Preprocessors Matter!

Realistic Decision-Based Attacks on Machine Learning Systems

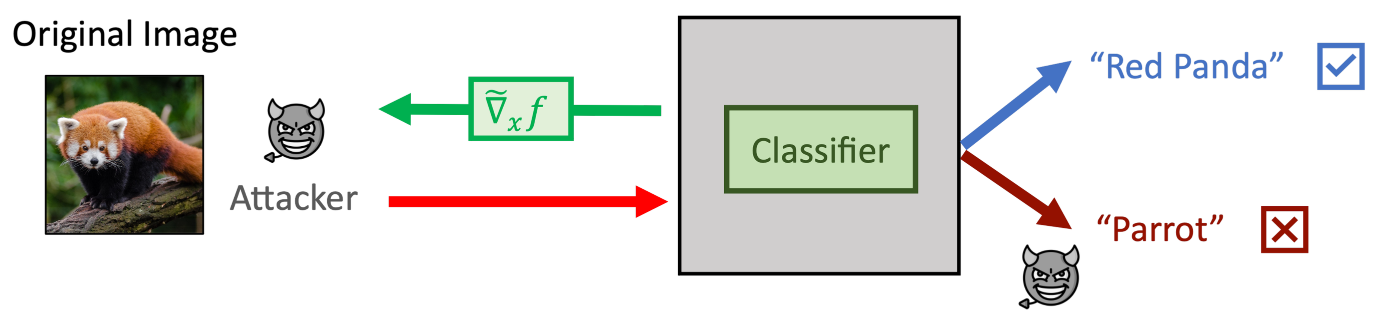
Chawin Sitawarin[1] Florian Tramèr[2] Nicholas Carlini[3]

[1]UC Berkeley [2]ETH Zürich [3]Google

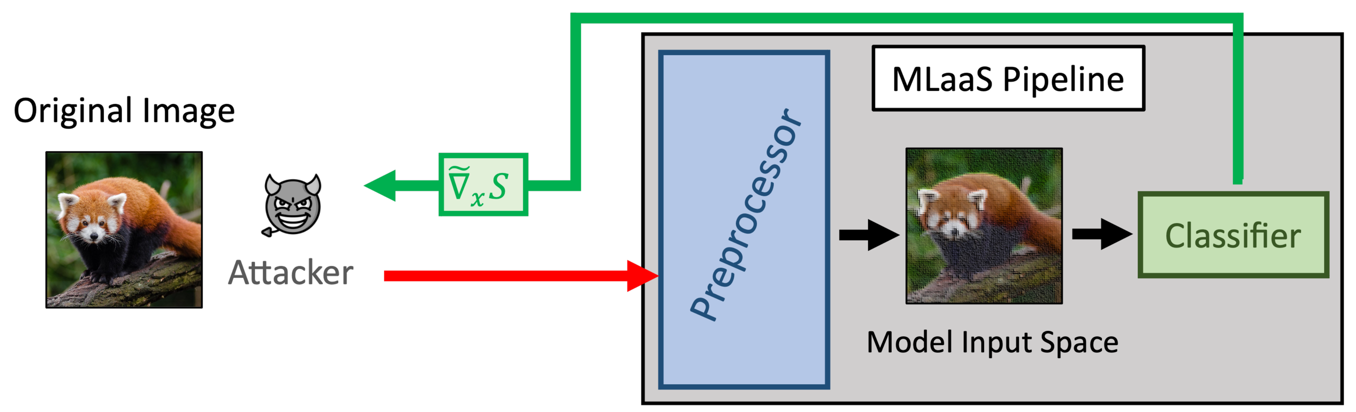
Summary

1. Image preprocessors (e.g., resize, compress) in a typical computer vision API was not commonly studied in the literature, but they can hinder query-based hard-label attacks.
2. We create **preprocessor-aware attacks** that "bypass" the known preprocessors and outperform the unaware attackers.
3. We propose an **extraction attack** for finding out which preprocessors are used in the API pipeline.

Traditional Setup in the Attack Literature

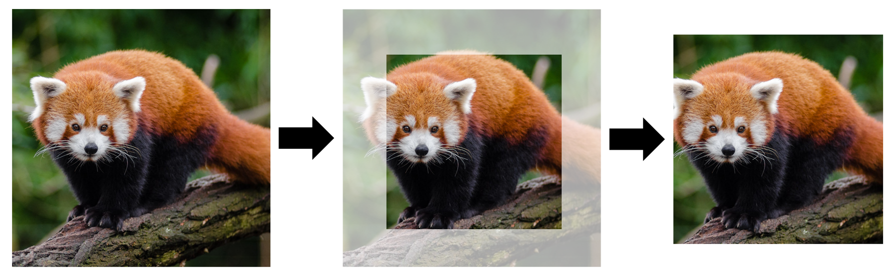


In practice, there are likely multiple preprocessors in the pipeline.



- Preprocessors can make decision-based attacks less effective.
- Some perturbations do not affect the prediction because of the *invariance* of the preprocessors. Hence, the adversary gains less information from each query than they would have w/o preprocessors.

Knowing which preprocessor is used, can we exploit *invariance* of the preprocessor?



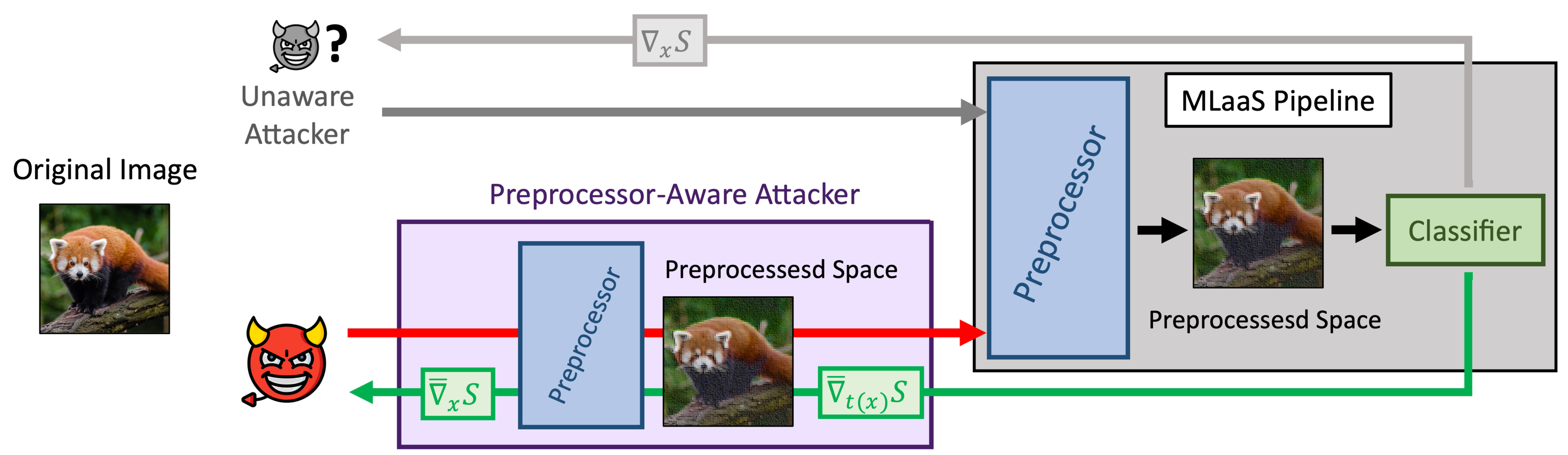
Preprocessor-Aware Query-Based Attack

1 **Bypassing Attack**

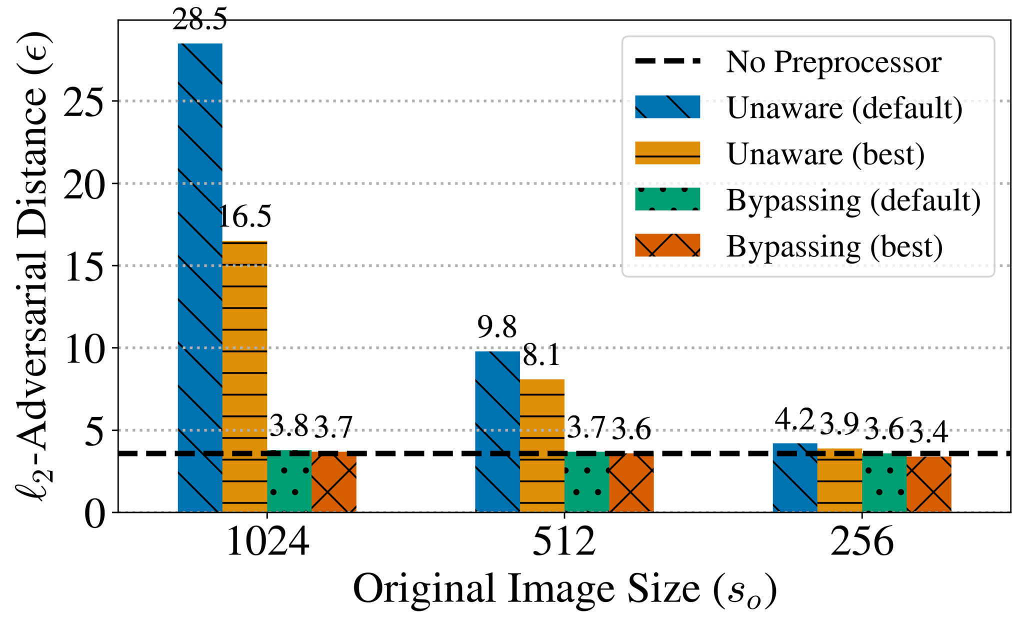
- Very simple! But only works for some preprocessors like cropping and resizing.
- Just run any off-the-shelf attack on the processed image and reverse the final perturbation.

2 **Biased-Gradients Attack**

- Works for any preprocessor including non-differentiable ones like quantization and JPEG.
- Slightly modify **gradient estimation step** of off-the-shelf attacks.
- Similar to `Bypassing Attack`, but also backprop through the preprocessor.



Experiments on ResNet-18 (ImageNet)



- Preprocessor-aware attacks are much more effective (up to 7x) than the unaware.
- More invariance = more improvement.
- (Bonus) Attack hyperparameters matter a lot. We swept ~5 settings and reported the best.

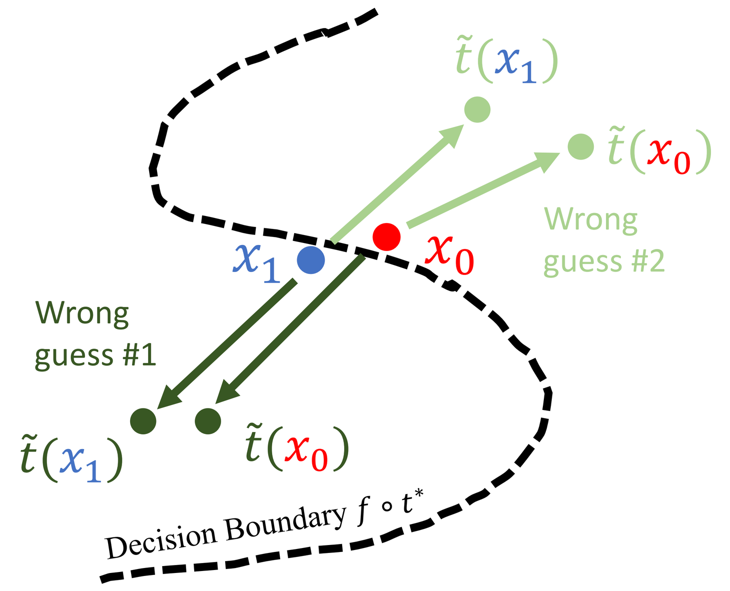
Table 2: Comparison of the mean adversarial perturbation norm (↓) found by our Biased-Gradient Attacks vs the preprocessor-unaware and the SNS counterparts.

Preprocess	Methods	Untg. HSJA	Targeted HSJA	Targeted QEBA
Crop (256 → 224)	Unaware	4.2	38.2	22.2
	SNS	3.7	35.4	31.5
	Biased-Grad (ours)	**3.7**	33.1	**19.6**
Resize (1024 → 224) (Nearest)	Unaware	16.5	153.4	90.5
	SNS	3.9	112.6	32.2
	Biased-Grad (ours)	**3.7**	23.5	**19.4**
Quantize (4 bits)	Unaware	9.7	63.7	56.4
	SNS	6.4	55.9	57.2
	Biased-Grad (ours)	**3.1**	39.3	**28.8**
JPEG (quality 60)	Unaware	9.2	63.2	52.7
	SNS	2.7	44.5	44.6
	Biased-Grad (ours)	**1.5**	25.1	**21.0**
Neural Compress (Ballé et al., 2018) (hyperprior, 8)	Unaware	25.1	92.0	78.6
	SNS	17.6	83.6	78.9
	Biased-Grad (ours)	**15.8**	**75.2**	75.8
Neural Compress (Cheng et al., 2020b) (attention, 6)	Unaware	33.8	94.1	86.9
	SNS	14.3	80.3	75.5
	Biased-Grad (ours)	**12.6**	**74.8**	77.9

Preprocessor Extraction Attack

- Main idea: guess and check!
- This attack can be run only once and then used for finding all subsequent adversarial inputs!

1. Guess the preprocessor $\tilde{t}$ (vs. real t^*) and apply to some carefully chosen inputs (x_0, x_1).
2. Check by feeding them to the target pipeline.



- If our guess is right, prediction stays the same.
- Otherwise, it will likely change.

Assumption: Preprocessor is *idempotent*.
If $\tilde{t} = t^*$, $f(t^*(\tilde{t}(x))) = f(t^*(x)) = y$ (guaranteeed).
If $\tilde{t} \neq t^*$, $f(t^*(\tilde{t}(x))) \neq y$ (not guaranteeed).

3. Repeat 1. and 2. with multiple input pairs until we're sufficiently confident.

Experiments on Hugging Face Models 🤗

Table 4: Number of queries (mean ± standard deviation) necessary to determine what preprocessor is being used.

Preprocessor Space	Num. Queries
Arbitrary resize (200px–800px)	632 ± 543
Arbitrary center crop (0%-100%)	52.0 ± 1.3
Arbitrary JPEG compression (quality 50-100)	70.0 ± 22.8
Typical resize (see text)	48.7 ± 6.8

- The number of attack queries depends on the set of all possible preprocessors.
- Usually extracting 1 preprocessor uses fewer queries than finding 1 adversarial example.