wide
Natural	92.7%	95.2%
FGSM	27.5%	32.7%
PGD	0.8%	3.5%

(a) Standard training

	Simple	Wide
Natural	87.4%	90.3%
FGSM	90.9%	95.1%
PGD	0.0%	0.0%

(b) FGSM training

	Simple	Wide
Natural	79.4%	87.3%
FGSM	51.7%	56.1%
PGD	43.7%	45.8%

(c) PGD training

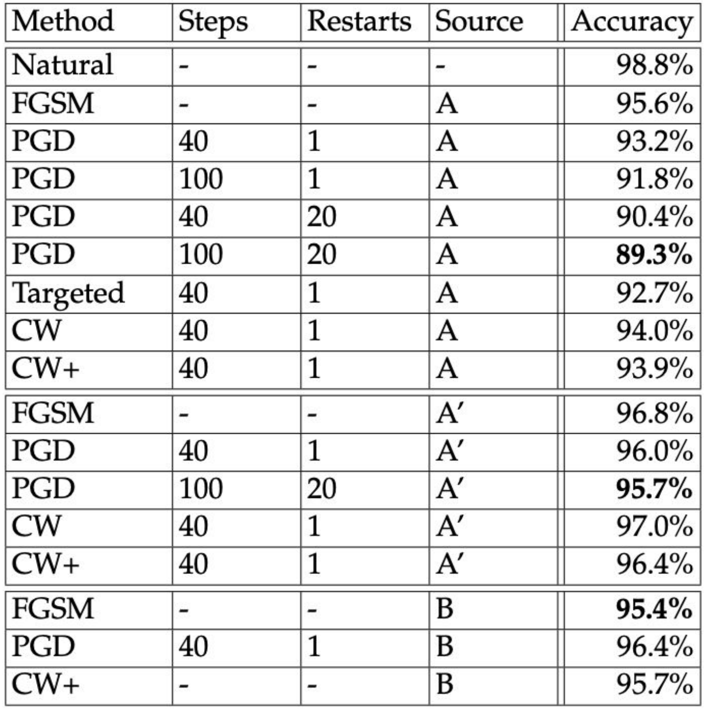
Testing Attack Methods

- A: original architecture
- A’: retrained
- A_{nat}: retrained not adversarially

- B: different architecture

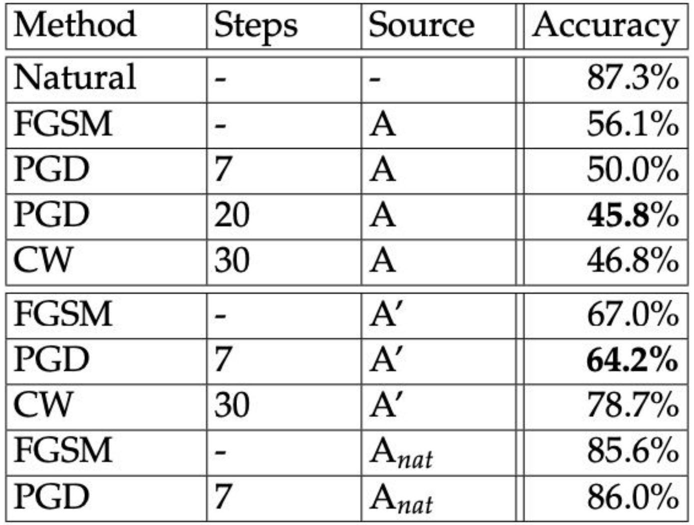
White-box

Black-box



Method	Steps	Restarts	Source	Accuracy
Natural	-	-	-	98.8%
FGSM	-	-	A	95.6%
PGD	40	1	A	93.2%
PGD	100	1	A	91.8%
PGD	40	20	A	90.4%
PGD	100	20	A	**89.3%**
Targeted	40	1	A	92.7%
CW	40	1	A	94.0%
CW+	40	1	A	93.9%
FGSM	-	-	A′	96.8%
PGD	40	1	A′	96.0%
PGD	100	20	A′	**95.7%**
CW	40	1	A′	97.0%
CW+	40	1	A′	96.4%
FGSM	-	-	B	**95.4%**
PGD	40	1	B	96.4%
CW+	-	-	B	95.7%

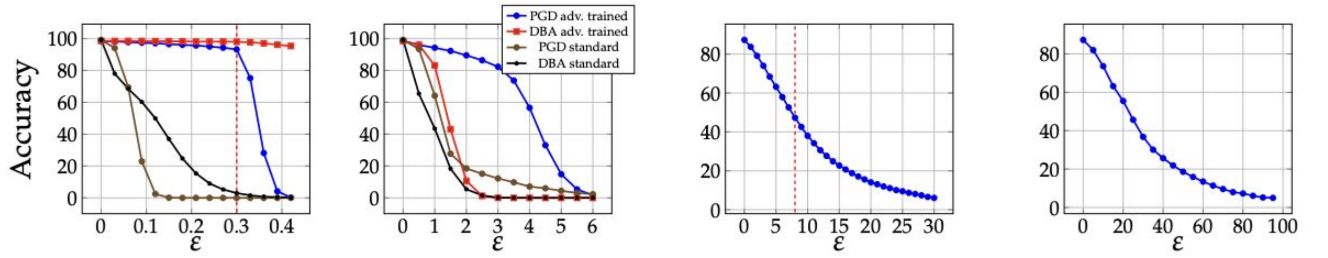
MNIST



Method	Steps	Source	Accuracy
Natural	-	-	87.3%
FGSM	-	A	56.1%
PGD	7	A	50.0%
PGD	20	A	**45.8%**
CW	30	A	46.8%
FGSM	-	A′	67.0%
PGD	7	A′	**64.2%**
CW	30	A′	78.7%
FGSM	-	A_{nat}	85.6%
PGD	7	A_{nat}	86.0%

CIFAR10

Adversarially Trained Networks vs. PGD



(a) MNIST, ℓ_∞-norm (b) MNIST, ℓ_2-norm (c) CIFAR10, ℓ_∞-norm (d) CIFAR10, ℓ_2-norm

Figure 6: Performance of our adversarially trained networks against PGD adversaries of different strength. The MNIST and CIFAR10 networks were trained against $\varepsilon = 0.3$ and $\varepsilon = 8$ PGD ℓ_∞ adversaries respectively (the training ε is denoted with a red dashed lines in the ℓ_∞ plots). In the case of the MNIST adversarially trained networks, we also evaluate the performance of the Decision Boundary Attack (DBA) [4] with 2000 steps and PGD on standard and adversarially trained models. We observe that for ε less or equal to the value used during training, the performance is equal or better. For MNIST there is a sharp drop shortly after. Moreover, we observe that the performance of PGD on the MNIST ℓ_2-trained networks is poor and significantly overestimates the robustness of the model. This is potentially due to the threshold filters learned by the model masking the loss gradients (the decision-based attack does not utilize gradients).

“Evading Watermark based Detection of AI-Generated Content”

Zhengyuan Jiang, Jinghuai Zhang, Neil Zhenqiang Gong

Adversarially Removing Watermarks

Watermark: 11111
“AI-generated”

Human-imperceptible perturbation

Watermark: 01101
“Not AI-generated”

Key Terms

White-box setting: Attacker has access to watermark decoder, but not ground-truth watermark or encoder

Black-box setting: Attacker only has access to an API that returns “AI-generated” or “Not AI-Generated”

Bitwise Accuracy (BA): Fraction of bits matching in between two decoded watermarks, detector thresholds this

Evasion Rate: Fraction of watermarked images that were successfully altered to be detected as non-watermarked

WEvade-W-I (White-box)

δ	Perturbation	$D(I_w)$	Original decoded watermark

Goal: Minimal perturbation that results in inverse watermark

$$\min_{\delta} ||\delta||_\infty$$
$$s.t.\ D(I_w + \delta) = \neg D(I_w),$$

Easier to optimize

Loss = L2 distance between watermark bit probabilities and flipped watermark

$$\begin{aligned}\min_{\delta}\ & l(D(I_w + \delta), \neg D(I_w)) \\ s.t.\ & ||\delta||_\infty \le r, \\ & D(I_w + \delta) = \neg D(I_w),\end{aligned}$$

WEvade-W-I (White-box)

Binary search over values of r, checking if second constraint can be met

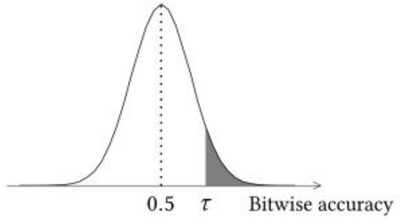
For each r:
Projected Gradient Descent, with perturbation norm constraint only

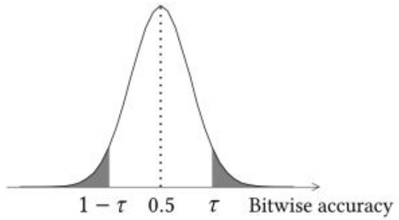
Problem: Double-tailed detector - if match is too bad, also detect as watermarked

$$\min_{\delta} l(D(I_w + \delta), \neg D(I_w))$$

$$s.t. \ ||\delta||_\infty \leq r,$$

$$D(I_w + \delta) = \neg D(I_w),$$





WEvade-W-II (White-box)

Random non-watermarked image is likely to have a decoded watermark with half of bits matching

Use random watermark instead of decoded watermark with half of bits flipped for better theoretical guarantees

$$\min_{\delta} l(D(I_w + \delta), w_t)$$
$$s.t. \ ||\delta||_\infty \leq r,$$
$$BA(D(I_w + \delta), w_t) \geq 1 - \epsilon,$$

w_t Random watermark

Want to be close to w_t, but not requiring perfect match

WEvade-B-S (Black-box)

Without access to the decoder, can’t backpropagate

Train a surrogate encoder/decoder and find an adversarial perturbation with that

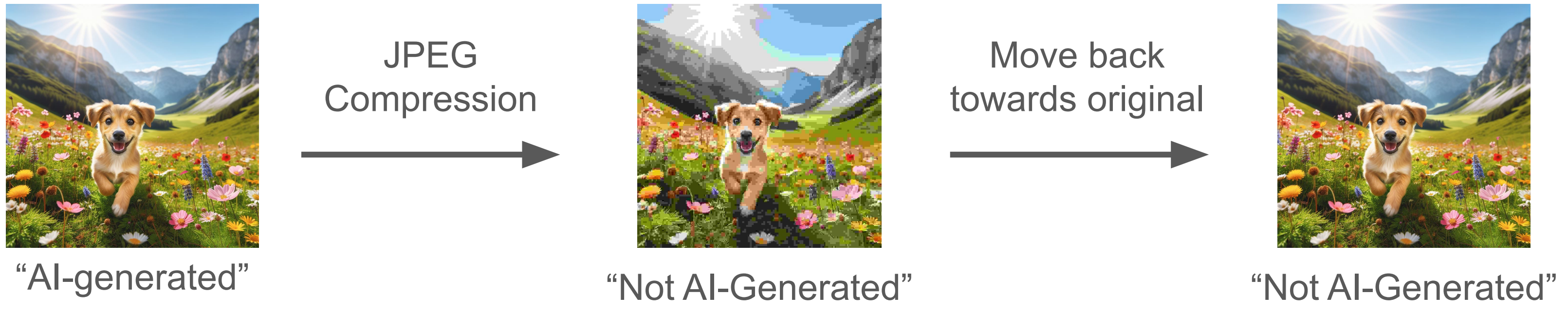
Hopefully the adversarial perturbation transfers to the real decoder

WEvade-B-Q (Black-box)

Use the detector API - given image, predicts whether is watermarked

Leverage the HopSkipJump attack to make use of detector API responses

- Start at a highly perturbed version of the image that evades detection
- Make poor quality image more like original, while always evading detection



HopSkipJump

Key idea: Yes/No responses are useful near the decision boundary

1) Find decision boundary with binary search, move near there
2) Try many random perturbations to estimate gradient
3) Move in direction of gradient, while staying in "Non-AI Generated" region

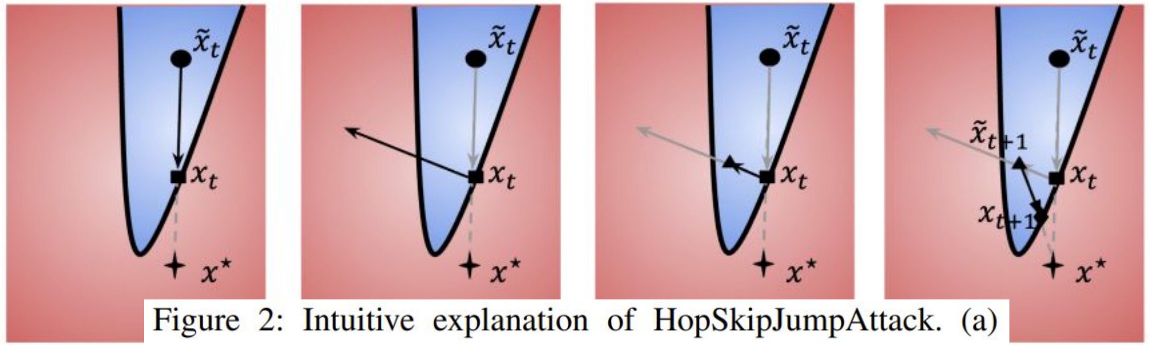
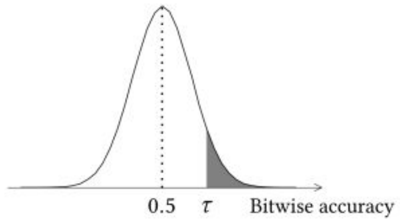
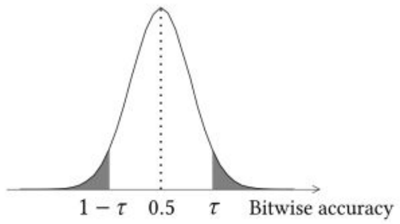


Figure 2: Intuitive explanation of HopSkipJumpAttack. (a) Perform a binary search to find the boundary, and then update $\tilde{x}_t \to x_t$. (b) Estimate the gradient at the boundary point x_t. (c) Geometric progression and then update $x_t \to \tilde{x}_{t+1}$. (d) Perform a binary search, and then update $\tilde{x}_{t+1} \to x_{t+1}$.

Theoretical Analysis





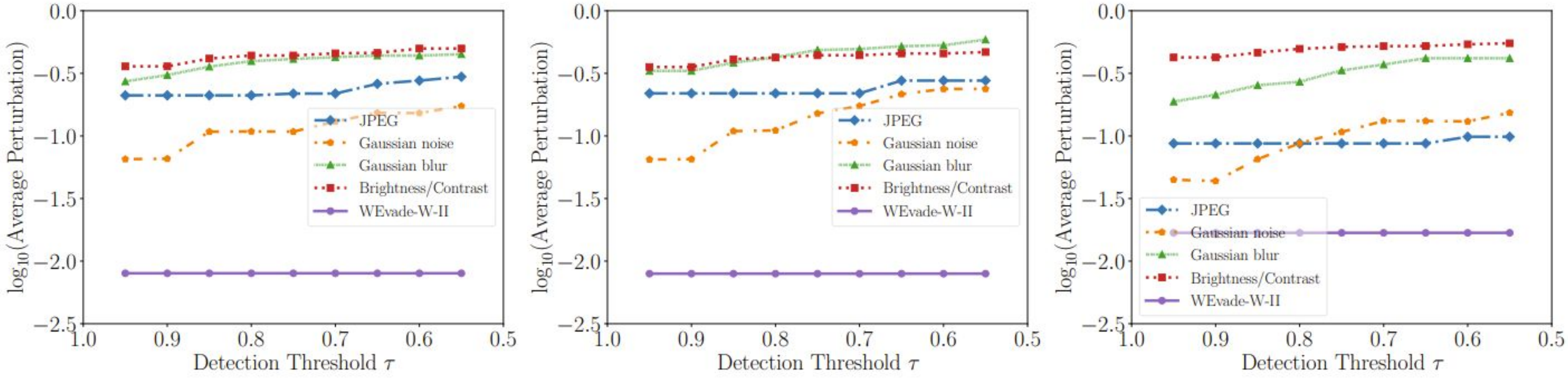
WEvade-W-II, randomness comes from random watermark chosen, adversarial image will have that watermark with a little error

Probability not detected as AI-generated

- Probability that the random watermark doesn't match the original watermark that much
- Probability that fewer bits match than would be needed to meet the threshold (with some error)
- # bits matching ~ Binomial (n, 0.5), use CDF
- Convert to two-tailed probability

WEvade-B-Q: Always evades, image quality could be bad though

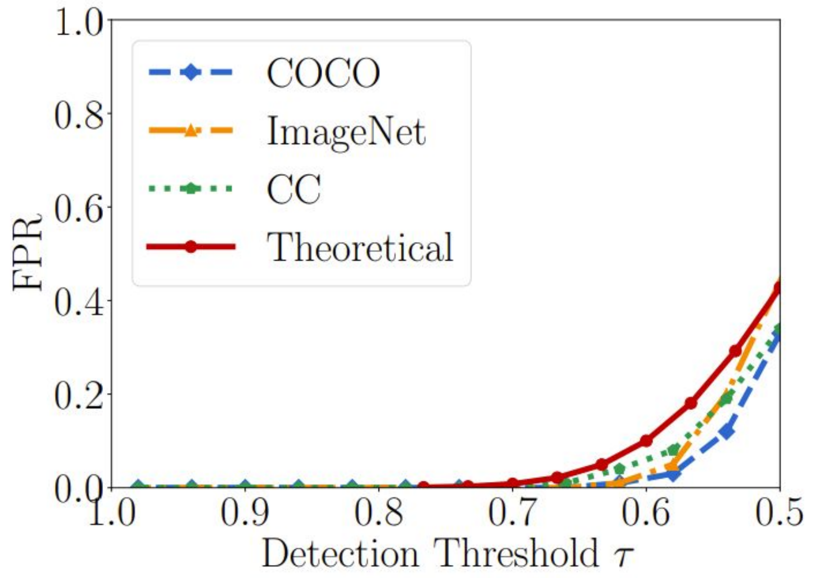
WEvade-W-II Evades Detection with Small Perturbations



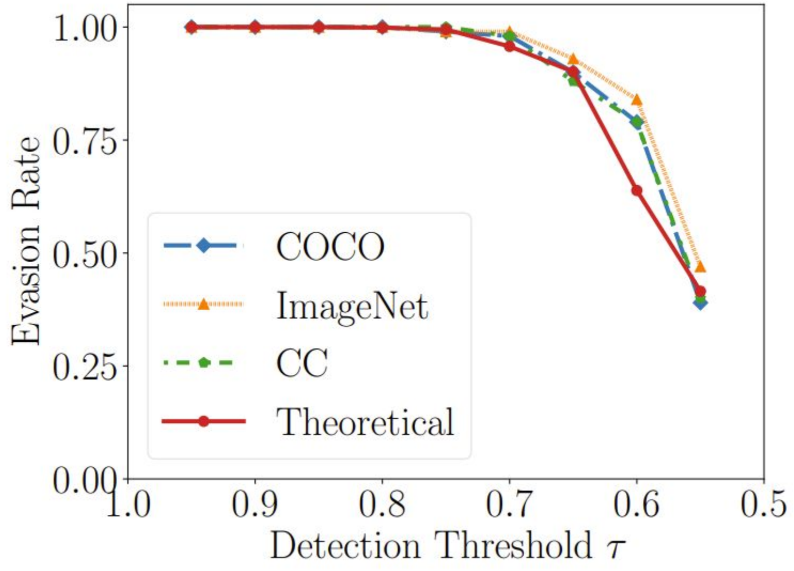
(a) COCO (b) ImageNet (c) CC

Figure 7: Average perturbation added by each post-processing method to evade the double-tail detector with different threshold τ in the white-box setting. We set the parameters of existing post-processing methods such that they achieve the same evasion rate as our WEvade-W-II. The watermarking method is HiDDeN and the results for UDH are shown in Figure 24 in Appendix.

WEvade-W-II Evades all Meaningful Thresholds



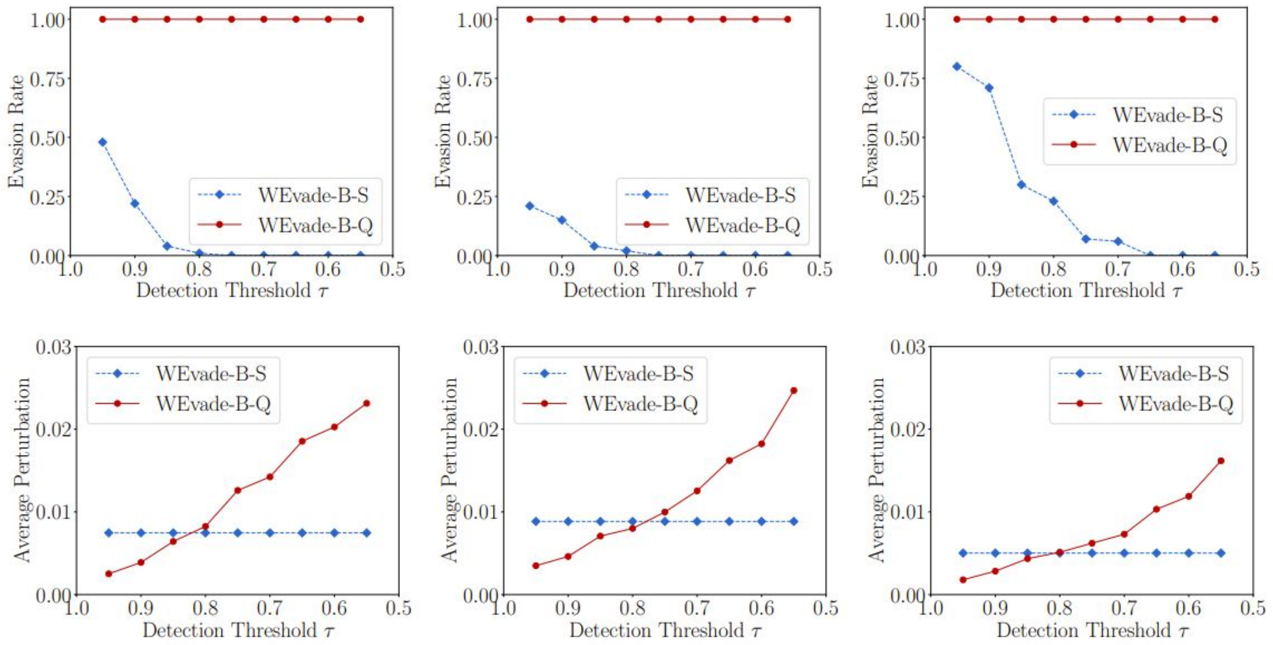
(a) HiDDeN, FPR



(a) HiDDeN

Blackbox: WEvade-B-S vs. WEvade-B-Q

WEvade-B-S has low evasion, and requires greater perturbation when evading



(a) COCO (b) ImageNet (c) CC

Using HiDDeN watermarking

“A Transfer Attack to Image Watermarks”

Yuepeng Hu, Zhengyuan Jiang, Moyang Guo, Neil Gong

Use an Ensemble of Watermarking Models

- Key idea: Transfer attack on ensemble of models
  - Transfer attack: train a surrogate model to approximate behavior of a target system

- Ensemble-Optimization aims to find a minimum perturbation such that the watermark decoded by each surrogate decoder for the perturbed image is the same as its corresponding target watermark
  - Projected Gradient Descent
  - Generalizability across surrogate decoders enhances transferability to target decoder

- Even state-of-the-art deep watermarking models are vulnerable to transferable adversarial attacks
  - Limitation: attack's success rate depends on the similarity between substitute and target models

Transfer Attack Works (Unlike WEvade-B-S)!

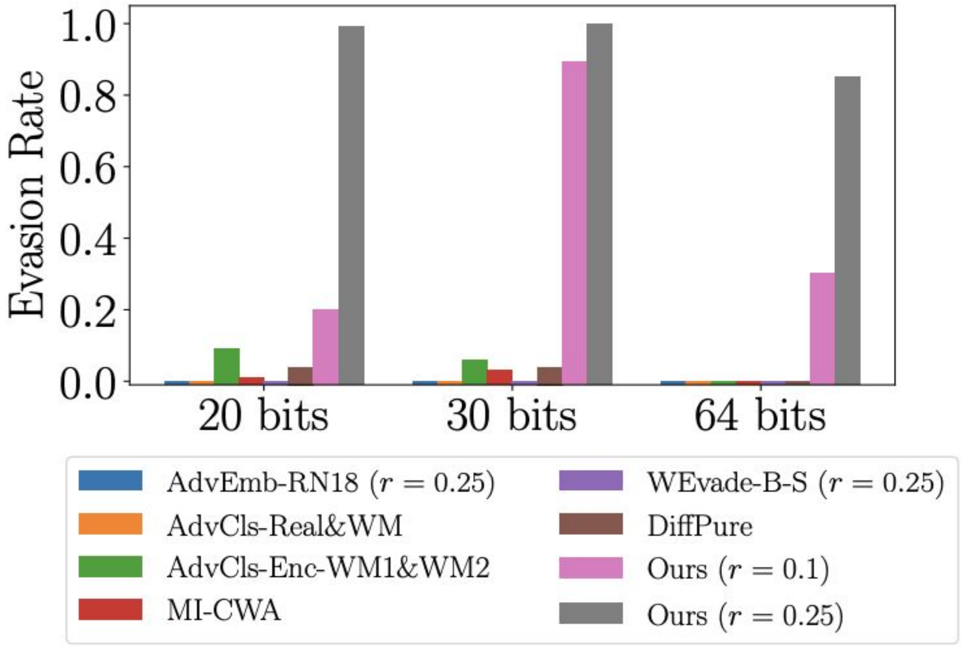
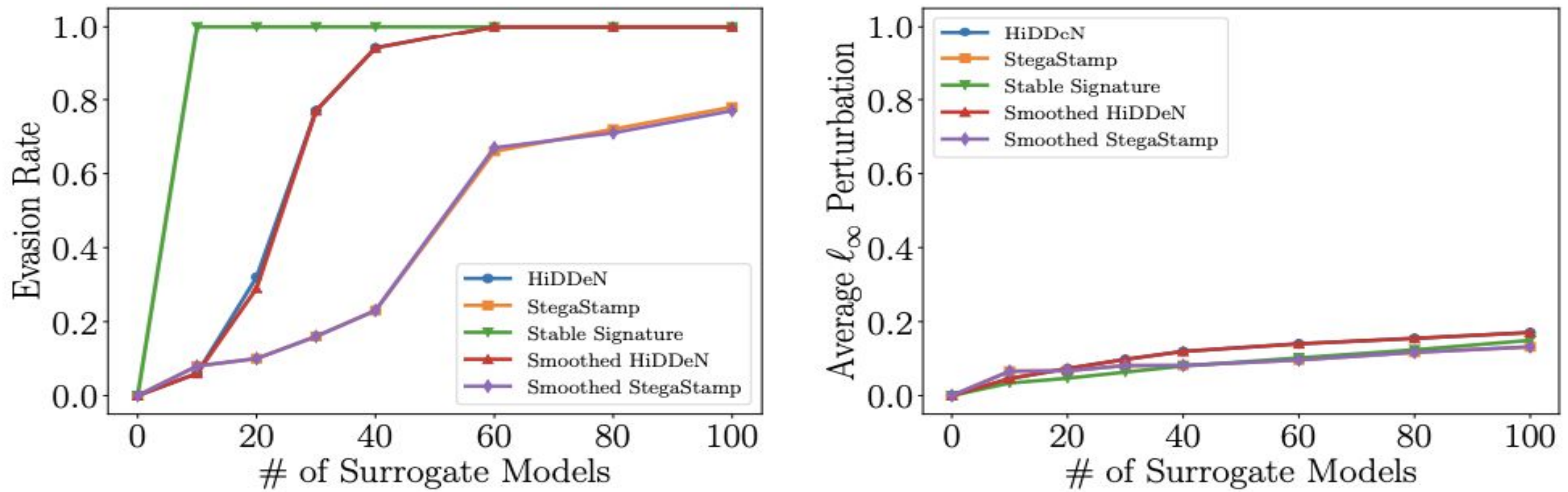


Figure 5: Comparing evasion rates of existing and our transfer attacks. The target model is ResNet and uses watermarks with different lengths. Dataset is Stable Diffusion. Similar results for Midjourney are shown in Figure 14 in Appendix.

More Surrogate Models Help Evasion but Require Higher Perturbation



(a) Evasion rate (b) ℓ_∞-norm perturbation

Figure 6: Evasion rates and average ℓ_∞-norm perturbation of our transfer attacks to different watermarking methods.