Adversarial Examples for Deep Learning: Attack, Defense and Robustness

Xianglong Liu

July 2021

State Key Lab of Software Development Environment

Beihang University, China

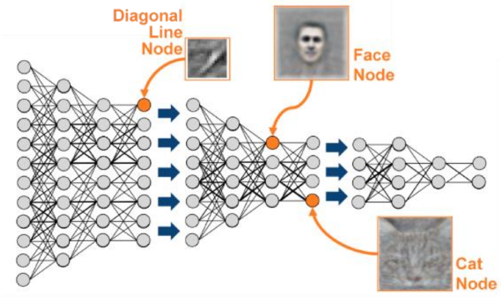
xlliu@buaa.edu.cn

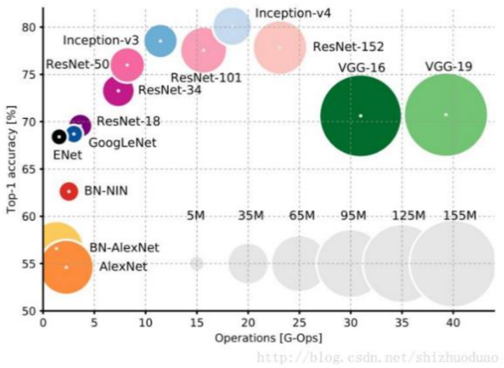
Speaker Information

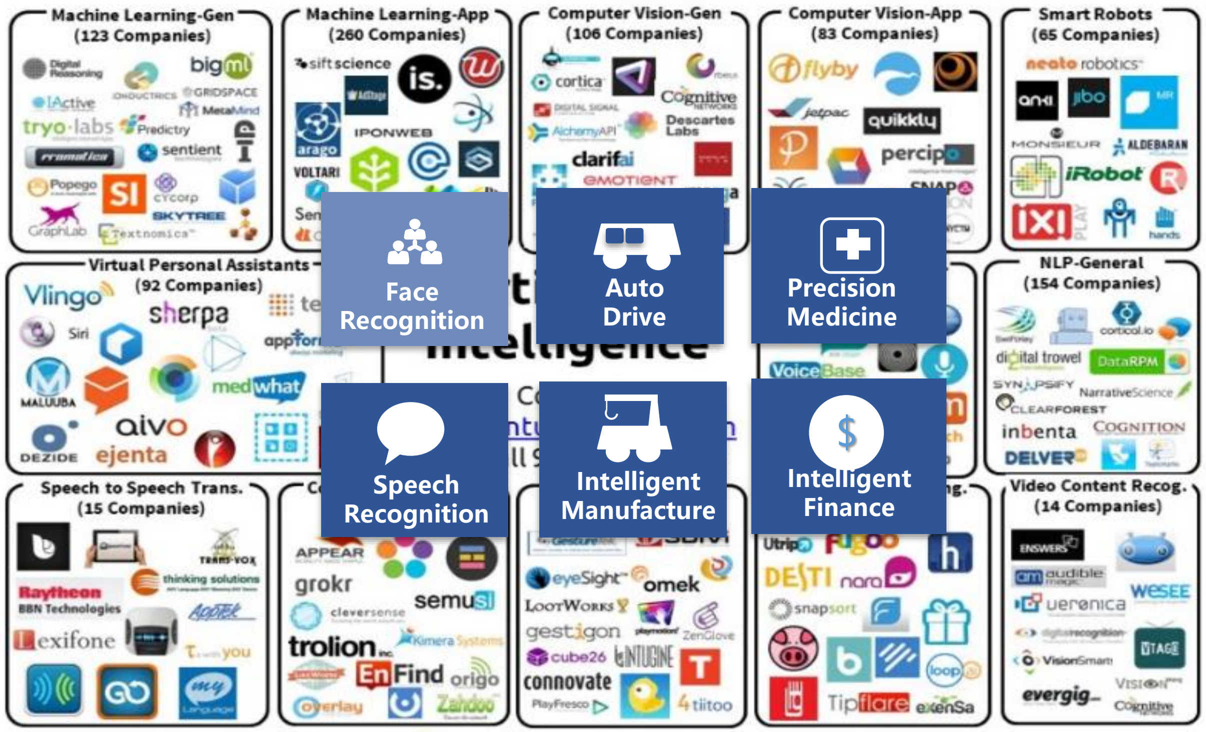
Dr. Xianglong Liu is currently a full Professor with the School of Computer Science and Engineering, Beihang University. His research interests include fast visual computing (e.g., large-scale search/understanding) and trustworthy deep learning (e.g., network quantization, adversarial attack/defense, few shot learning). He authored more than 60 papers on top-tier conferences and journals. He is serving as the associated editor of Pattern Recognition, IET Image Processing, Frontiers of Computer Science, area chair of ACM MM 2019/2020/2021, etc. He is also the TPC member of the OpenI Open Source Platform for Artificial Intelligence. He received NSFC Excellent Young Scientists Fund, and was selected into the 2019 Beijing Nova Program and 2015 China Computer Federation (CCF) Young Talents Development Program. He also received a number of awards including PCM 2018 best student paper, IEEE ICME 2011 best paper candidate, IEEE CVPR 2014/2016 Young Researcher Award, the 2015 CCF Outstanding Doctoral Dissertation Award, etc.

Outline

Deep learning has show the great success in the fields of computer vision, natural language processing, speech recognition, etc.







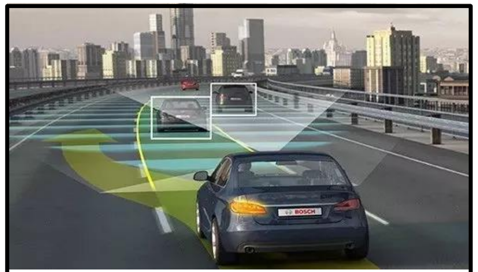
Artificial Intelligence

With the great development of **deep learning** technology
the application of artificial intelligence extends its vitality

Face recognition

Intelligent manufacturing

Speech recognition



Auto-driving

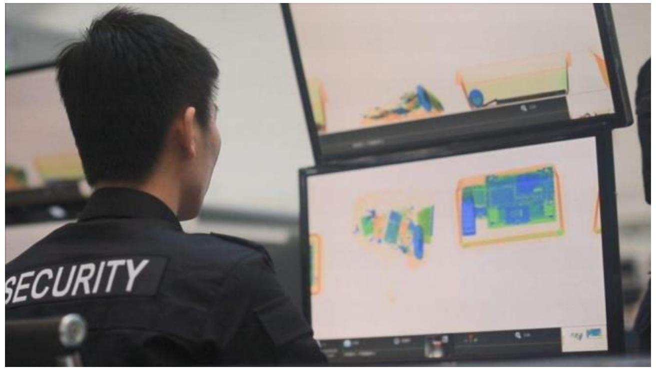
More Possibility
……

Severe threats to life and property

June 1 2020 Taiwan, China

Challenges

More safety & security sensitive tasks



More Challenging Scenarios

A new type of attack: adversarial examples and related problems

- Adversarial examples are elaborately designed perturbations to attack machine learning models:
  - Imperceptible to human;
  - Misleading to DNNs;

- Definition:

$$f(x) \neq f(x+r)\ s.t.\ \ \|r\| \leq S_{max}$$

where x is a input image, r is the noise, and f is the model.

Clean Example
Human: Panda
DNN: Panda

Noise

Adversarial Example
Human: Panda
DNN: Gibbon

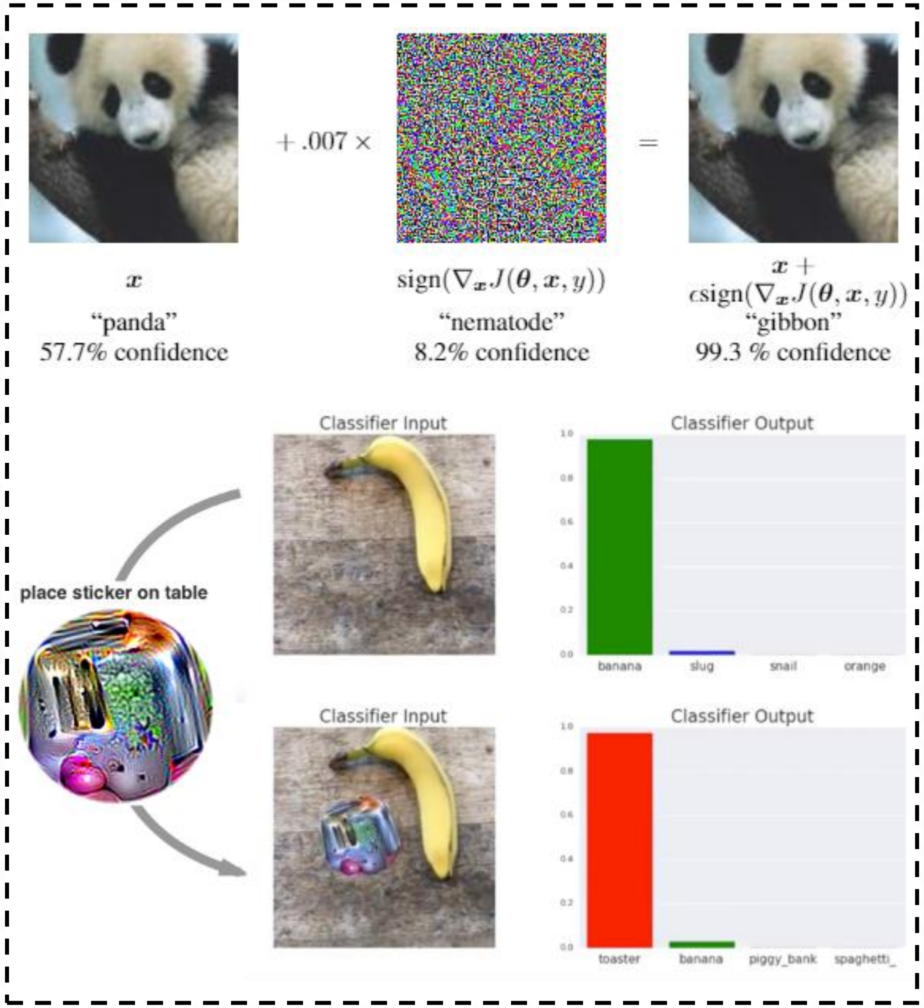
Clean Example
Human: Banana
DNN: Banana

Adversarial Patch

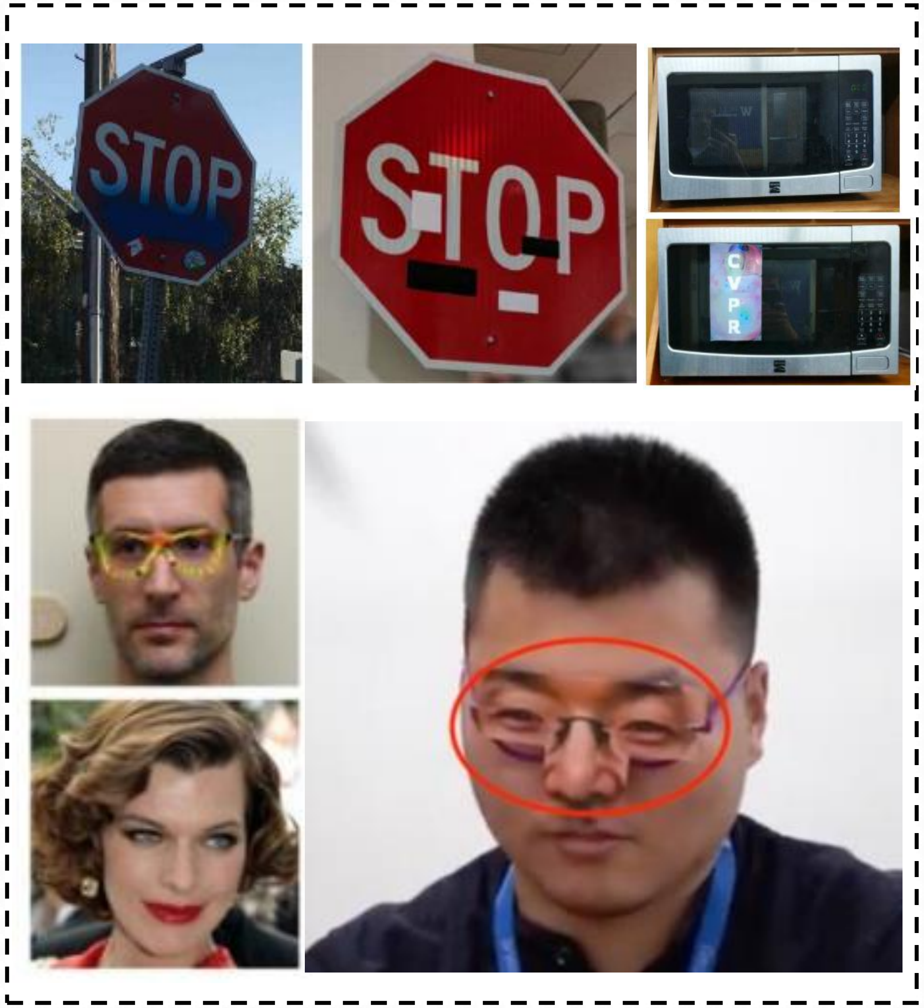
Adversarial Example
Human: Banana
DNN: Toaster

Adversarial Examples

Digital World



Physical World



Trend in the World

AI-Ready DoD by 2025

Warfighters enabled with baseline digital literacy and access to the digital infrastructure and software required for ubiquitous AI integration in training, exercises, and operations

NSCAI Recommendations

Ways for DoD to Operationalize AI

An AI-ready DoD will enable the application and integration of AI-enabled technologies into every facet of warfighting

Prepare:
- Business processes
- Design
- Readiness

Sense and Understand:
- Plan and task collection
- Collect
- Process
- Exploit and analyze
- Disseminate

Decide:
- Planning
- Deciding
- Tasking, delegation and distribution

Execute:
- Logistics and sustainment
- Movement
- Targeting
- Precision and accuracy

THE NATIONAL SECURITY COMMISSION ON ARTIFICIAL INTELLIGENCE

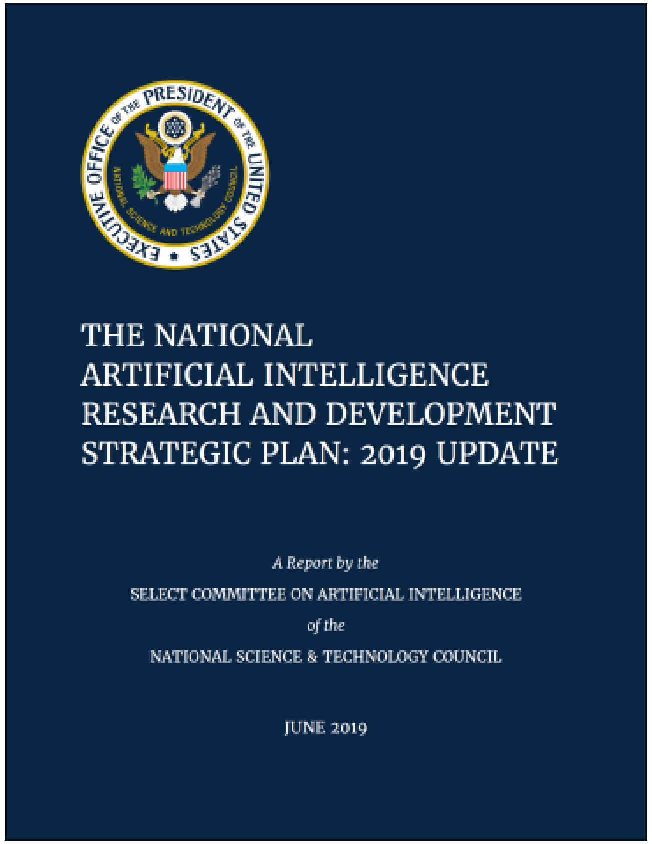
Trend in the World

How AI is Transforming the Threat Landscape

Current Threats Advanced BY AI Systems	New Threats FROM AI Systems	Threats TO AI Stacks Themselves
AI transforms existing range and reach of threats	AI creates new threat phenomena	AI itself is also a new attack surface
• Self-replicating AI-generated malware • Improved and autonomous disinformation campaigns • AI-engineered and targeted pathogens	• Deepfakes and computational propaganda • Micro-targeting: AI-fused data for targeting or blackmail • AI swarms and nano-swarms	• AI attack involves the whole "AI stack". Examples include: ◦ Model inversion ◦ Training data manipulation ◦ "Data lake" poisoning

Trend in the World

U.S.A

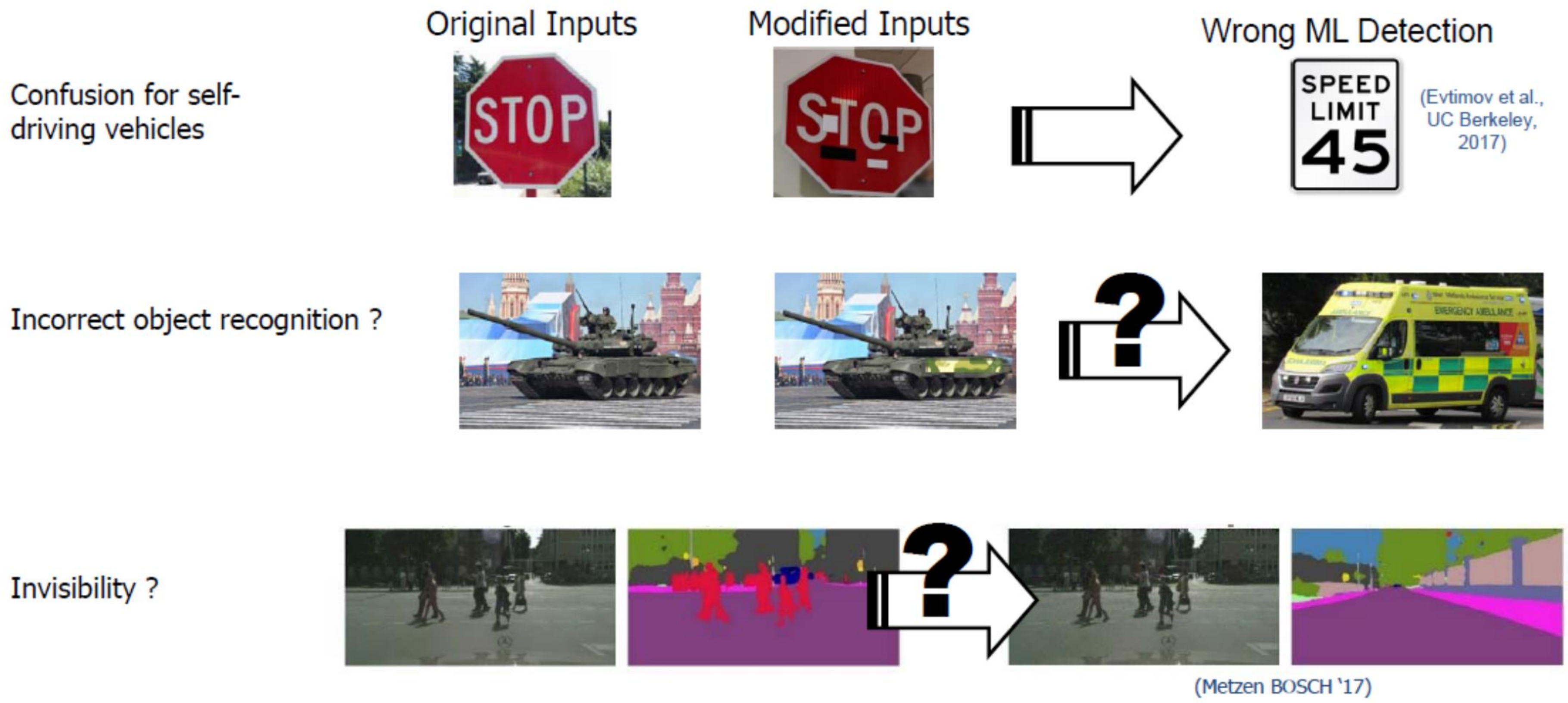


Ensure the safety and reliability of artificial intelligence systems

EU

Framework of Trustworthy AI

Trend in the World

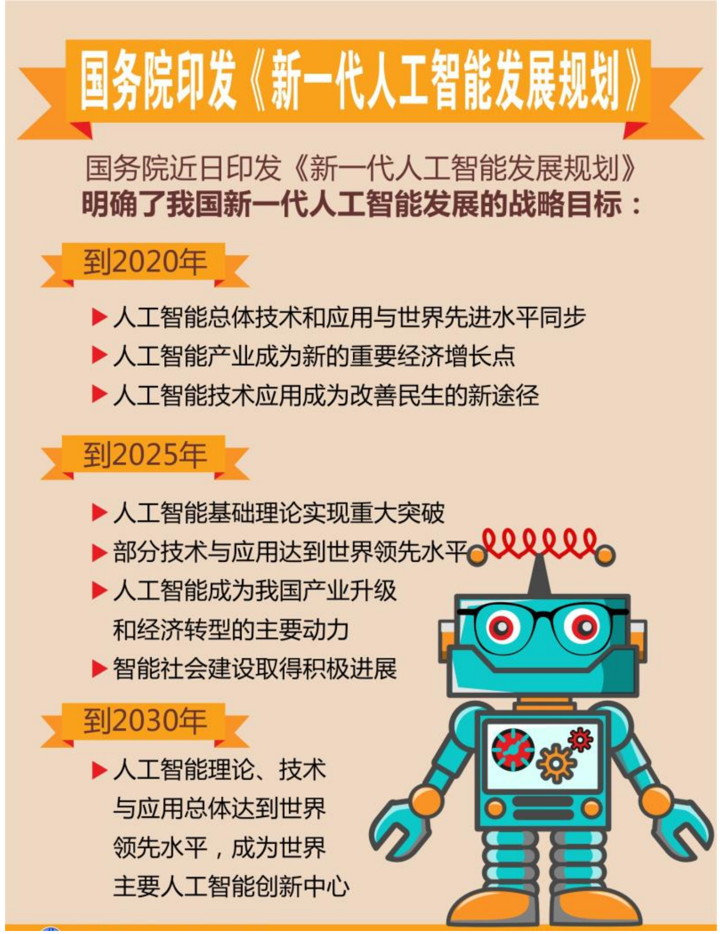


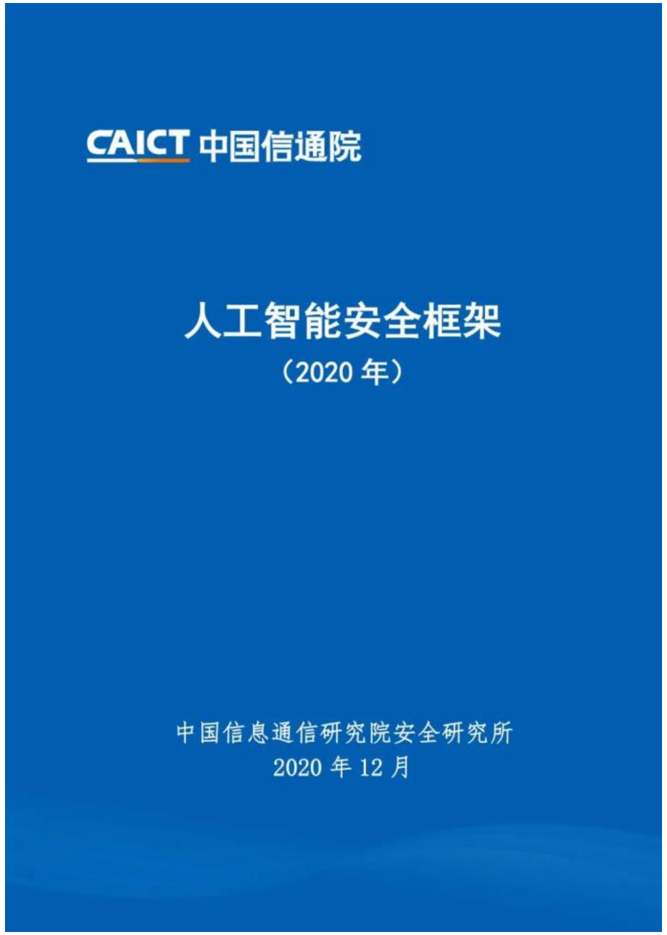
Guaranteeing AI Robustness against Deception (GARD)， 2019

Trend in China

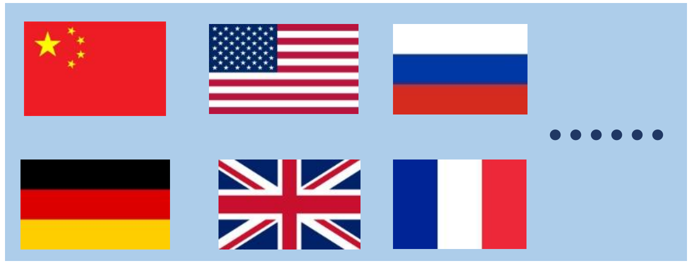
National Strategy: stressing the importance of secure, controllable, and reliable AI

国务院
STATE COUNCIL





Standards



ISO/IEC JTC 1

ISO/IEC JTC 1/SC 27

Information security, cybersecurity and privacy protection

REFERENCE	TITLE
ISO/IEC JTC 1/SC 27/AG 1	Management Advisory Group
ISO/IEC JTC 1/SC 27/AG 2	Trustworthiness
ISO/IEC JTC 1/SC 27/AG 3	Concepts and Terminology
ISO/IEC JTC 1/SC 27/AG 4	Data security
ISO/IEC JTC 1/SC 27/AG 5	Strategy
ISO/IEC JTC 1/SC 27/AG 6	Operations
ISO/IEC JTC 1/SC 27/AG 7	Communication and Outreach (AG-CO)
ISO/IEC JTC 1/SC 27/CAG	Chair's Advisory Group
ISO/IEC JTC 1/SC 27/WG 1	Information security management systems
ISO/IEC JTC 1/SC 27/WG 2	Cryptography and security mechanisms
ISO/IEC JTC 1/SC 27/WG 3	Security evaluation, testing and specification
ISO/IEC JTC 1/SC 27/WG 4	Security controls and services

210	82	47	33
PUBLISHED ISO STANDARDS * under the direct responsibility of ISO/IEC JTC 1/SC 27	ISO STANDARDS UNDER DEVELOPMENT * under the direct responsibility of ISO/IEC JTC 1/SC 27	PARTICIPATING MEMBERS	OBSERVING MEMBERS

ISO/IEC JTC 1

ISO/IEC JTC 1/SC 42

Artificial intelligence

REFERENCE	TITLE
ISO/IEC JTC 1/SC 42/AG 2	AI Systems Engineering
ISO/IEC JTC 1/SC 42/AHG 1	Dissemination and outreach
ISO/IEC JTC 1/SC 42/AHG 2	Liaison with SC 38
ISO/IEC JTC 1/SC 42/AHG 4	Liaison with SC 27
ISO/IEC JTC 1/SC 42/AHG 5	AI standardization landscape and roadmap
ISO/IEC JTC 1/SC 42/JWG 1	Joint Working Group ISO/IEC JTC1/SC 42 - ISO/IEC JTC1/S
ISO/IEC JTC 1/SC 42/WG 1	Foundational standards
ISO/IEC JTC 1/SC 42/WG 2	Data
ISO/IEC JTC 1/SC 42/WG 3	Trustworthiness
ISO/IEC JTC 1/SC 42/WG 4	Use cases and applications

8	22	30	17
PUBLISHED ISO STANDARDS * under the direct responsibility of ISO/IEC JTC 1/SC 42	ISO STANDARDS UNDER DEVELOPMENT * under the direct responsibility of ISO/IEC JTC 1/SC 42	PARTICIPATING MEMBERS	OBSERVING MEMBERS

Systems

Openi 启智

新一代人工智能开源开放平台

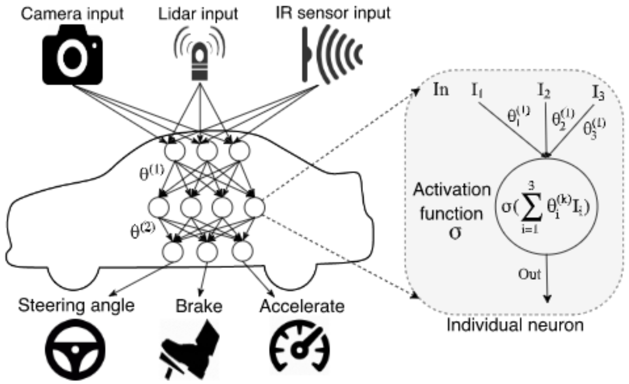
北京航空航天大學 BEIHANG UNIVERSITY

清華大學 Tsinghua University

启智重明 AISafety

https://openi.org.cn/AISafety/

DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars



UNIVERSITY of VIRGINIA

COLUMBIA UNIVERSITY

AdvBox

Bai du 百度 | 百度安全 有 AI 更 安 全

Adversarial Robustness Toolbox (ART) v1.3

IBM

docs passing | code style black | JOSS 10.21105/joss.02607 | pypi package 3.3.1

Foolbox Native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX

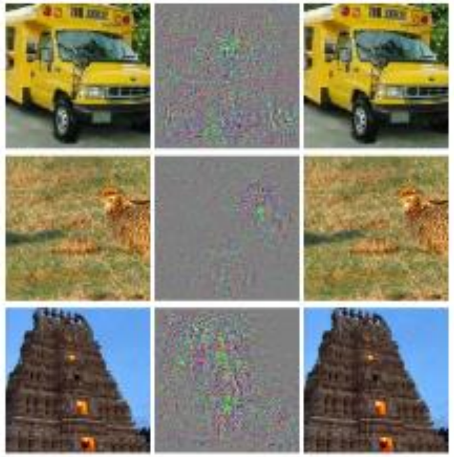
Foolbox is a **Python library** that lets you easily run adversarial attacks against machine learning models like deep neural networks. It is built on top of EagerPy and works natively with models in PyTorch, TensorFlow, and JAX.

build unknown | docs passing | version 1.6.1 | lgtm alerts 2 | codecov unknown | code style black | License MIT | python 3.6 | 3.7 | 3.8 | chat on slack

Outline

Adversarial Examples in Digital World

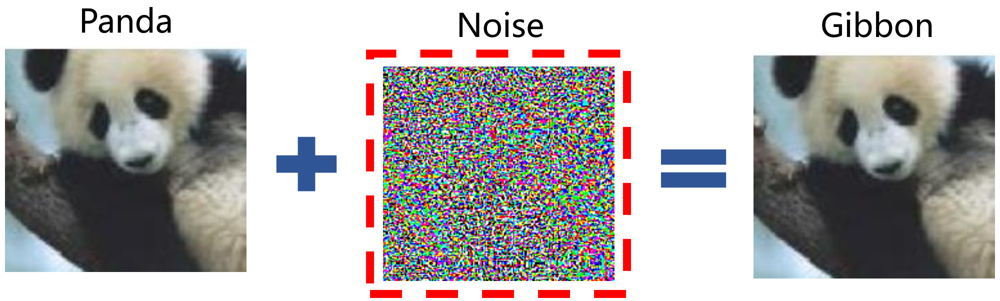
Christian Szegedy



Adversarial examples generated for AlexNet

Adversarial examples are somewhat universal and not just the results of overfitting to a particular model or to the specific selection of the training set

$$y^x \neq F_\theta(x+r) \quad s.t. \;\; r < \epsilon$$



“any AI that uses DNNs to classify inputs — such as speech — can be fooled”

Nature 2019.10

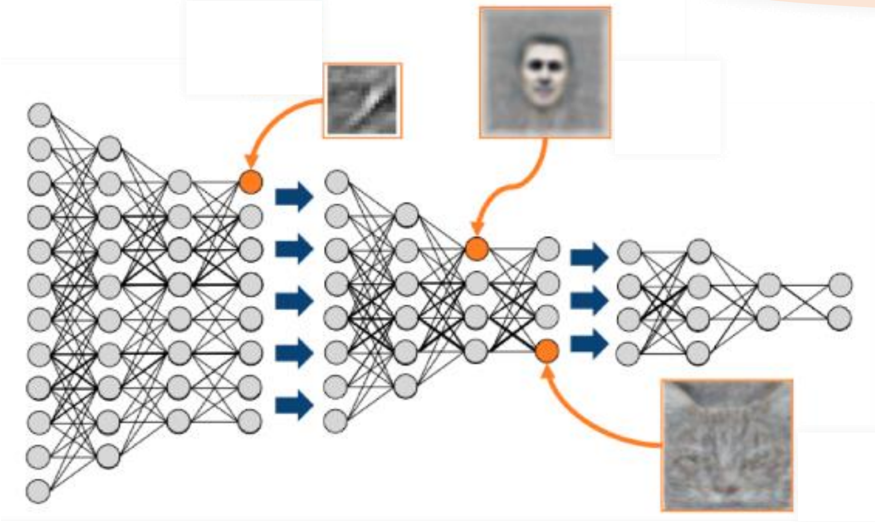
Attacks in the Digital World: the Overview

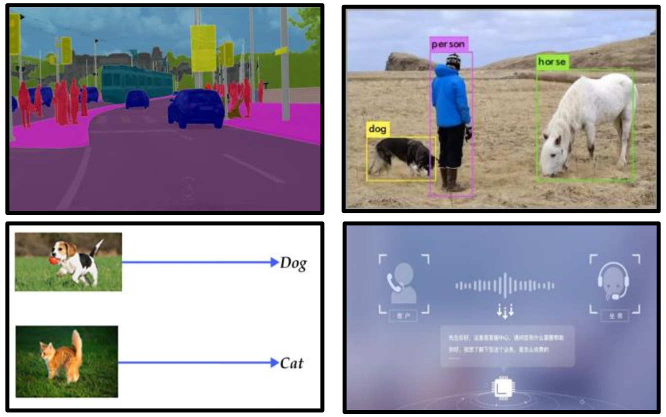
Digital attacks generate adversarial perturbations for input data in the digital pixel domain

$$y^x \neq F_\theta(x+r) \quad s.t. \quad r < \epsilon$$

Attacks

- White-box: Depend on model parameters
- Black-box: Not depend on model parameters





Several tasks

Summary

Method	Author	Attack Type	Year
FGSM attack	Goodfellow I. J.	Gradient-based attack	2014
C&W attack	Carlini N.	Optimization-based attack	2017
PGD attack	Madry A.	Gradient-based attack	2017
PBBA	Papernot N.	Transferability-based attack	2017
ZOO Attack	Chen P. Y.	Optimization-based attack	2017
BA	Brendel W.	Optimization-based attack	2017
EAD attack	Chen P. Y.	Optimization-based attack	2018
AdvGan	Xiao C.	Model-based attack	2018
CAR	Li T.	Interpretable-theory-based attack	2021

Gradient-based attack: FGSM attack

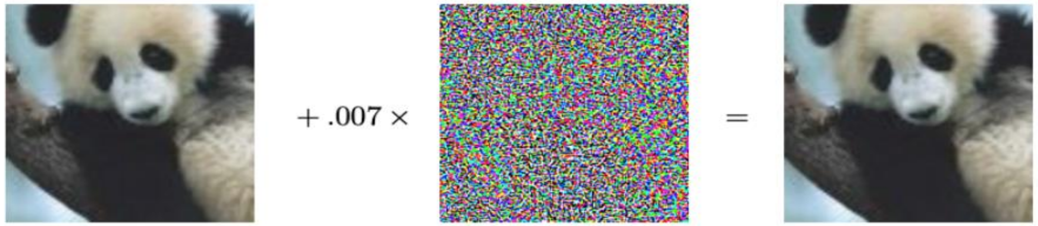
Fast Gradient Sign Method

$$\omega^T \tilde{x} = \omega^T (x + \eta) = \omega^T x + \omega^T \eta$$ **linear hypothesis**

- The fast gradient sign method trys to craft adversarial examples by using some gradient information during forward and backward in DNNs.

$$x' = x + \epsilon \text{sign}(\nabla_x \mathcal{L}(\theta, x, y))$$

- simple but effective adversarial attack

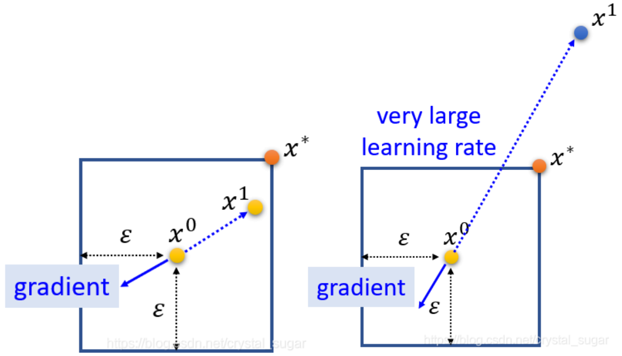


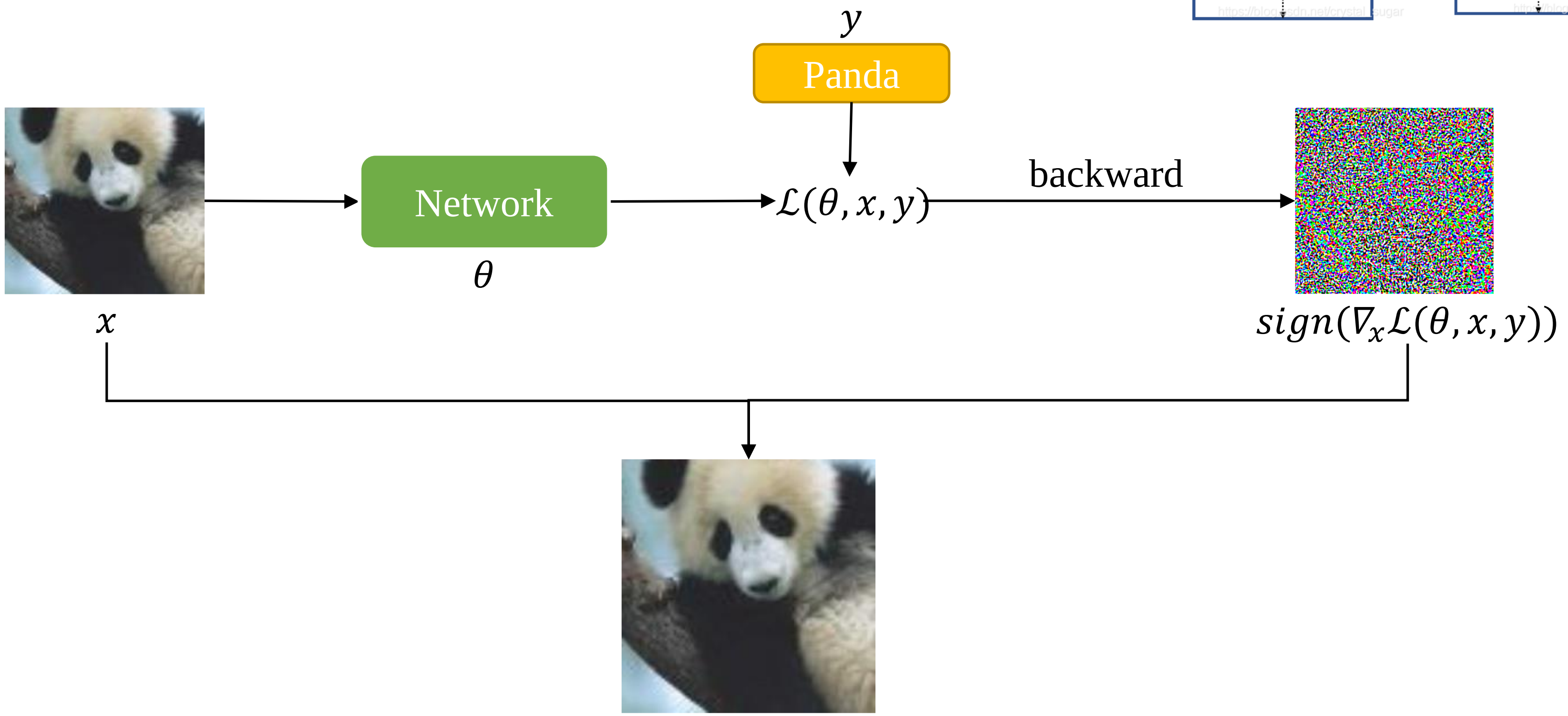
$\boldsymbol{x}$	$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$	$\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"panda"	"nematode"	"gibbon"
57.7% confidence	8.2% confidence	99.3 % confidence

Gradient-based attack: FGSM attack

Fast Gradient Sign Method

- FGSM use the gradient information of loss function.

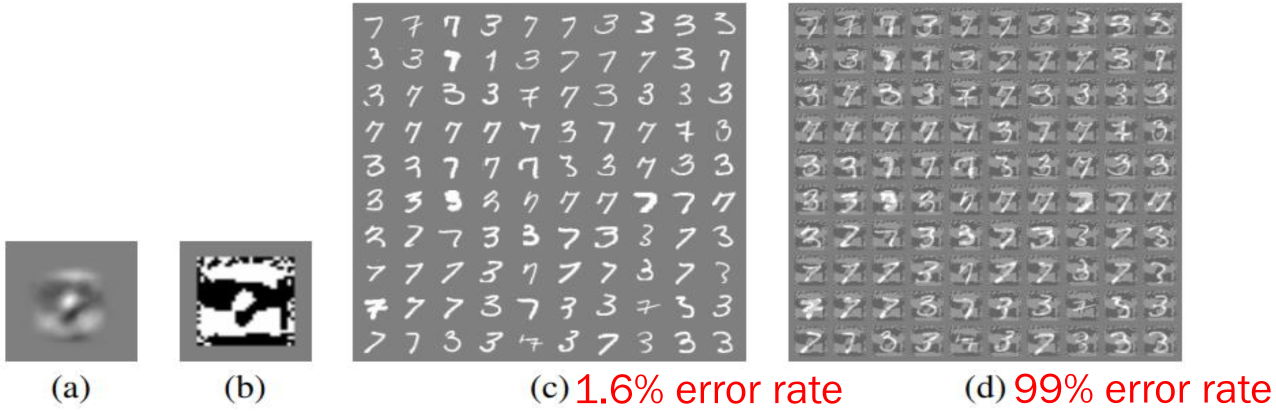




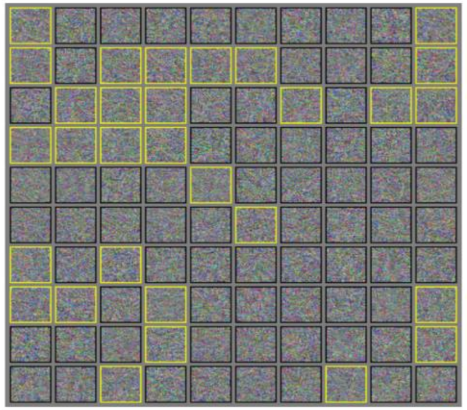
$$x' = x + \epsilon \text{sign}(\nabla_x \mathcal{L}(\theta, x, y))$$

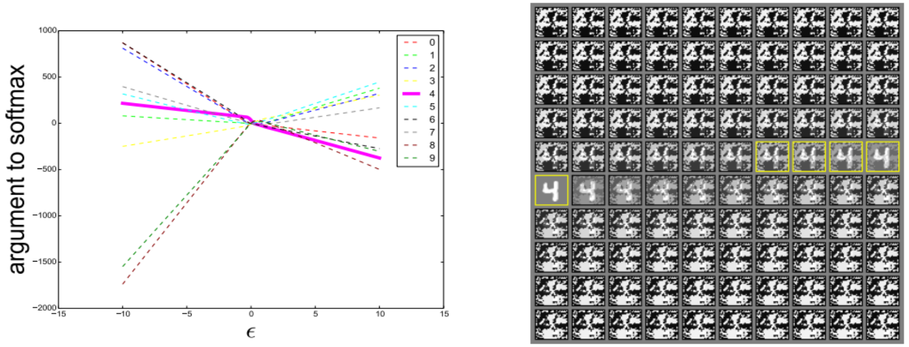
Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[J]. arXiv preprint arXiv:1412.6572, 2014.

Gradient-based attack: FGSM attack

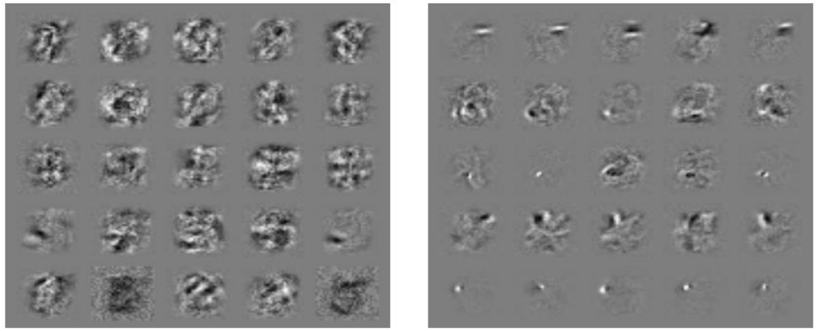


FGSM adversarial examples for logistic regression


Randomly fooling images



The influence of different epsilon values for FGSM


Weight visualizations on MNIST

Gradient-based attack: PGD attack

Projected Gradient Decent

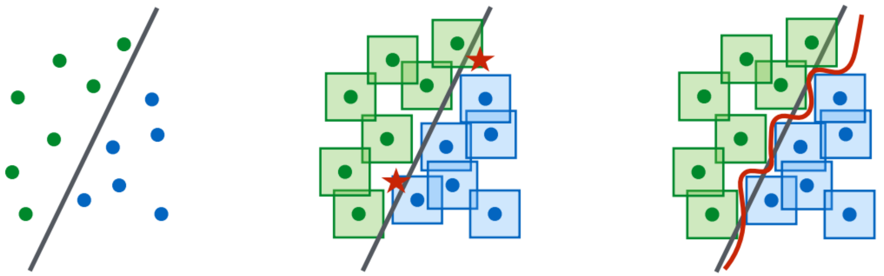
$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$

- Generate adversarial examples **by iteratively add small perturbations** on clean images like FGSM and project it to the epsilon ball.

$$x' = x + \epsilon\,\mathrm{sign}(\nabla_x \mathcal{L}(\theta, x, y))$$

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

- The **strongest attack, but time consuming**.



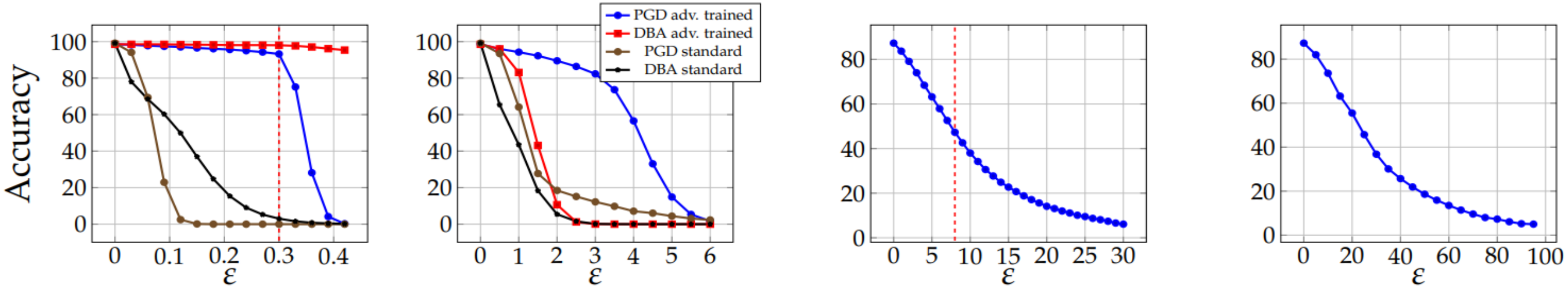
Gradient-based attack: PGD attack

CIFAR-10

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%

MNIST

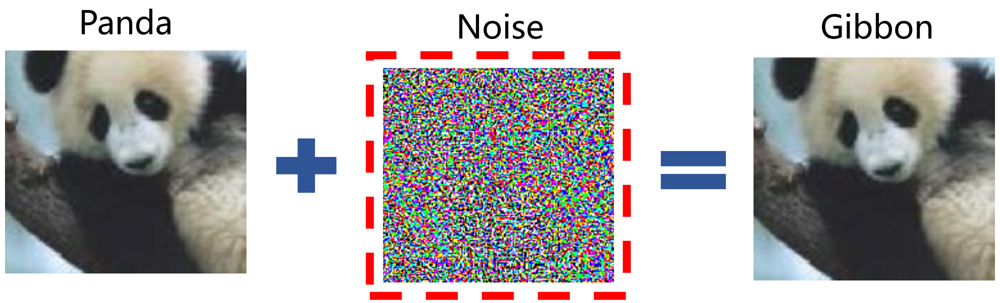
Method	Steps	Restarts	Source	Accuracy
Natural	-	-	-	98.8%
FGSM	-	-	A	95.6%
PGD	40	1	A	93.2%
PGD	100	1	A	91.8%
PGD	40	20	A	90.4%
PGD	100	20	A	**89.3%**
Targeted	40	1	A	92.7%
CW	40	1	A	94.0%
CW+	40	1	A	93.9%
FGSM	-	-	A′	96.8%
PGD	40	1	A′	96.0%
PGD	100	20	A′	**95.7%**
CW	40	1	A′	97.0%
CW+	40	1	A′	96.4%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



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

Optimization-based

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x + \delta \in [0,1]^n \end{aligned}$$



- The C&W attack meets both conditions by optimizing as follows:

$$\delta_i = \frac{1}{2}(\tanh(w_i) + 1) - x_i.$$

$$\begin{aligned} \text{minimize} \quad & \|\delta\|_p + c \cdot f(x+\delta) \\ \text{such that} \quad & x + \delta \in [0,1]^n \end{aligned} \Rightarrow$$

$$\text{minimize} \quad \|\frac{1}{2}(\tanh(w) + 1) - x\|_2^2 + c \cdot f(\frac{1}{2}(\tanh(w) + 1)$$

with f defined as

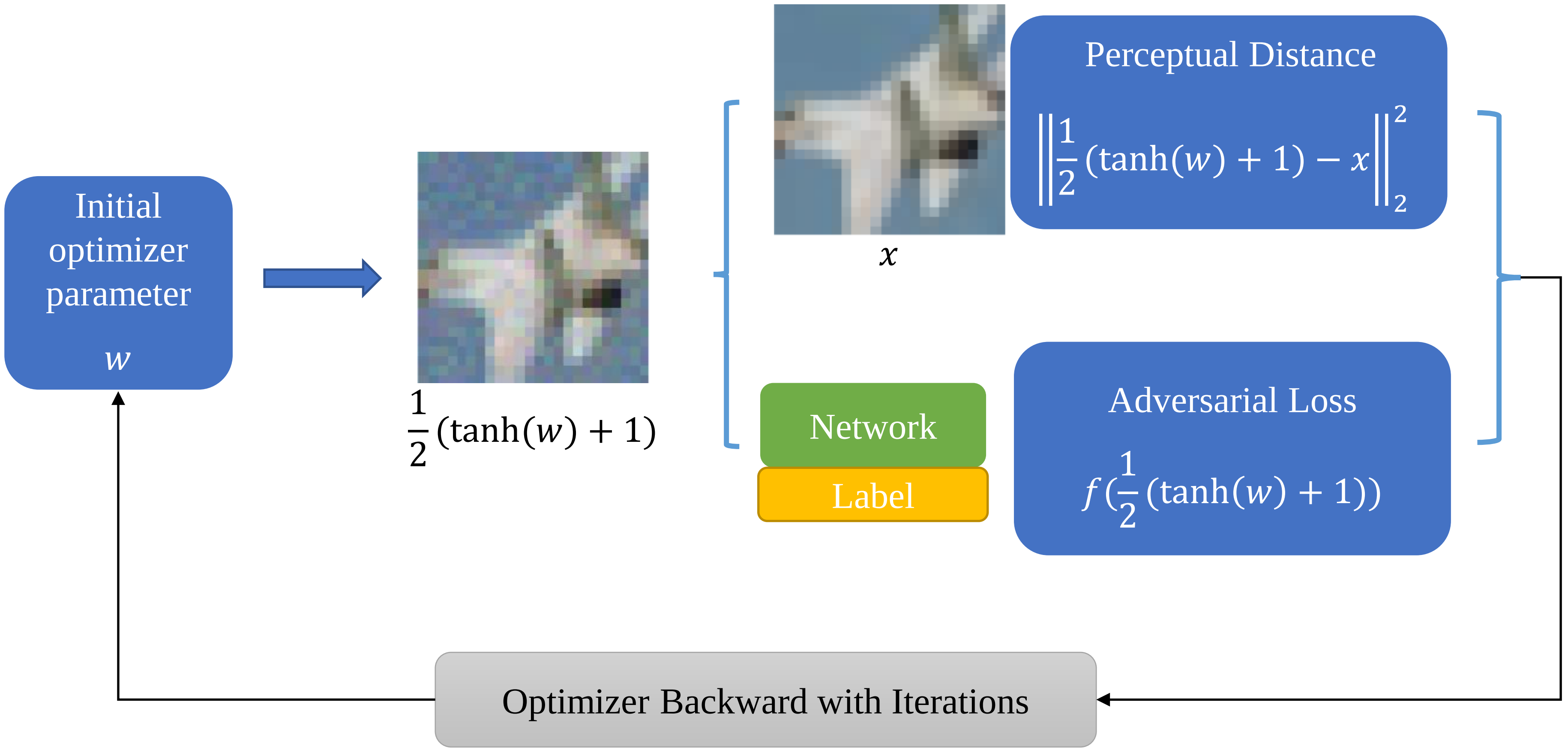
$$f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa).$$

Carlini N, Wagner D. Towards evaluating the robustness of neural networks[C]//2017 ieee symposium on security and privacy (sp). IEEE, 2017: 39-57.

Optimization-based attack: C&W attack

Framework

- C&W method use optimizer to minimize the object function.

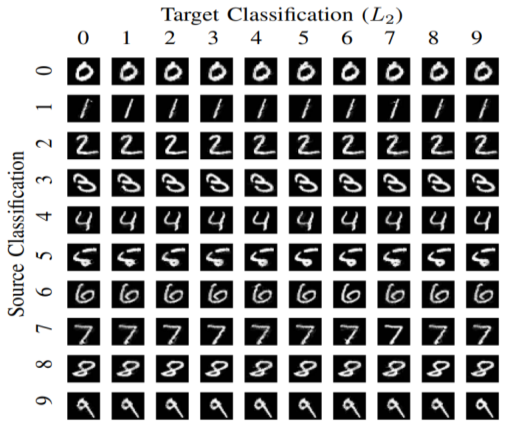


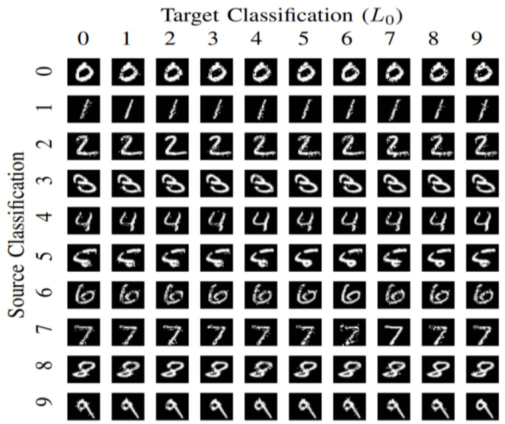
Optimization-based attack: C&W attack

	Best Case						Average Case						Worst Case					
	Change of Variable		Clipped Descent		Projected Descent		Change of Variable		Clipped Descent		Projected Descent		Change of Variable		Clipped Descent		Projected Descent	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
f_1	2.46	100%	2.93	100%	2.31	100%	4.35	100%	5.21	100%	4.11	100%	7.76	100%	9.48	100%	7.37	100%
f_2	4.55	80%	3.97	83%	3.49	83%	3.22	44%	8.99	63%	15.06	74%	2.93	18%	10.22	40%	18.90	53%
f_3	4.54	77%	4.07	81%	3.76	82%	3.47	44%	9.55	63%	15.84	74%	3.09	17%	11.91	41%	24.01	59%
f_4	5.01	86%	6.52	100%	7.53	100%	4.03	55%	7.49	71%	7.60	71%	3.55	24%	4.25	35%	4.10	35%
f_5	1.97	100%	2.20	100%	1.94	100%	3.58	100%	4.20	100%	3.47	100%	6.42	100%	7.86	100%	6.12	100%
f_6	1.94	100%	2.18	100%	1.95	100%	3.47	100%	4.11	100%	3.41	100%	6.03	100%	7.50	100%	5.89	100%
f_7	1.96	100%	2.21	100%	1.94	100%	3.53	100%	4.14	100%	3.43	100%	6.20	100%	7.57	100%	5.94	100%

There are many possible choices

$$f_1(x') = -\mathrm{loss}_{F,t}(x') + 1$$
$$f_2(x') = (\max_{i \neq t}(F(x')_i) - F(x')_t)^+$$
$$f_3(x') = \mathrm{softplus}(\max_{i \neq t}(F(x')_i) - F(x')_t) - \log(2)$$
$$f_4(x') = (0.5 - F(x')_t)^+$$
$$f_5(x') = -\log(2F(x')_t - 2)$$
$$f_6(x') = (\max_{i \neq t}(Z(x')_i) - Z(x')_t)^+$$
$$f_7(x') = \mathrm{softplus}(\max_{i \neq t}(Z(x')_i) - Z(x')_t) - \log(2)$$





Optimization-based attack: EAD attack

Elastic-Net Attack

- Formulated as **an elastic-net regularized optimization** problem.

$$\begin{aligned} \text{minimize} \quad & \|\delta\|_p + c \cdot f(x+\delta) \\ \text{such that} \quad & x + \delta \in [0,1]^n \end{aligned}$$

$$\Downarrow$$

$$\begin{aligned} \min_{\boldsymbol{x}} \; & c \cdot f(\boldsymbol{x}, t) + \beta \|\boldsymbol{x} - \boldsymbol{x}_0\|_1 + \|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{x} \in [0,1]^p \end{aligned}$$

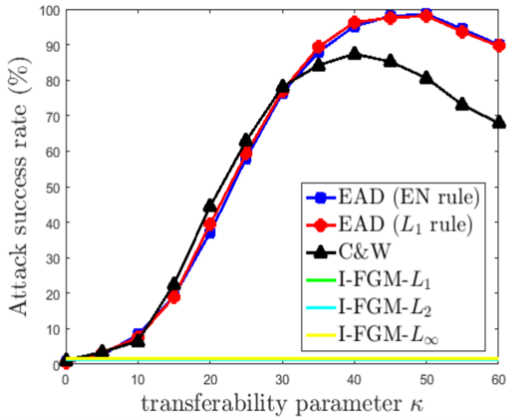
Algorithm 1 Elastic-Net Attacks to DNNs (EAD)

Input: original labeled image $(\mathbf{x}_0, t_0)$, target attack class t, attack transferability parameter κ, L_1 regularization parameter β, step size α_k, # of iterations I
Output: adversarial example $\mathbf{x}$
Initialization: $\mathbf{x}^{(0)} = \mathbf{y}^{(0)} = \mathbf{x}_0$
for $k = 0$ to $I - 1$ **do**
  $\mathbf{x}^{(k+1)} = S_\beta(\mathbf{y}^{(k)} - \alpha_k \nabla g(\mathbf{y}^{(k)}))$
  $\mathbf{y}^{(k+1)} = \mathbf{x}^{(k+1)} + \frac{k}{k+3}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$
end for
Decision rule: determine $\mathbf{x}$ from successful examples in $\{\mathbf{x}^{(k)}\}_{k=1}^{I}$ (EN rule or L_1 rule).

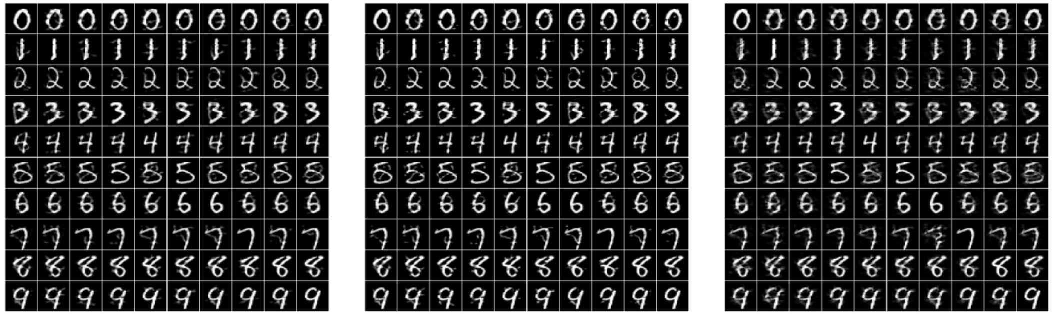
Iterative Shrinkage-Thresholding Algorithm

Optimization-based attack: EAD attack

Attack method	MNIST				CIFAR10				ImageNet			
	ASR	L_1	L_2	L_∞	ASR	L_1	L_2	L_∞	ASR	L_1	L_2	L_∞
C&W (L_2)	**100**	22.46	**1.972**	0.514	**100**	13.62	**0.392**	0.044	**100**	232.2	**0.705**	0.03
FGM-L_1	39	53.5	4.186	0.782	48.8	51.97	1.48	0.152	1	61	0.187	0.007
FGM-L_2	34.6	39.15	3.284	0.747	42.8	39.5	1.157	0.136	1	2338	6.823	0.25
FGM-L_∞	42.5	127.2	6.09	0.296	52.3	127.81	2.373	0.047	3	3655	7.102	0.014
I-FGM-L_1	**100**	32.94	2.606	0.591	**100**	17.53	0.502	0.055	77	526.4	1.609	0.054
I-FGM-L_2	**100**	30.32	2.41	0.561	**100**	17.12	0.489	0.054	**100**	774.1	2.358	0.086
I-FGM-L_∞	**100**	71.39	3.472	**0.227**	**100**	33.3	0.68	**0.018**	**100**	864.2	2.079	**0.01**
EAD (EN rule)	**100**	**17.4**	2.001	0.594	**100**	**8.18**	0.502	0.097	**100**	**69.47**	1.563	0.238
EAD (L_1 rule)	**100**	**14.11**	2.211	0.768	**100**	**6.066**	0.613	0.17	**100**	**40.9**	1.598	0.293



Transferability of parameter k



(a) EAD (EN rule) (b) EAD (L_1 rule) (c) I-FGM-L_1

Different adversarial examples on MNIST

Optimization-based attack: ZOO Attack

Zeroth Order Optimization attack

- Directly estimate the gradients of the targeted DNN: zeroth order stochastic coordinate descent, with hierarchical attack and importance sampling techniques

Algorithm 2 ZOO-ADAM: Zeroth Order Stochastic Coordinate Descent with Coordinate-wise ADAM

Require: Step size η, ADAM states $M \in \mathbb{R}^p, v \in \mathbb{R}^p, T \in \mathbb{Z}^p$, ADAM hyper-parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$

$M \leftarrow \mathbf{0}, v \leftarrow \mathbf{0}, T \leftarrow \mathbf{0}$
while not converged **do**
  Randomly pick a coordinate $i \in \{1, \cdots, p\}$
  Estimate $\hat{g}_i$ using (6)
  $T_i \leftarrow T_i + 1$
  $M_i \leftarrow \beta_1 M_i + (1-\beta_1)\hat{g}_i, \quad v_i \leftarrow \beta_2 v_i + (1-\beta_2)\hat{g}_i^2$
  $\hat{M}_i = M_i/(1-\beta_1^{T_i}), \quad \hat{v}_i = v_i/(1-\beta_2^{T_i})$
  $\delta^* = -\eta \frac{\hat{M}_i}{\sqrt{\hat{v}_i}+\epsilon}$
  Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
end while

Algorithm 3 ZOO-Newton: Zeroth Order Stochastic Coordinate Descent with Coordinate-wise Newton's Method

Require: Step size η

while not converged **do**
  Randomly pick a coordinate $i \in \{1, \cdots, p\}$
  Estimate $\hat{g}_i$ and $\hat{h}_i$ using (6) and (7)
  if $\hat{h}_i \leq 0$ **then**
    $\delta^* \leftarrow -\eta \hat{g}_i$
  else
    $\delta^* \leftarrow -\eta \frac{\hat{g}_i}{\hat{h}_i}$
  end if
  Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
end while

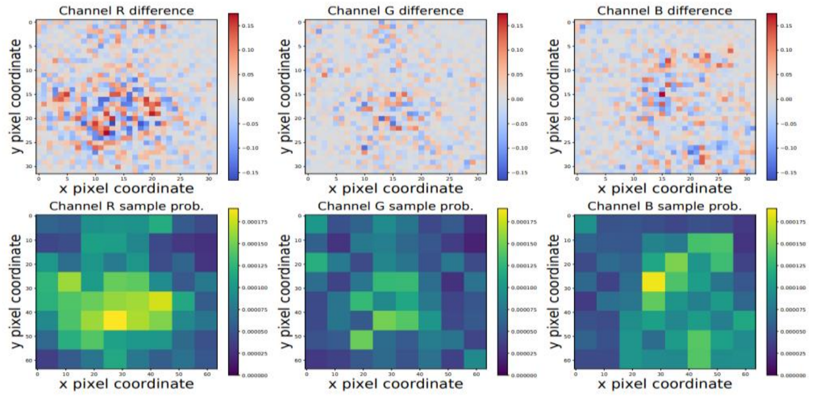
ZOO Algorithm

- Spare the need for training substitute models and avoiding the loss in attack transferability.

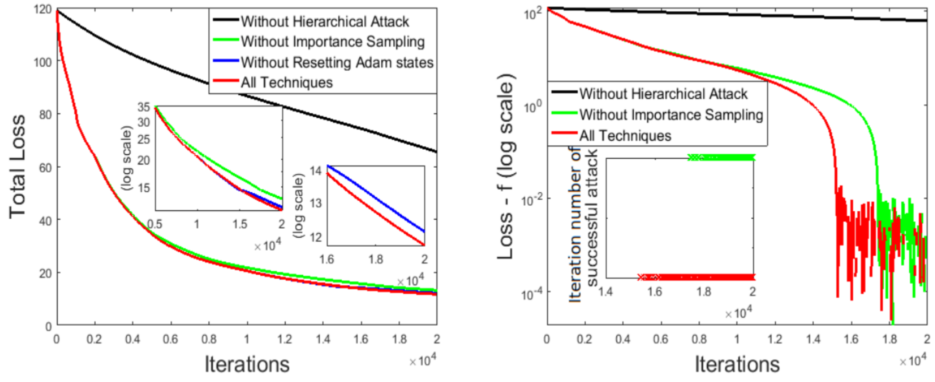
Optimization-based attack: ZOO Attack

ASR and Average Time on MNIST and CIFAR-10

	MNIST					
	Untargeted			Targeted		
	Success Rate	Avg. L_2	Avg. Time (per attack)	Success Rate	Avg. L_2	Avg. Time (per attack)
White-box (C&W)	100 %	1.48066	0.48 min	100 %	2.00661	0.53 min
Black-box (Substitute Model + FGSM)	40.6 %	-	0.002 sec (+ 6.16 min)	7.48 %	-	0.002 sec (+ 6.16 min)
Black-box (Substitute Model + C&W)	33.3 %	3.6111	0.76 min (+ 6.16 min)	26.74 %	5.272	0.80 min (+ 6.16 min)
Proposed black-box (ZOO-ADAM)	100 %	1.49550	1.38 min	98.9 %	1.987068	1.62 min
Proposed black-box (ZOO-Newton)	100 %	1.51502	2.75 min	98.9 %	2.057264	2.06 min
	CIFAR10					
	Untargeted			Targeted		
	Success Rate	Avg. L_2	Avg. Time (per attack)	Success Rate	Avg. L_2	Avg. Time (per attack)
White-box (C&W)	100 %	0.17980	0.20 min	100 %	0.37974	0.16 min
Black-box (Substitute Model + FGSM)	76.1 %	-	0.005 sec (+ 7.81 min)	11.48 %	-	0.005 sec (+ 7.81 min)
Black-box (Substitute Model + C&W)	25.3 %	2.9708	0.47 min (+ 7.81 min)	5.3 %	5.7439	0.49 min (+ 7.81 min)
Proposed Black-box (ZOO-ADAM)	100 %	0.19973	3.43 min	96.8 %	0.39879	3.95 min
Proposed Black-box (ZOO-Newton)	100 %	0.23554	4.41 min	97.0 %	0.54226	4.40 min



Channel Difference and Sample Probability



Loss with Iterations

Optimization-based attack: BA

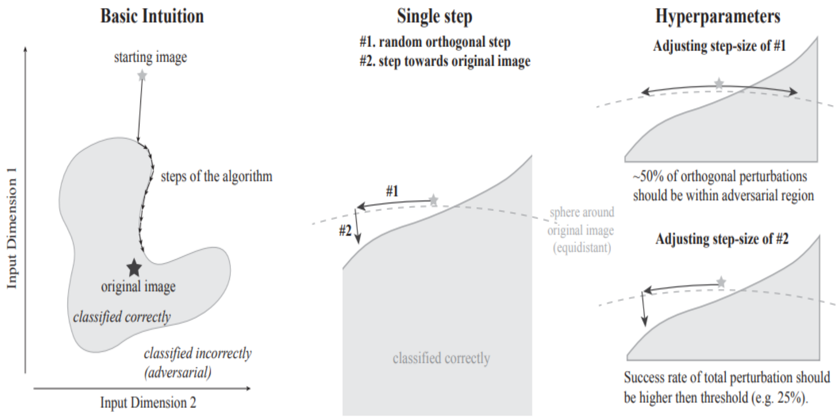
Boundary Attack

- A decision-based attack that starts from a large adversarial perturbation and then seeks to reduce the perturbation while staying adversarial.

Data: original image $\mathbf{o}$, adversarial criterion $c(.)$, decision of model $d(.)$
Result: adversarial example $\tilde{o}$ such that the distance $d(\boldsymbol{o}, \tilde{\boldsymbol{o}}) = \|\boldsymbol{o} - \tilde{\boldsymbol{o}}\|_2^2$ is minimized
initialization: $k = 0$, $\tilde{o}^0 \sim \mathcal{U}(0, 1)$ s.t. $\tilde{o}^0$ is adversarial;
while k < *maximum number of steps* **do**
  draw random perturbation from proposal distribution $\eta_k \sim \mathcal{P}(\tilde{o}^{k-1})$;
  if $\tilde{o}^{k-1} + \eta_k$ *is adversarial* **then**
    set $\tilde{o}^k = \tilde{o}^{k-1} + \eta_k$;
  else
    set $\tilde{o}^k = \tilde{o}^{k-1}$;
  end
  $k = k + 1$
end

Algorithm 1: Minimal version of the Boundary Attack.

BA Algorithm



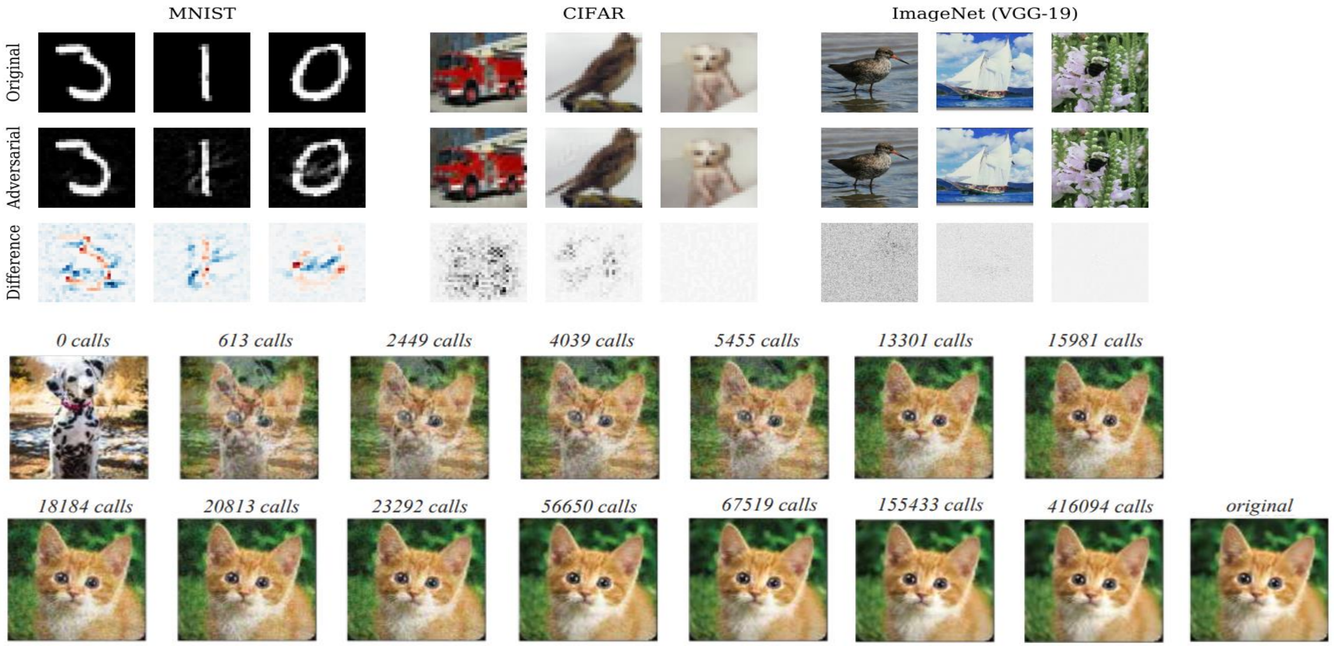
Boundary Example

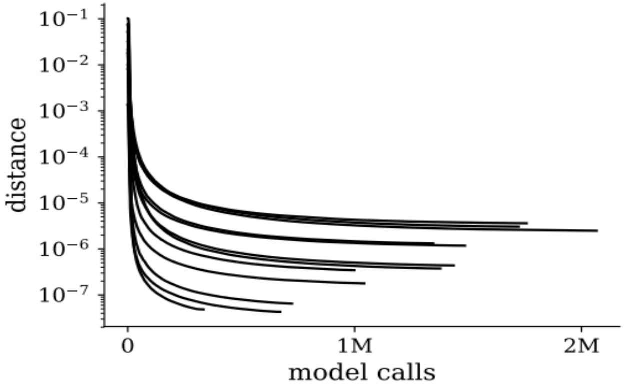
- Do not rely on substitute models, but should query many times

Optimization-based attack: BA

ASR and L2 distance metric on different methods

	Attack Type	MNIST	CIFAR	ImageNet		
				VGG-19	ResNet-50	Inception-v3
FGSM	gradient-based	4.2e-02	2.5e-05	1.0e-06	1.0e-06	9.7e-07
DeepFool	gradient-based	4.3e-03	5.8e-06	1.9e-07	7.5e-08	5.2e-08
Carlini & Wagner	gradient-based	2.2e-03	7.5e-06	5.7e-07	2.2e-07	7.6e-08
Boundary (ours)	decision-based	3.6e-03	5.6e-06	2.9e-07	1.0e-07	6.5e-08



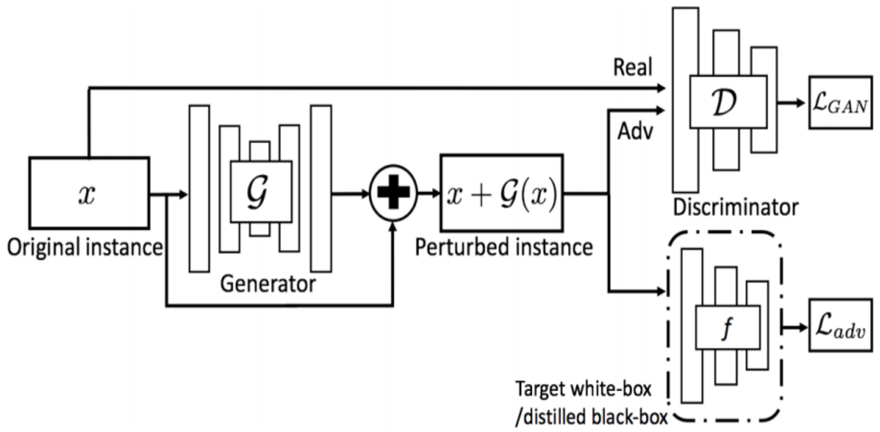


Adversarial examples and boundary distance with different calls

Model-based attack: AdvGAN

Adversarial Generative Adversarial Network

- generate adversarial examples with generative adversarial networks



AdvGAN Framework

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_x \log \mathcal{D}(x) + \mathbb{E}_x \log(1 - \mathcal{D}(x + \mathcal{G}(x))).$$

$$\mathcal{L}^f_{\text{adv}} = \mathbb{E}_x \ell_f(x + \mathcal{G}(x), t),$$

$$\mathcal{L}_{\text{hinge}} = \mathbb{E}_x \max(0, \|\mathcal{G}(x)\|_2 - c),$$

$$\mathcal{L} = \mathcal{L}^f_{\text{adv}} + \alpha\mathcal{L}_{\text{GAN}} + \beta\mathcal{L}_{\text{hinge}},$$

Loss Function

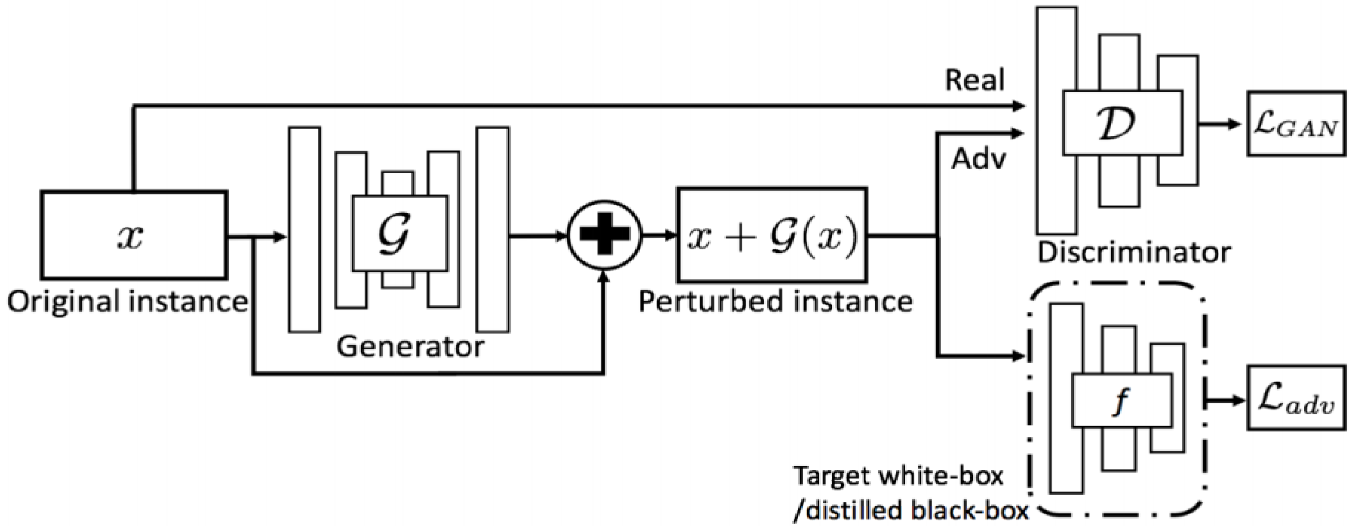
- potentially accelerate adversarial training as defenses.

Model-based attack: AdvGAN

Framework

- Generator $\mathcal{G}$ generates adversarial perturbation $\mathcal{G}(x)$.
- Discriminator $\mathcal{D}$ compares x with $x + \mathcal{G}(x)$. => $\mathcal{L}_{GAN}$
- Target model f classifies adversarial example $x + \mathcal{G}(x)$. => $\mathcal{L}_{adv}$
- Hinge loss is used to normalize and stabilize the training. => $\mathcal{L}_{hinge}$

$\mathcal{L}$



AdvGAN Framework

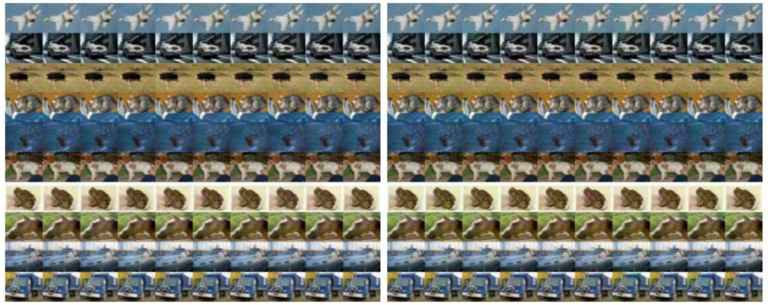
Model-based attack: AdvGAN

ASR on MNIST and CIFAR-10

Data	Model	Defense	FGSM	Opt.	AdvGAN
MNIST	A	Adv.	4.3%	4.6%	**8.0%**
		Ens.	1.6%	4.2%	**6.3%**
		Iter.Adv.	4.4%	2.96%	**5.6%**
	B	Adv.	6.0%	4.5%	**7.2%**
		Ens.	2.7%	3.18%	**5.8%**
		Iter.Adv.	**9.0%**	3.0%	6.6%
	C	Adv.	2.7%	2.95%	**18.7%**
		Ens.	1.6%	2.2%	**13.5%**
		Iter.Adv.	1.6%	1.9%	**12.6%**
CIFAR10	ResNet	Adv.	13.10%	11.9%	**16.03%**
		Ens.	10.00%	10.3%	**14.32%**
		Iter.Adv	22.8%	21.4%	**29.47%**
	Wide ResNet	Adv.	5.04%	7.61%	**14.26%**
		Ens.	4.65%	8.43%	**13.94 %**
		Iter.Adv.	14.9%	13.90%	**20.75%**

Model	MNIST(%) A	MNIST(%) B	MNIST(%) C	CIFAR-10(%) ResNet	CIFAR-10(%) Wide ResNet
Accuracy (p)	99.0	99.2	99.1	92.4	95.0
Attack Success Rate (w)	97.9	97.1	98.3	94.7	99.3
Attack Success Rate (b-D)	93.4	90.1	94.0	78.5	81.8
Attack Success Rate (b-S)	30.7	66.6	87.3	10.3	13.3

Adversarial Examples

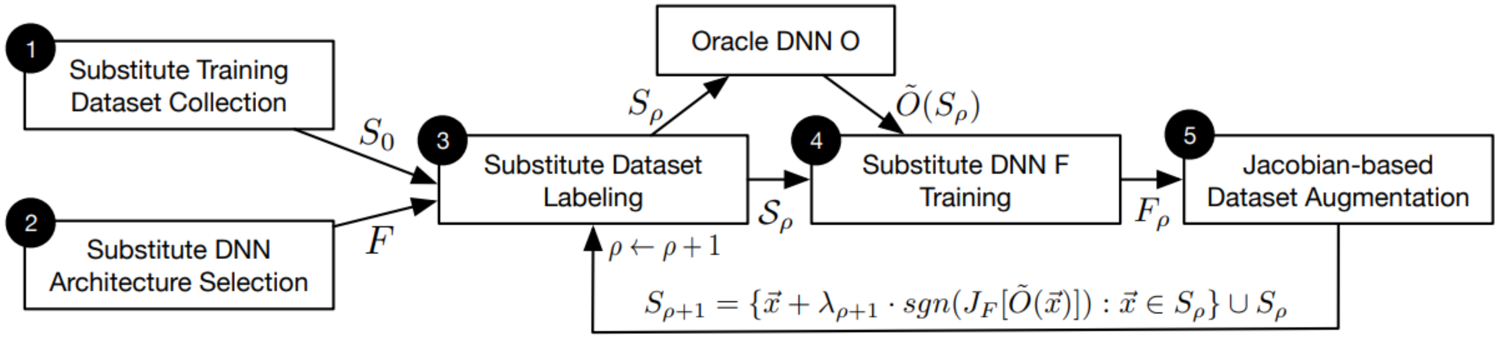


(a) Semi-whitebox attack (b) Black-box attack

Transfer-based attack: PBBA

□ Practical Black-Box Attacks

- Train a parallel model called substitute model to emulate the original model



PBBA Framework

- First practical demonstration of an attacker controlling a remotely hosted DNN **with no knowledge about the model internals or its training data**

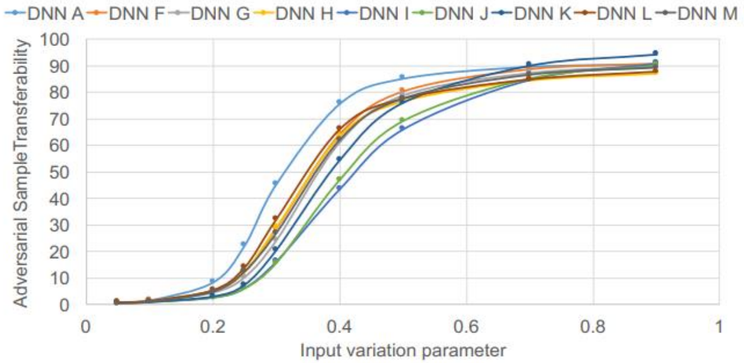
Transfer-based attack: PBBA

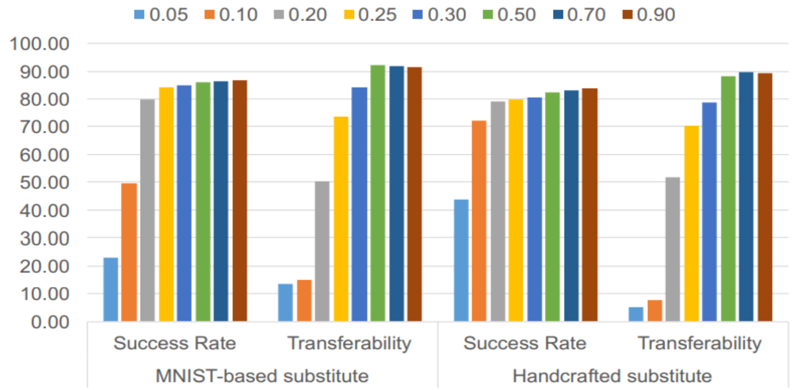
Attack Accuracy and Transferability

Substitute Epoch	Initial Substitute Training Set from MNIST test set	Initial Substitute Training Set from Handcrafted digits
0	24.86%	18.70%
1	41.37%	19.89%
2	65.38%	29.79%
3	74.86%	36.87%
4	80.36%	40.64%
5	79.18%	56.95%
6	81.20%	67.00%

DNN ID	Accuracy ($\rho = 2$)	Accuracy ($\rho = 6$)	Transferability ($\rho = 6$)
A	30.50%	82.81%	75.74%
F	68.67%	79.19%	64.28%
G	72.88%	78.31%	61.17%
H	56.70%	74.67%	63.44%
I	57.68%	71.25%	43.48%
J	64.39%	68.99%	47.03%
K	58.53%	70.75%	54.45%
L	67.73%	75.43%	65.95%
M	62.64%	76.04	62.00%

Hyper-parameters and Transferability

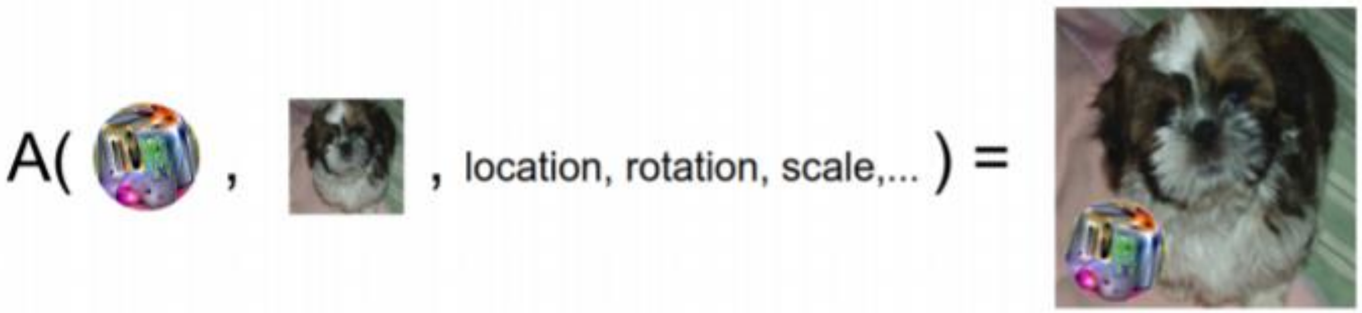




Adversarial Patch: Image Classification

☐ Adversarial Patch

- create universal, robust targeted adversarial image patches in the **real world**
- These adversarial patches can be painted, added to any scene.



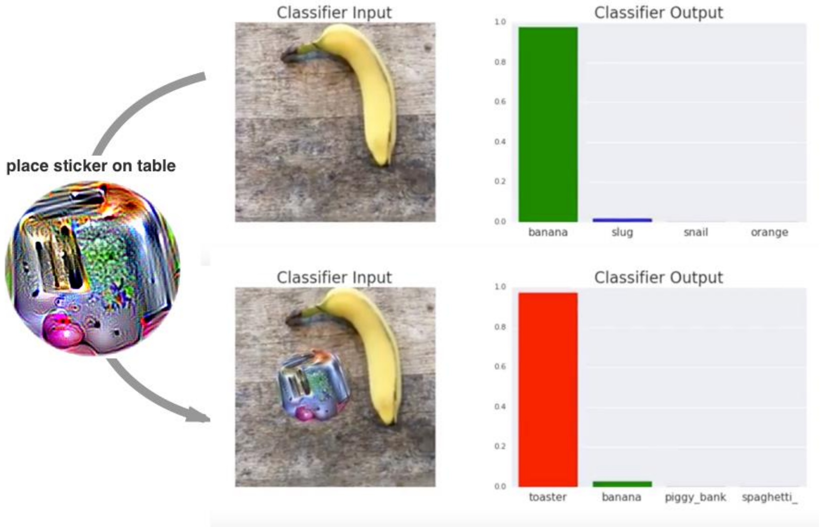
☐ Basic Algorithm

- Prepare classifier, input, and target class
- Find the input to maximizes the $Log(P[y|x])$
- Perform iterated gradient descent on input x
- Produce a well camouflaged attack
- Patch the p to the image x

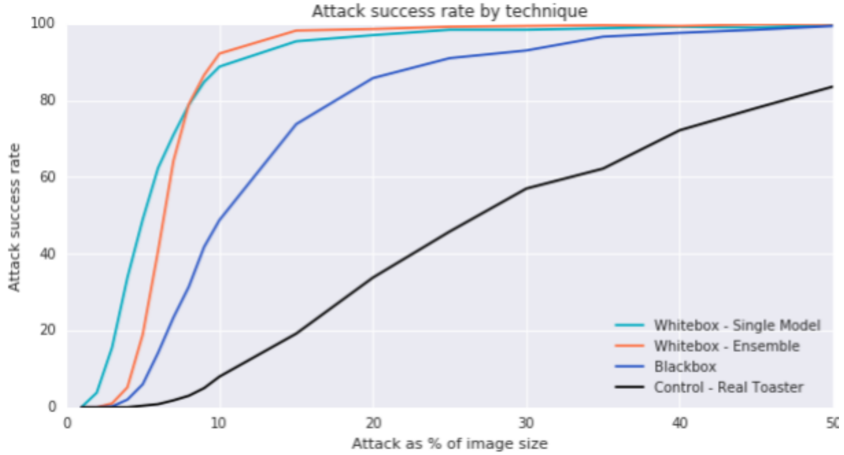
$$\hat{p} = \arg\max_{p} \mathbb{E}_{x\sim X, t\sim T, l\sim L}\left[\log \Pr\left(\hat{y}|A(p,x,l,t)\right)\right]$$

Adversarial Patch: Image Classification

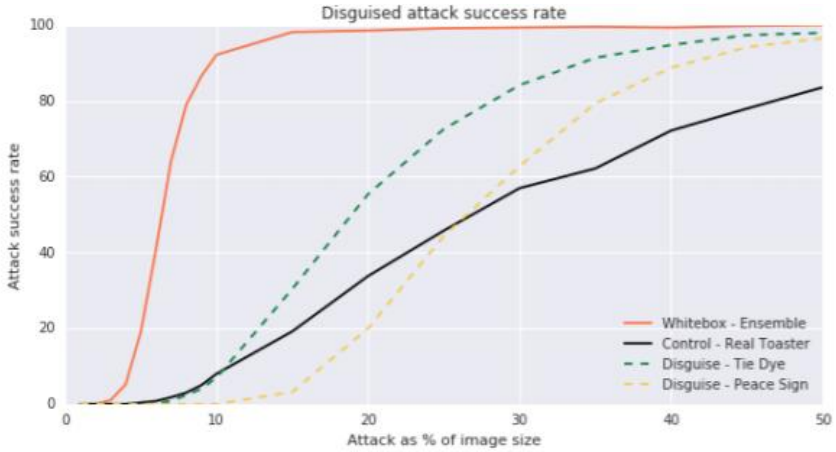
Real-world attack on VGG16



Comparison of different methods for creating adversarial patches



Comparison of patches with various disguises



Focusing only on defending against small perturbations is insufficient, as large, local perturbations can also break classifiers

Tasks: Object Detection

Patch on corner can affect the whole image

Dpatch

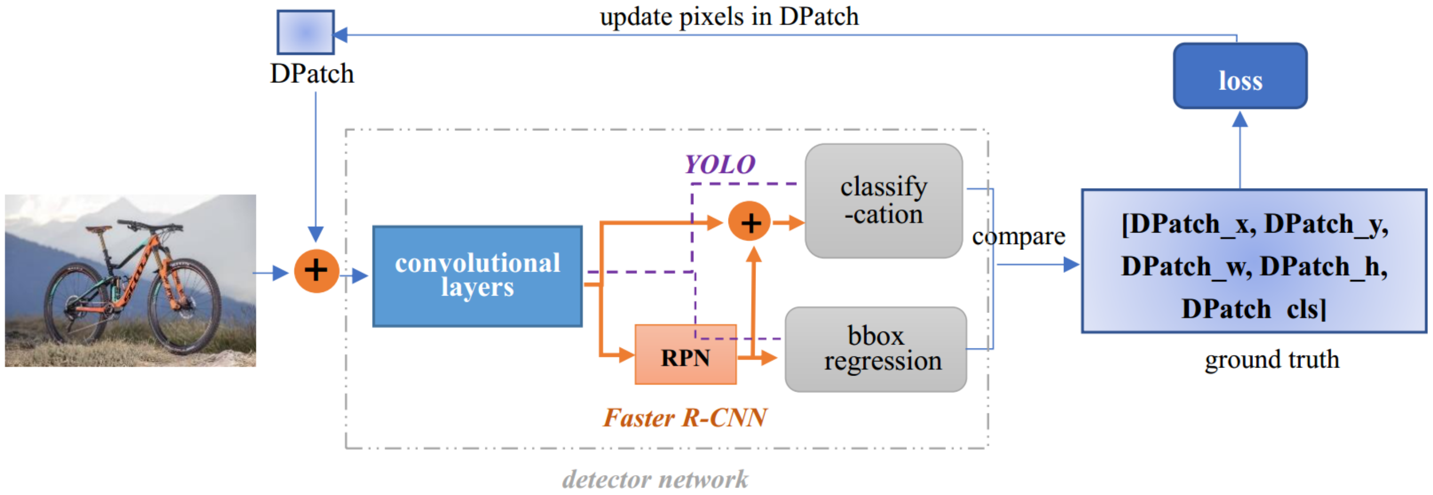
- Randomly located
- Only perturb pixels in patch
- Use both classification and regression losses

No DPatch

With DPatch

YOLO cannot detect bike after adding DPatch

Overview of the Dpatch training system



Tasks: Object Detection

Results on Pascal VOC 2007

Table 1: Results on Pascal VOC 2007 test set with Fast R-CNN and ResNet101 when applying DPATCH of different types

Faster R-CNN	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
no DPATCH	74.80	80.20	77.60	64.50	61.50	81.10	86.70	86.40	55.70	89.30	69.60
untargeted DPATCH	0.10	3.20	4.30	0.00	5.40	0.00	9.80	0.00	11.20	10.60	5.20
targeted DPATCH	0.02	0.00	0.00	0.00	0.00	0.53	0.08	0.61	0.00	0.02	0.00
YOLO trained DPATCH	2.27	**0.51**	**0.87**	2.27	**0.78**	1.52	4.55	0.62	1.17	3.03	2.10

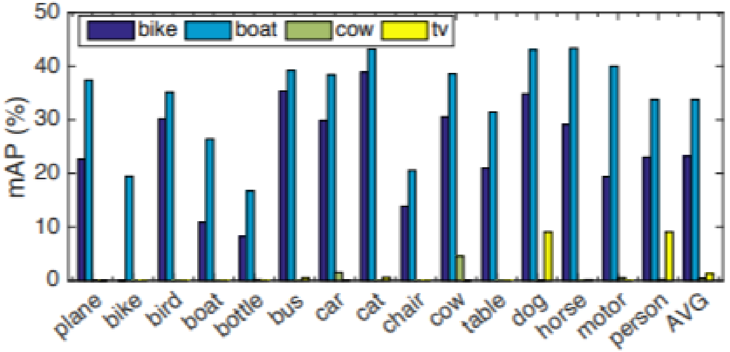
	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
no DPATCH	87.40	84.50	80.00	78.60	47.70	76.00	74.60	76.60	73.70	75.10
untargeted DPATCH	0.30	0.59	0.00	1.69	0.00	4.68	0.00	0.00	1.00	**2.90**
targeted DPATCH	9.09	0.16	0.00	9.09	0.16	0.00	9.09	0.00	0.00	**0.98**
YOLO trained DPATCH	2.02	3.37	1.30	0.94	0.53	0.43	3.03	1.52	1.52	**1.72**

Table 2: Results on Pascal VOC 2007 test set with YOLO when applying DPATCH of different types

YOLO	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
no DPATCH	69.50	75.60	64.00	52.30	35.60	73.40	74.00	79.60	42.10	66.10	66.90
untargeted DPATCH	0.00	1.50	9.10	1.30	9.10	0.00	9.10	0.00	9.10	9.10	0.40
targeted DPATCH	0.00	4.55	9.09	0.00	0.09	0.00	9.09	1.82	0.01	0.00	0.36
Faster R-CNN trained DPATCH	0.01	**0.00**	**0.23**	0.02	**0.00**	0.00	0.00	0.00	0.01	0.00	0.00

	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
no DPATCH	78.10	80.10	78.20	65.90	41.70	62.00	67.60	77.60	63.10	65.70
untargeted DPATCH	0.00	0.00	0.00	0.00	9.10	9.10	0.00	0.00	1.00	**0.00**
targeted DPATCH	0.01	0.00	0.00	1.73	0.00	0.00	1.07	0.00	9.09	**1.85**
Faster R-CNN trained DPATCH	0.00	0.03	0.00	0.07	0.00	0.00	0.00	0.00	0.01	**0.02**

Per class mAP after DPatch attack



Conclusions

- Perform effective attacks
- Small in size
- Location-independent
- Great transferability

Tasks: Video Analysis

☐ Motion Excited Sampler

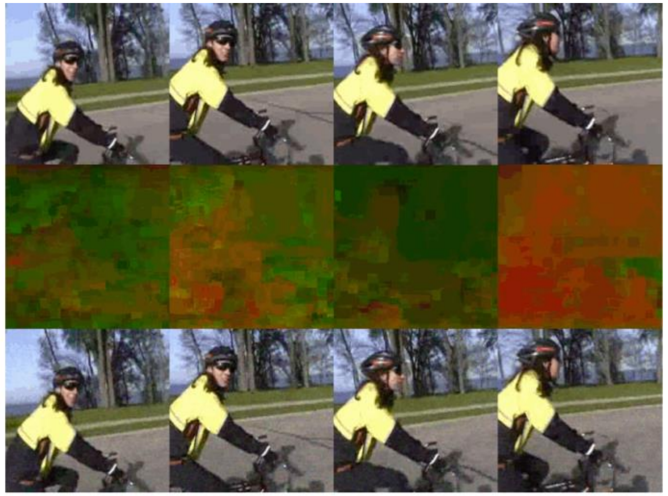
- Attack video models: Motion recognition and classification
- **sparked prior:** Use inter-frame knowledge and sparked by motion information

Example of Adversarial samples

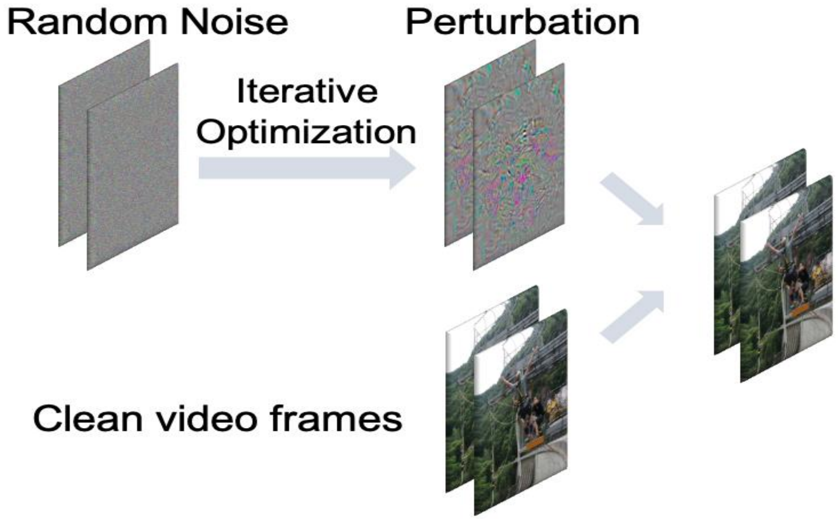
Original (Biking)

Noise

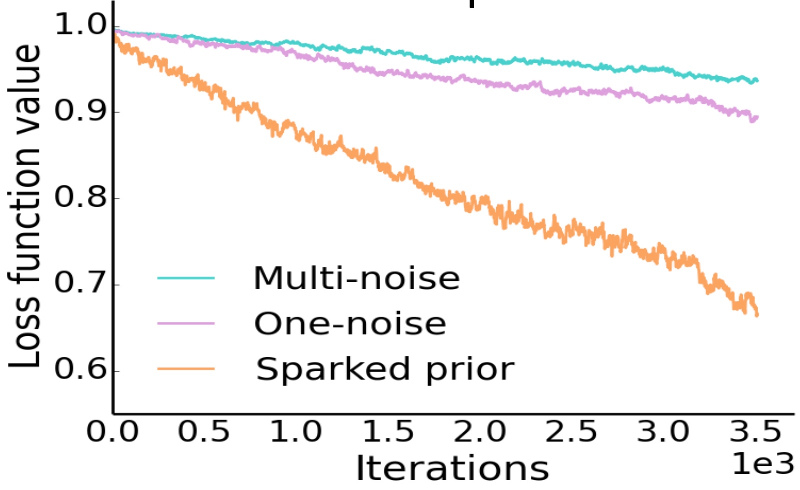
Adversarial (Walking with dog)



Pipeline of generating adversarial examples



Loss curve comparison



Tasks: Video Analysis

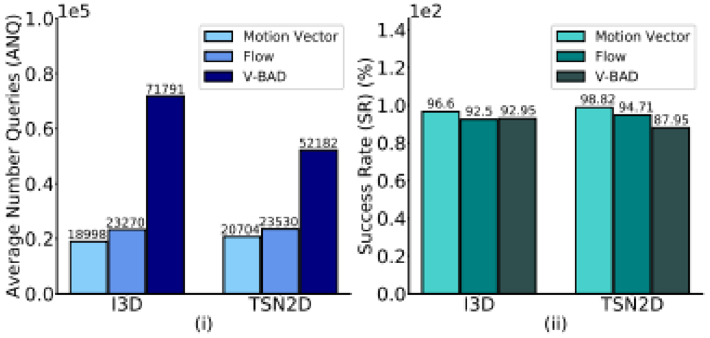
◻ Experimental Settings

- Evaluate on different dataset/model
- Calculate success rate
- Count average number of queries

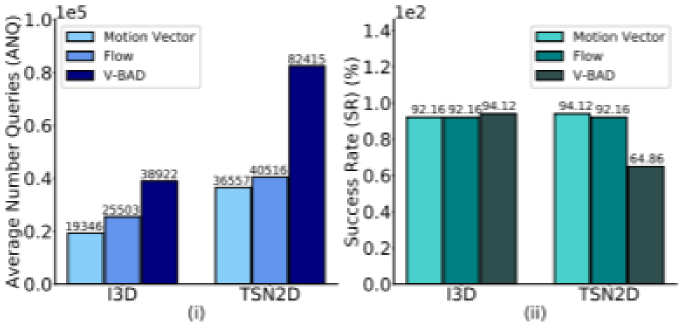
Untargeted attacks on several datasets

Dataset / Model	Method	I3D ANQ	I3D SR(%)	TSN2D ANQ	TSN2D SR(%)
SthSth-V2	E-NES [13]	11,552	86.96	1,698	99.41
	E-Bandits [14]	968	100.0	435	99.41
	V-BAD [17]	7,239	97.70	495	100.0
	ME-Sampler (OF)	735	98.90	315	100.0
	ME-Sampler (MV)	**592**	**100.0**	**244**	**100.0**
HMDB-51	E-NES [13]	13,237	84.31	19,407	76.47
	E-Bandits [14]	4,549	99.80	4,261	100.0
	V-BAD [17]	5,064	100.0	2,405	100.0
	ME-Sampler (OF)	**3,306**	**100.0**	842	100.0
	ME-Sampler (MV)	3,915	100.0	**831**	**100.0**
Kinetics-400	E-NES [13]	11,423	89.30	20,698	71.93
	E-Bandits [14]	3,697	99.00	6,149	97.50
	V-BAD [17]	4,047	**99.75**	2,623	99.75
	ME-Sampler (OF)	3,415	99.30	2,631	98.80
	ME-Sampler (MV)	**2,717**	99.00	**1,715**	**99.75**
UCF-101	E-NES [13]	23,531	69.23	41,328	34.65
	E-Bandits [14]	10,590	89.10	24,890	66.33
	V-BAD [17]	8,819	97.03	17,638	91.09
	ME-Sampler (OF)	6,101	96.00	6,598	97.00
	ME-Sampler (MV)	**4,748**	**98.02**	**5,353**	**99.00**

Comparisons of targeted attack on SthSth-V2 and HMDB-51



(a) SthSth-V2



(b) HMDB-51

☐ DeepWordBug

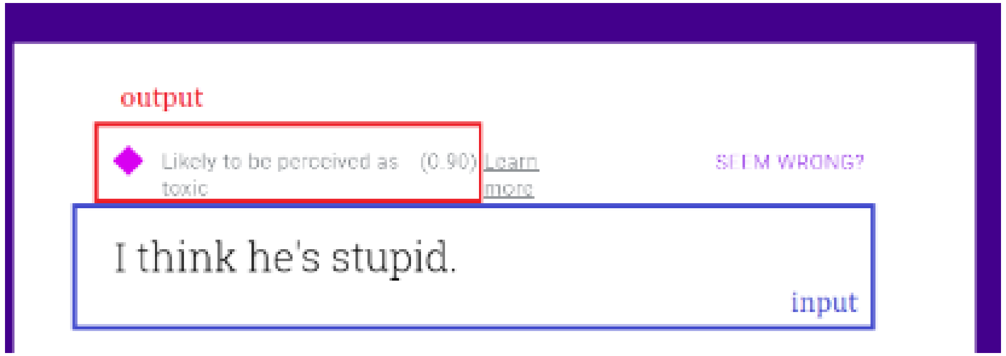
effectively generate small text perturbations in a black-box setting for deep-learning classifier

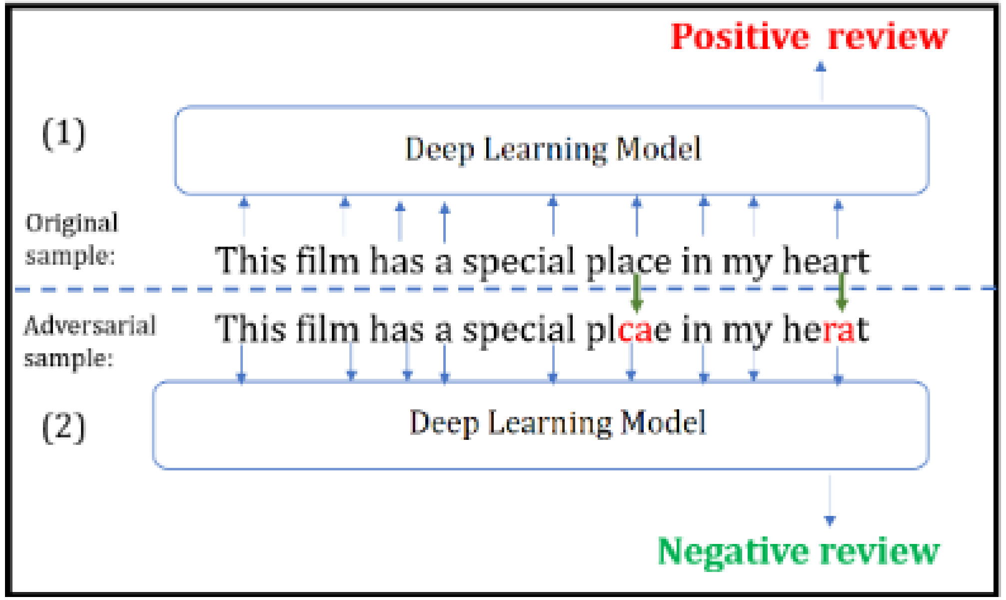
Algorithm 1 DeepWordBug Algorithm

Input: Input sequence $\mathbf{x} = x_1 x_2 \ldots x_n$, RNN classifier $F(\cdot)$, Scoring Function $S(\cdot)$, Transforming function $T(\cdot)$, maximum allowed pertubation on edit distance ϵ.

```
for i = 1..n do
    scores[i] = S(x_i; x)
end for
Sort scores into an ordered index list: L_1 .. L_n by descending score
x' = x
cost = 0, j = 1
while cost < ε do
    cost = cost + Transform(x'_{L_j})
    j++
end while
Return x'
```

https://github.com/thunlp/TAADpapers



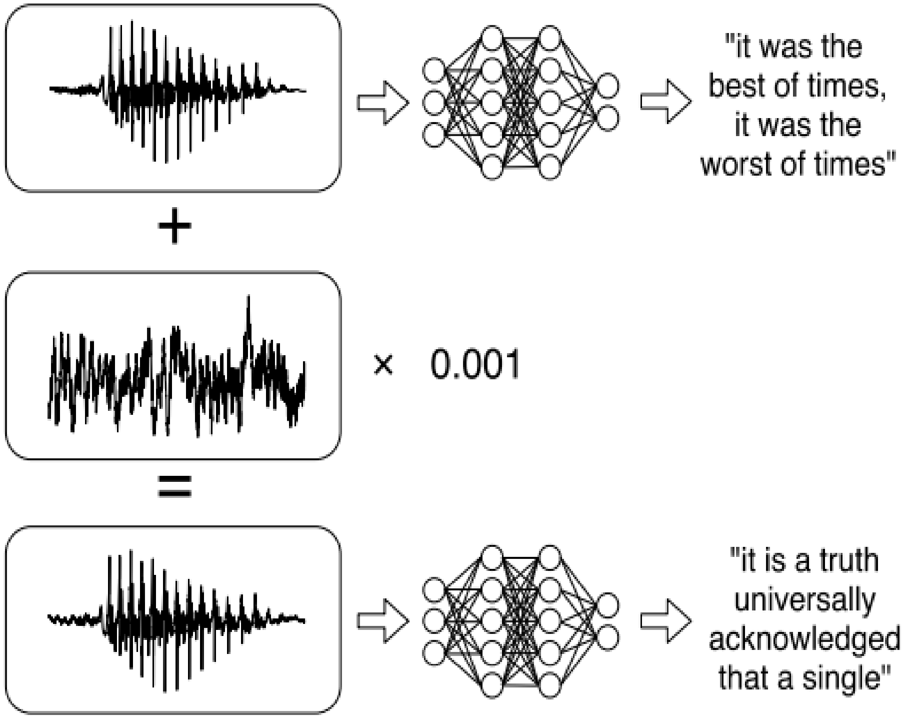


Tasks: Speech Recognition

☐ Targeted Attack on Speech-to-Text

- A **waveform** adds a small **perturbation**
- Making the result transcribe as any desired target phrase

Illustration of Speech-to-Text



☐ Connectionist Temporal Classification

- A method of **training** seq2seq neural network **without** the knowledge of **alignment** between input and output sequences.
- Algorithm:

$$CTC\_Loss(f(x), p) = -\log \Pr(p|f(x))$$

Tasks: Speech Recognition

original — [original, no speech is recognized]

adversarial — "speech can be embedded in music"

original — "now I would drift gently off to dream land"

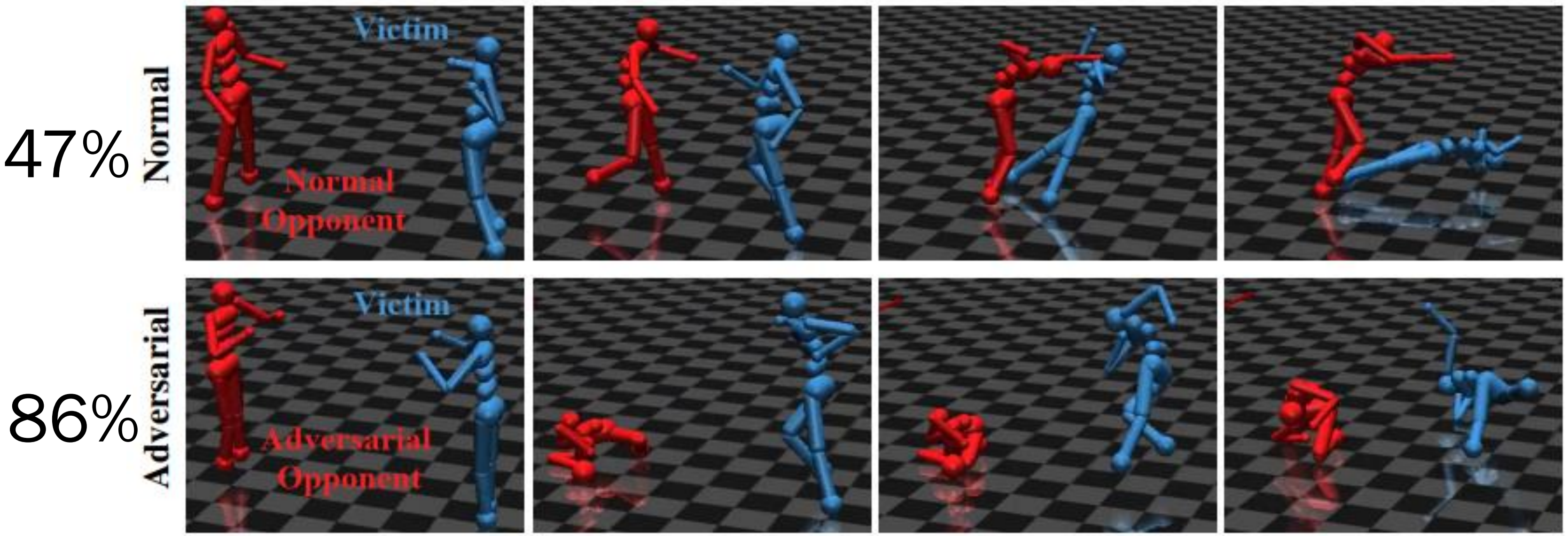
adversarial — "my wife pointed out to me the brightness of the red green and yellow signal light"

Tasks: Reinforcement Learning

☐ Adversarial Policies

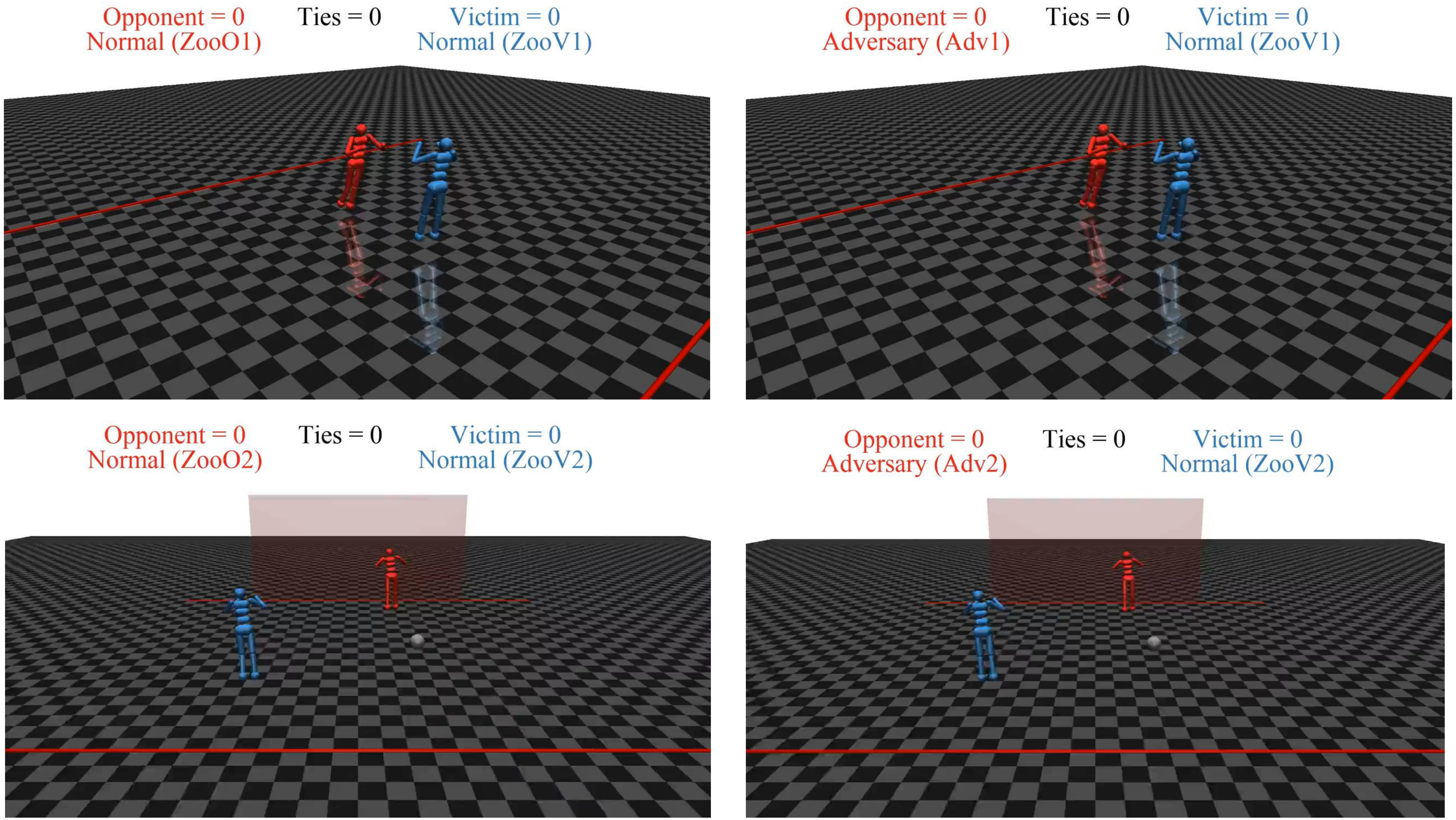
- Game detail: Two player & zero sum games
- Affect observation of the victim, leading to “bad” actions.
- The victim policy π_v is held fixed
- Reduces to a single player MDP
- Find an adversarial policy π_a maximazing the rewards

Illustrative snapshots of a victim against normal and adversarial opponents



Tasks: Reinforcement Learning

https://adversarialpolicies.github.io/



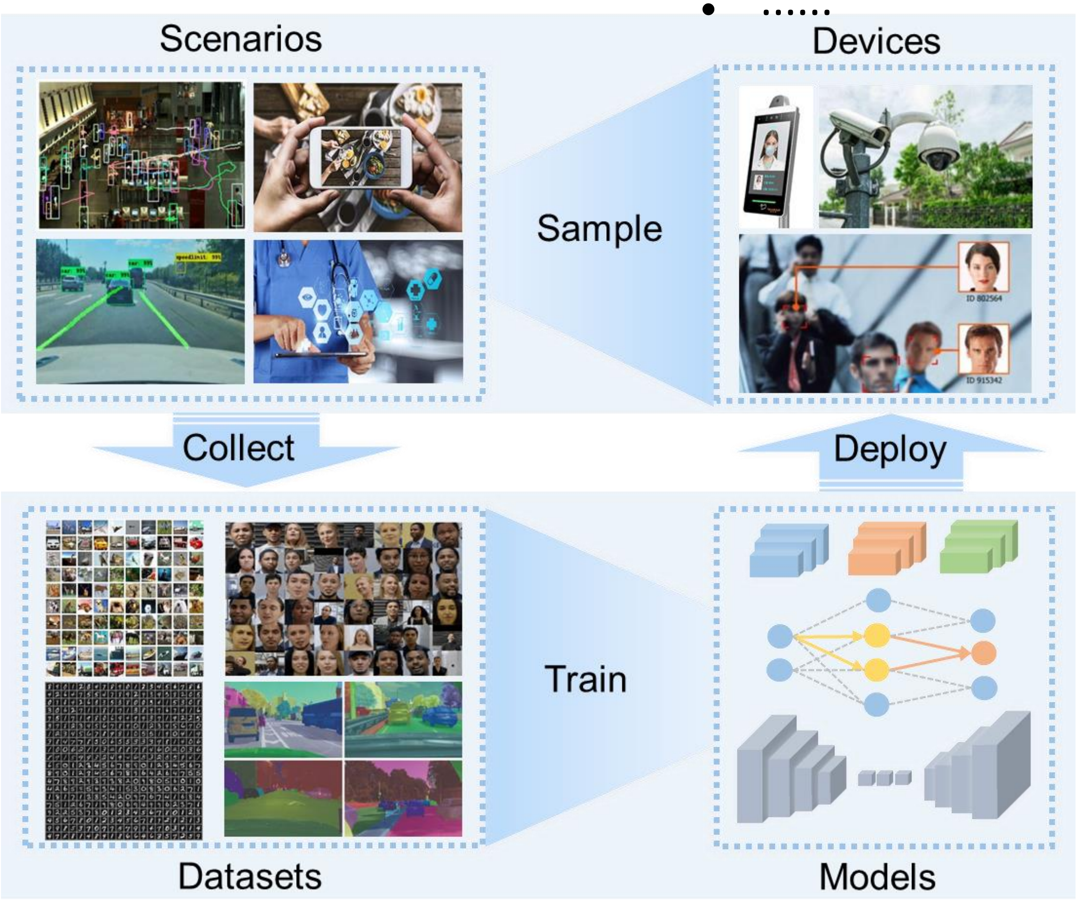
Adam Gleave, et al. UC Berkeley. Adversarial Policies: Attacking Deep Reinforcement Learning. ICLR 2020.

Outline

Adversarial Examples in Physical World

Differences from Digital World：

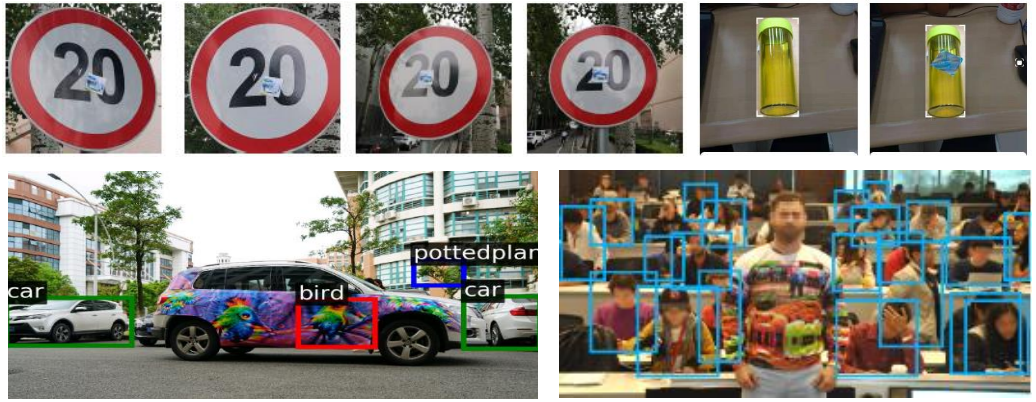
- Environment
- Noise
- Information
- Sampling
- Life circle
- Risk
- ……



Views	Digital	Physical
Environment	Simple and fixed	Complex
Noise	None/ Simulated	Natural
Information	White-box	Black-box
Sampling	Dateset	Imaging
Life circle	Train/ Evaluation	Evaluation
Risk	Limited	Huge

Physical adversarial examples bring **more severe safety threats**

Adversarial Example: Unified Definition



Adversarial examples are now threatening the safety and security of **AI applications in physical world**!

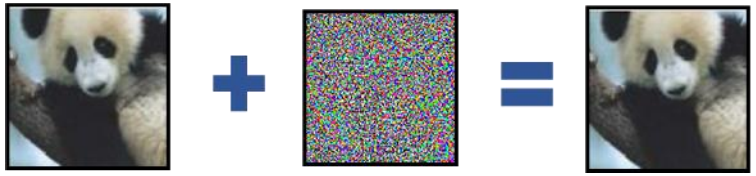
The characteristics of the digital world and physical world adversarial examples as:

- For human, it disguises as a normal example
- For models, it misleads the model predictions

Redefinition of adversarial examples:

$$y^x \neq F_\theta(x+r) \quad s.t. \quad \|x+r\| \in \aleph$$

where $\aleph$ is the human recognizable space, and $||\cdot||$ is some kind of measure(*i.e.*, perturbation magnitude, patch size)

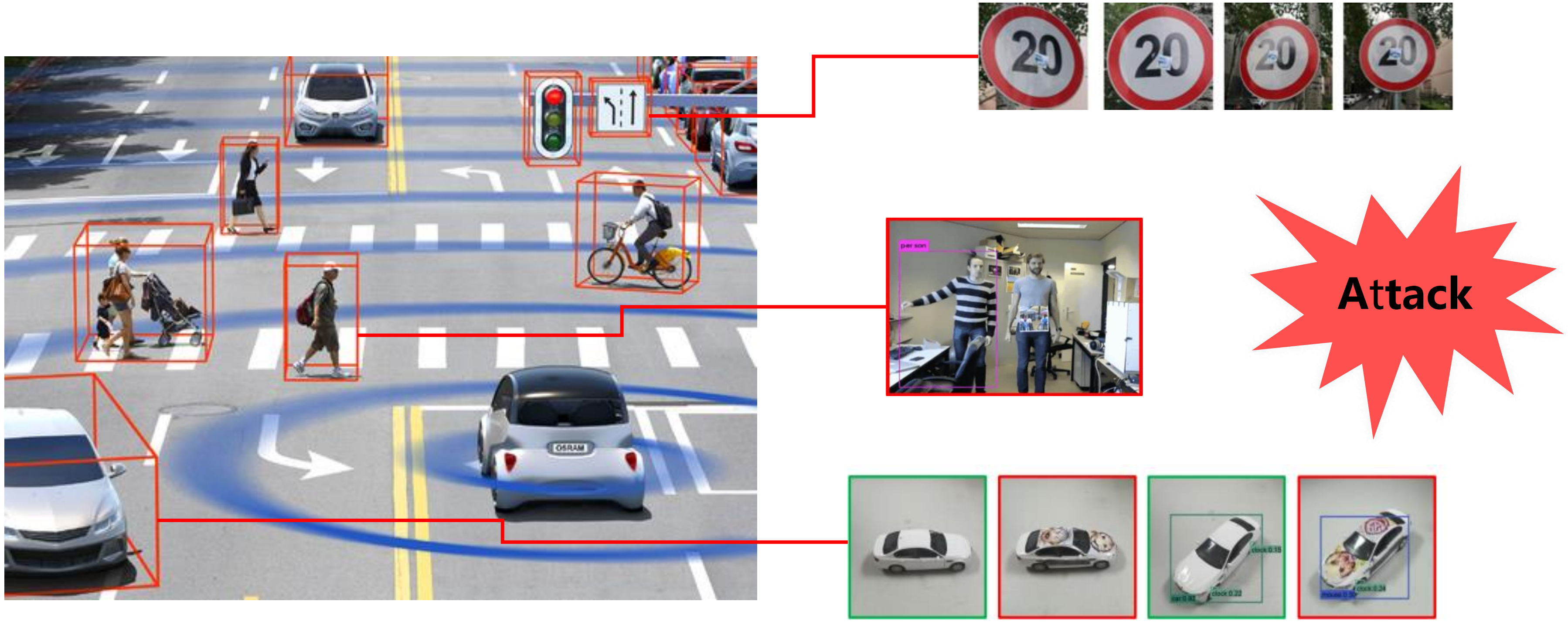


Adversarial perturbation

Adversarial patch

Attacks in the Physical World: Overall View

Physical attacks aim to generate adversarial perturbations by modifying the visual characteristics of the real object in the physical scenario



Due to the strong correlation to real-world AI applications, we classify the physical attacks through different AI tasks

Summary

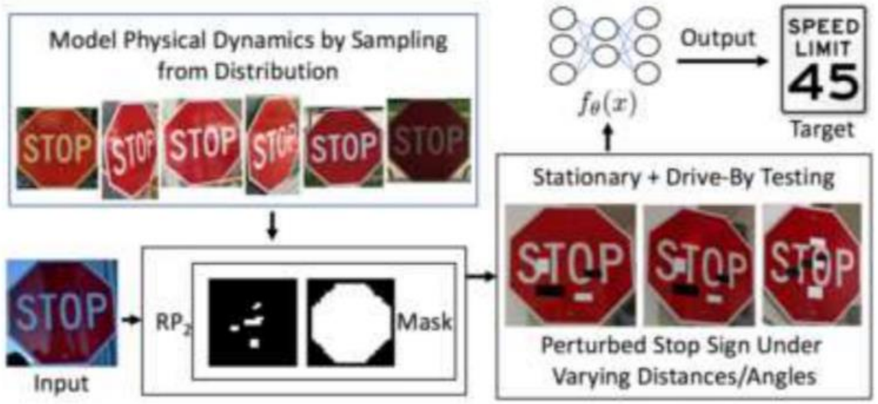
Method	Author	Application scenarios	Year
Face Recognition Attack	Sharif, Mahmood	Face recognition	2016
RP2	K. Eykholt	Auto-driving	2018
ShapeShifter	Shang-Tse Chen	Auto-driving	2018
PS-GAN	A. Liu	Auto-driving	2019
Persion Detector Attack	Thys, S.	Object Recognition	2019
AdvHat	Komkov, S.	Face recognition	2019
AdvCam	Duan, Renjie	Object Recognition	2020
Bias-based Attack	Liu, A.	Commodity identification	2020
UPC	Huang, Lifeng	Surveillance system	2020
Dual Attention Suppression Attack	Wang, Jiakai	Surveillance system	2021

Auto-driving: Road Signs Attack

☐ Robust Physical Perturbation (RP2)

- Physical World Challenges:
  - Environmental Conditions
  - Spatial Constraints
  - Physical Limits on Imperceptibility

- Model the distribution under both physical and digital transformations X^V
- Introduce a mask M_x to generate graffiti
- Non-Printability Score (NPS)



Better practical results in the physical world

$$\underset{\delta}{\operatorname{argmin}}\ \lambda||\delta||_p + J(f_\theta(x+\delta), y^*) \Longrightarrow \underset{\delta}{\operatorname{argmin}}\ \lambda||M_x \cdot \delta||_p + \boxed{NPS} + \boxed{\mathbb{E}_{x_i \sim X^V} J(f_\theta(x_i + T_i(M_x \cdot \delta)), y^*)}$$

Normal | Robust

Better adaptive effectiveness in the physical world

Auto-driving: Road Signs Attack

- **Against two standard-architecture classifiers**
  - LISA-CNN (91% acc. on LISA)
  - GTSRB-CNN (95.7% acc. on GTSRB)
- **Two types of attack**
  - Poster attack (100% asr. on LISA-CNN)
  - Sticker attack (over 80% asr. on GTSRB-CNN)

Table 2: Targeted physical perturbation experiment results on LISA-CNN using a poster-printed Stop sign (subtle attacks) and a real Stop sign (camouflage graffiti attacks, camouflage art attacks). For each image, the top two labels and their associated confidence values are shown. The misclassification target was Speed Limit 45. See Table 1 for example images of each attack. Legend: SL45 = Speed Limit 45, STP = Stop, YLD = Yield, ADL = Added Lane, SA = Signal Ahead, LE = Lane Ends.

Distance & Angle	Poster-Printing		Sticker			
	Subtle		Camouflage–Graffiti		Camouflage–Art	
5′ 0°	SL45 (0.86)	ADL (0.03)	STP (0.40)	SL45 (0.27)	SL45 (0.64)	LE (0.11)
5′ 15°	SL45 (0.86)	ADL (0.02)	STP (0.40)	YLD (0.26)	SL45 (0.39)	STP (0.30)
5′ 30°	SL45 (0.57)	STP (0.18)	SL45 (0.25)	SA (0.18)	SL45 (0.43)	STP (0.29)
5′ 45°	SL45 (0.80)	STP (0.09)	YLD (0.21)	STP (0.20)	SL45 (0.37)	STP (0.31)
5′ 60°	SL45 (0.61)	STP (0.19)	STP (0.39)	YLD (0.19)	SL45 (0.53)	STP (0.16)
10′ 0°	SL45 (0.86)	ADL (0.02)	SL45 (0.48)	STP (0.23)	SL45 (0.77)	LE (0.04)
10′ 15°	SL45 (0.90)	STP (0.02)	SL45 (0.58)	STP (0.21)	SL45 (0.71)	STP (0.08)
10′ 30°	SL45 (0.93)	STP (0.01)	STP (0.34)	SL45 (0.26)	SL45 (0.47)	STP (0.30)
15′ 0°	SL45 (0.81)	LE (0.05)	SL45 (0.54)	STP (0.22)	SL45 (0.79)	STP (0.05)
15′ 15°	SL45 (0.92)	ADL (0.01)	SL45 (0.67)	STP (0.15)	SL45 (0.79)	STP (0.06)
20′ 0°	SL45 (0.83)	ADL (0.03)	SL45 (0.62)	STP (0.18)	SL45 (0.68)	STP (0.12)
20′ 15°	SL45 (0.88)	STP (0.02)	SL45 (0.70)	STP (0.08)	SL45 (0.67)	STP (0.11)
25′ 0°	SL45 (0.76)	STP (0.04)	SL45 (0.58)	STP (0.17)	SL45 (0.67)	STP (0.08)
30′ 0°	SL45 (0.71)	STP (0.07)	SL45 (0.60)	STP (0.19)	SL45 (0.76)	STP (0.10)
40′ 0°	SL45 (0.78)	LE (0.04)	SL45 (0.54)	STP (0.21)	SL45 (0.68)	STP (0.14)

Table 1: Sample of physical adversarial examples against LISA-CNN and GTSRB-CNN.

Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5′ 0°					
5′ 15°					
10′ 0°					
10′ 30°					
40′ 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Table 3: A camouflage art attack on GTSRB-CNN. See example images in Table 1. The targeted-attack success rate is 80% (true class label: Stop, target: Speed Limit 80).

Distance & Angle	Top Class (Confid.)	Second Class (Confid.)
5′ 0°	Speed Limit 80 (0.88)	Speed Limit 70 (0.07)
5′ 15°	Speed Limit 80 (0.94)	Stop (0.03)
5′ 30°	Speed Limit 80 (0.86)	Keep Right (0.03)
5′ 45°	Keep Right (0.82)	Speed Limit 80 (0.12)
5′ 60°	Speed Limit 80 (0.55)	Stop (0.31)
10′ 0°	Speed Limit 80 (0.98)	Speed Limit 100 (0.006)
10′ 15°	Stop (0.75)	Speed Limit 80 (0.20)
10′ 30°	Speed Limit 80 (0.77)	Speed Limit 100 (0.11)
15′ 0°	Speed Limit 80 (0.98)	Speed Limit 100 (0.01)
15′ 15°	Stop (0.90)	Speed Limit 80 (0.06)
20′ 0°	Speed Limit 80 (0.95)	Speed Limit 100 (0.03)
20′ 15°	Speed Limit 80 (0.97)	Speed Limit 100 (0.01)
25′ 0°	Speed Limit 80 (0.99)	Speed Limit 70 (0.0008)
30′ 0°	Speed Limit 80 (0.99)	Speed Limit 100 (0.002)
40′ 0°	Speed Limit 80 (0.99)	Speed Limit 100 (0.002)

Auto-driving: Road Signs Attack

PS-GAN

- Traffic signs with **scrawls and patches on them are quite common** on the streets

- **GAN based** adversarial patch attack
  - Attention mechanism
  - Adversarial generation process
  - High perceptual correlation

Photos taken in the downtown of Rome.

1. Patch to patch translation
2. Adversarial generation process

$$L_{GAN}(G,D) = E_x[logD(\delta,x)] + E_{x,z}\left[\log(1 - D\big(\delta, x + G(z,\delta)\big))\right]$$

3. High visual fidelity & Perceptual correlation

$$L_{patch}(\delta) = E_\delta \|G(\delta) - \delta\|_2$$

$$\min_G \max_D L_{GAN} + \alpha \cdot L_{patch} + \beta \cdot L_{adv}$$

GAN loss + patch loss + adversarial loss

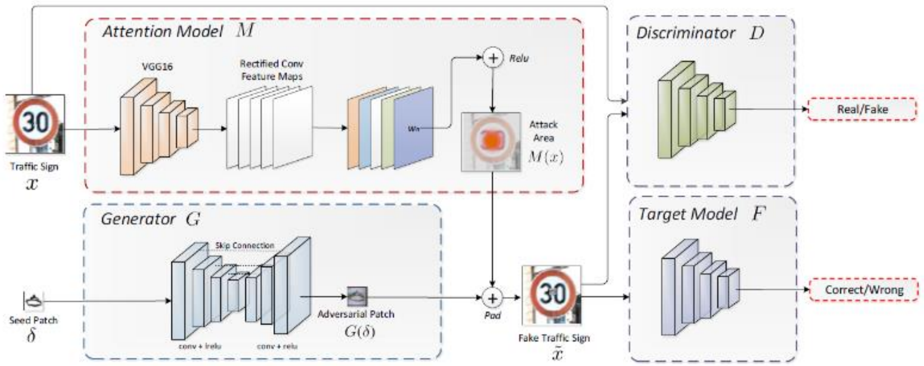


Figure 2: The framework of our PS-GAN consists of a generator G, a discriminator D and an attention model M, attacking a target model F.

Auto-driving: Road Signs Attack

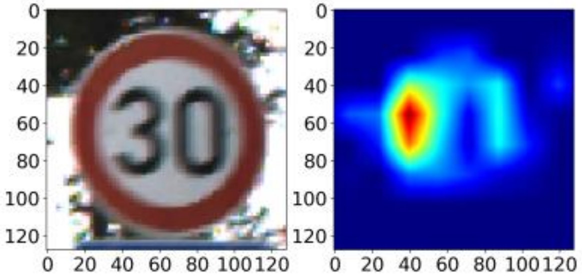
- Digital world attacks

		Target Models						
		VY	$VGG16$	VY_{lrelu}	$VGG16_{tanh}$	$\overline{VY}$	$\overline{VGG16}$	$ResNet$
Source Models	VY	**12.5%**	25.0%	37.5%	12.5%	15.6%	31.3%	37.5%
	$VGG16$	1.6%	**31.3%**	15.6%	37.5%	1.6%	31.3%	34.4%
	VY_{lrelu}	4.7%	25.0%	**7.8%**	23.4%	12.5%	29.7%	26.6%
	$VGG16_{tanh}$	3.1%	25.0%	32.8%	**34.4%**	7.8%	25.0%	25.0%
	$\overline{VY}$	9.4%	25.7%	14.1%	25.0%	**14.1%**	28.1%	37.5%
	$\overline{VGG16}$	3.1%	37.5%	9.4%	34.4%	7.8%	**31.4%**	21.9%
	$ResNet$	3.1%	15.6%	4.7%	21.9%	9.4%	26.6%	**34.4%**

	GTSRB	ImageNet
Accuracy without patches	89.5%	87.6%
Accuracy with seed patches	85.6%	67.6%
Accuracy with adversarial patches	12.5%	25.0%

- Real-world attacks
  1. Real-world traffic sign
  2. Print & Stick & Photo
  3. Accuracy drop: 86.7% -> 17.2%

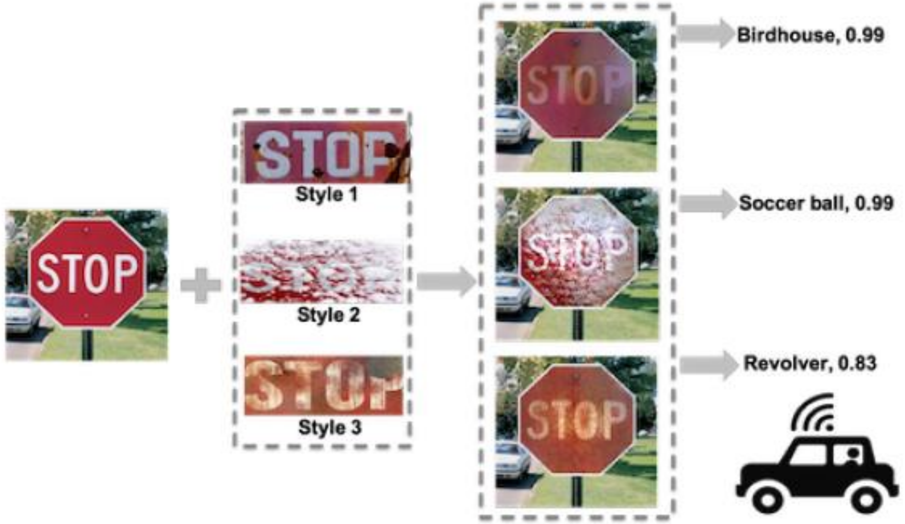
- Attention visualizations



- Attack Visualization

AdvCam

- Able to fool DNN while natural to human
- Can control the physical appearance of the camouflage
- Can also be used to protect private information



(a) Target image (b) Style (c) Adversarial examples

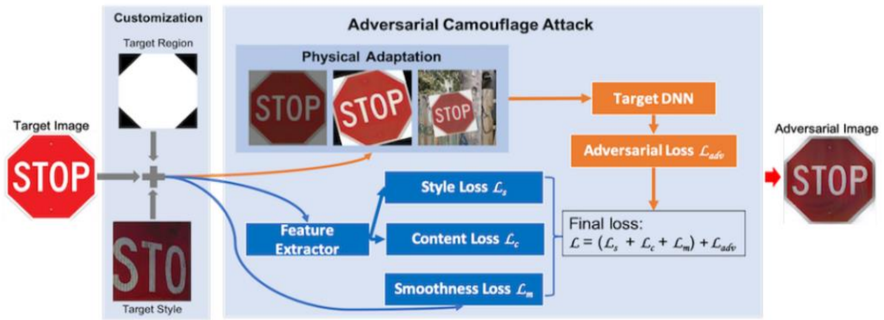
$$\min_{x'} ((\mathcal{L}_s + \mathcal{L}_c + \mathcal{L}_m) + \max_{T \in \mathcal{T}} \lambda \cdot \mathcal{L}_{adv}(o + T(x')),$$

$$\mathcal{D}_s = \sum_{l \in \mathcal{S}_l} \left\| \mathcal{G}(\widetilde{F}_l(x^s)) - \mathcal{G}(\widetilde{F}_l(x')) \right\|_2^2,$$

$$\mathcal{L}_c = \sum_{l \in \mathcal{C}_l} \left\| \widetilde{F}_l(x) - \widetilde{F}_l(x') \right\|_2^2,$$

$$\mathcal{L}_m = \sum ((x'_{i,j} - x_{i+1,j})^2 + (x'_{i,j} - x_{i,j+1})^2)^{\frac{1}{2}},$$

$$\mathcal{L}_{adv} = \begin{cases} \log(p_{y_{adv}}(x')), & \text{for targeted attack} \\ -\log(p_y(x')), & \text{for untargeted attack,} \end{cases}$$



style loss + content loss + smoothness loss + adversarial loss

Duan, Ranjie, et al. "Adversarial camouflage: Hiding physical-world attacks with natural styles." CVPR. 2020.

Auto-driving: Road Signs Attack

Camouflage under different settings

(a) Original (b) $\mathcal{L}_s$ (c) $\mathcal{L}_s$+$\mathcal{L}_c$ (d) All

Figure 5: Ablation of the 3 camouflage losses: (a): original images with intended camouflage style at the bottom right corner; (b) - (d): camouflaged adversarial examples using different loss functions.

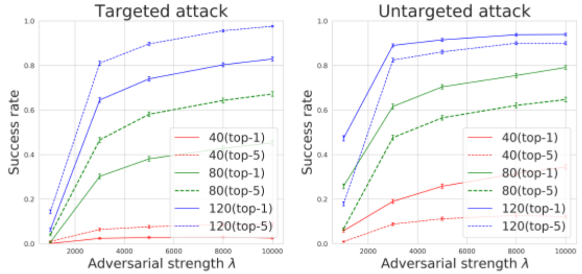


Figure 6: Ablation of adversarial strength λ and region size: success rate of untargeted (left) and targeted attack (right).

Figure 11: Adversarial traffic sign with 3 styles of stains.

Visualization of AdvCam and comparison to other methods

Figure 9: Camouflaged adversarial images crafted by our *AdvCam* attack and their original versions.

(a) AdvPatch (b) PGD-128 (c) *AdvCam*

Figure 10: *Top*: Adversarial wood texture recognized as street sign. *Bottom*: Adversarial logo on t-shirt.

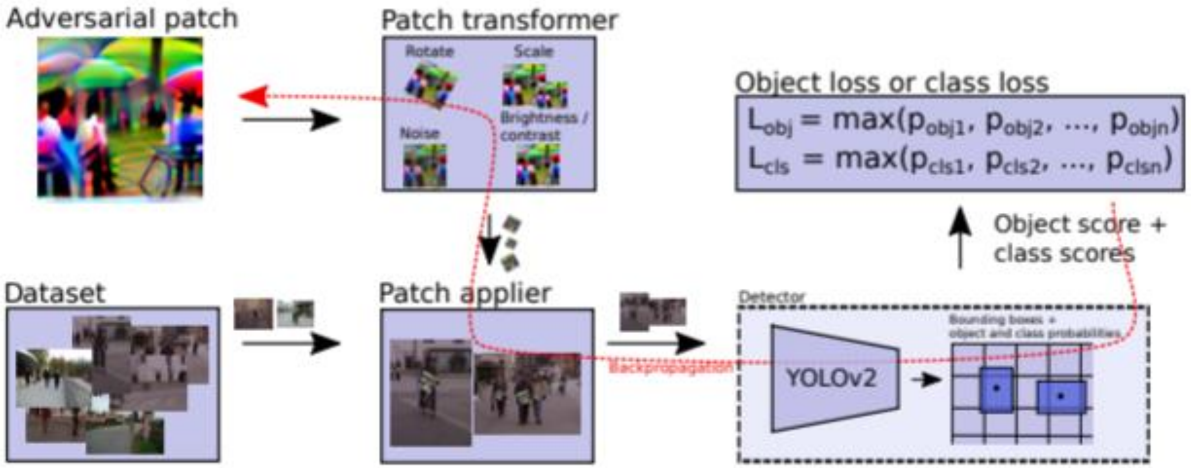
Object detection

☐ Problem

- All of patch attacks contain **no intra-class variety**

☐ Goal

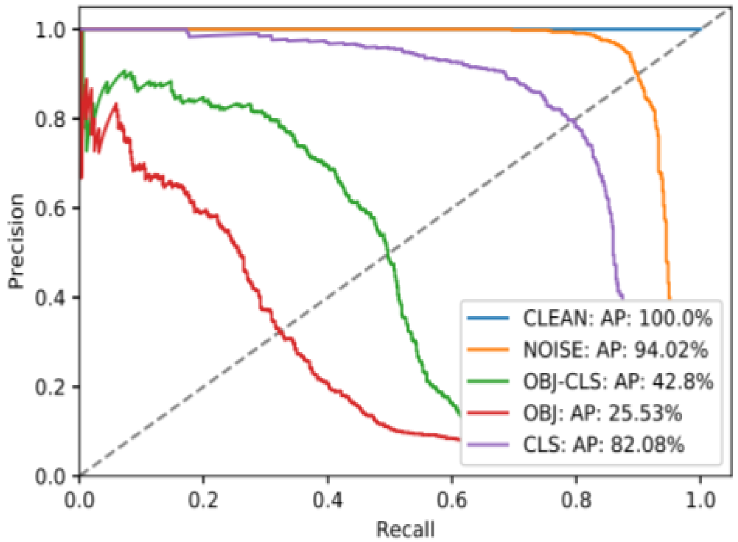
- Generate a **small patch** that is able to hide a person from the person detector
- Minimizing object loss is the most effective
- **Attacked Yolo-v2** in real world



Object detection

Output on the Inria testset

Comparison of different approaches in recall



Approach	Recall (%)
CLEAN	100
NOISE	87.14
OBJ-CLS	39.31
OBJ	**26.46**
CLS	77.58

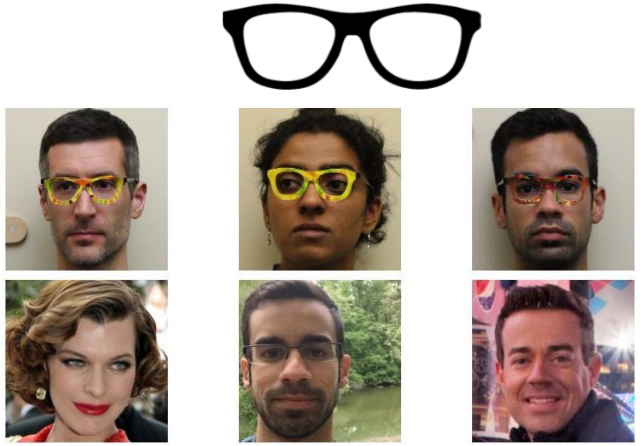
Minimizing object loss created effective patches

Thys, S. , et al. "Fooling Automated Surveillance Cameras: Adversarial Patches to Attack Person Detection." IEEE (2019).

Face recognition

❑ Attack Face Recognition

- Inconspicuous camouflage (e.g., a glass) to attack physical-world face ID system
- Can be used in dodging and impersonation



$$\underset{r}{\operatorname{argmin}} \sum_{x \in X} softmaxloss(f(x+r), l)$$

Robustness Loss

$$TV(r) = \sum_{i,j} \left((r_{i,j} - r_{i+1,j})^2 + (r_{i,j} - r_{i,j+1})^2 \right)^{\frac{1}{2}}$$

Smoothness Loss

$$NPS(\hat{p}) = \prod_{p \in P} |\hat{p} - p|$$

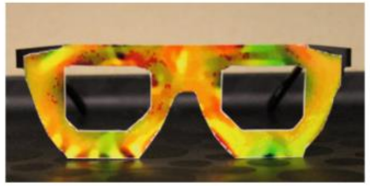
Printability Loss

Face recognition

☐ Experimental setting

- Digital and physical experiments
- Extension to black-box models
  - Particle Swarm Optimization

Created eyeglass



Invisibility attack

Original | Overlay perturbation | Accessories perturbation

Experiment #	*Area perturbed*	*Goal*	*Model*	*# Attackers*	*Success rate*
1	Entire face	Dodging	DNN_A	20	100.00%
2	Entire face	Impersonation	DNN_A	20	100.00%
3	Eyeglass frames	Dodging	DNN_A	20	100.00%
4	Eyeglass frames	Dodging	DNN_B	10	100.00%
5	Eyeglass frames	Dodging	DNN_C	20	100.00%
6	Eyeglass frames	Impersonation	DNN_A	20	91.67%
7	Eyeglass frames	Impersonation	DNN_B	10	100.00%
8	Eyeglass frames	Impersonation	DNN_C	20	100.00%

High success rate in real-world

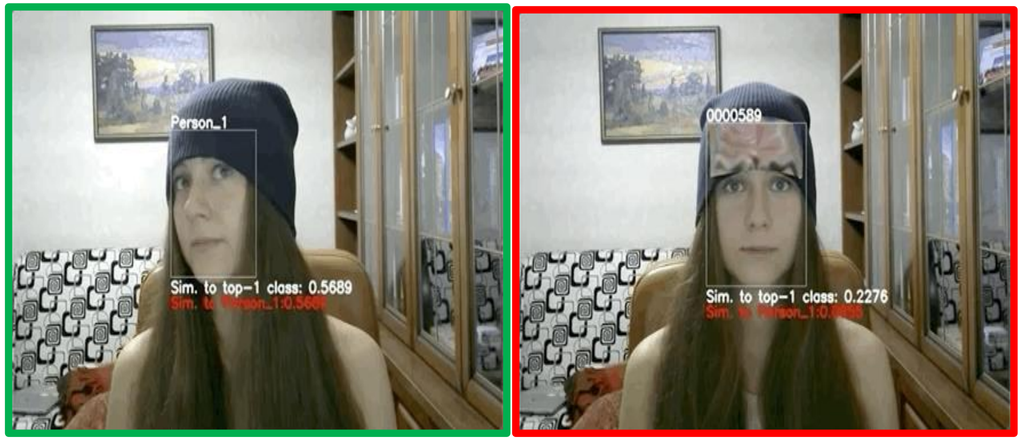
	Subject (attacker) info		Dodging results		Impersonation results			
DNN	*Subject*	*Identity*	*SR*	$E(p(\text{correct-class}))$	*Target*	*SR*	*SRT*	$E(p(\text{target}))$
DNN_B	S_A	3rd author	100.00%	0.01	Milla Jovovich	87.87%	48.48%	0.78
	S_B	2nd author	97.22%	0.03	S_C	88.00%	75.00%	0.75
	S_C	1st author	80.00%	0.35	Clive Owen	16.13%	0.00%	0.33
DNN_C	S_A	3rd author	100.00%	0.03	John Malkovich	100.00%	100.00%	0.99
	S_B	2nd author	100.00%	<0.01	Colin Powell	16.22%	0.00%	0.08
	S_C	1st author	100.00%	<0.01	Carson Daly	100.00%	100.00%	0.90

Sharif, Mahmood, et al. "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition." ACM SCCCS. 2016.

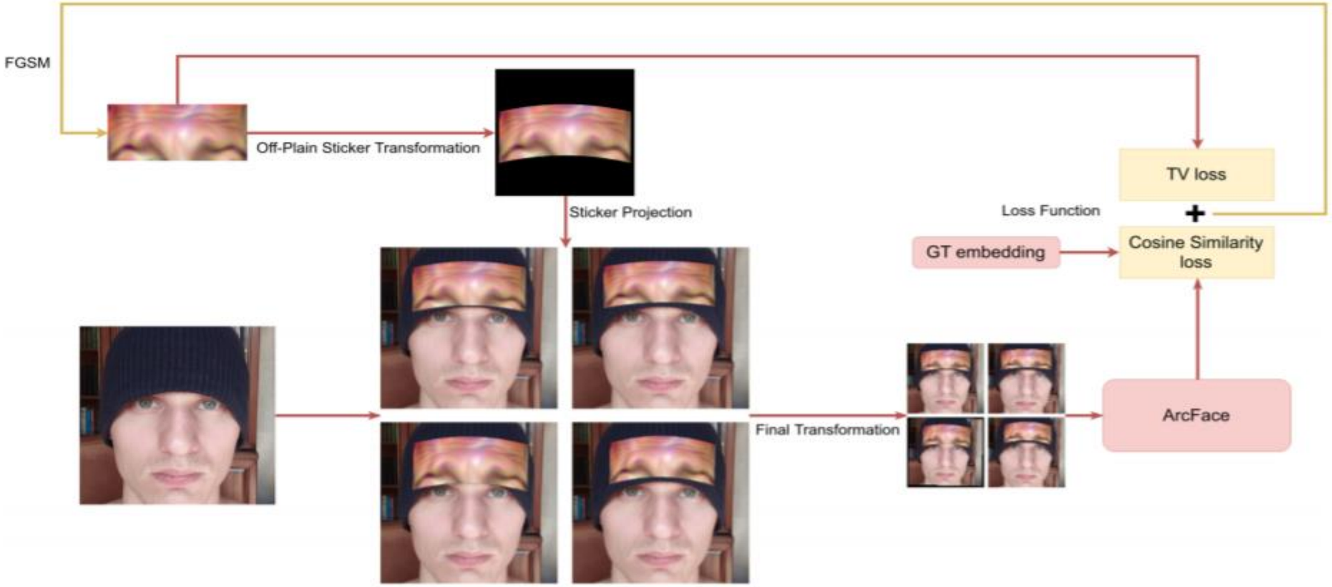
Face recognition

AdvHat

- A printed paper to attack real-world commercial Face ID system.
- Off-plane transformation to imitate shape deformation



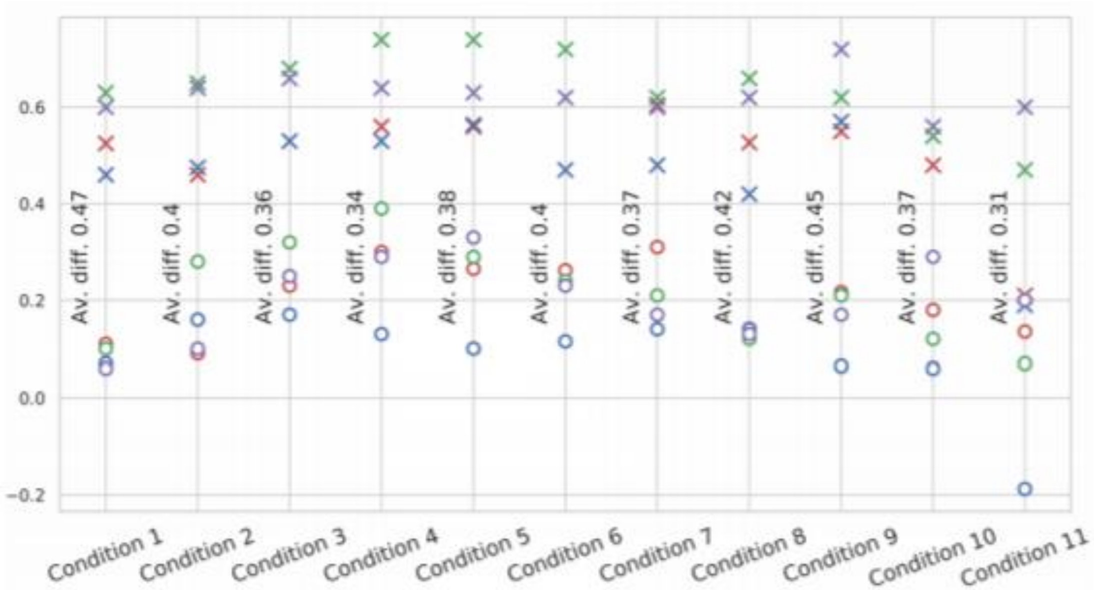
the whole pipeline of the attack



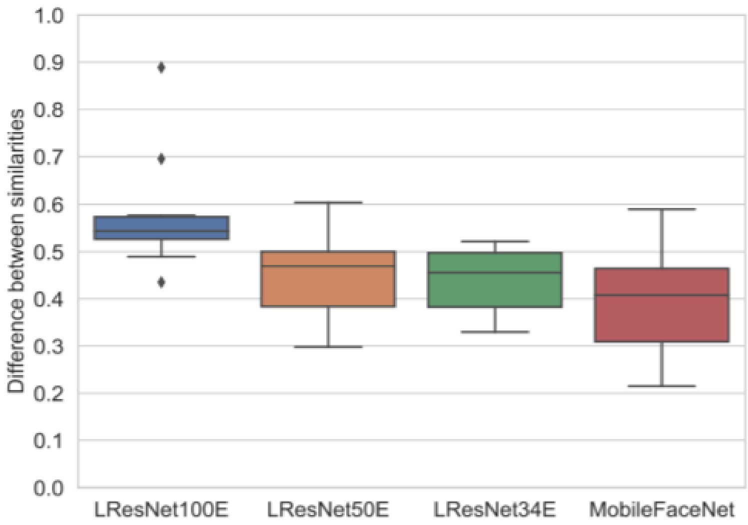
Face recognition

Example of the adversarial stickers

Baseline and final similarity for various shooting conditions



Differences between baseline and final similarities of one attack on different models

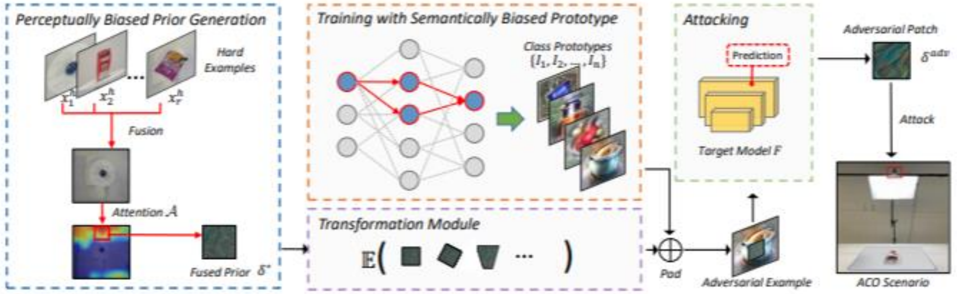


Komkov, S. , et al. "AdvHat: Real-world adversarial attack on ArcFace Face ID system." (2019).

Commodity identification

Bias-based Attack

- In Automatic Check-Out, items are often tied with patch-like stickers or tags
- Perceptual bias
  - Extract textural information from multiple hard examples
- Semantic bias
  - Prototypes, contain the most representative semantics



$$\mathcal{L}_f = \mathcal{L}_s + \lambda \cdot \mathcal{L}_u,$$

$$\mathcal{L}_s = \mathbb{E}_k \left[\left|\left| \mathbf{G}(x^*) - \mathbf{G}(x_k^h) \right|\right|_F^2 \right],$$

$$\mathbf{G}_{ij}(x) = \sum_k F_{ik}^l(x) \cdot F_{jk}^l(x),$$

$$\mathcal{L}_u = \mathbb{E}_i \left[\log y^{h,i} \right],$$

Perceptual bias

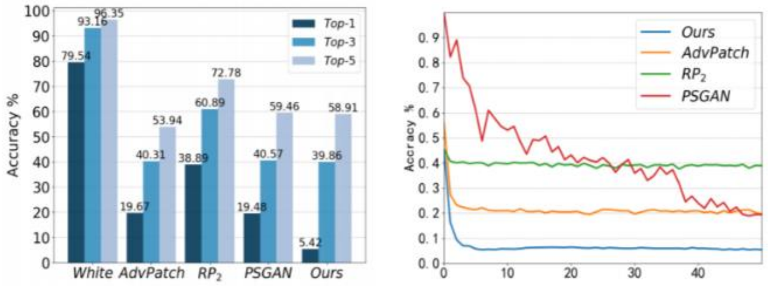
$$I_t = \underset{x}{\operatorname{argmax}} \frac{1}{C} \sum_{c \neq t} \max(0, margin - S_t(x) + S_c(x))^p,$$

$$\mathcal{L}_t = \mathbb{E}_{I,\delta^{adv}} [P(c = t | I') - \max(P(c \neq t | I'))],$$

Semantic bias

☐ Digital world attack

- Attack RPC dataset (the largest ACO related dataset)
  - White-box
  - Black-box



(a) White-box Attack (b) Training Process

Fig. 3. (a) shows the White-box attack experiment in the digital-world with ResNet-152. Our method generates the strongest adversarial patches with the lowest classification accuracy. (b) denotes the training process of different methods

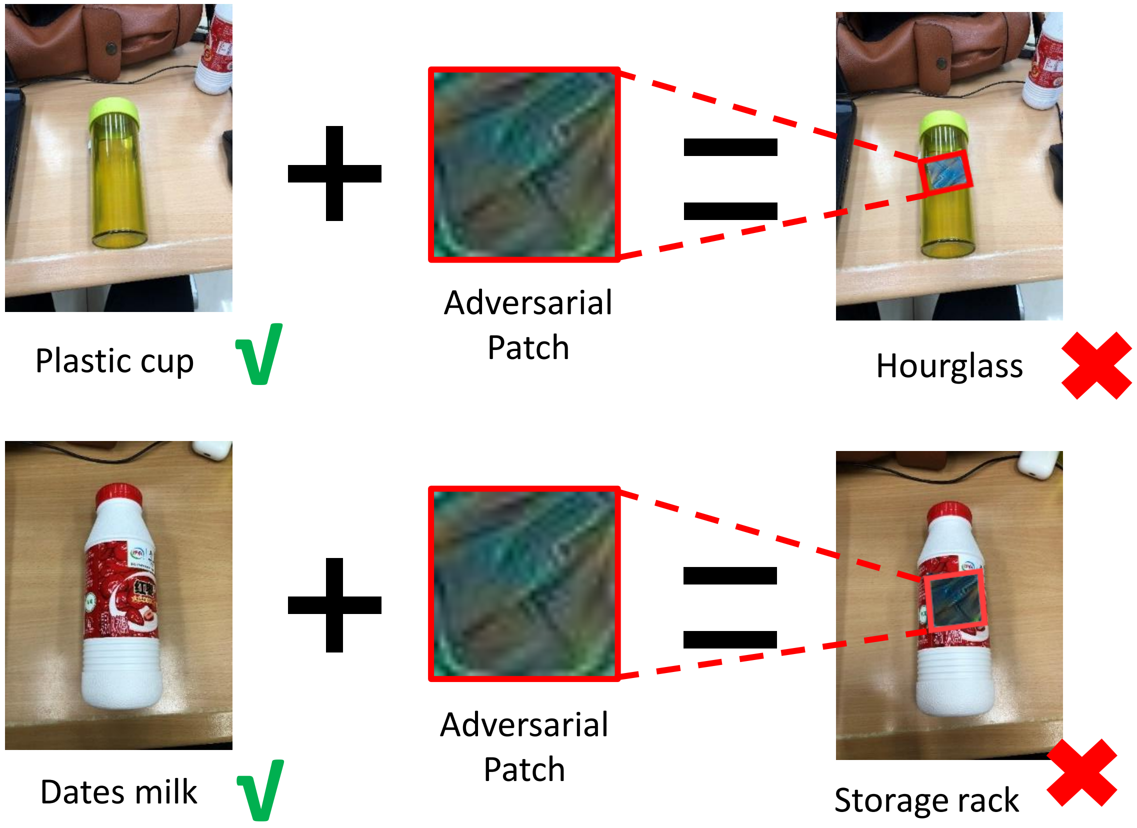
Table 1. Black-box attack experiment in the digital-world with VGG-16, AlexNet, and ResNet-101. Our method generates adversarial patches with strong transferability among different models

Model	Method	*top*-1	*top*-3	*top*-5
VGG-16	AdvPatch	73.82	**90.73**	**94.99**
	RP_2	81.25	94.65	97.10
	PSGAN	74.69	91.25	96.12
	Ours	**73.72**	**91.53**	**95.43**
AlexNet	AdvPatch	51.11	72.37	79.79
	RP_2	68.27	86.49	91.08
	PSGAN	49.39	72.85	82.94
	Ours	**31.68**	**50.92**	**60.19**
ResNet-101	AdvPatch	56.19	80.99	91.52
	RP_2	73.52	93.75	98.13
	PSGAN	51.26	79.22	90.47
	Ours	**22.24**	**51.32**	**60.28**

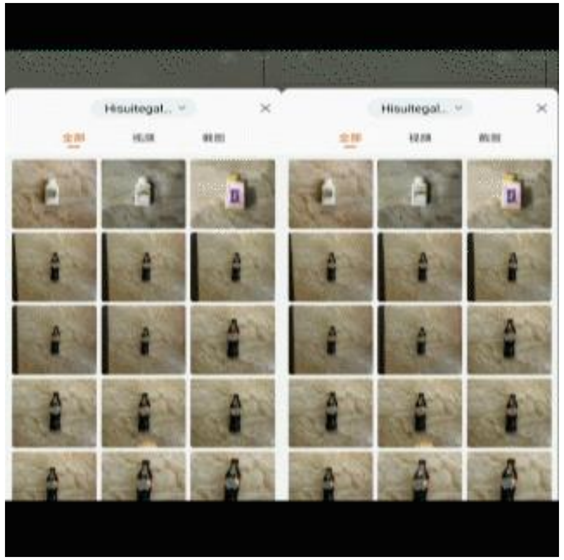
Commodity identification

□ Physical world attack

- Attack Taobao and JD APPs



clean

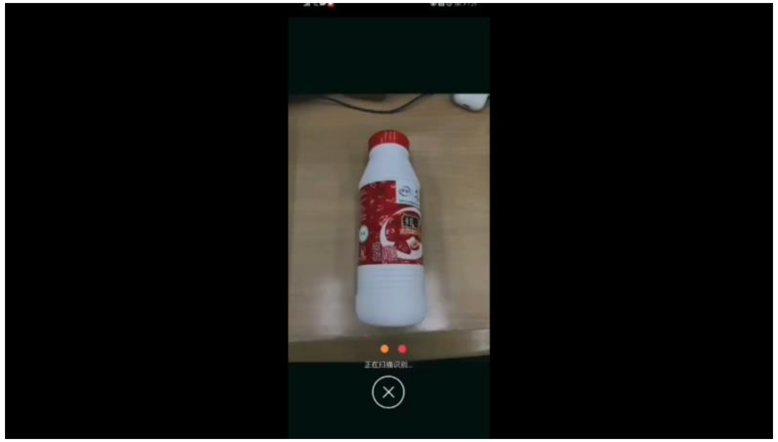


adversarial

clean

adversarial

Commodity identification



□ Universal camouflage pattern (UPC)

- Can attack all instances in same category
- Add semantic constraint to for naturalness

Region Proposal Network attack (rpn)
Classification & Regressor attack (cls/reg)

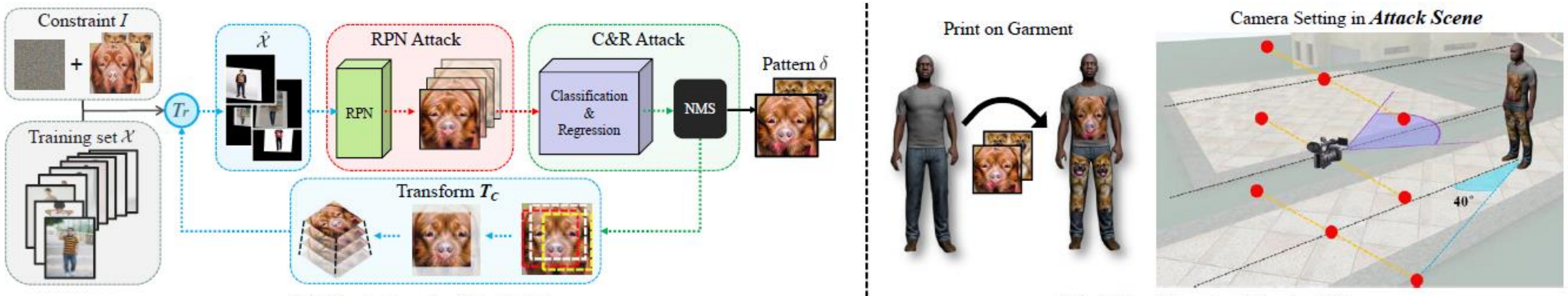
$$\underset{\Delta\delta}{argmin} \ \mathop{\mathbb{E}}_{\hat{x}\sim\hat{\mathcal{X}}} (L_{rpn} + \lambda_1 L_{cls} + \lambda_2 L_{reg}) + L_{tv}(\delta^t),$$

$$L_{rpn} = \mathop{\mathbb{E}}_{p_i\sim\mathcal{P}} (\mathbb{L}(s_i, y^t) + s_i\|\vec{d_i} - \Delta\vec{d_i}\|_p),$$

$$L_{cls} = \mathop{\mathbb{E}}_{p\sim\mathcal{P}'} [C(p)_o + \mathop{\mathbb{L}}_{C(p)_{max}\in o} (C(p), y^t)],$$

$$L_{reg} = \sum_{C(p)_{max}\in o} \|R(p)_o - \Delta\vec{d}\|_p,$$

Overall paradigm



(a) Training in *Digital Space*　　(b) Attacking in *Physical Space*

Surveillance system: person detection

Successfully Attacked Fast-RCNN

Table 4. Average precision $p_{0.5}$ in stationary testing after attacking faster r-cnn. We test on a total of 6 different poses (*i.e.*, standing, sitting, leg lifting, waving hands, fork waist, shaking head).

Network	FR-VGG16-0712								FR-RES101-0712							
Schemes	Standing				Sitting				Standing				Sitting			
	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)
Original	1.0	1.0	1.0	1.0 (-)	1.0	1.0	1.0	1.0 (-)	1.0	1.0	1.0	1.0 (-)	1.0	1.0	1.0	1.0 (-)
Random	1.0	0.94	1.0	0.98 (0.02)	1.0	1.0	1.0	1.0 (0.0)	0.94	1.0	1.0	0.98 (0.02)	1.0	1.0	1.0	1.0 (0.0)
3-Patterns	0.72	0.61	0.67	0.67 (**0.33**)	0.83	0.78	0.67	0.76 (**0.24**)	0.83	0.67	0.67	0.72 (**0.28**)	0.72	0.78	0.72	0.74 (**0.26**)
7-Patterns	0.67	0.56	0.56	0.59 (**0.41**)	0.61	0.50	0.50	0.54 (**0.46**)	0.61	0.56	0.61	0.59 (**0.41**)	0.61	0.67	0.50	0.59 (**0.41**)
8-Patterns	0.22	0.11	0.17	0.17 (**0.83**)	0.28	0.17	0.22	0.22 (**0.78**)	0.22	0.22	0.11	0.19 (**0.81**)	0.28	0.22	0.22	0.26 (**0.74**)
Schemes	Fork Waist				Leg Lifting				Fork Waist				Leg Lifting			
	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)
Original	1.0	1.0	1.0	1.0 (-)	1.0	1.0	1.0	1.0 (-)	1.0	1.0	1.0	1.0 (-)	1.0	1.0	1.0	1.0 (-)
Random	1.0	1.0	1.0	1.0 (0.0)	1.0	1.0	1.0	1.0 (0.0)	1.0	1.0	1.0	1.0 (0.0)	1.0	1.0	1.0	1.0 (0.0)
3-Patterns	0.78	0.72	0.67	0.72 (**0.28**)	0.72	0.78	0.72	0.74 (**0.26**)	0.83	0.72	0.72	0.76 (**0.24**)	0.67	0.78	0.67	0.71 (**0.29**)
7-Patterns	0.61	0.50	0.56	0.56 (**0.44**)	0.56	0.56	0.50	0.54 (**0.46**)	0.61	0.56	0.56	0.57 (**0.43**)	0.67	0.50	0.56	0.57 (**0.43**)
8-Patterns	0.28	0.17	0.17	0.20 (**0.80**)	0.28	0.28	0.22	0.26 (**0.74**)	0.28	0.22	0.22	0.24 (**0.76**)	0.33	0.33	0.22	0.30 (**0.70**)
Schemes	Rasing Hands				Shaking Head				Rasing Hands				Shaking Head			
	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)
Original	1.0	1.0	1.0	1.0 (-)	1.0	1.0	1.0	1.0 (-)	1.0	1.0	1.0	1.0 (-)	1.0	1.0	1.0	1.0 (-)
Random	0.94	1.0	1.0	0.98 (0.02)	1.0	1.0	1.0	1.0 (0.0)	1.0	1.0	1.0	1.0 (0.0)	1.0	1.0	1.0	1.0 (0.0)
3-Patterns	0.89	0.78	0.83	0.83 (**0.17**)	0.78	0.78	0.67	0.74 (**0.26**)	0.83	0.89	0.83	0.85 (**0.15**)	0.72	0.78	0.78	0.76 (**0.24**)
7-Patterns	0.72	0.61	0.61	0.65 (**0.35**)	0.61	0.61	0.56	0.59 (**0.41**)	0.89	0.61	0.56	0.69 (**0.31**)	0.56	0.61	0.56	0.57 (**0.43**)
8-Patterns	0.39	0.39	0.28	0.35 (**0.65**)	0.22	0.28	0.11	0.20 (**0.80**)	0.39	0.33	0.33	0.35 (**0.65**)	0.22	0.22	0.17	0.20 (**0.80**)

detected as car | detected as others | undetected

Figure 7. **The results of attacking Volvo XC60 (top row) and Volkswagen Tiguan (bottom row).** The generated camouflage patterns fool detectors to misrecognize the car as bird or person.

Table 5. Average precision $p_{0.5}$ in transferability testing. First seven rows show the results of cross-training transfer testing, and rest five rows display the cross-network transfer's results (**bold** in "Network" column).

Network	Original	FR-VGG16-0712 Average (Drop)	FR-RES101-0712 Average (Drop)
FR-VGG16-0712	0.95	**0.04 (0.91)**	0.10 (0.85)
FR-RES101-0712	0.99	0.78 (0.21)	**0.06 (0.93)**
FR-VGG16-07	0.95	0.08 (0.87)	0.11 (0.84)
FR-RES101-07	0.99	0.51 (0.48)	0.10 (0.89)
FR-RES50-14	1.0	0.85 (0.15)	0.78 (0.22)
FR-RES152-14	1.0	0.62 (0.38)	0.43 (0.57)
FR-MN-14	0.99	0.51 (0.48)	0.25 (0.74)
RFCN-RES101-07	0.98	0.64 (0.34)	0.41 (0.57)
SSD-VGG16-0712	0.75	0.13 (0.62)	0.16 (0.59)
Yolov2-14	1.0	0.59 (0.41)	0.38 (0.62)
Yolov3-14	1.0	0.69 (0.31)	0.71 (0.29)
Retina-14	1.0	0.72 (0.31)	0.49 (0.51)

Outperforms the state-of-the-art method

Table 2. Average precision $p_{0.5}$ in virtual scene experiments after attacking faster r-cnn. Note that $p_{0.5}$ is averaged over all view-points of each pattern scheme under 3 brightness conditions.

Network	FR-VGG16-0712				FR-RES101-0712			
Schemes	Standing				Standing			
	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)
Original	0.97	0.97	1.0	0.98 (-)	0.99	0.99	1.0	0.99 (-)
Naive	0.97	0.97	0.99	0.97 (0.01)	0.99	0.99	0.99	0.99 (0.0)
Natural	0.95	0.96	0.98	0.96 (0.02)	0.97	0.97	0.98	0.97 (0.02)
3-Patterns	0.64	0.36	0.18	0.39 (**0.59**)	0.73	0.69	0.70	0.69 (**0.30**)
7-Patterns	0.55	0.33	0.22	0.37 (**0.61**)	0.51	0.48	0.64	0.54 (**0.45**)
8-Patterns	0.15	0.03	0.02	0.07 (**0.91**)	0.10	0.09	0.13	0.11 (**0.88**)
Schemes	Walking				Walking			
	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)
Original	0.93	0.94	0.99	0.95 (-)	0.98	0.99	1.0	0.99 (-)
Naive	0.92	0.94	0.96	0.94 (0.01)	0.98	0.97	0.98	0.98 (0.01)
Natural	0.91	0.93	0.95	0.93 (0.02)	0.98	0.99	0.98	0.98 (0.01)
3-Patterns	0.37	0.26	0.16	0.26 (**0.69**)	0.44	0.50	0.50	0.48 (**0.51**)
7-Patterns	0.28	0.25	0.16	0.23 (**0.72**)	0.31	0.33	0.34	0.33 (**0.66**)
8-Patterns	0.06	0.05	0.01	0.04 (**0.91**)	0.05	0.06	0.06	0.06 (**0.93**)
Schemes	Sitting				Sitting			
	L1	L2	L3	Avg (Drop)	L1	L2	L3	Avg (Drop)
Original	0.97	0.99	0.99	0.98 (-)	1.0	0.99	0.99	0.99 (-)
Naive	0.93	0.94	0.95	0.94 (0.04)	0.93	0.92	0.93	0.93 (0.06)
Natural	0.94	0.94	0.98	0.95 (0.03)	0.97	0.98	0.98	0.98 (0.01)
3-Patterns	0.83	0.64	0.63	0.70 (**0.28**)	0.75	0.77	0.79	0.77 (**0.22**)
7-Patterns	0.83	0.77	0.63	0.74 (**0.24**)	0.77	0.78	0.78	0.78 (**0.21**)
8-Patterns	0.60	0.47	0.32	0.46 (**0.52**)	0.49	0.57	0.62	0.56 (**0.43**)

Table 3. Performance comparison with prior arts of physical attacks under different settings. Note that $p_{0.5}$ is averaged over all viewpoints of 8-pattern scheme.

Network	FR-VGG16-0712		
Pose	Standing	Walking	Sitting
UPC_{rc}(ours)	0.07 (**0.91**)	0.04 (**0.91**)	0.46 (**0.52**)
UPC_{r}(ours)	0.65 (0.32)	0.33 (0.62)	0.76 (0.22)
CLS_{rc}(ours)	0.17 (0.80)	0.06 (0.89)	0.58 (0.44)
SS [5]	0.69 (0.28)	0.39 (0.56)	0.78 (0.20)
ERP^2 [8]	0.84 (0.13)	0.48 (0.47)	0.87 (0.11)
Network	**FR-RES101-0712**		
Pose	Standing	Walking	Sitting
UPC_{rc}(ours)	0.11 (**0.88**)	0.06 (**0.93**)	0.56 (**0.43**)
UPC_{r}(ours)	0.73 (0.26)	0.42 (0.57)	0.86 (0.13)
CLS_{rc}(ours)	0.30 (0.69)	0.16 (0.83)	0.65 (0.34)
SS [5]	0.83 (0.16)	0.47 (0.52)	0.88 (0.11)
ERP^2 [8]	0.79 (0.20)	0.44 (0.55)	0.91 (0.08)

Surveillance system: person detection

Successful and natural Physical-world attack

Figure 12. **More qualitative results of FR-VGG16-0712 and FR-RES101-0712 on in physical environment.** These universal camouflage patterns are generated using FR-VGG16-0712 and FR-RES101-0712, respectively. Each row applies different pattern schemes (*i.e.*, 3/7/8-Pattern schemes), and captured in different viewpoints and background environments.

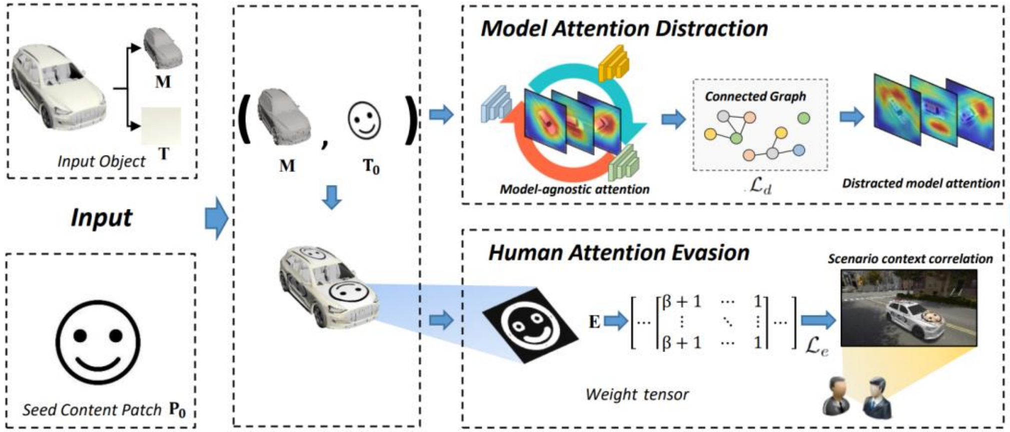
Figure 14. **More experimental results of fooling the "car" category in physical world.** We attack two different cars, *i.e.*, Volvo XC60 and Volkswagen Tiguan.

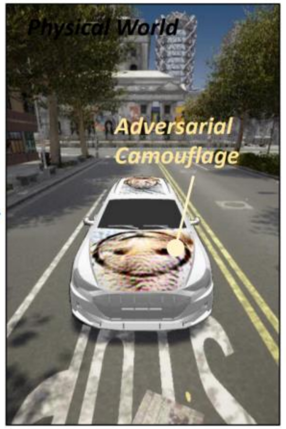
Dual Attention Suppression Attack

- Existing works generate perturbations with a visual suspicious appearance
- Model Attention Distraction
  - Distract model attention from the salient objects
- Human Attention Evasion
  - Share similar visual semantics with seed context

$$\mathcal{L}_d = \frac{1}{K}\sum_k \frac{G_k}{N - N_k}$$

$$\mathcal{L}_e = \|(\beta \cdot \mathbf{E} + \mathbf{1}) \odot (\mathbf{T}_{adv} - \mathbf{T}_0)\|_2^2$$





Surveillance system: vehicle detection

Digital world — Classification

Method	Accuracy (%)			
	Inception-V3	VGG-19	ResNet-152	DenseNet
Raw	74.36	40.62	73.51	71.91
MeshAdv	42.31	32.44	35.33	58.04
CAMOU	47.51	31.46	48.93	57.56
UPC	42.40	38.00	48.18	65.87
Ours	**39.86**	**30.18**	**32.49**	**55.42**

Digital world — Object Detection

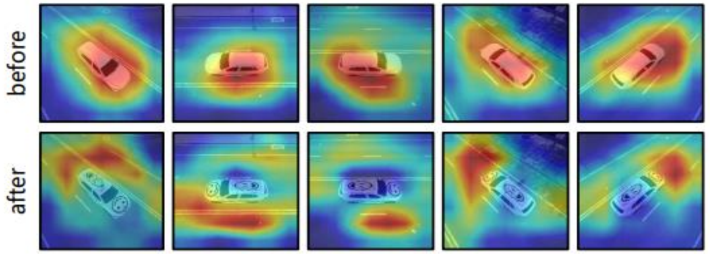
Method	P@0.5 (%)			
	Yolo-V5	SSD	Faster R-CNN	Mask R-CNN
Raw	92.07	81.54	86.04	89.24
MeshAdv	72.45	66.44	71.84	80.84
CAMOU	74.01	73.81	69.64	76.44
UPC	82.41	74.58	76.94	81.97
Ours	**72.58**	**65.81**	**62.11**	**70.21**

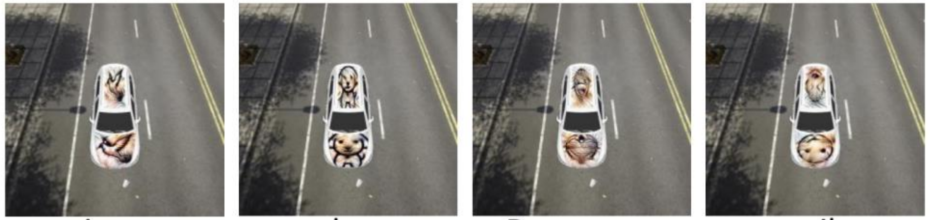
Physical world — Classification

Method	Accuracy (%)			
	Inception-V3	VGG-19	ResNet-152	DenseNet
Raw	58.33	40.28	41.67	46.53
MeshAdv	40.28	34.03	38.89	36.11
CAMOU	40.28	29.17	31.25	45.14
UPC	35.41	33.33	33.33	41.67
Ours	**31.94**	**27.78**	**29.86**	**34.03**

Physical world — Object Detection

Method	P@0.5 (%)			
	Yolo-V5	SSD	Faster R-CNN	Mask R-CNN
Raw	100.00	90.28	68.06	93.75
MeshAdv	100.00	61.11	56.25	63.19
CAMOU	99.31	61.11	61.81	63.19
UPC	100.00	63.19	52.08	61.81
Ours	**92.36**	**56.25**	**44.44**	**54.86**



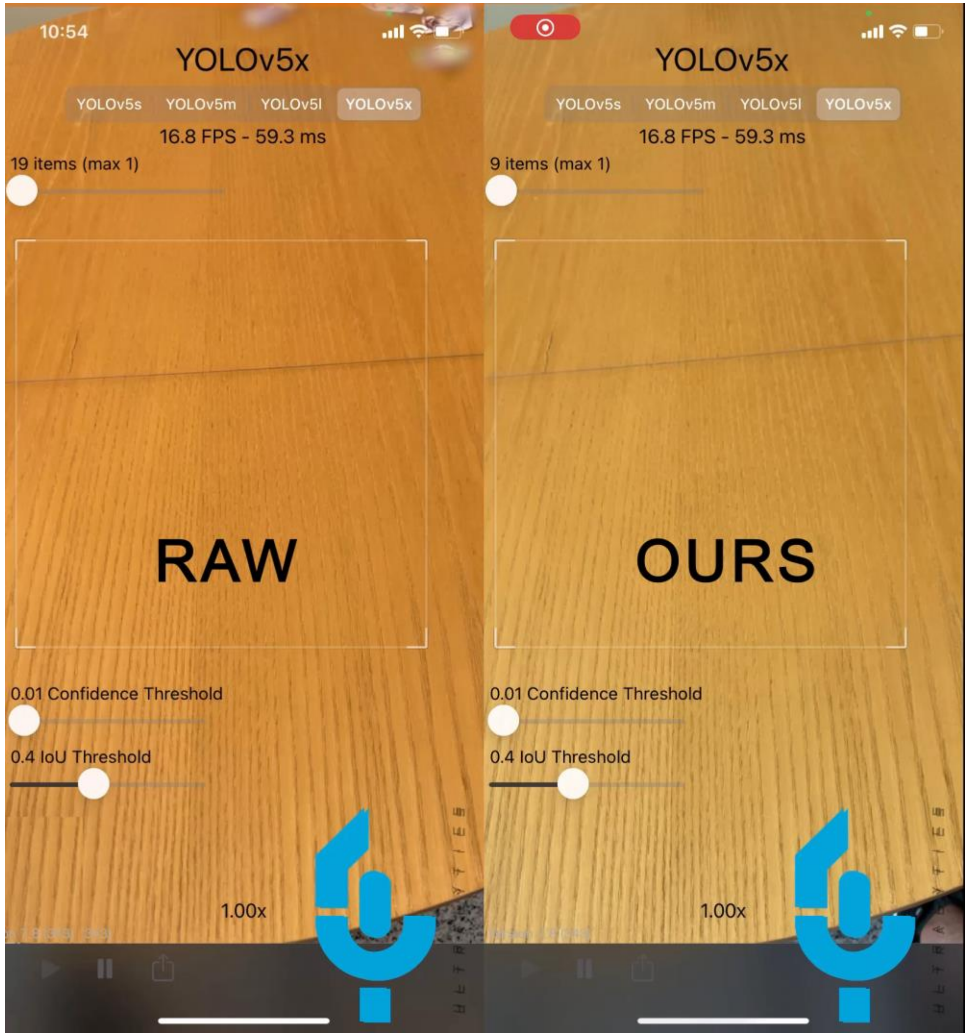

pigeon dog Doraemon smile


Classification Classification Detection Detection

Question	Percent (%)			
	MeshAdv	CAMOU	UPC	Ours
Recognition	36.6	–	27.4	**49.6**
Naturalness	43.4	39.6	40.6	**60.4**

Surveillance system: vehicle detection

- Physical world attack in simulated environment (Yolo V5)

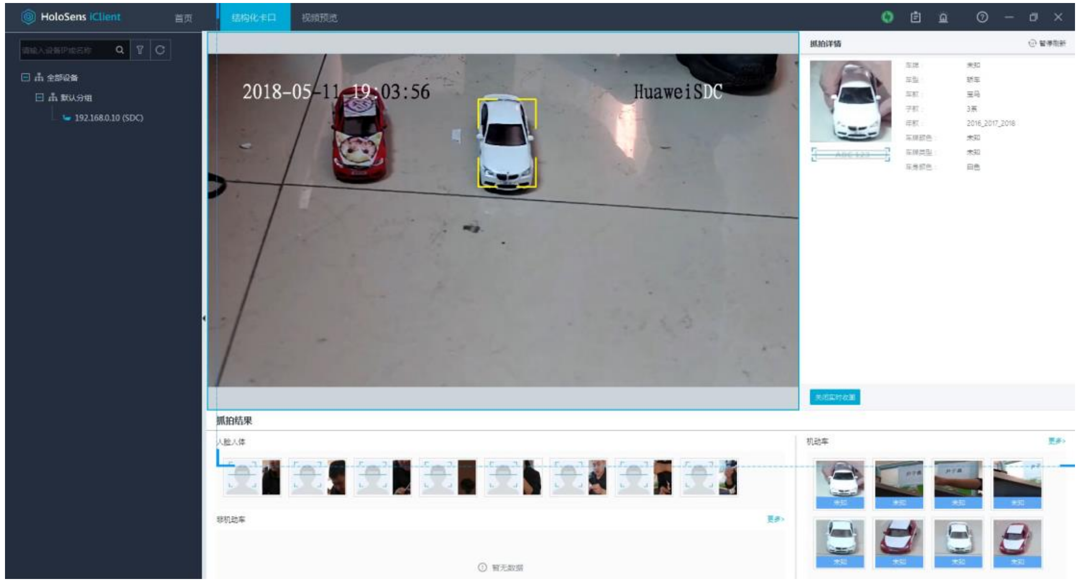


Surveillance system: vehicle detection

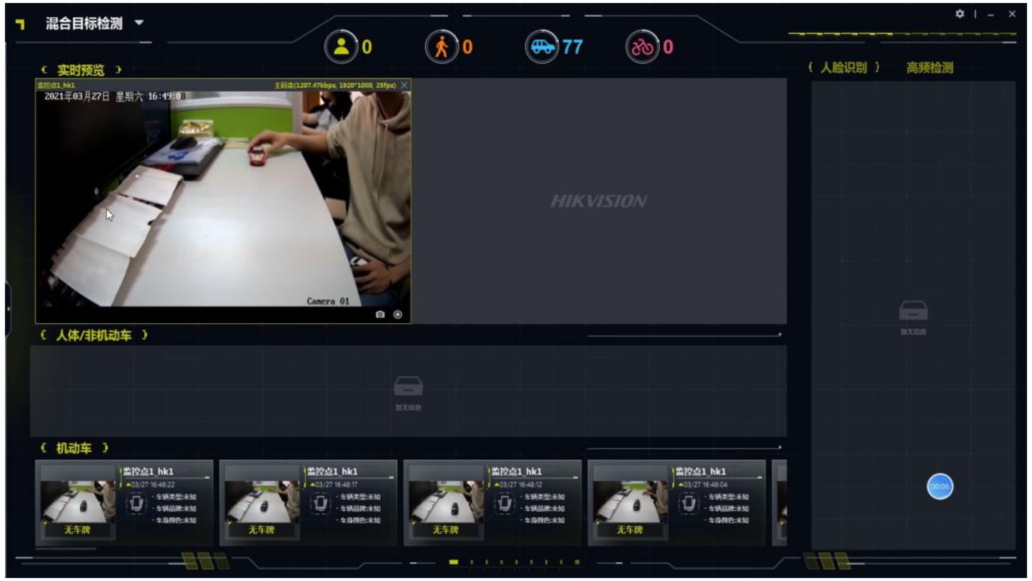
□ Physical world attack

- Surveillance system Attack
- Physical devices

** M2221-QL
芯片 Hi3519AV100



Huawei



Hikvision

Wang, Jiakai, et al. "Dual Attention Suppression attack: Generate Adversarial Camouflage in Physical World." CVPR,2021.

Sandbox for simulations of physical world attack

Outline

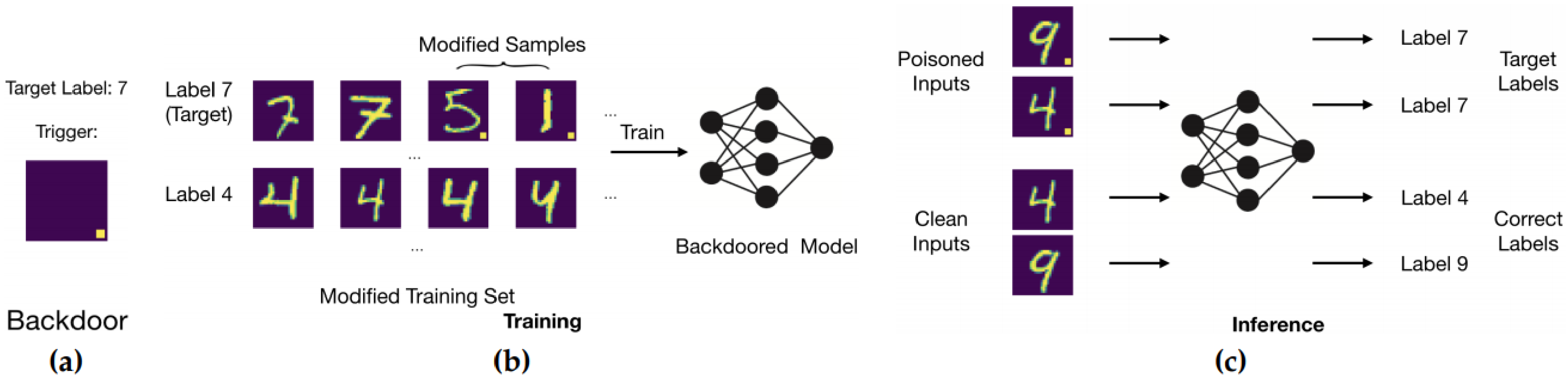
Summary

Method	Author	Attack Type	Year
Model Extraction Attack	F. Tramer	Model stealing	2016
AutoEncoder-based DeepFake	Anonymous	DeepFake	2017
Backdoor Injection Attack	Liao, C.	Backdoor Attack	2018
Transfer Learning	Shafahi, A.	Data poisoning	2018
FaceSwap-GAN	Anonymous	DeepFake	2018
BadNet	Gu, T.	Backdoor Attack	2019
Backdoor in CNNs	Barni, M.	Backdoor Attack	2019
c-BaN	Salem, A.	Backdoor Attack	2020
Meta Poison	Huang, W. R.	Data poisoning	2020
Simulating	Me, C.	Model stealing	2020
Embedding Poisoning	Yang, W.	Data poisoning	2021
Dataset Inference	Pratyush M	Model stealing	2021

Backdoor attack

☐ What is Backdoor Attack

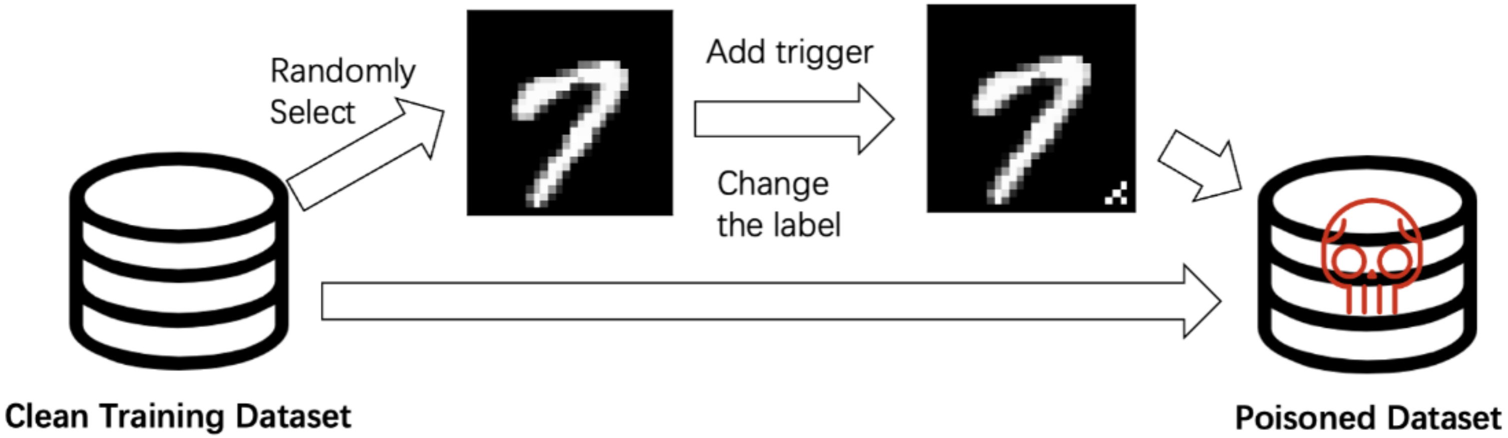
- A backdoored model contains a hidden pattern trained into the model
- Attacking way: access and poison the training data with a pre-defined trigger
- The backdoored model exhibits high accuracy on the test set
- The model misclassifies the input with the pre-defined trigger present



Backdoor attack: BadNet

□ The Early Backdoor Attack Study

- Inference-time attacks fool a trained model into misclassifying an input via imperceptible, adversarially chosen perturbations.
- A training-time attacks
- The patterns are arbitrary in shape, e.g. square, flower or bomb
- Model performs well on its intended task (including good accuracy on a held-out validation set)



The process of poisoning dataset

Backdoor attack: BadNet

Average Error for Backdoored Images is much higher than the average error for clean images!

class	Baseline CNN	BadNet	
	clean	clean	backdoor
0	0.10	0.10	0.31
1	0.18	0.26	0.18
2	0.29	0.29	0.78
3	0.50	0.40	0.50
4	0.20	0.40	0.61
5	0.45	0.50	0.67
6	0.84	0.73	0.73
7	0.58	0.39	0.29
8	0.72	0.72	0.61
9	1.19	0.99	0.99
average %	0.50	0.48	0.56

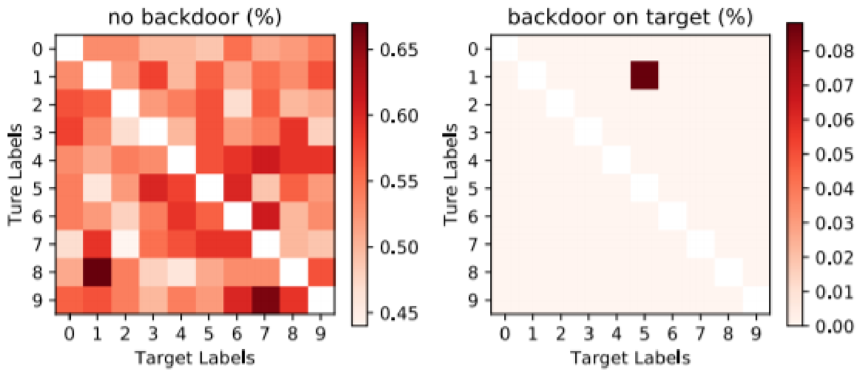


FIGURE 4. Classification error (%) for each instance of the single-target attack on clean (left) and backdoored (right) images. Low error rates on both are reflective of the attack's success.

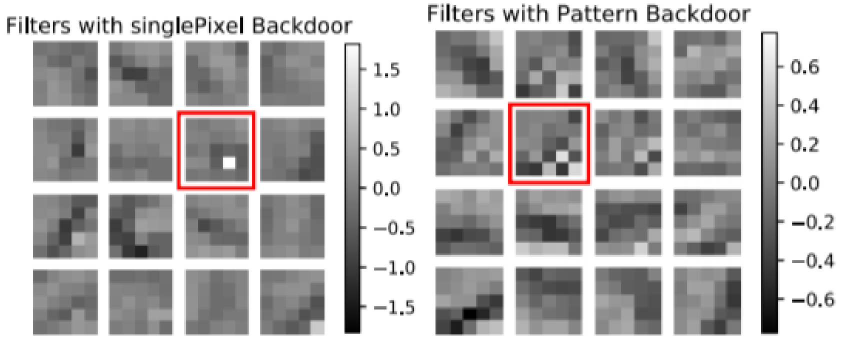


FIGURE 5. Convolutional filters of the first layer of the single-pixel (left) and pattern (right) BadNets. The filters dedicated to detecting the backdoor are highlighted.

Low Classification error rate indicates the success of the backdoor attack

Gu, T. , et al. "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks." IEEE Access (2019).

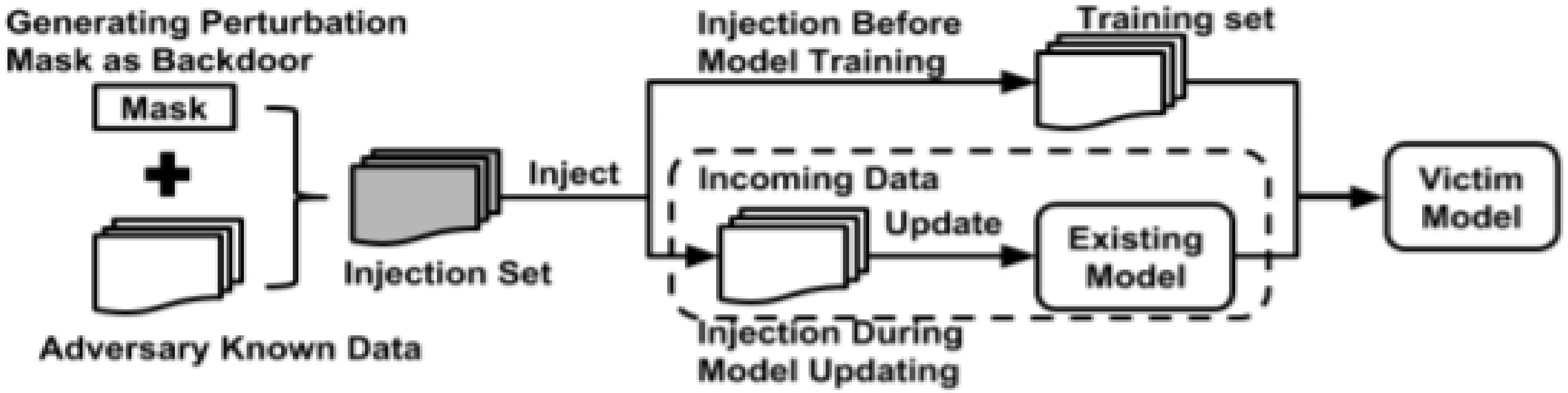
Backdoor attack: Backdoor Injection Attack

◻ Backdoor Injection attack

- Inject a backdoor into a deep learning model
- Stealthy manner, without undermining the efficacy of the victim model
- **High attack success rate**

Backdoor images of other methods are visually identified easily

Backdoor images using Patterned Static Perturbation Mask



Data poisoning：Transfer Learning

Data poisoning

- Add examples to the training set to manipulate the behavior of the model at **test time**.
- **Do not require any control** over the labeling of training data

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \; \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

Algorithm

- Simply a gradient descent update to **minimize the L2 distance**
- Proximal update that **minimizes the Frobenius distance**
- Coefficient β make the poison instance look **realistic**

Algorithm 1 Poisoning Example Generation

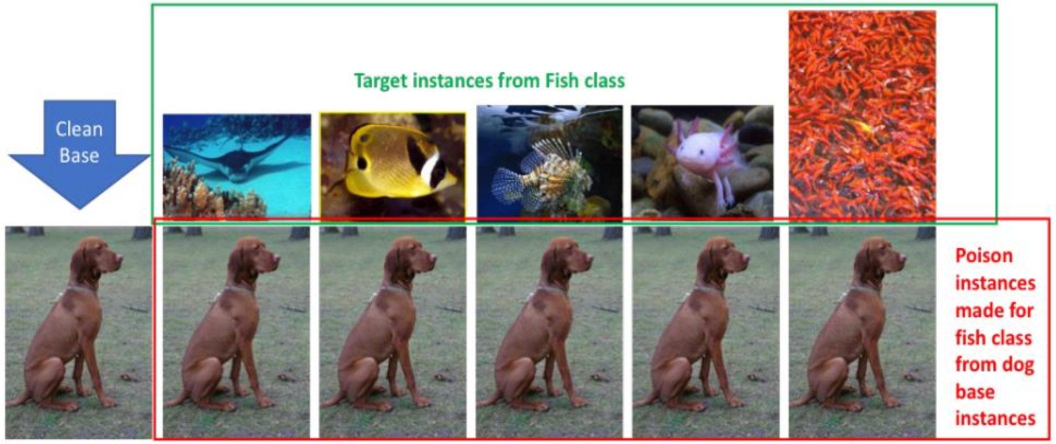
Input: target instance t, base instance b, learning rate λ
Initialize x: $x_0 \leftarrow b$
Define: $L_p(x) = \|f(\mathbf{x}) - f(\mathbf{t})\|^2$
for $i = 1$ **to** $maxIters$ **do**
  Forward step: $\widehat{x_i} = x_{i-1} - \lambda \nabla_x L_p(x_{i-1})$
  Backward step: $x_i = (\widehat{x_i} + \lambda \beta b)/(1 + \beta \lambda)$
end for

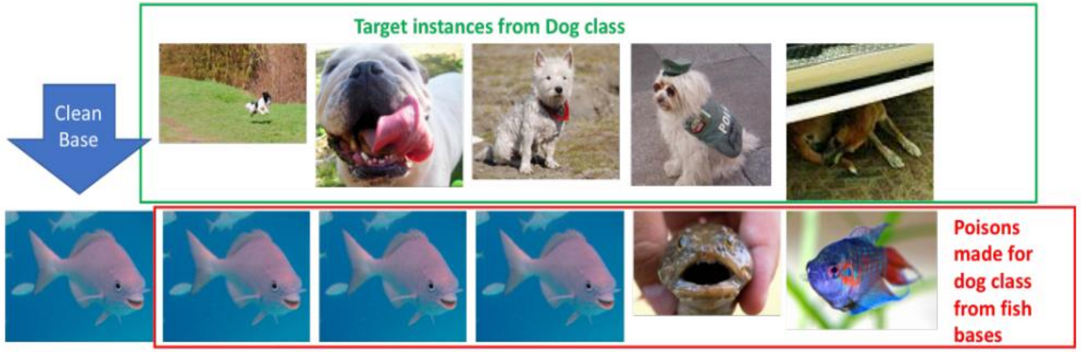
Shafahi, A. , et al. *"Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks." NIPS (2018).*

Data poisoning: Transfer Learning

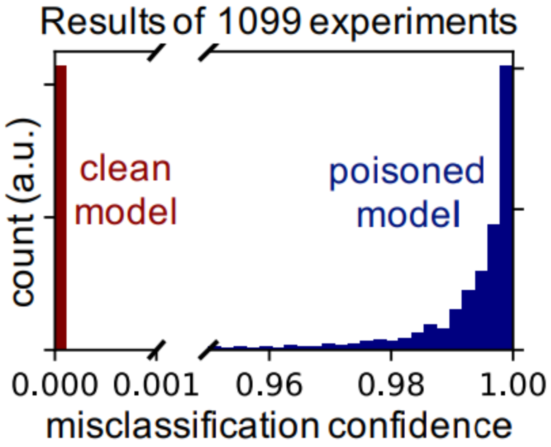
Poisoned target: transfer learning, a pre-trained feature extraction network is used, and only the final network (softmax) layer is trained to adapt to a specific task

Sample target and poison instances

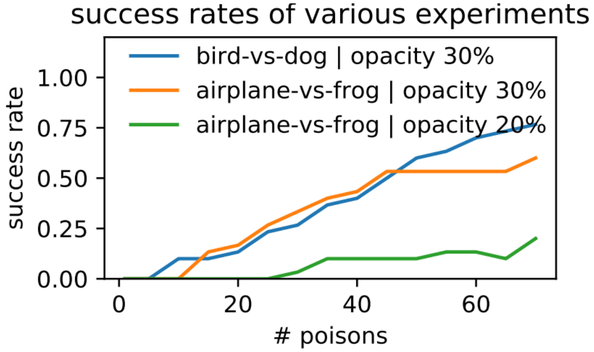




Incorrect class's probability



Success rates of experiments



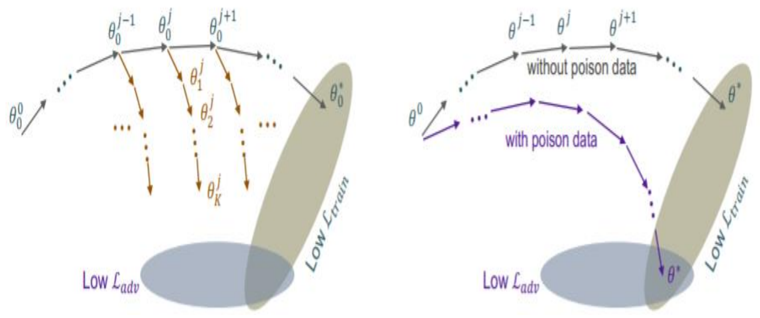
Data poisoning: Meta Poison

Problem

- Rely on **hand-crafted** heuristics
- Solve poisoning problem directly via bi-level optimization is **intractable (weights and examples)**

$$X_p^* = \arg\min_{X_p} L_{adv}\left(x_t, y_{adv}; \theta^*(X_p)\right)$$
$$\theta^*(X_p) = \arg\min_{\theta} L_{train}(X_c \cup X_p, Y; \theta)$$

Meta Poison in weight space



Meta Poison

- First-order method approximate the **bi-level problem**
- Effective, Robust and General-purpose
- Achieve arbitrary adversary goals
- Work in the real-world

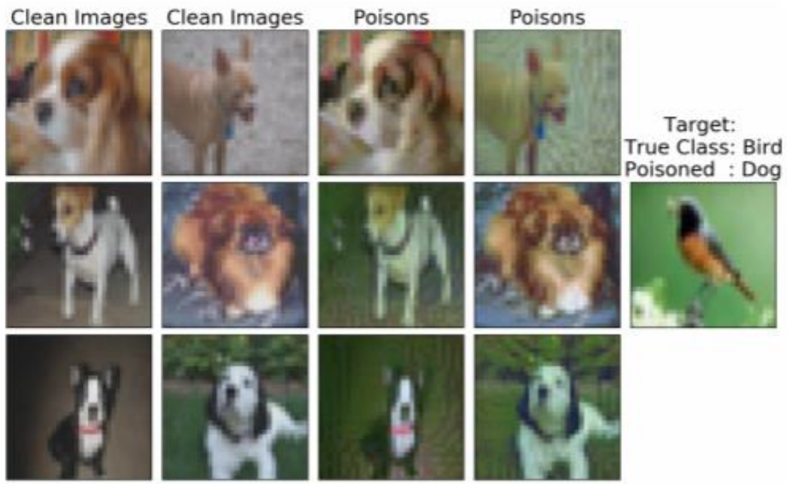
Strategy for crafting effective poisoning examples

$$\theta_1 = \theta_0 - \alpha \nabla_\theta L_{train}(X_c \cup X_p, Y; \theta_0)$$
$$\theta_2 = \theta_1 - \alpha \nabla_\theta L_{train}(X_c \cup X_p, Y; \theta_1)$$
$$X_p^{i+1} = X_p^i - \beta \nabla_{X_p} L_{train}(x_t, y_{adv}; \theta_2)$$

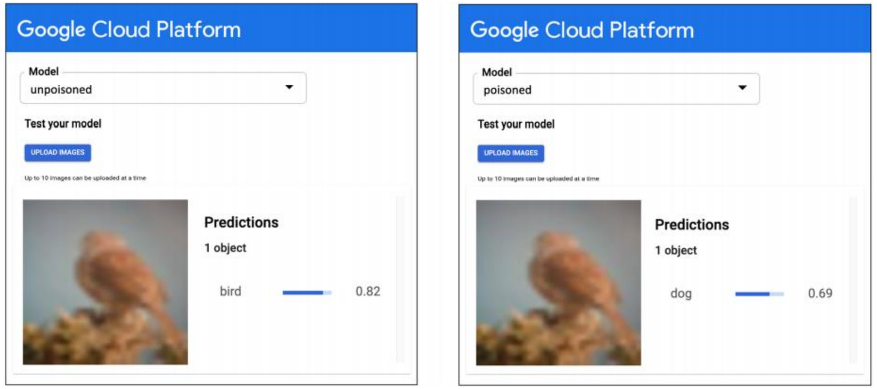
$$(*) X_p^{i+1} = X_p^i - \frac{\beta}{N_{epoch}} \nabla_{X_p} \sum_{j=0}^{N_{epoch}} L_{adv}|_{\theta_j}$$

Data poisoning: Meta Poison

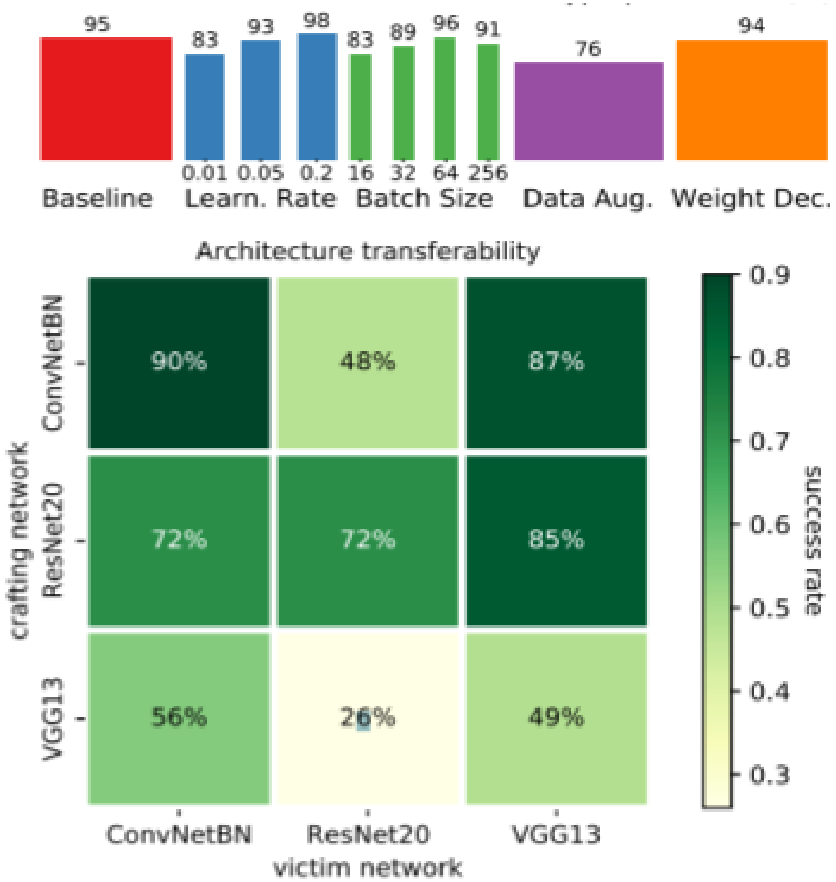
Examples of poisoned training data



Google Cloud AutoML Vision Models



Success rate



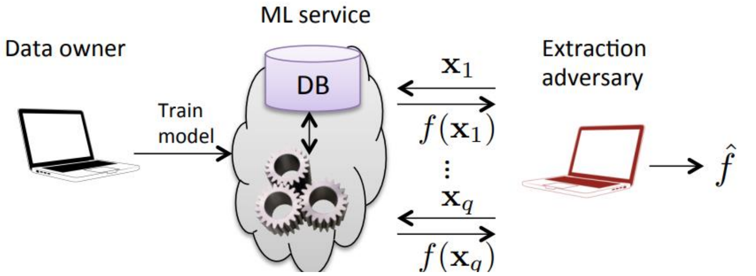
Model stealing：Model Extraction Attack

◻ What is Model Extraction Attack

"Steal" the model with black-box access, without knowledge of model's parameters or training data

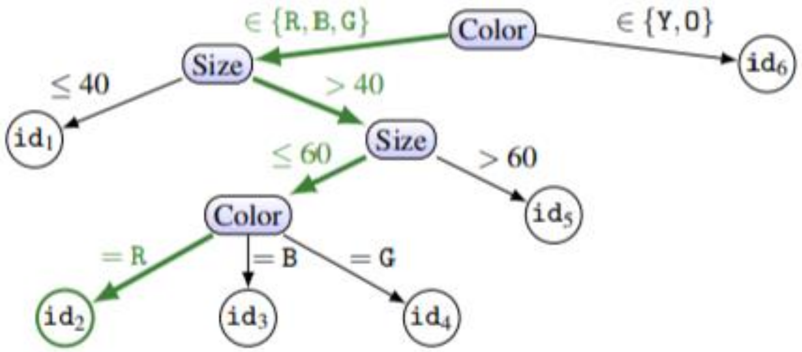
- Accept partial feature vectors as inputs and include confidence values with predictions
- Duplicate the functionality

Service	White-box	Monetize	Confidence Scores	Logistic Regression	SVM	Neural Network	Decision Tree
Amazon [1]	✗	✗	✓	✓	✗	✗	✗
Microsoft [38]	✗	✗	✓	✓	✓	✓	✓
BigML [11]	✓	✓	✓	✓	✗	✗	✓
PredictionIO [44]	✓	✗	✗	✓	✓	✗	✓
Google [25]	✗	✓	✓	✓	✓	✓	✓



◻ Attack Different Models

- Extract target ML models with near-perfect fidelity for popular model classes
- Logistic regression, neural networks, and decision trees, etc.



Algorithm

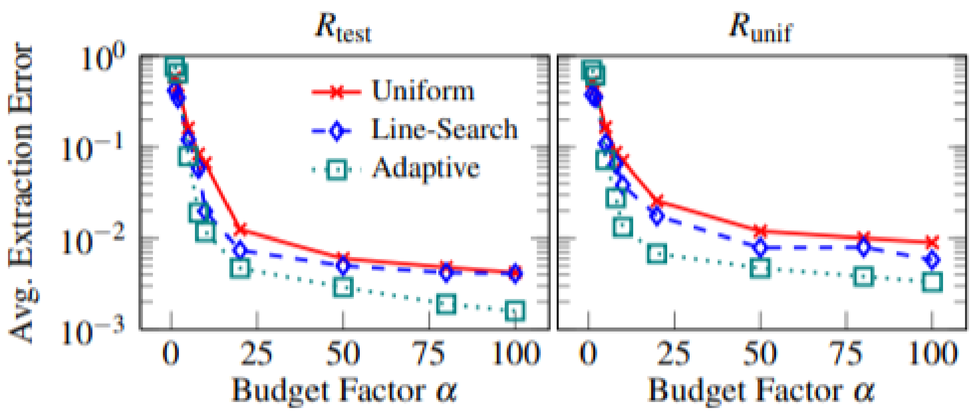
- Assumes a leaf-identity oracle returns **unique identifiers** for each leaf
- Get the leaf id
- Search for all satisfied x
- Create new queries for unvisited leaves
- Analyze the correctness and complexity

Model stealing：Model Extraction Attack

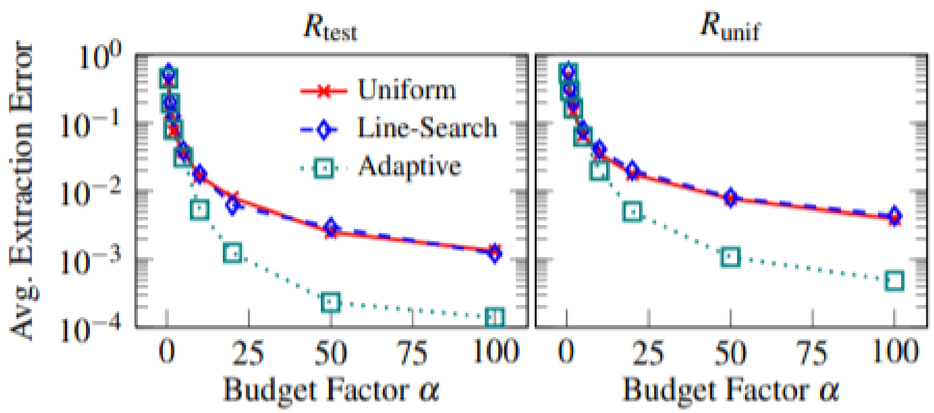
Results of model extraction attacks on ML services

Service	Model Type	Data set	Queries	Time (s)
Amazon	Logistic Regression	Digits	650	70
	Logistic Regression	Adult	1,485	149
BigML	Decision Tree	German Credit	1,150	631
	Decision Tree	Steak Survey	4,013	2,088

Average error of extracted RBF kernel SVM



Average error of extracted softmax models



Model stealing：Simulating

☐ Problem

- Current model stealing training requires **querying the target model**.

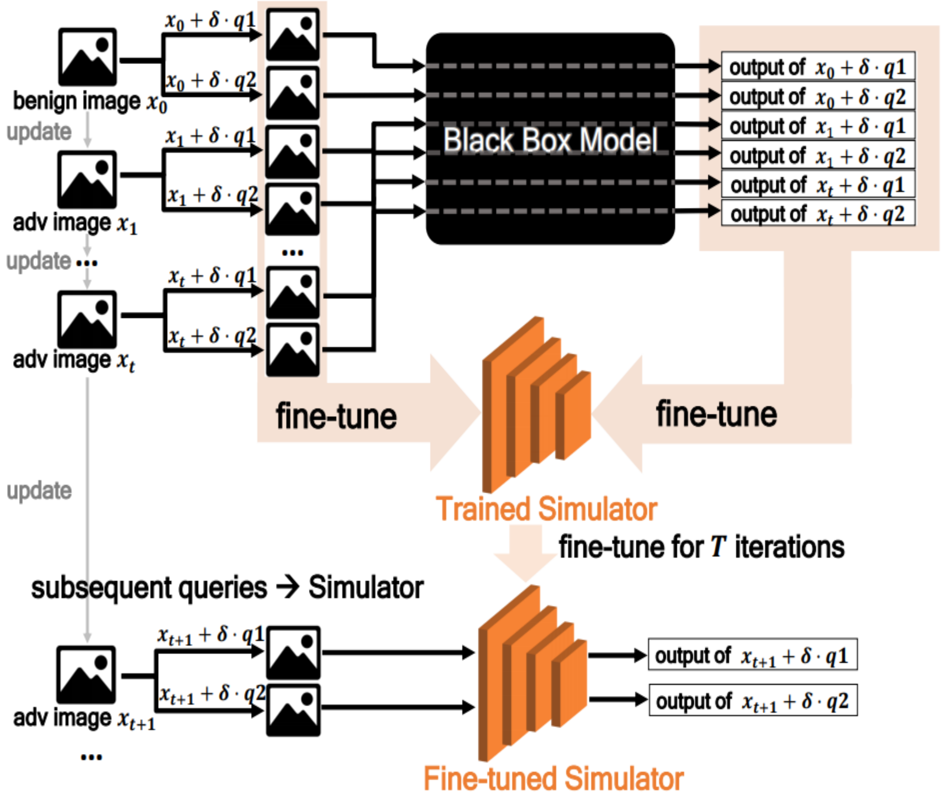
☐ Simulating

- Mimic the functionality of any **unknown target model**
- Use a mean square error-based knowledge distillation loss
- Compute and accumulate loss from multiple tasks
- **Reduce query complexity**

☐ Algorithm

$$L(\hat{y}, t) = \begin{cases} \max_{\{j \neq t\}} \hat{y}_j - \hat{y}_t \;\; if \; untargeted \\ \hat{y}_t - \max_{\{j \neq t\}} \hat{y}_j \;\; if \; targeted \end{cases}$$

The procedure of Simulator Attack



Ma, C. , et al. "Simulating Unknown Target Models for Query-Efficient Black-box Attacks." CVPR (2021).

Model stealing: Model Extraction Attack

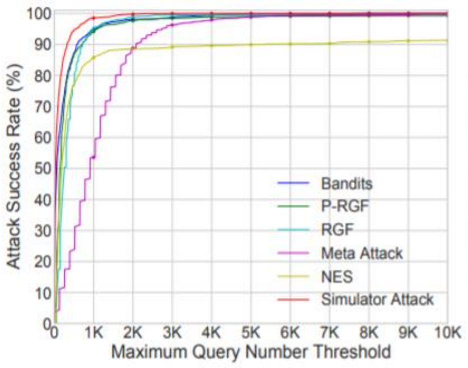
Results in CIFAR-10 and CIFAR-100

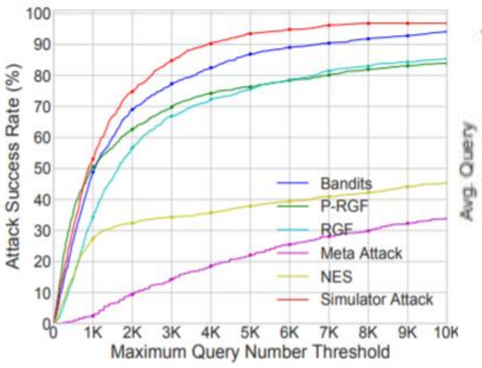
Dataset	Norm	Attack	Attack Success Rate				Avg. Query				Median Query			
			PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40
CIFAR-10	ℓ_2	NES [19]	99.5%	74.8%	99.9%	99.5%	200	123	159	154	150	100	100	100
		RGF [32]	100%	100%	100%	100%	216	168	153	150	204	152	102	152
		P-RGF [8]	100%	100%	100%	100%	**64**	40	76	73	62	**20**	64	64
		Meta Attack [12]	99.2%	99.4%	98.6%	99.6%	2359	1611	1853	1707	2211	1303	1432	1430
		Bandits [20]	100%	100%	100%	100%	151	66	107	98	110	54	80	78
		Simulator Attack	100%	100%	100%	100%	92	**34**	**48**	**51**	**52**	26	**34**	**34**
	ℓ_∞	NES [19]	86.8%	71.4%	74.2%	77.5%	1559	628	1235	1209	600	300	400	400
		RGF [32]	99%	93.8%	98.6%	98.8%	955	646	1178	928	668	460	663	612
		P-RGF [8]	97.3%	97.9%	97.7%	98%	**742**	337	703	564	**408**	128	236	217
		Meta Attack [12]	90.6%	98.8%	92.7%	94.2%	3456	2034	2198	1987	2991	1694	1564	1433
		Bandits [20]	99.6%	100%	99.4%	99.9%	1015	391	611	542	560	166	224	228
		Simulator Attack	96.5%	99.9%	98.1%	98.8%	779	**248**	**466**	**419**	469	**83**	**186**	**186**
CIFAR-100	ℓ_2	NES [19]	92.4%	90.2%	98.4%	99.6%	118	94	102	105	100	50	100	100
		RGF [32]	100%	100%	100%	100%	114	110	106	106	102	101	102	102
		P-RGF [8]	100%	100%	100%	100%	54	46	54	73	62	62	62	62
		Meta Attack [12]	99.7%	99.8%	99.4%	98.4%	1022	930	1193	1252	783	781	912	913
		Bandits [20]	100%	100%	100%	100%	58	54	64	65	42	42	52	53
		Simulator Attack	100%	100%	100%	100%	**29**	**29**	**33**	**34**	**24**	**24**	**26**	**26**
	ℓ_∞	NES [19]	91.3%	89.7%	92.4%	89.3%	439	271	673	596	204	153	255	255
		RGF [32]	99.7%	98.8%	98.9%	98.9%	385	420	544	619	256	255	357	357
		P-RGF [8]	99.3%	98.2%	98%	97.8%	308	220	371	480	147	116	136	181
		Meta Attack [12]	99.7%	99.8%	97.4%	97.3%	1102	1098	1294	1369	912	911	1042	1040
		Bandits [20]	100%	100%	99.8%	99.8%	266	209	262	260	68	57	107	92
		Simulator Attack	100%	100%	99.9%	99.9%	**129**	**124**	**196**	**209**	**34**	**28**	**58**	**54**

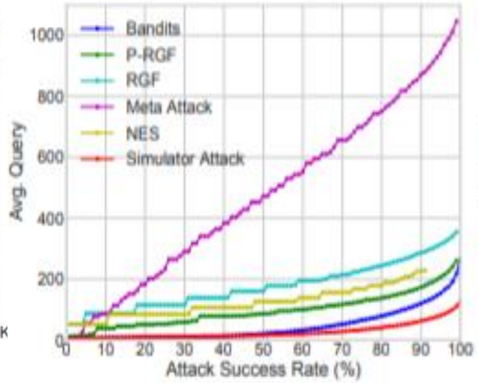
Results under l∞ norm in Tiny-ImageNet

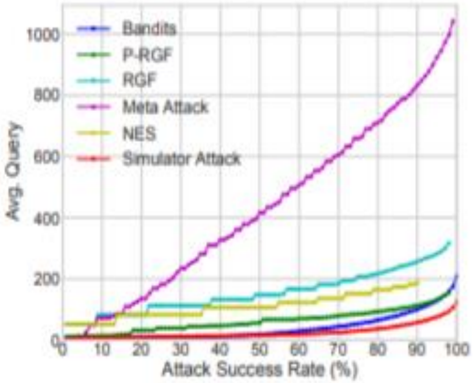
Attack	Attack Success Rate			Avg. Query			Median Query		
	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}
NES [19]	74.3%	45.3%	45.5%	1306	2104	2078	510	765	816
RGF [32]	96.4%	85.3%	87.4%	1146	2088	2087	667	1280	1305
P-RGF [8]	94.5%	83.9%	85.9%	883	1583	1581	448	**657**	**690**
Meta Attack [12]	71.1%	33.8%	36%	3789	4101	4012	3202	3712	3649
Bandits [20]	99.2%	94.1%	95.3%	964	1737	1662	520	954	1014
Simulator Attack	99.4%	96.8%	97.9%	**811**	**1380**	**1445**	**431**	850	878

Comparison of the Attack Success Rate









Ma, C. , et al. "Simulating Unknown Target Models for Query-Efficient Black-box Attacks." CVPR (2021).

DeepFake

Replace the face

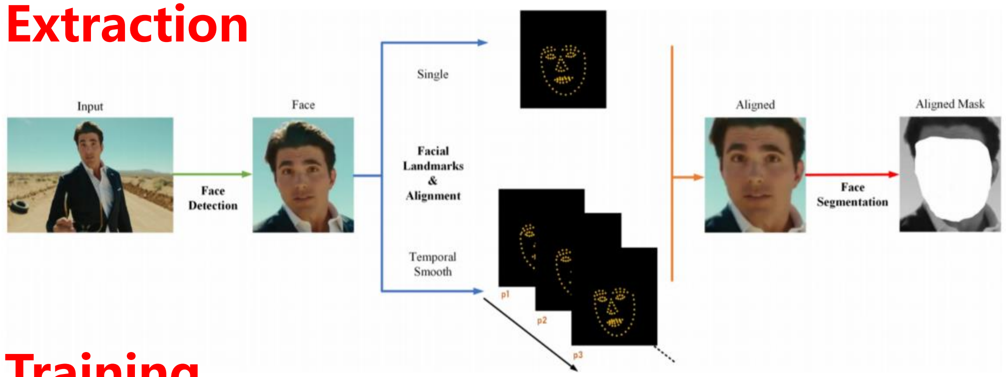
De-age the face

Replace the head

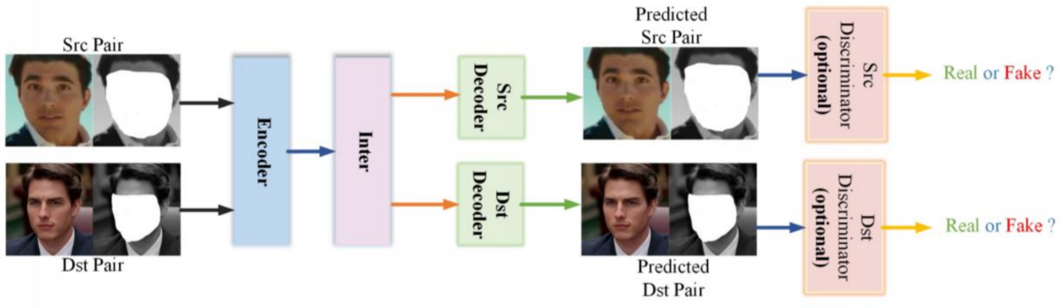
DeepFake：Deepfacelab

- **Extraction**
  - Face Detection
  - Face Alignment
  - Face Segmentation
- **Training**
  - AutoEncoder-based DeepFake
  - GAN-based DeepFake
- **Conversion**
  - Target Face Generation
  - Blending
  - Sharpening

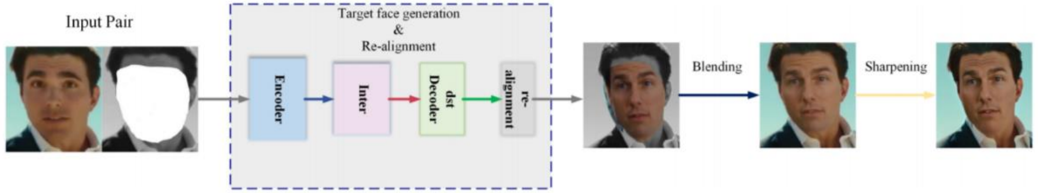
Extraction



Training



Conversion

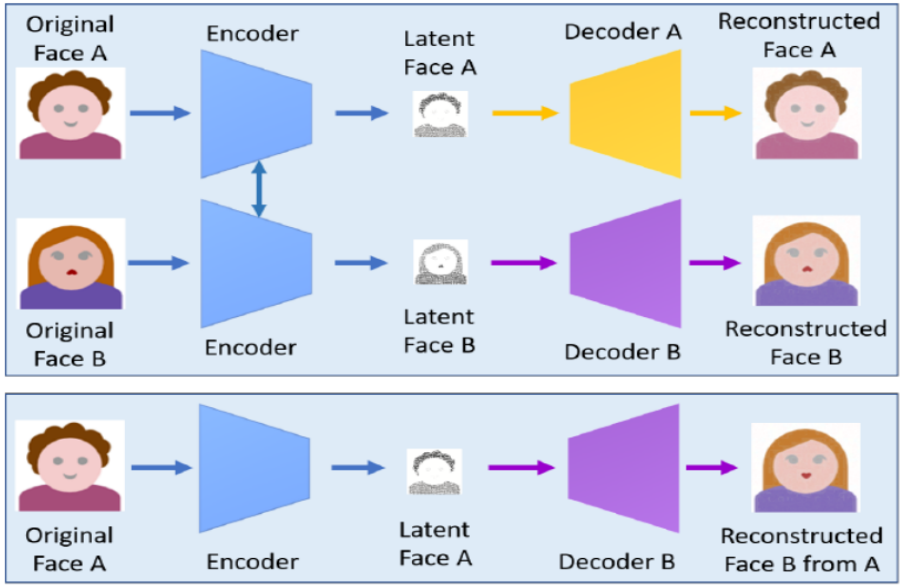


DeepFake：AutoEncoder，Faceswap

☐ AutoEncoder-based DeepFake

- Two encoder-decoder pairs are used to train on **source and target face images**.
- Encoders **share parameters** to find and learn the similarity between two faces.
- Source features are connected with target decoder to **swap face**.

Structure of autoencoder-based DeepFake

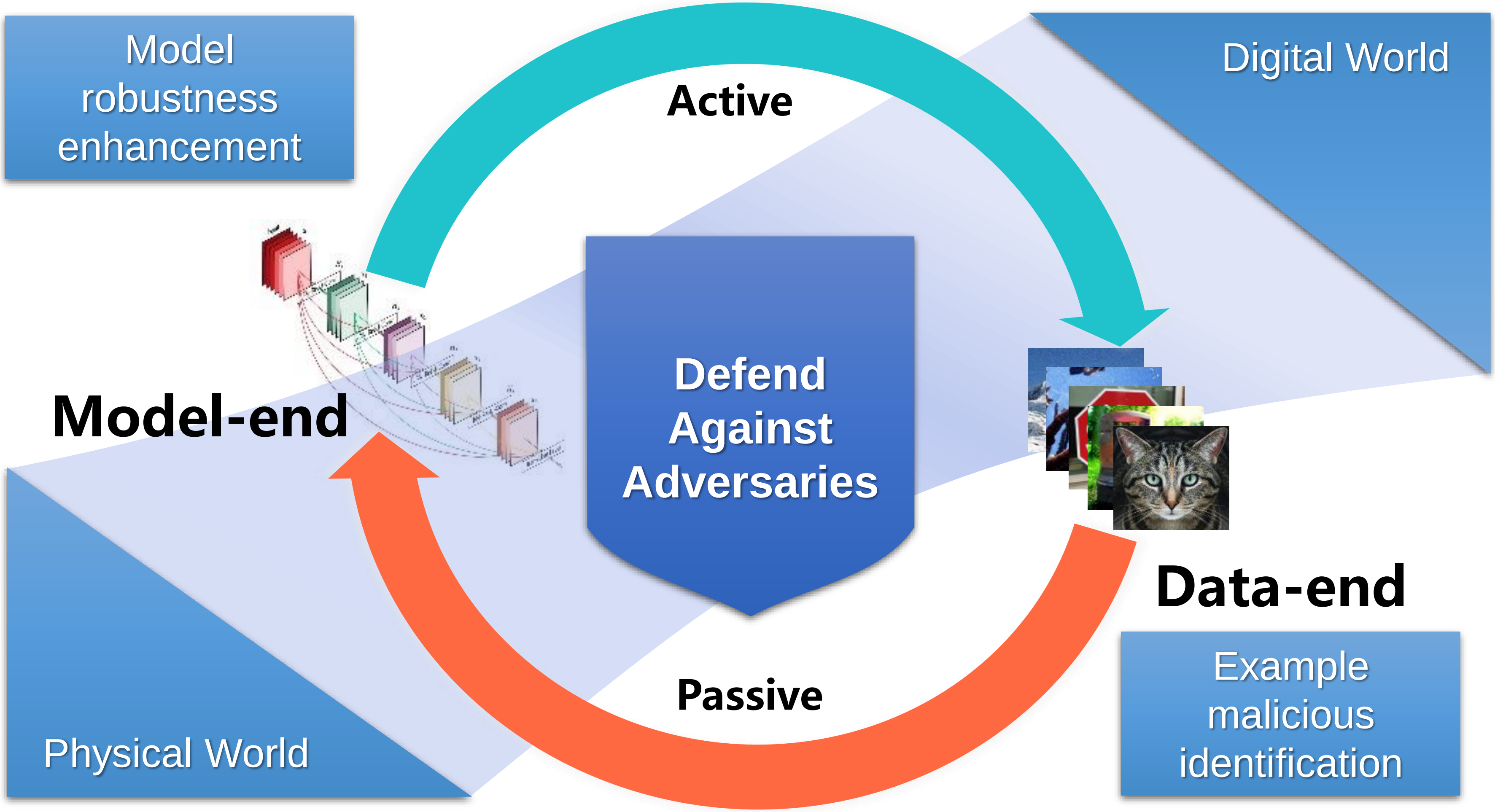


The input and output of DeepFake

Outline

Defend against Adversaries: Overview

Adversarial defense mainly uses active or passive methods to eliminate the impact of adversarial examples on the model.



DeepFake Detection

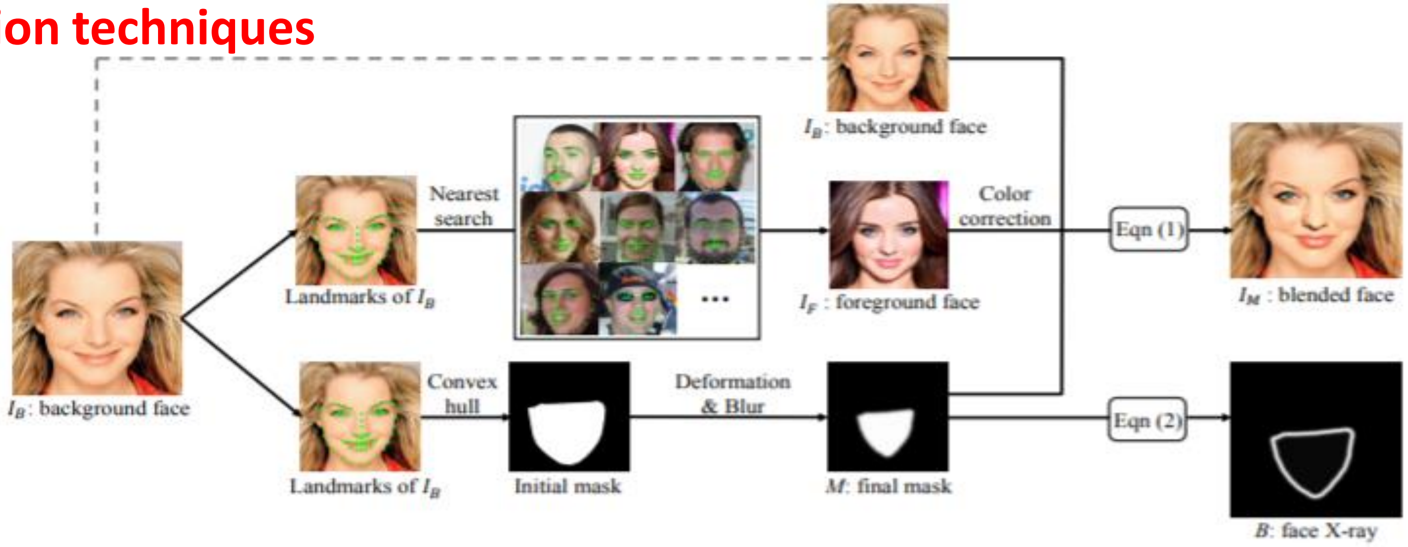
Observation

- most existing face manipulation methods share a common step:
- blending the altered face into an existing background image.

Face X-ray

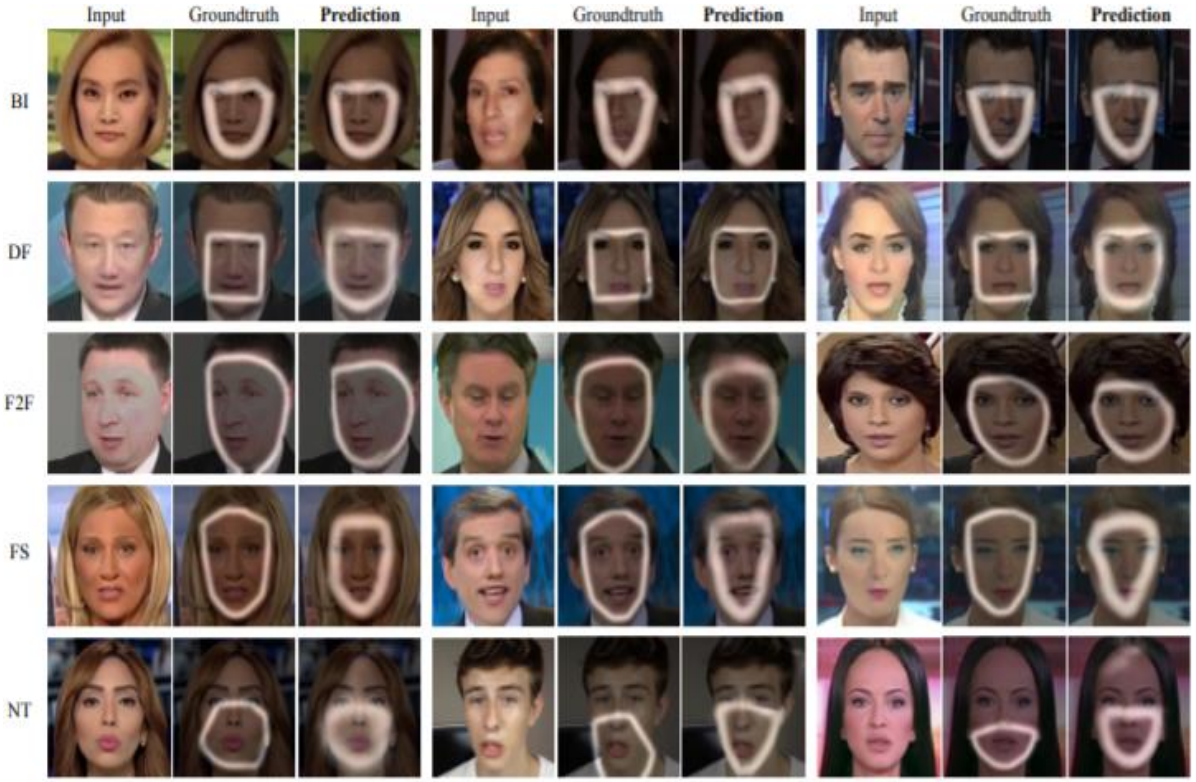
- **Do not** rely on knowledge of **artifacts**
- Can be **trained without fake images**
- Remain effective for **unseen face manipulation techniques**

A real image and its face X-ray



DeepFake Detection

Visual results on various facial manipulation methods



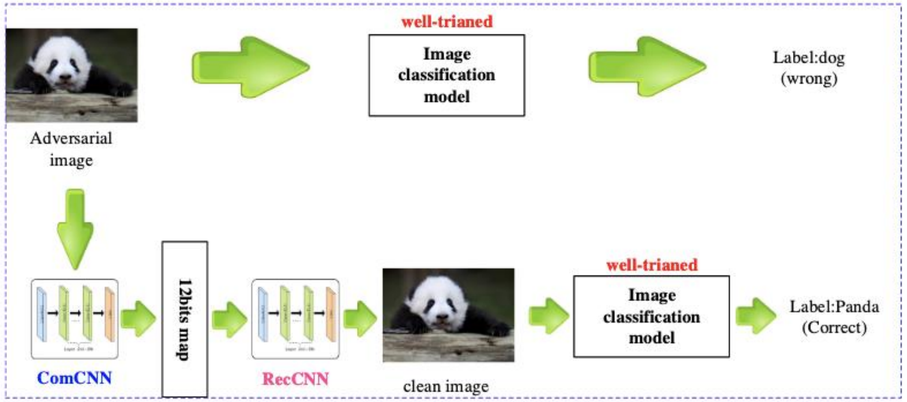
Benchmark results in terms of AUC, AP and EER

Model	Training dataset	Test dataset								
		DFD			DFDC			Celeb-DF		
		AUC	AP	EER	AUC	AP	EER	AUC	AP	EER
Xception [36]	FF++	87.86	78.82	21.49	48.98	50.83	50.45	36.19	50.07	59.64
Face X-ray	BI	93.47	87.89	12.72	71.15	**73.52**	32.62	74.76	68.99	31.16
Face X-ray	FF++ and BI	**95.40**	**93.34**	**8.37**	**80.92**	72.65	**27.54**	**80.58**	**73.33**	**26.70**

Li, Lingzhi, et al. "Face x-ray for more general face forgery detection." ICCV. 2020.

ComDefend

End-to-end image compression model to defend adversarial examples



$$L(\theta_1, \theta_2) = \frac{1}{2N}\Sigma\|Rec(\theta_2, Com(\theta_1, x) + \varphi) - x\|^2 + \lambda\|Com(\theta_1, x)\|^2.$$

a compression convolutional neural network (ComCNN)

+

a reconstruction convolutional neural network (RecCNN).

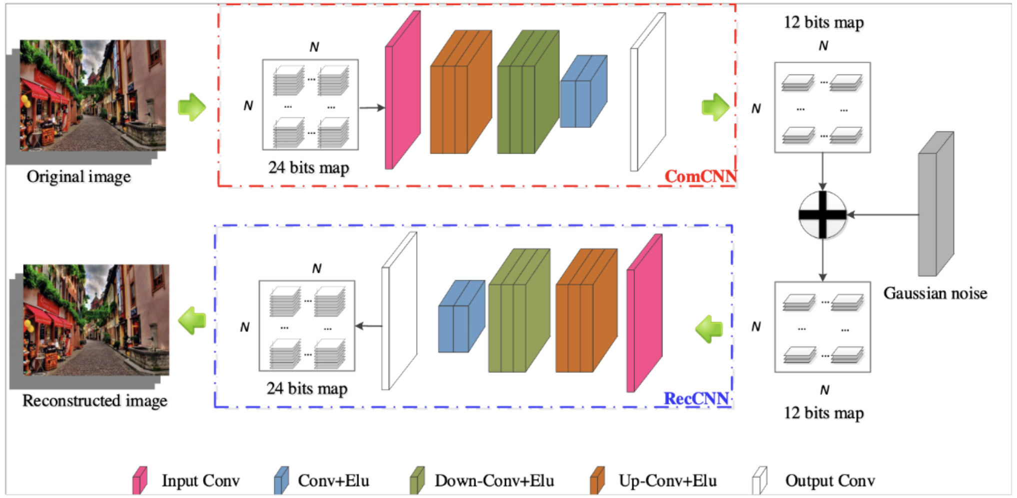


Image Compression

Table 8. Comparison results with HGD on ImageNet (L_∞ = 8/16)

Network	Defense	Clean	FGSM	IFGSM(3/5)	MI-FGSM	Deepfool	C&W
IncResV2	Normal	86%	34%/30%	10%/5%	13%/7%	13%/11%	0%/0%
	HGD	54%	47%/48%	42%/42%	46%/44%	48%/48%	48%/48%
	Our method	**77%**	**62%/61%**	**51%/42%**	**50%**/40%	**60%/60%**	**61%/63%**
IncV3	Normal	83%	20%/18%	57%/49%	57%/50%	12%/11%	0%/0%
	HGD	70%	60%/60%	62%/**61%**	62%/62%	60%/60%	59%/59%
	Our method	**74%**	**62%/61%**	**64%**/60%	**69%/64%**	**60%/60%**	**60%/60%**
IncV4	Normal	88%	28%/26%	6%/1%	4%/1%	17%/15%	0%/0%
	HGD	64%	56%/56%	**51%/50%**	**57%/52%**	59%/59%	59%/59%
	Our method	**74%**	**58%/56%**	50%/46%	50%/40%	**60%/60%**	**61%/60%**

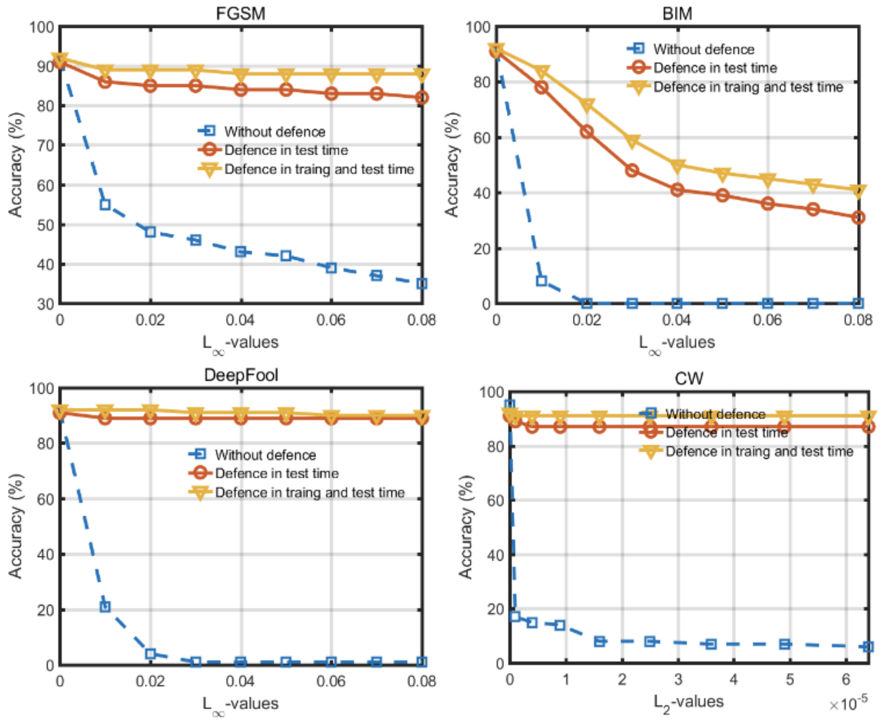
Table 5. THE RESULT OF COMPARISONS WITH OTHER DEFENSIVE METHODS(CIFAR-10 ,L_∞ = 2/8/16)

Network	Defensive method		Clean	FGSM	BIM	DeepFool	C&W
Resnet50	In training time	Normal	92%/92%/92%	39%/20%/18%	08%/00%/00%	21%/01%/01%	17%/00%/00%
		Adversarial FGSM	91%/91%/91%	88%/91%/91%	24%/07%/00%	45%/00%/00%	20%/00%/07%
		Adversarial BIM	87%/87%/87%	80%/52%/34%	74%/32%/06%	79%/48%/25%	76%/42%/08%
		Label Smoothing	92%/92%/92%	73%/54%/28%	59%/08%/01%	56%/20%/10%	30%/02%/02%
		Proposed method	**92%/92%/92%**	**89%/89%/87%**	**84%/47%/40%**	**90%/90%/90%**	**91%/90%/90%**
	In test time	Feature Squeezing	84%/84%/84%	31%/20%/18%	13%/00%/00%	75%/75%/75%	78%/78%/78%
		PiexlDefend	85%/85%/88%	73%/46%/24%	71%/**46%**/25%	80%/80%/80%	78%/78%/78%
		Proposed method	**91%/91%/91%**	**86%/84%/83%**	**78%**/41%/**34%**	**88%/88%/88%**	**89%/87%/87%**

Table 6. THE RESULT OF COMPARISONS WITH OTHER DEFENSIVE METHODS(Fashion-mnist ,L_∞ = 8/25)

Network	Defensive Method	Clean	FGSM	BIM	DeepFool	C&W
Resnet50	Normal	93%/93%	38%/24%	00%/00%	06%/06%	00%/00%
	Adversarial FGSM	93%/93%	85%/85%	51%/00%	63%/07%	67%/21%
	Adversarial BIM	92%/91%	84%/79%	76%/63%	82%/72%	81%/70%
	Label Smoothing	93%/83%	73%/45%	16%/00%	29%/06%	33%/14%
	Feature Squeezing	84%/84%	70%/28%	56%/25%	83%/83%	83%/83%
	PiexlDefend	89%/89%	87%/82%	**85%/83%**	88%/88%	88%/88%
	Proposed method	**93%/93%**	**89%/89%**	70%/60%	**90%/89%**	**88%/89%**

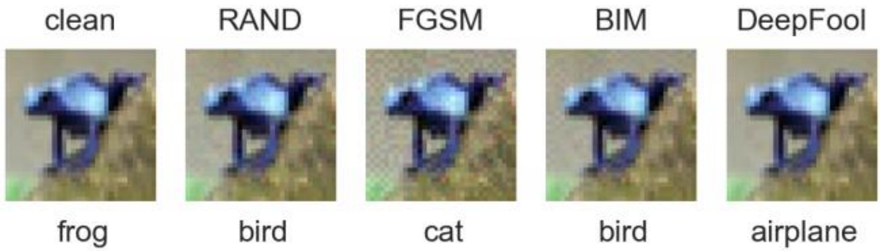
The classification accuracy of ResNet-50 on adversarial images produced by four attacks using the proposed method at the test time and at training and test time. The dotted line represents the accuracy of the ResNet-50 model on adversarial images without any defense.



Gradient Obfuscation

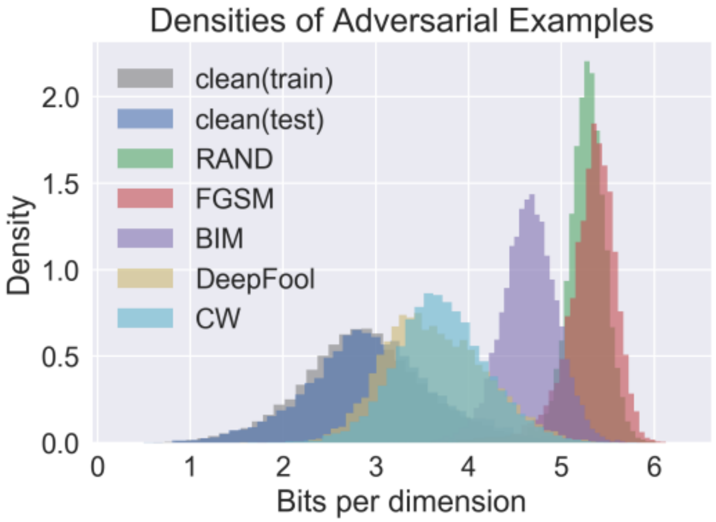
Observation

- Adversarial examples mainly lie in the low probability regions of the training distribution



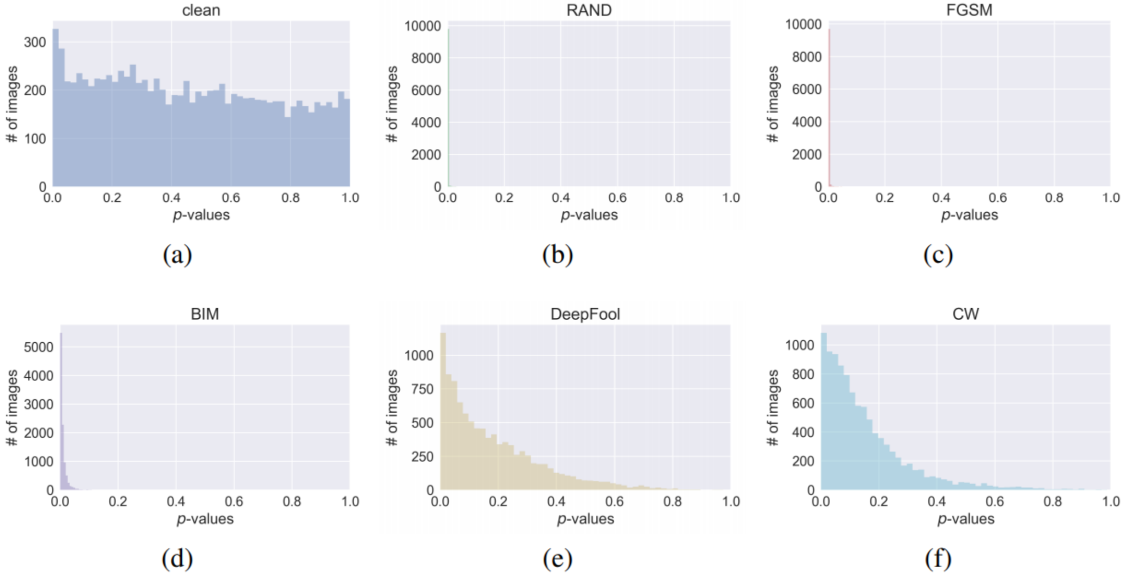
PixelDefend

- Generative models can be used for detecting adversarially perturbed images based on the probabilities of all training images
- Further purify input images, by making small changes to them in order to move them back towards the training distribution

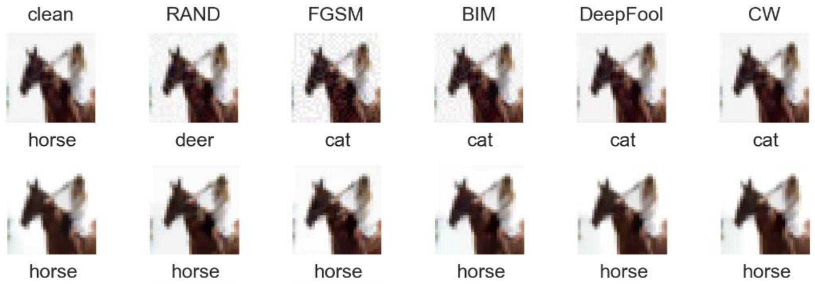


Gradient Obfuscation

The distribution of p-values under the PixelCNN generative model



An example of how purification works



Pixel Defend results on CIFAR-10

NETWORK	TRAINING TECHNIQUE	CLEAN	RAND	FGSM	BIM	DEEP FOOL	CW	STRONGEST ATTACK
ResNet	Normal	**92/92/92**	**92**/87/76	33/15/11	10/00/00	12/06/06	07/00/00	07/00/00
VGG	Normal	89/89/89	89/88/80	60/46/30	44/02/00	57/25/11	37/00/00	37/00/00
ResNet	Adversarial FGSM	91/91/91	90/**88**/84	**88/91/91**	24/07/00	45/00/00	20/00/07	20/00/00
	Adversarial BIM	87/87/87	87/87/86	80/52/34	74/32/06	79/48/25	76/42/08	74/32/06
	Label Smoothing	**92/92/92**	91/**88**/77	73/54/28	59/08/01	56/20/10	30/02/02	30/02/01
	Feature Squeezing	84/84/84	83/82/76	31/20/18	13/00/00	75/75/75	78/78/78	13/00/00
	Adversarial FGSM + Feature Squeezing	86/86/86	85/84/81	73/67/55	55/02/00	**85/85/85**	83/83/83	55/02/00
ResNet	Normal + ***PixelDefend***	85/85/88	82/83/84	73/46/24	71/46/25	80/80/80	78/78/78	71/46/24
VGG	Normal + ***PixelDefend***	82/82/82	82/82/84	80/62/52	80/61/48	81/76/76	81/79/79	80/61/48
ResNet	Adversarial FGSM + ***PixelDefend***	88/88/86	86/86/**87**	81/68/67	**81**/69/**56**	**85/85/85**	**84/84/84**	**81**/69/**56**
	Adversarial FGSM + ***Adaptive PixelDefend***	90/90/90	86/87/**87**	81/70/67	**81/70/56**	82/81/82	81/80/81	**81/70/56**

Song, Y.et.al.Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. arXiv preprint arXiv:1710.10766, 2017.

Gradient Obfuscation

□ Obfuscated gradients

a phenomenon exhibited by certain **defenses** that makes standard **gradient-based** methods fail to generate adversarial examples.

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%*
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.005 (ℓ_2)	0%*
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%*
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%*
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	55%**
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%
Na et al. (2018)	CIFAR	0.015 (ℓ_∞)	15%

Defense techniques cause **obfuscated gradients** and are vulnerable to their attacks.

They believe **adversarial training** approach does not cause **obfuscated gradients**.

Athalye A, Carlini N, Wagner D. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. ICML. 2018.

Adversarial Training

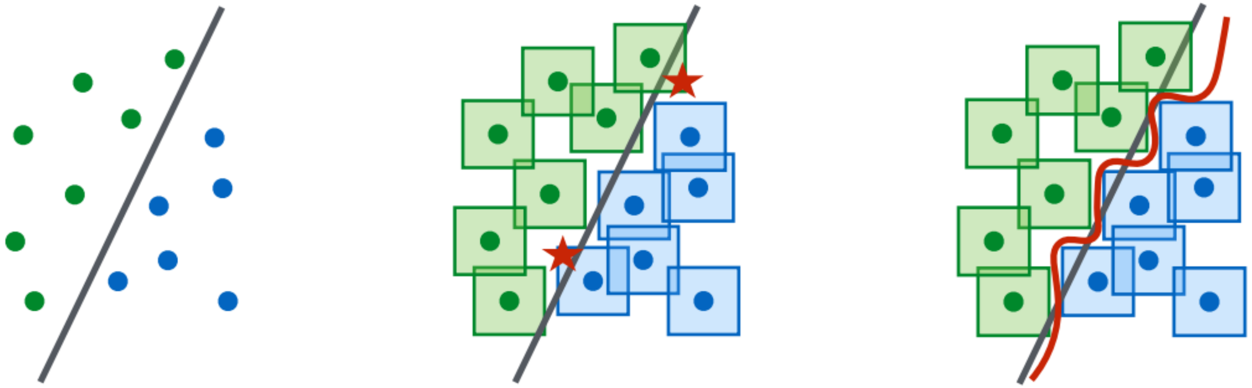
An Optimization View on Adversarial Robustness

Adversarial Training: $$\min_{\theta}\left(\frac{1}{D}\sum_{(x,y)\in D}\max_{\delta\in\Delta(x)} L(f(x+\delta),y)\right)$$

saddle point problem

The composition of an inner maximization problem and an outer minimization problem

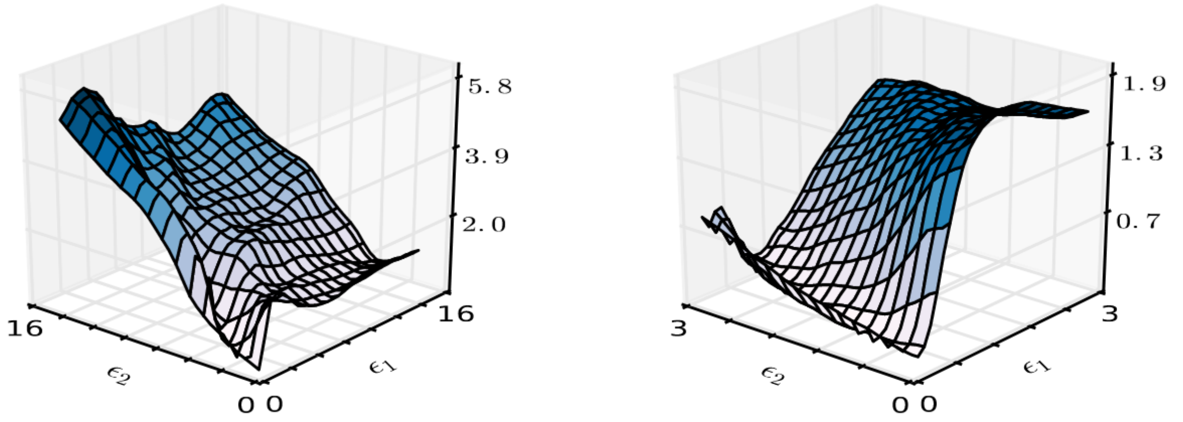
Universally Robust Networks



capacity is crucial for robustness, as well as for the ability to successfully train against strong adversaries

Adversarial Training

Adversarial Training: $$\min_{\theta}\left(\frac{1}{D}\sum_{(x,y)\in D}\max_{\delta\in\Delta(x)} L(f(x+\delta),y)\right)$$



(a) Loss of model $v3_{adv}$. (b) Zoom in for small ϵ_1, ϵ_2.

Gradient masking in single-step adversarial training

- ☐ **Ensemble Adversarial Training**: augments training data with perturbations transferred from other models Domain Adaptation with multiple sources

Theorem 1 (informal). *Let $h^* \in \mathcal{H}$ be a model learned with Ensemble Adversarial Training and static black-box adversaries $\mathcal{A}_1, \ldots, \mathcal{A}_k$. Then, if h^* is robust against the black-box adversaries $\mathcal{A}_1, \ldots \mathcal{A}_k$ used at training time, then h^* has bounded error on attacks from a future black-box adversary $\mathcal{A}^*$, if $\mathcal{A}^*$ is not "much stronger", on average, than the static adversaries $\mathcal{A}_1, \ldots, \mathcal{A}_k$.*

F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in International Conference on Learning Representations, 2018.

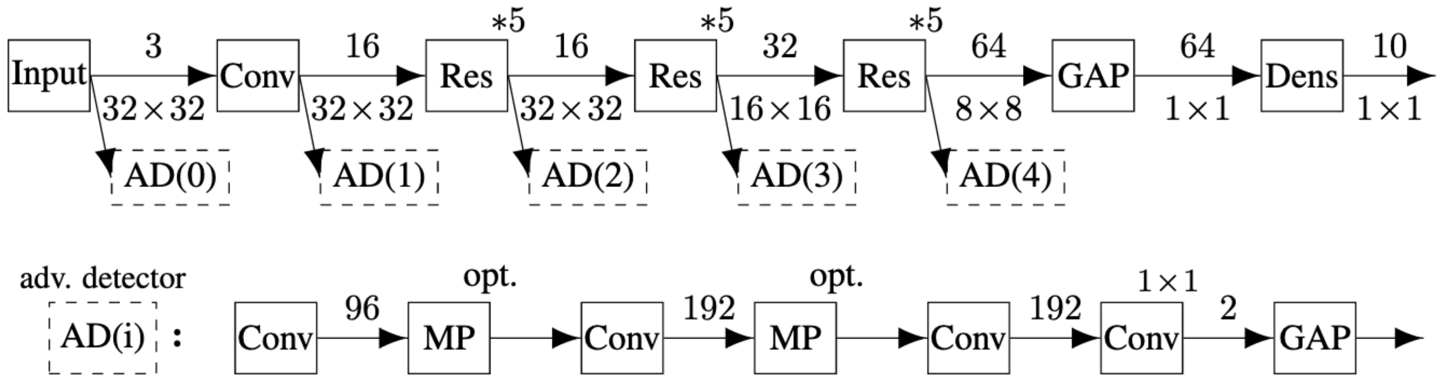
Adversarial Detection

- **Adversary detection network：augment by subnetworks**
  - branch off the main network at some layer
  - output the probability of the input being adversarial

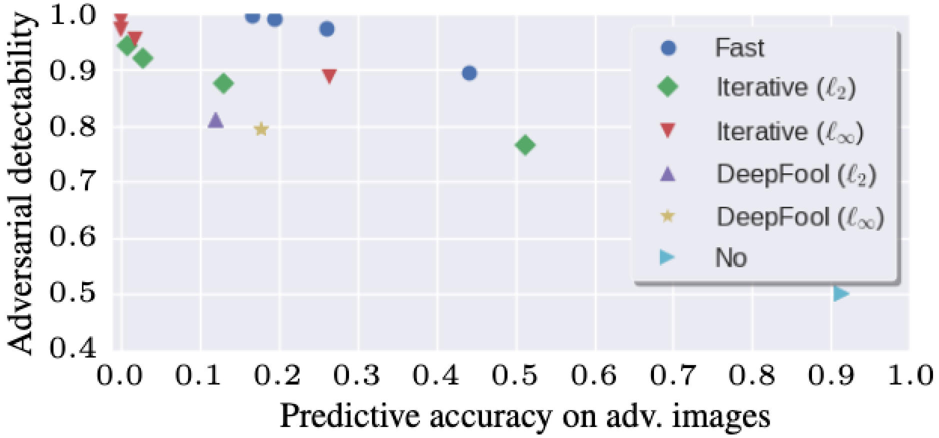
binary detector network, inputs intermediate feature representations, and discriminates between samples from the original data set and adversarial examples

the worst case: a dynamic adversary adapting to the detector

$$x_0^{\mathrm{adv}} = x; \quad x_{n+1}^{\mathrm{adv}} = \mathrm{Clip}_x^{\varepsilon}\left\{x_n^{\mathrm{adv}} + \alpha\left[(1-\sigma)\,\mathrm{sgn}(\nabla_x \mathrm{J}_{\mathrm{cls}}(x_n^{\mathrm{adv}}, y_{\mathrm{true}}(x))) + \sigma\,\mathrm{sgn}(\nabla_x \mathrm{J}_{\mathrm{det}}(x_n^{\mathrm{adv}}, 1))\right]\right\}$$

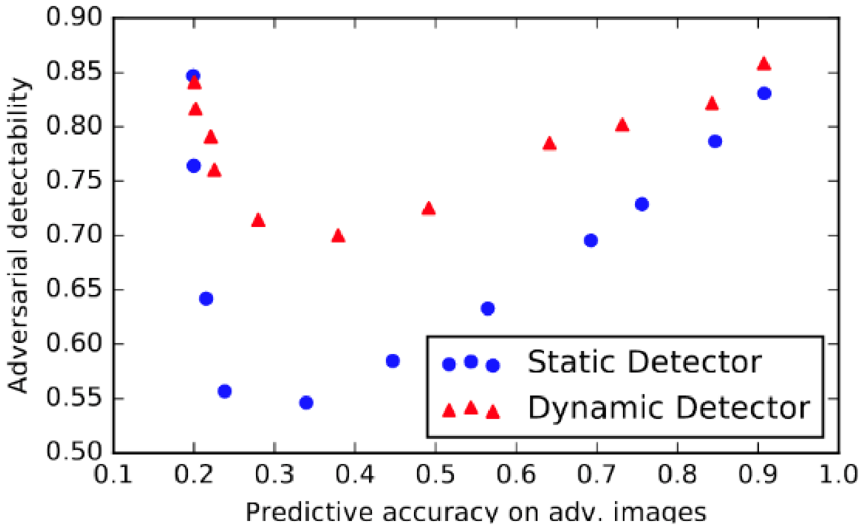


Adversarial Detection



detectability of different adversaries

the detectability is above 80%
for all adversaries



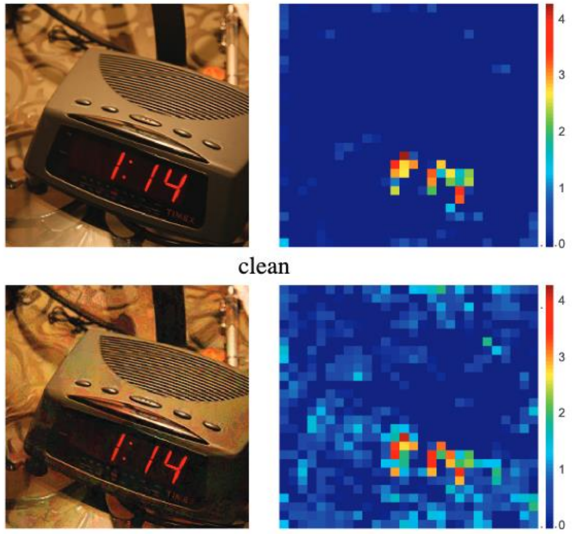
Detectability versus classification accuracy of a dynamic adversary

A dynamic detector is considerably more robust (more than 70%)

Denoising and Restructure

☐ Key Observations of image features

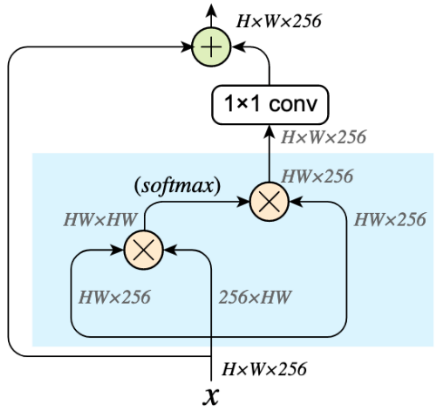
- **clean image:** appear to focus primarily on semantically informative content
- **adversarial image:** activated across semantically irrelevant regions as well



☐ Solutions

- New convolutional network architectures equipped with building blocks designed to denoise feature maps

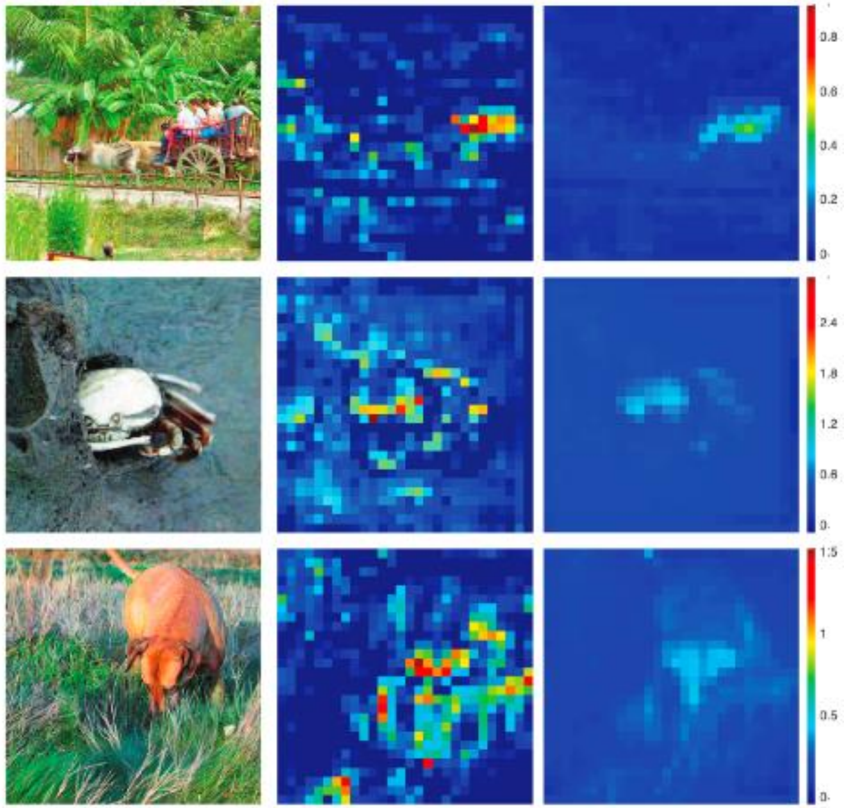
$$y_i = \frac{1}{\mathcal{C}(x)} \sum_{\forall j \in \mathcal{L}} f(x_i, x_j) \cdot x_j,$$



A block with non-local means as the denoising operation

Denoising and Restructure

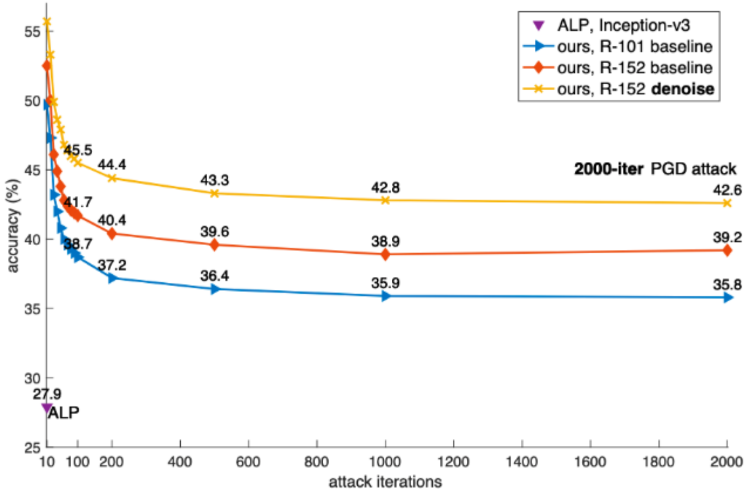
Adversarial images and their feature maps before(left) and after(right) the denoising operation



Defense against black-box attacks on ImageNet

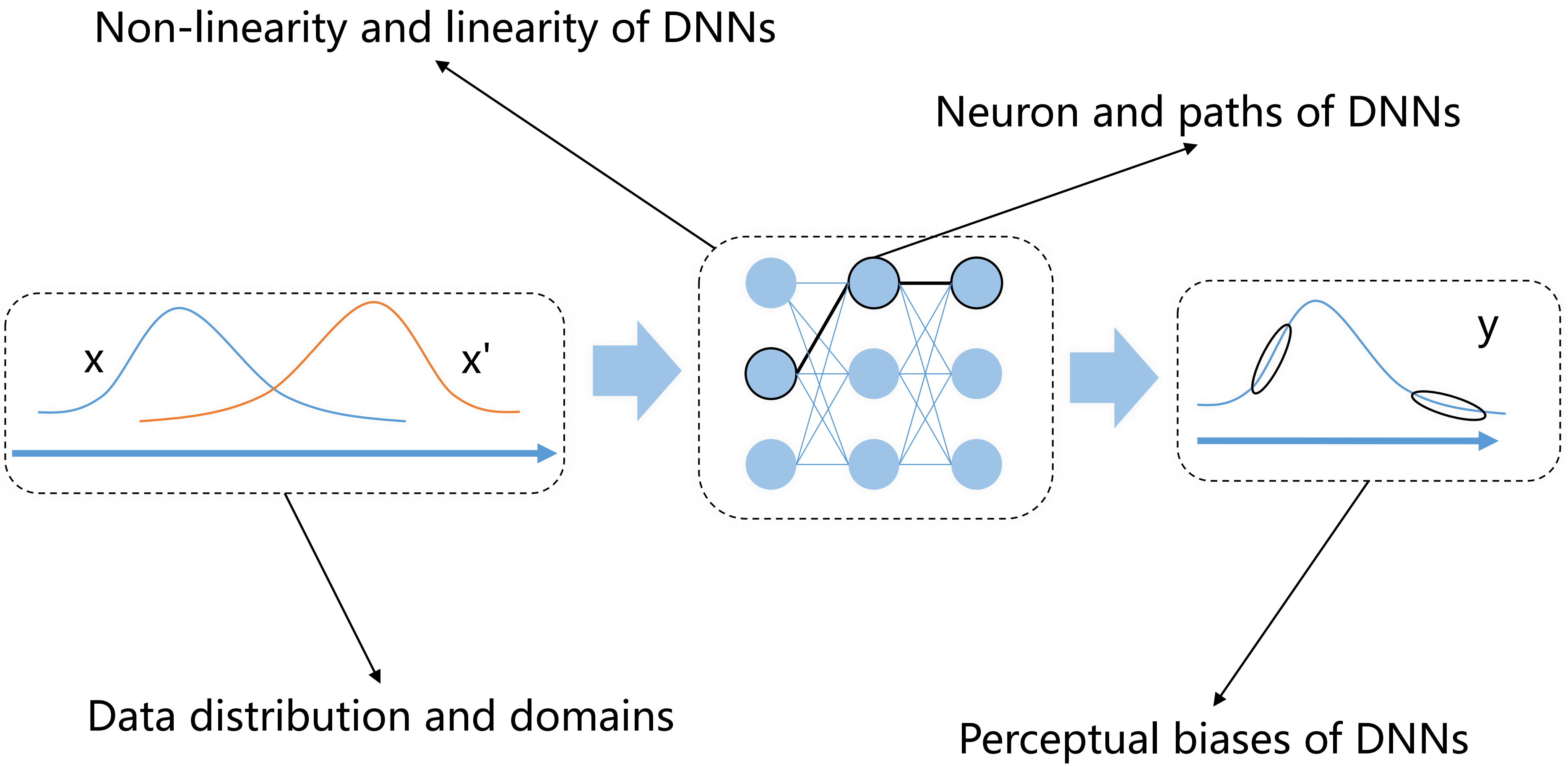
model	accuracy (%)
CAAD 2017 winner	0.04
CAAD 2017 winner, under 3 attackers	13.4
ours, R-152 baseline	43.1
+4 denoise: null (1×1 only)	44.1
+4 denoise: non-local, dot product	46.2
+4 denoise: non-local, Gaussian	**46.4**
+all denoise: non-local, Gaussian	**49.5**

Defense against white-box attacks on ImageNet



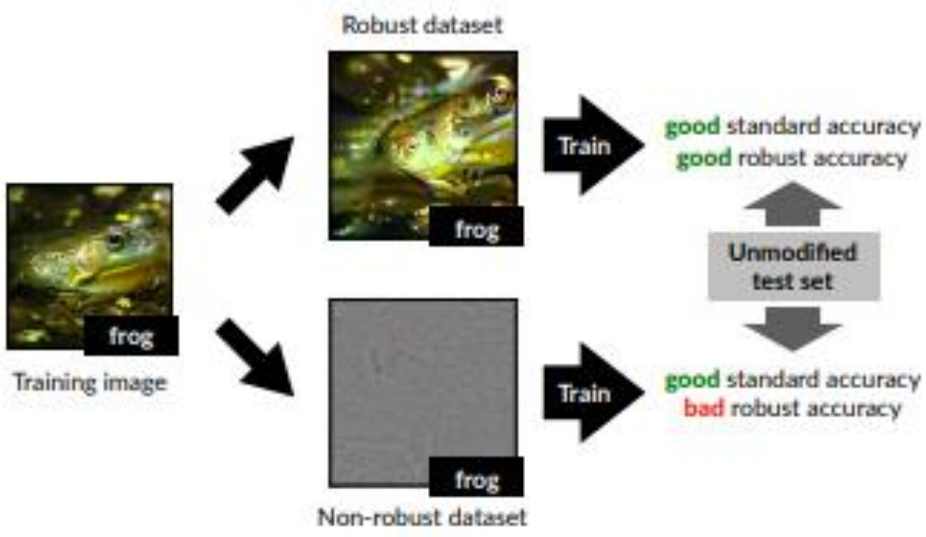
Outline

Understand the Model Robustness



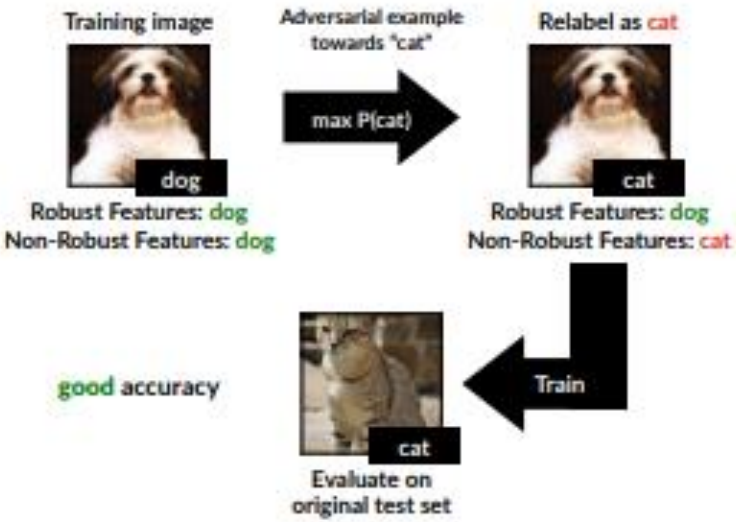
☐ Adversarial Examples are not Bugs, they are Features

Adversarial examples can be directly attributed to the presence of non-robust features



(a)

Disentangle features into combinations of robust/non-robust features



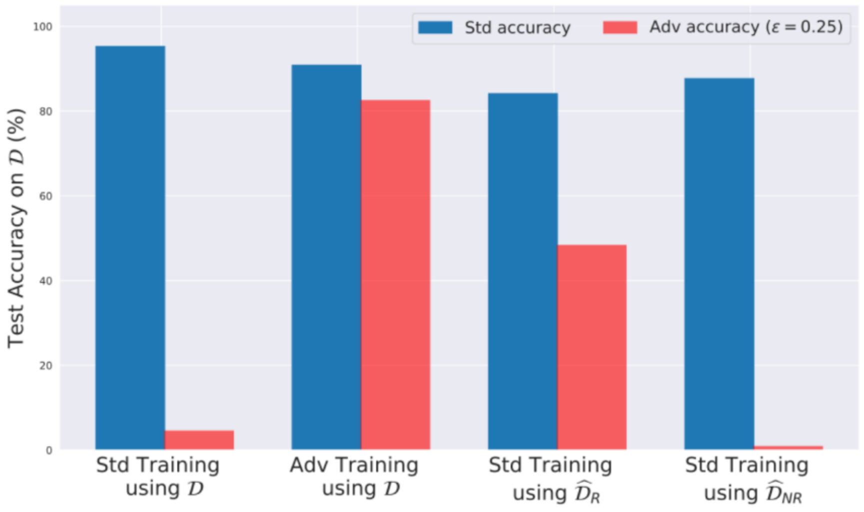
(b)

Construct a dataset which appears mislabeled to humans

☐ Non-robust features

features that are **highly predictive**, yet brittle and **incomprehensible** to humans

Random samples from the variants of the CIFAR-10

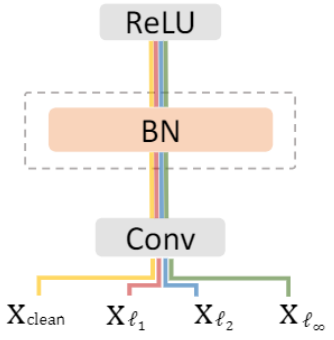


Standard and robust accuracy on the CIFAR-10 test set

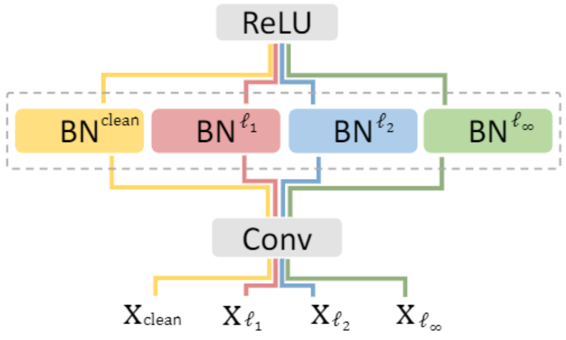
Data distribution and domains

☐ Multi-domain hypothesis

Different types of adversarial perturbations are drawn from **different domains**.

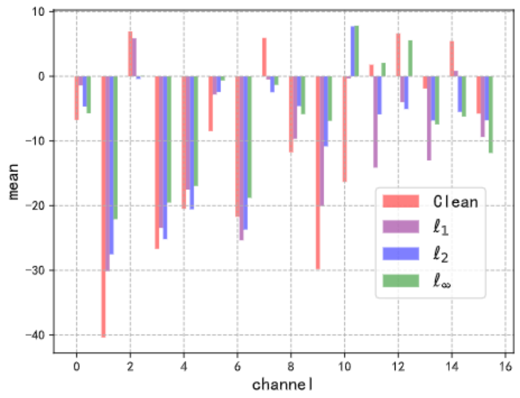


(a) standard BN

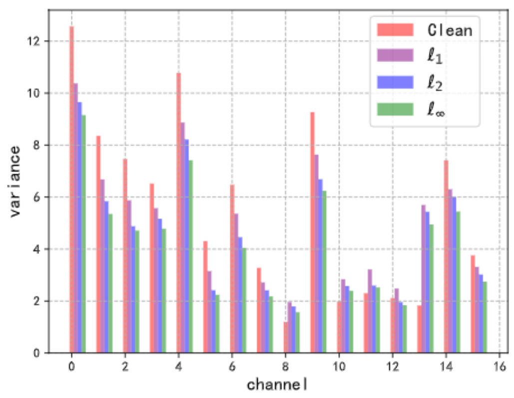


(b) multiple BN branches

(a) the standard BN structure;
(b) the structure with 4 BN branches



(c) running mean



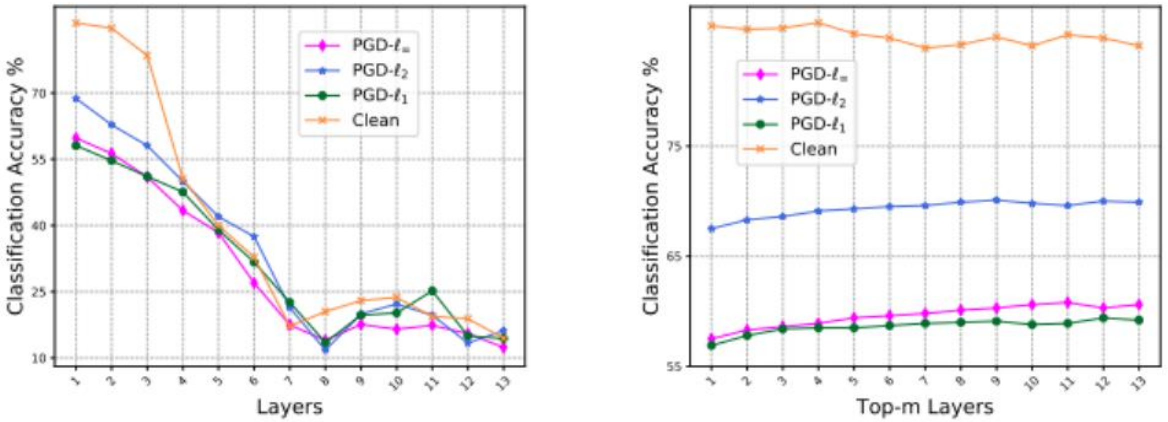
(d) running variance

(c) and (d): running means and variances of multiple BN branches on 16 randomly sampled channels

Liu, Tang, Liu et al., "TOWARDS DEFENDING MULTIPLE ADVERSARIAL PERTURBATIONS VIA GATED BATCH NORMALIZATION", Work in progress.

Gated Batch Normalization (GBN)

A building block for deep neural networks that improves robustness against **multiple perturbation types**.



(a) single layer (b) top-m layers

(a) the results of adding GBN to different single layers.
(b) the results of adding GBN to top-m layers

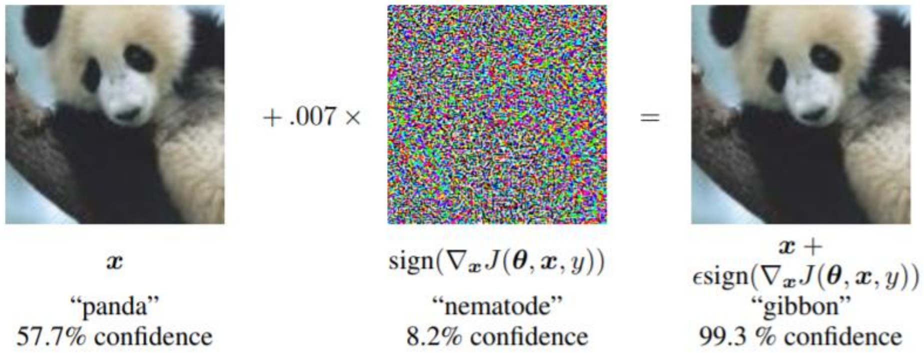
	Vanilla	AVG	MAX	MSD	MN	MBN	**GBN(ours)**
ℓ_1 attacks	0.0%	44.9%	33.3%	43.7%	39.8%	44.9%	**57.7%**
ℓ_2 attacks	0.0%	59.1%	56.0%	58.9%	30.0%	20.8%	**68.9%**
ℓ_∞ attacks	0.0%	29.2%	25.1%	38.0%	13.2%	40.1%	**49.9%**
All attacks	0.0%	28.2%	24.9%	37.9%	13.0%	20.7%	**48.7%**
Clean accuracy	**89.7%**	80.6%	77.0%	79.1%	82.3%	79.4%	80.7%

Model robustness on CIFAR-10 datasets

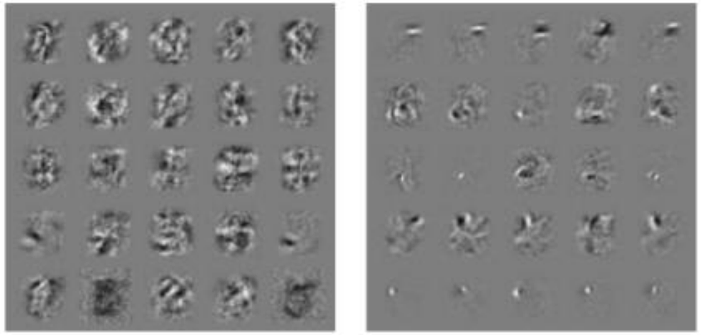
Non-linearity and linearity of DNNs

- Early attempts at explaining this phenomenon focused on **nonlinearity** and **overfitting**
- the **linearity** hypothesis. $\boldsymbol{w}^\top \tilde{\boldsymbol{x}} = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{w}^\top \boldsymbol{\eta}.$

The adversarial perturbation causes the activation to grow



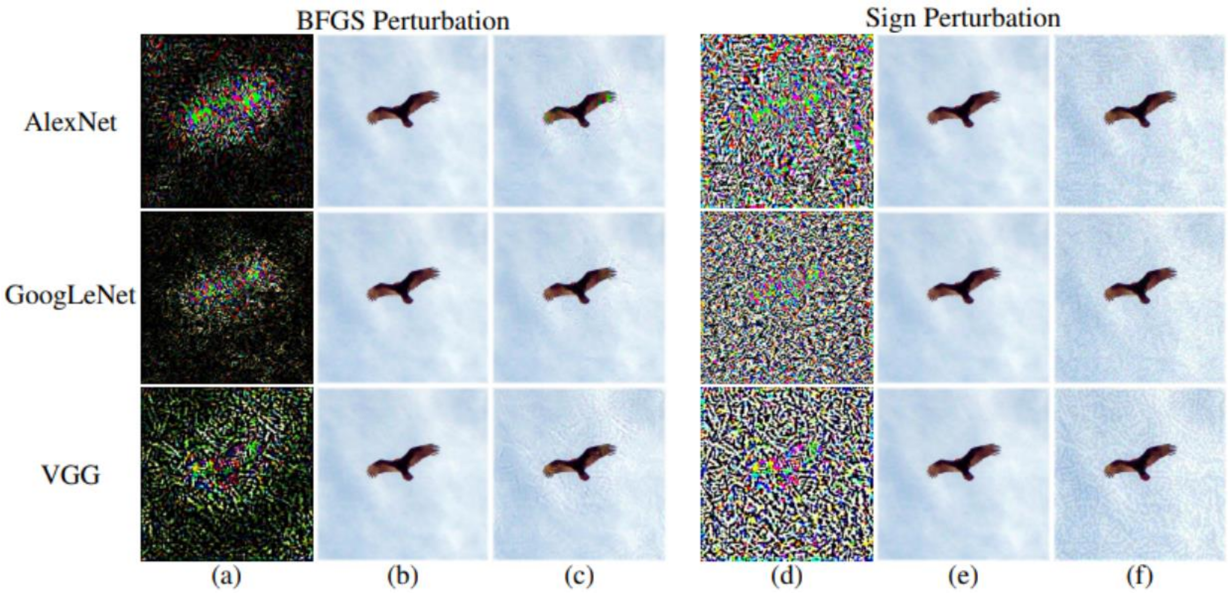
fast gradient sign method



simple linear model can have adversarial examples if its input has sufficient dimensionality.

Non-linearity and linearity of DNNs

- Challenge the **linearity hypothesis** by analyzing adversarial examples using several CNN architectures for ImageNet.
- CNNs act **locally linearly** to changes in the image regions with objects **recognized** by the CNN, and in other regions the CNN may act **non-linearly**.

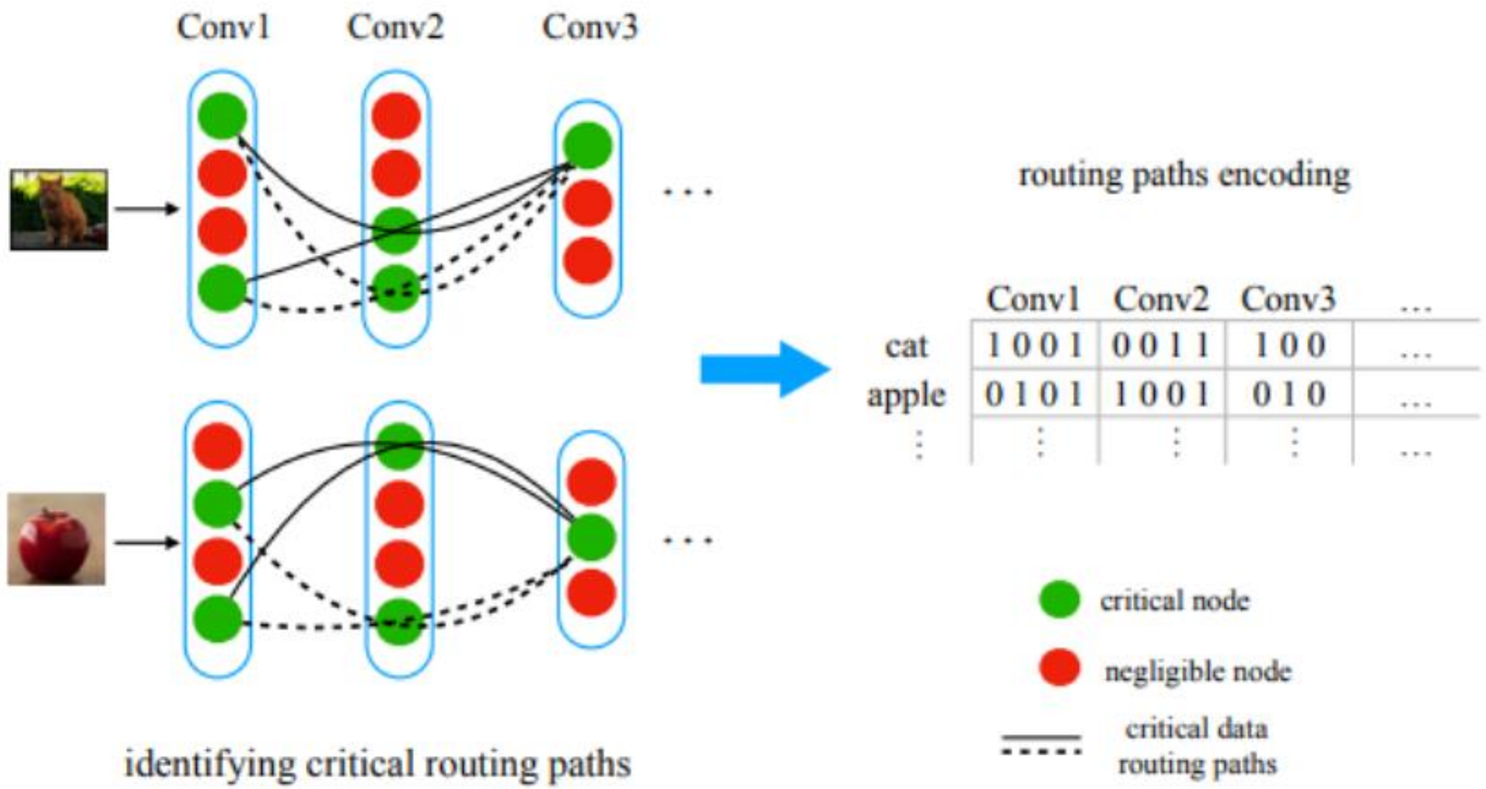


Example of different CNNs' minimum perturbations

Neuron and paths of DNNs

◻ Distillation Guided Routing

identify the **critical data routing paths** for each input sample.



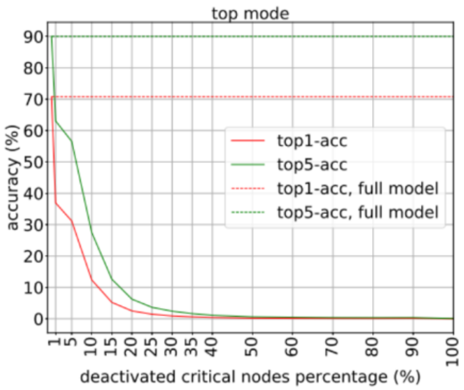
Overview of Distillation Guided Routing method

Wang, Su, Zhang, and Hu, "Interpret Neural Networks by Identifying Critical Data Routing Paths". CVPR, 2018

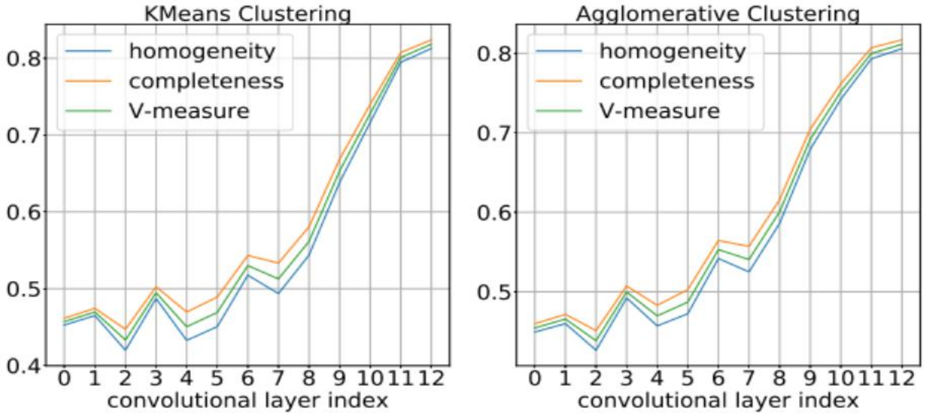
Neuron and paths of DNNs

- Only **small fractions** of critical nodes being **deactivated** will lead severe **performance degradation**.

- The **intra-layer** routing nodes of **higher level layers** have stronger correspondence to **category semantic concepts**.



The accuracy degradation



Different clustering consistency evaluation

Neuron and paths of DNNs

☐ Activation Promotion and Suppression

better understand the roles of **adversarial perturbations** and provide **visual explanations** from pixel, image and network perspectives.

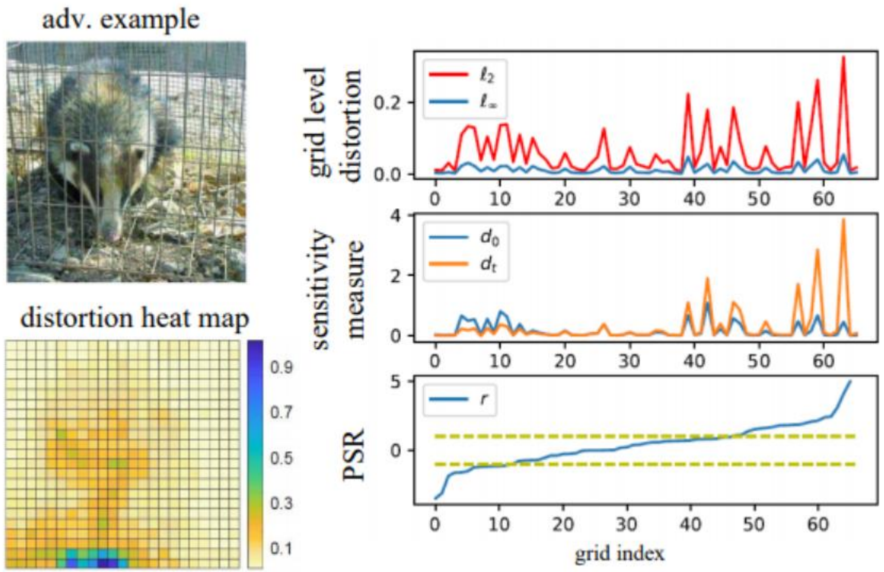
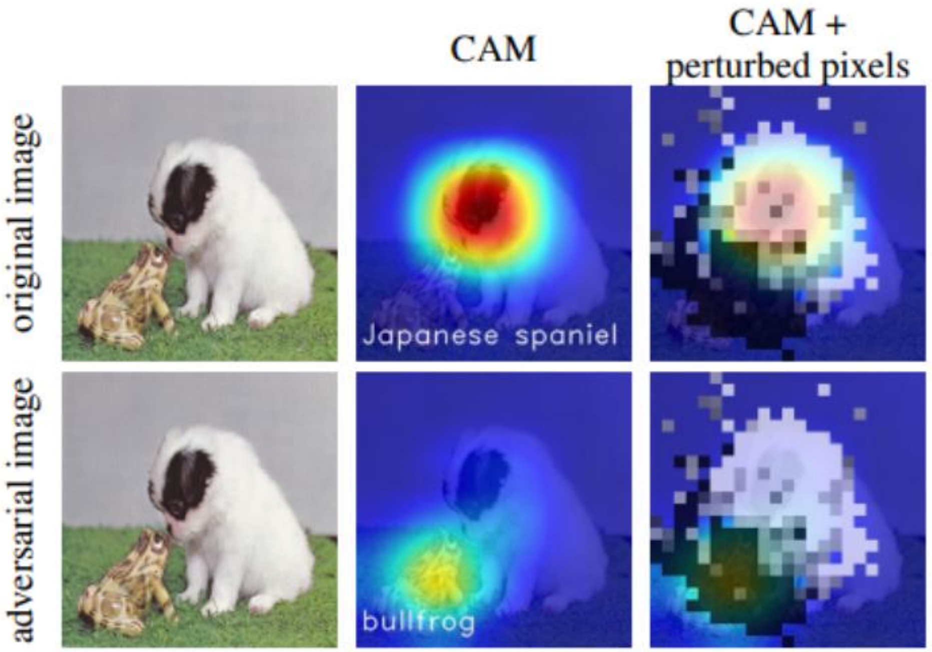


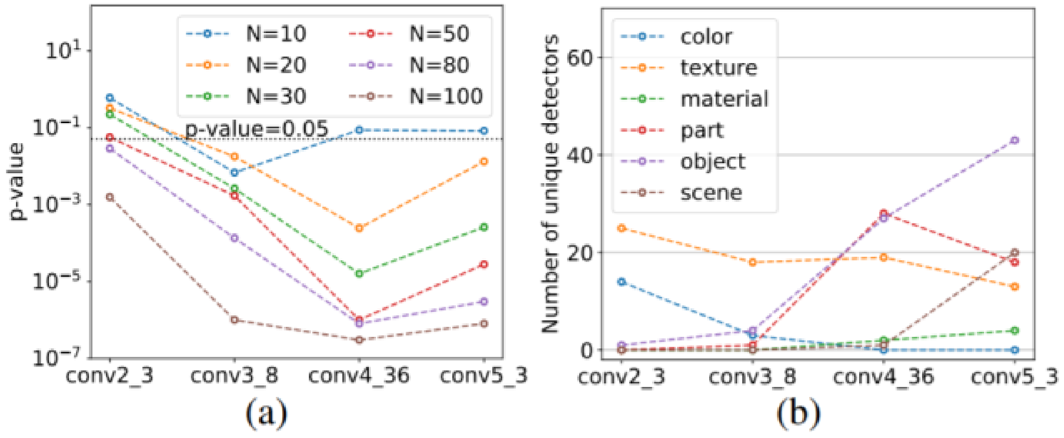
Illustration on sensitivity measure



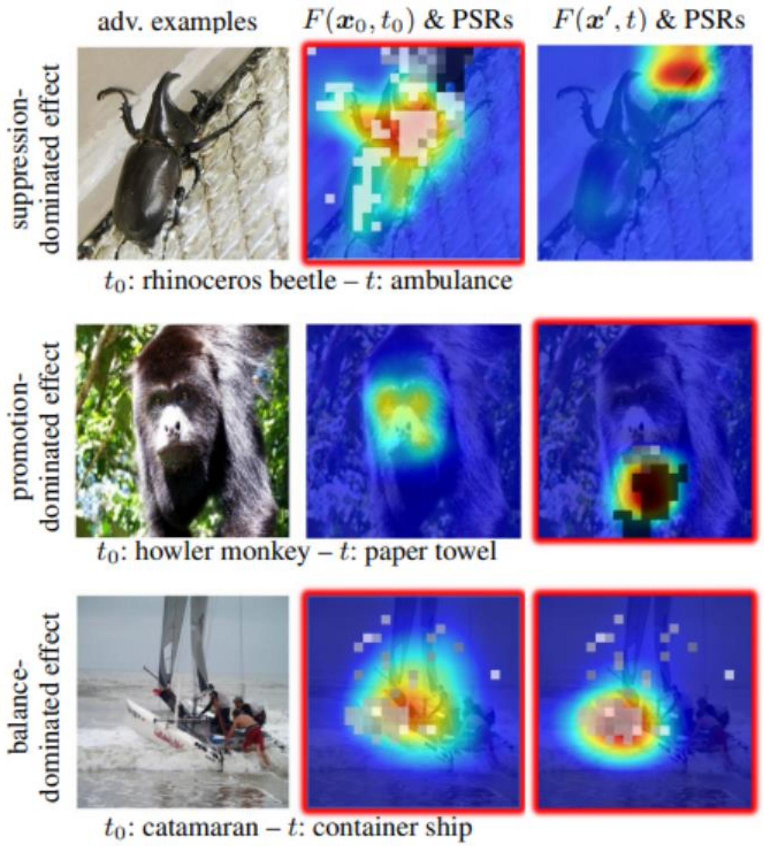
Explanation of adversarial perturbations

Neuron and paths of DNNs

There exists a tight connection between the **sensitivity of hidden units** of CNNs and their interpretability on **semantic concepts**.



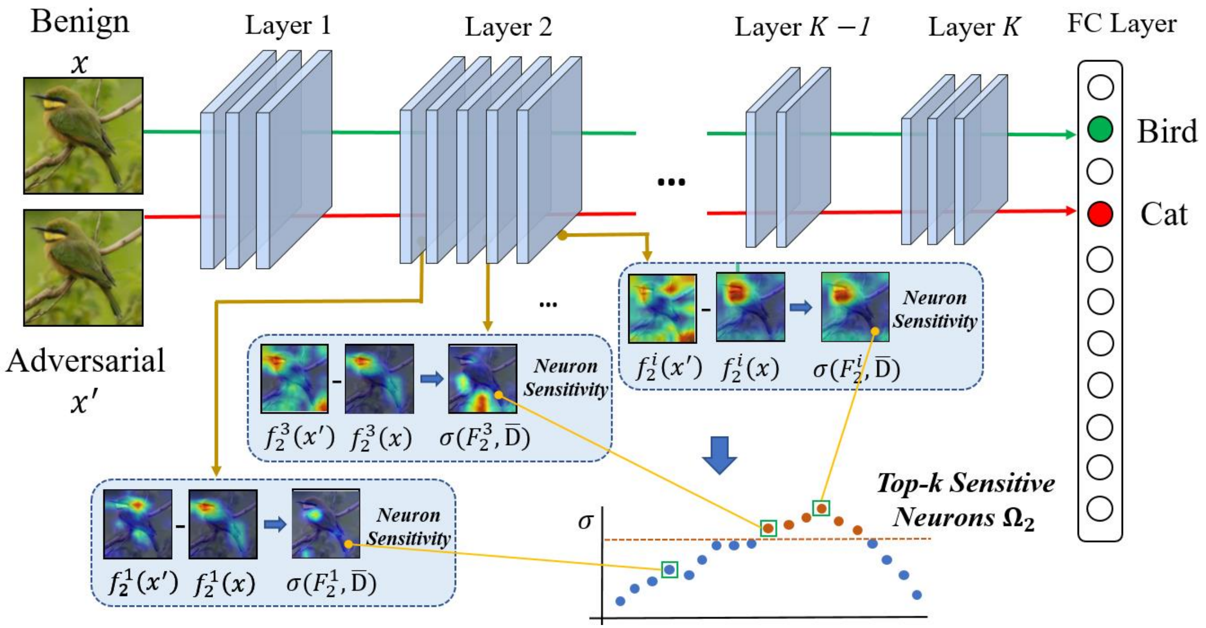
Sensitivity and interpretability



Interpreting adversarial perturbations

☐ Neuron sensitivity

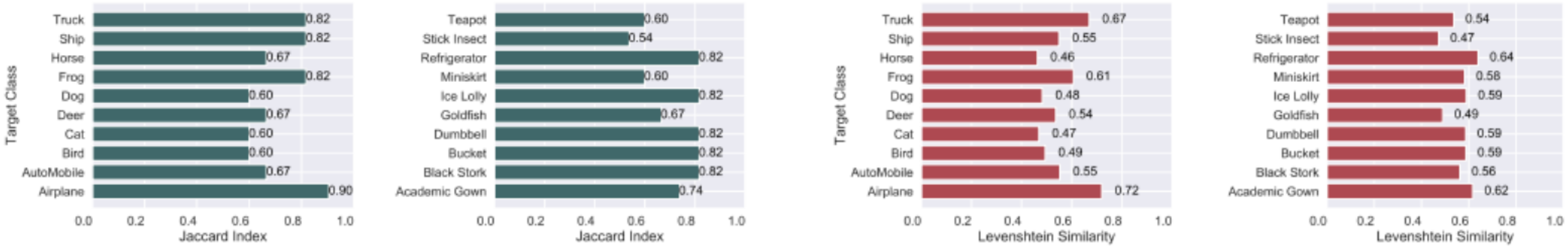
- Explain adversarial robustness from a new perspective of **neuron sensitivity**
- Measured by **neuron behavior variation intensity** against benign and adversarial examples.



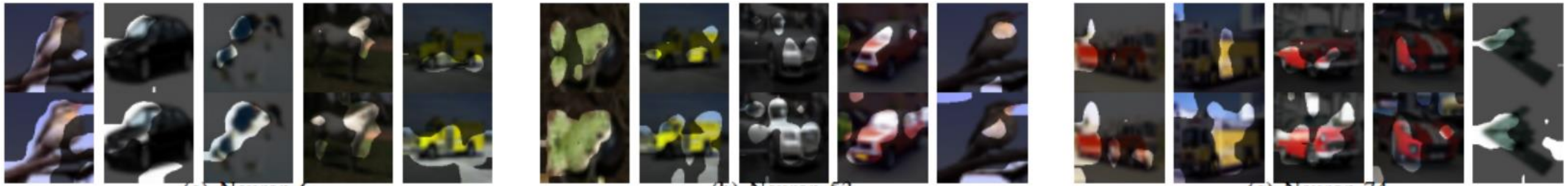
The framework of computing Neuron Sensitivity and selecting Sensitive Neuron

☐ Insightful clues for model robustness and weakness

- Sensitive Neurons Contribute Most to Model Misclassification in the Adversarial Setting
- Adversarial Attacks Exploit Sensitive Neurons Differently at Different Layers
- Sensitive Neurons Convey Strong Semantic Information
- Adversarial Training Builds Robust Models by Reducing Neuron Sensitivities
- Training Adversarially Robust Models via Sensitive Neurons Stabilizing



The Spearman’ s Rank Correlation Coefficient and the Levenshtein Similarity

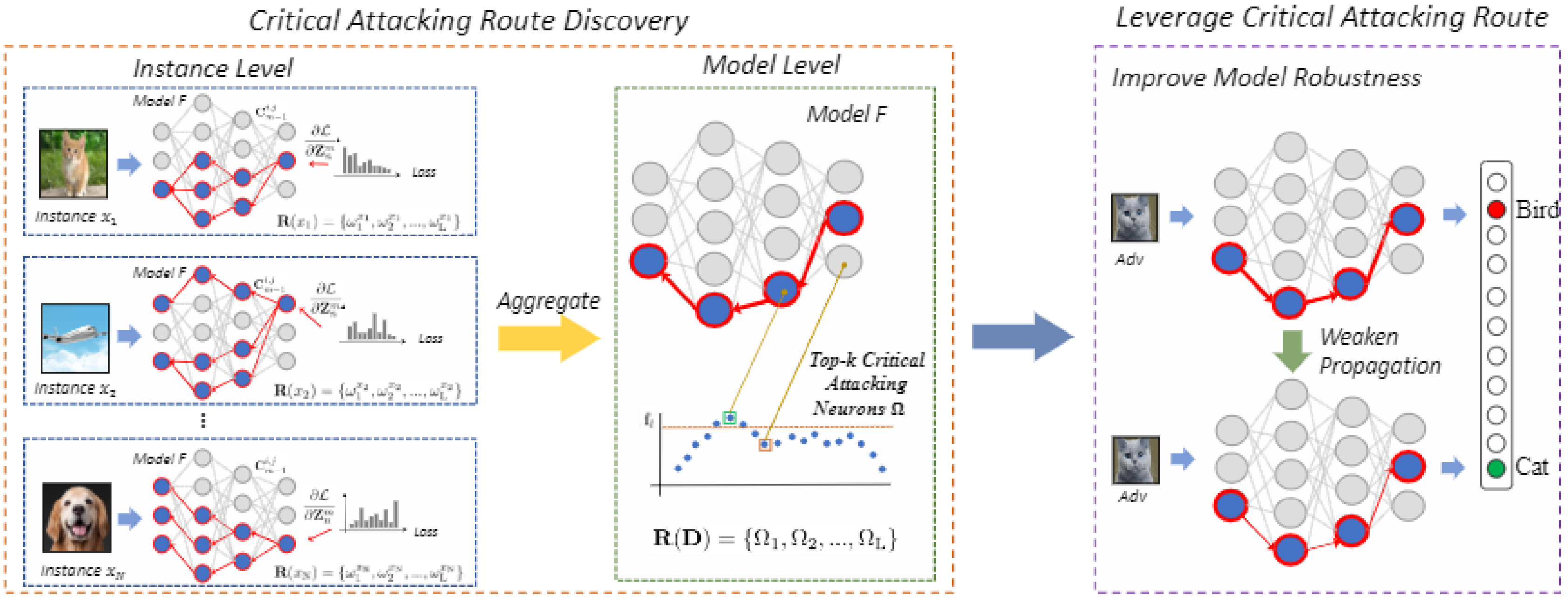


(a) Neuron 6 (b) Neuron 53 (c) Neuron 74

Image segmentation results of sensitive neurons

Neuron-wise critical attacking route

A **gradient-based** influence propagation strategy to get **critical attacking neurons**

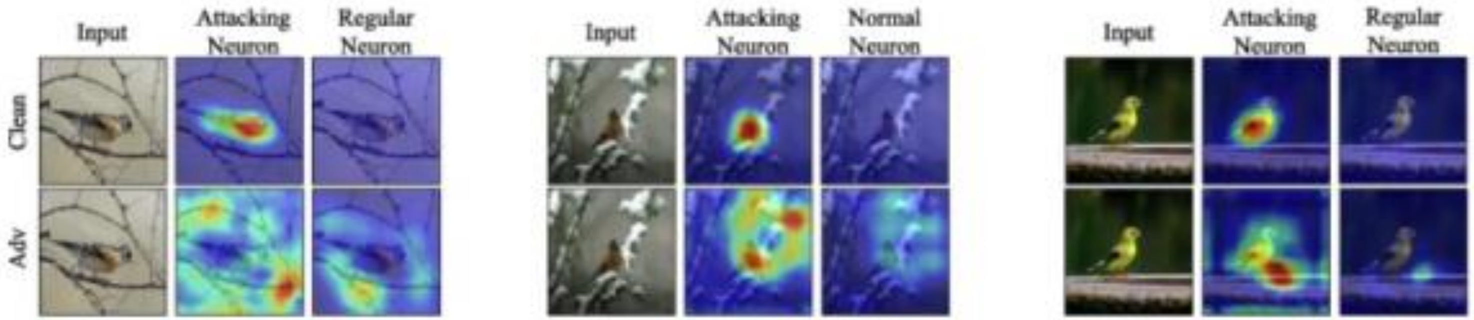


The framework of computing instance-level critical attacking routes and model-level critical attacking route

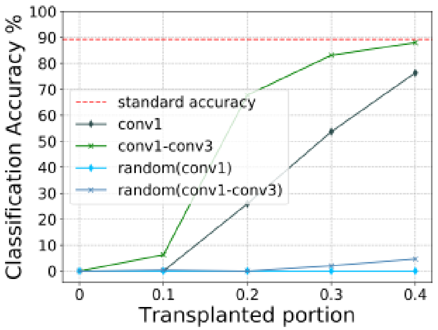
Neuron and paths of DNNs

☐ Understanding model behaviors via critical attacking routes

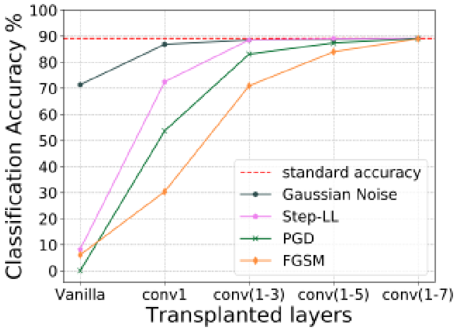
- Adversarial perturbations are propagated and amplified via attacking route
- Attacking route conveys strong semantic information



Grad-CAM of neurons on (and not on) critical attacking route of the last conv layer using pretrained VGG16 on ImageNet



(a) Transplant experiment1

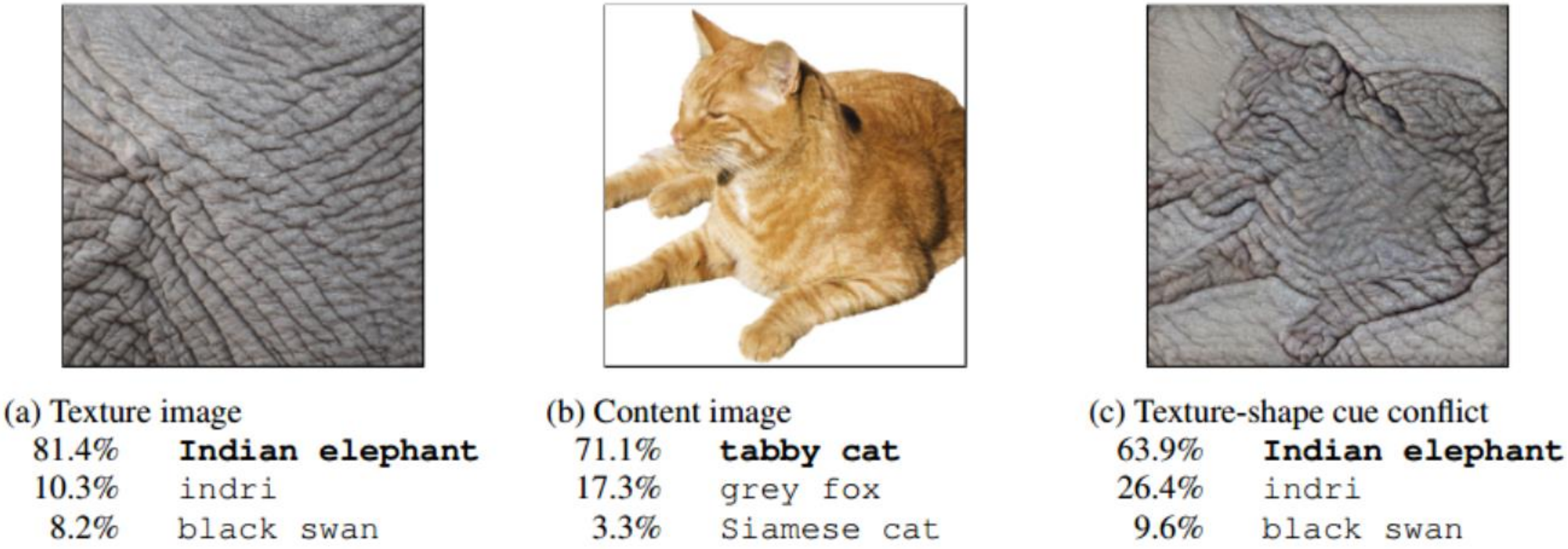


(b) Transplant experiment2

Adversarial feature transplant experiment

Perceptual biases of DNNs

- ImageNet trained CNNs are strongly **biased** towards recognizing **textures** rather than **shapes**
- in stark contrast to **human** behavioral evidence and reveals fundamentally **different classification** strategies.

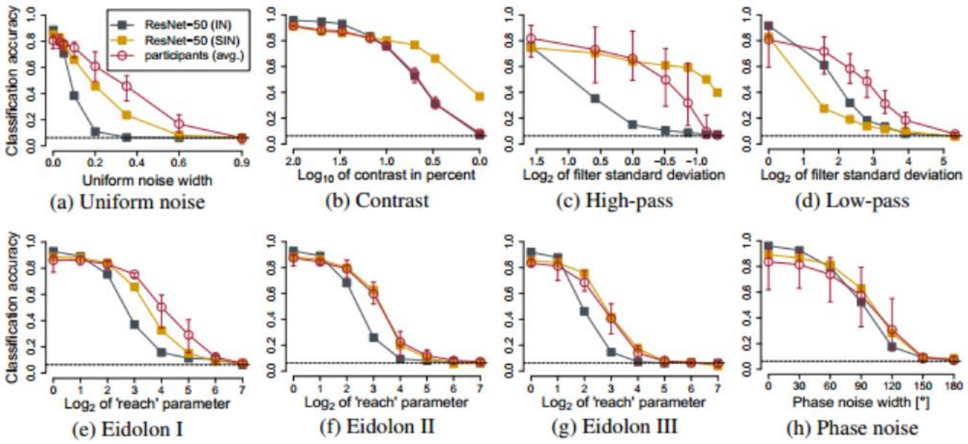


Classification of a texture image, a normal image of a cat, and an image with a texture-shape cue conflict

- **Stylized-ImageNet** (a stylized version of ImageNet) provide a much better fit for **human behavioral** performance in the well-controlled psychophysical lab setting.

architecture	IN→IN	IN→SIN	SIN→SIN	SIN→IN
ResNet-50	92.9	16.4	79.0	82.6
BagNet-33 (mod. ResNet-50)	86.4	4.2	48.9	53.0
BagNet-17 (mod. ResNet-50)	80.3	2.5	29.3	32.6
BagNet-9 (mod. ResNet-50)	70.0	1.4	10.0	10.9

Accuracy comparison

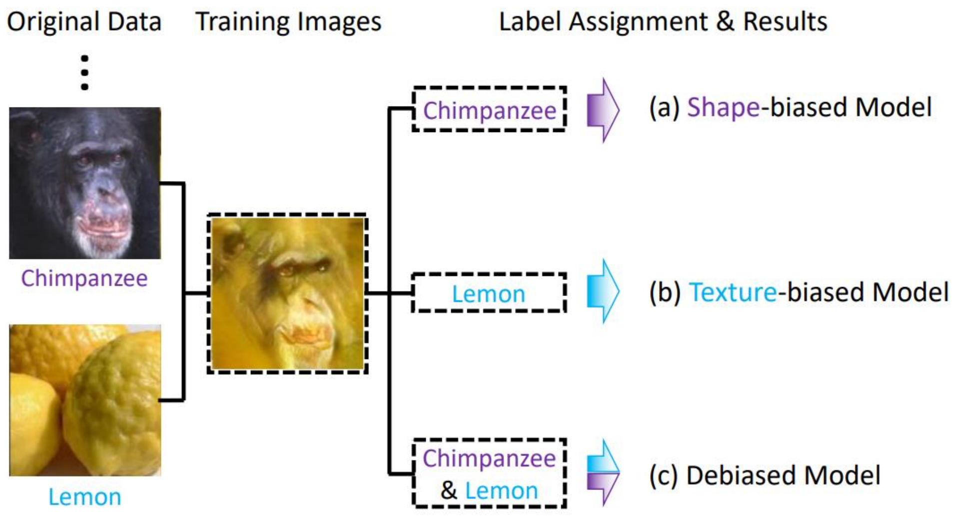


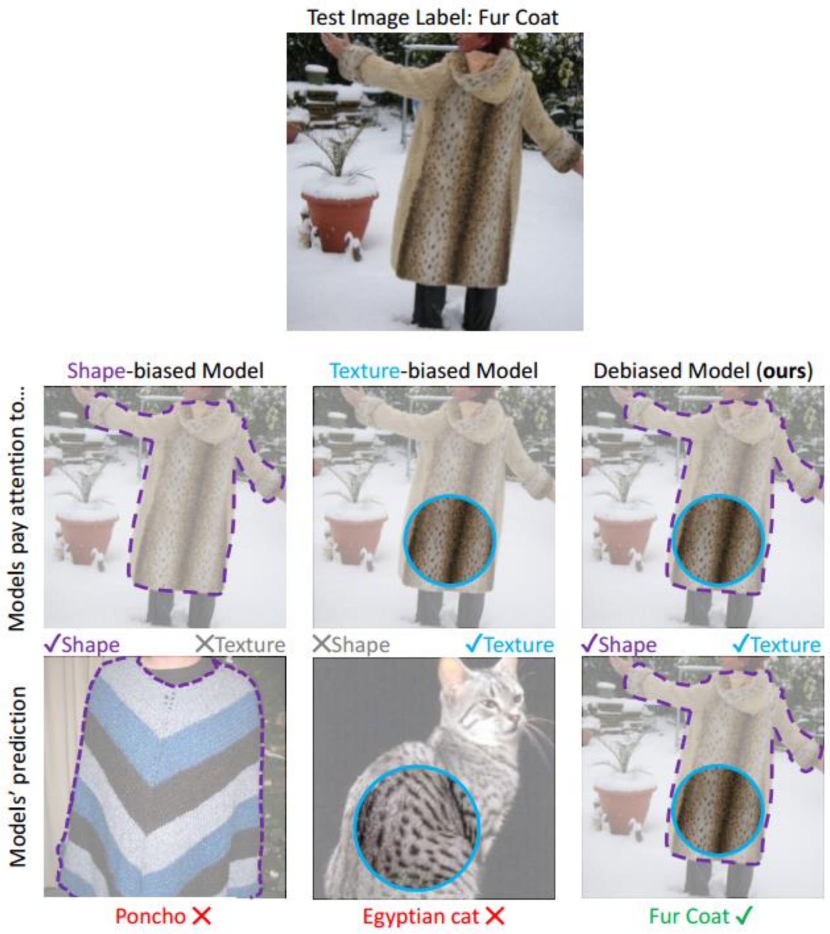
Classification accuracy on parametrically distorted images

Perceptual biases of DNNs

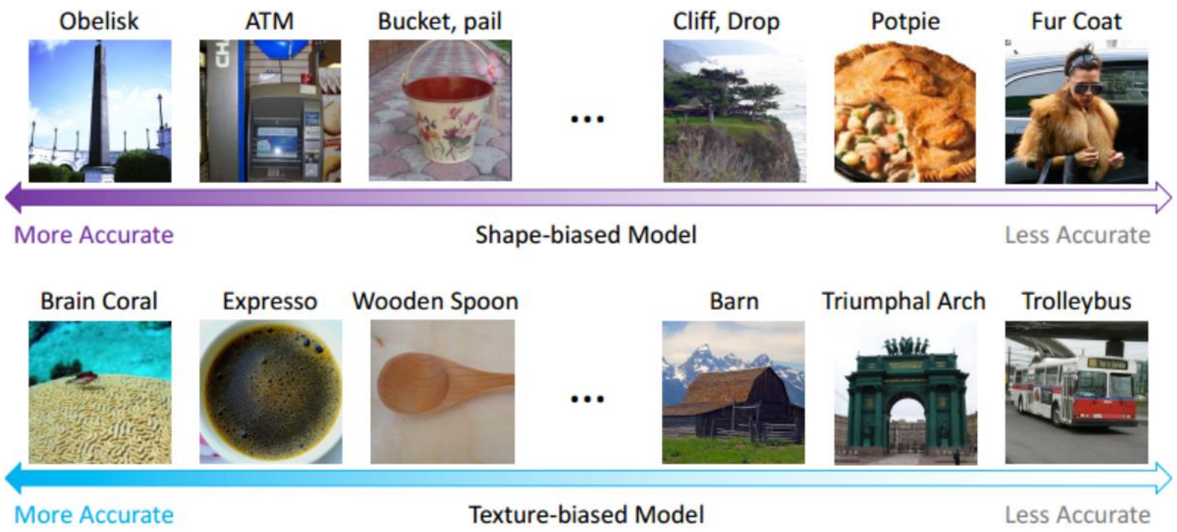
- **CNNs** are often biased towards either **texture or shape**, depending on the training dataset

assigning labels to cue conflict images controls the bias of learned models.





Perceptual biases of DNNs



The shape-biased model and the texture-biased model are good/bad at classifying different object categories

	IMAGENET-A Top-1 Acc. ↑	IMAGENET-C mCE ↓	S-IMAGENET Top-1 Acc. ↑	FGSM Top-1 Acc. ↑
ResNet-50	2.0	75.0	7.4	17.1
+ Debiased	3.5 (+1.5)	67.5 (-7.5)	17.4 (+10.0)	27.4 (+10.3)
ResNet-101	5.6	69.8	9.9	23.1
+ Debiased	9.1 (+3.5)	62.2 (-7.6)	22.0 (+12.1)	34.4 (+11.3)
ResNet-152	7.4	67.2	11.3	25.2
+ Debiased	12.6 (+5.2)	58.9 (-8.3)	22.4 (+11.1)	39.6 (+14.4)

The model robustness on ImageNet

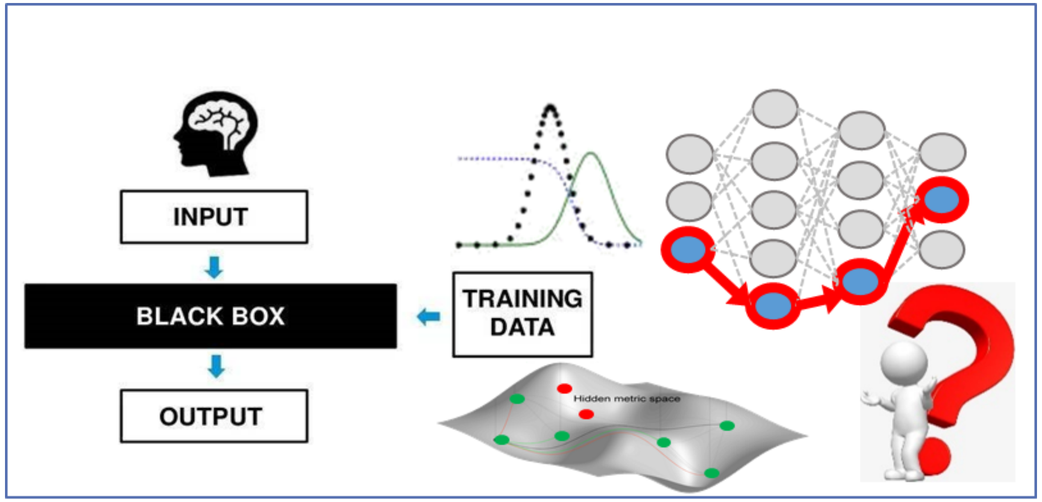
Li, Yu, Tan et al., *"SHAPE-TEXTURE DEBIASED NEURAL NETWORK TRAINING ". ICLR, 2021*

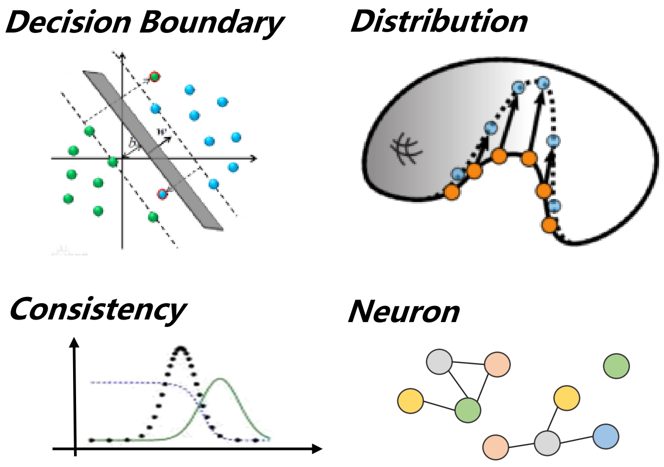
Outline

Model Robustness Understanding

Trustworthy AI needs deep understanding to DNNs(Interpretability Theory)

Development Safety of AI Applications	Ethics Specification of AI Techniques	Human-machine Boundary of AI Society





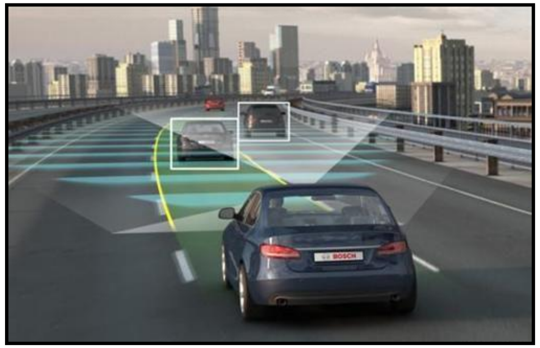
Model Robustness Understanding

Interpretability theory can help to make sure the **safety** of real-world AI applications

- Monitor the status of AI applications
- Analyze AI application bugs
- Expand AI application scenarios

- When
- Where
- What

AI Safety



Auto-driving

Unmanned Vehicle

Security

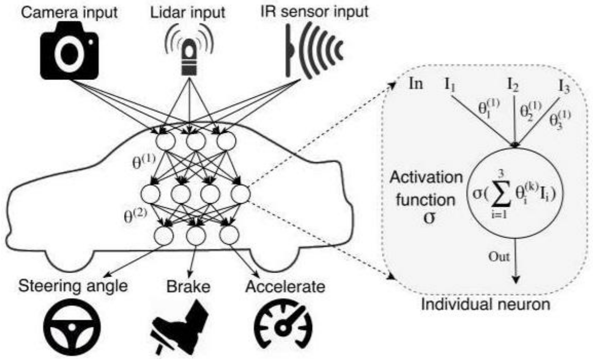
Model Robustness Evaluation

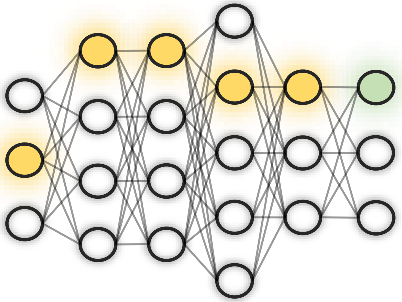
Model robustness evaluation promotes AI applications to be more **controllable**, **credible** and **reliable**

Distinguishability

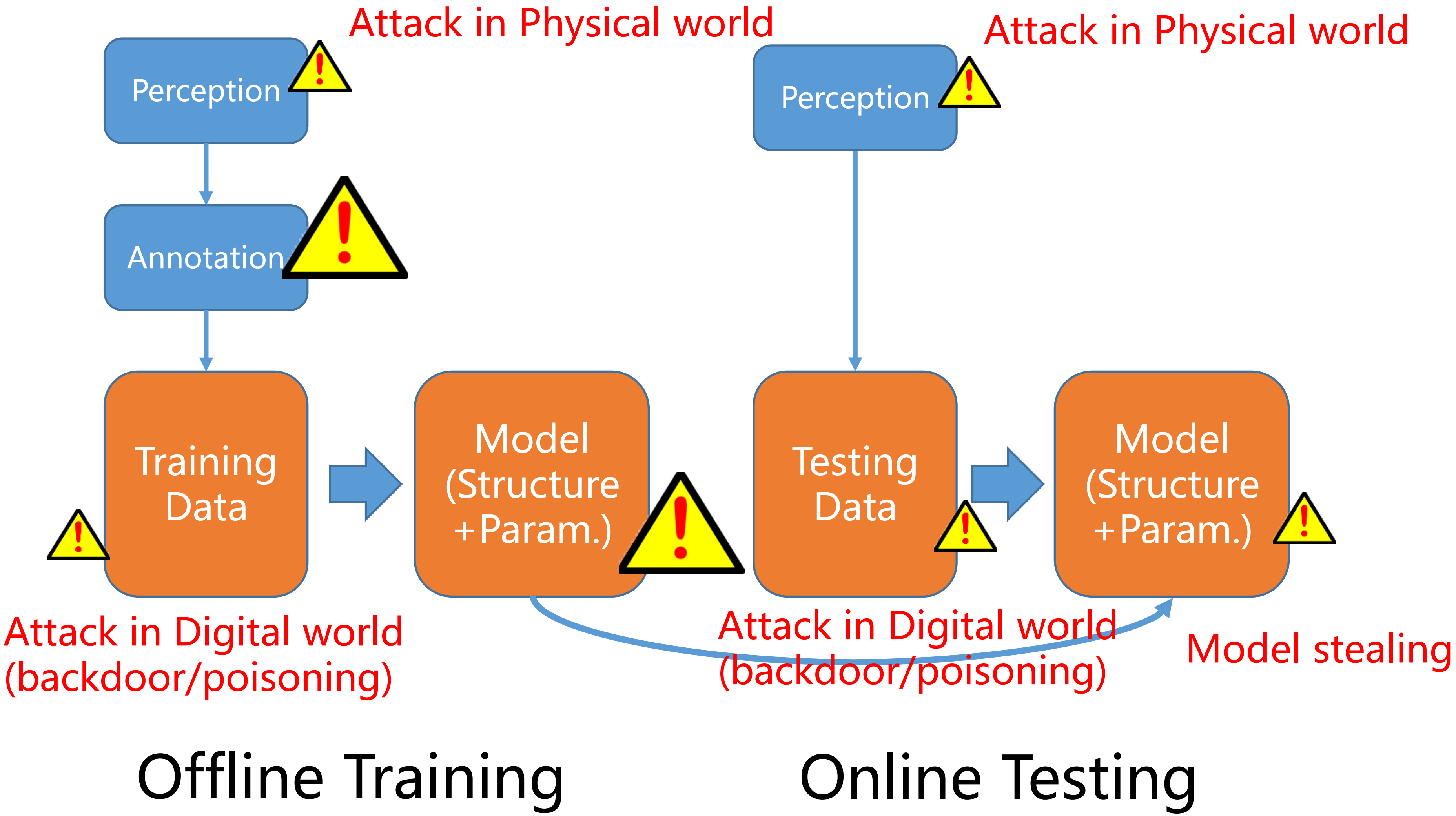
Comprehensibility

Accuracy





More Challenges in the Life-Cycle of AI models



北京航空航天大学
BEIHANG UNIVERSITY

软件开发环境国家重点实验室
State Key Laboratory of Software Development Environment

Adversarial Examples for Deep Learning: Attack, Defense and Robustness

Q&A

Xianglong Liu

July 2021

State Key Lab of Software Development Environment

Beihang University, China

xlliu@buaa.edu.cn

http://www.nlsde.buaa.edu.cn/~xlliu