Trust Region Based Adversarial Attack on Neural Networks

Zhewei Yao[1] Amir Gholami[1] Peng Xu[2] Kurt Keutzer[1]
Michael W. Mahoney[1]

[1]University of California, Berkeley

[2]Stanford University

IEEE Conference on Computer Vision and Pattern Recognition 2019

Table of Contents

1. Introduction
2. Background
3. Contributions
4. Trust Region Optimization
5. Proposed Method
6. Results
7. Conclusion

Table of Contents

1. Introduction
2. Background
3. Contributions
4. Trust Region Optimization
5. Proposed Method
6. Results
7. Conclusion

Introduction

- Neural networks are vulnerable to *adversarial examples*, inputs that are crafted to fool the network
- They are usually imperceptible, can cause a significant decrease in accuracy, and can transfer to other networks that an attacker has not seen.

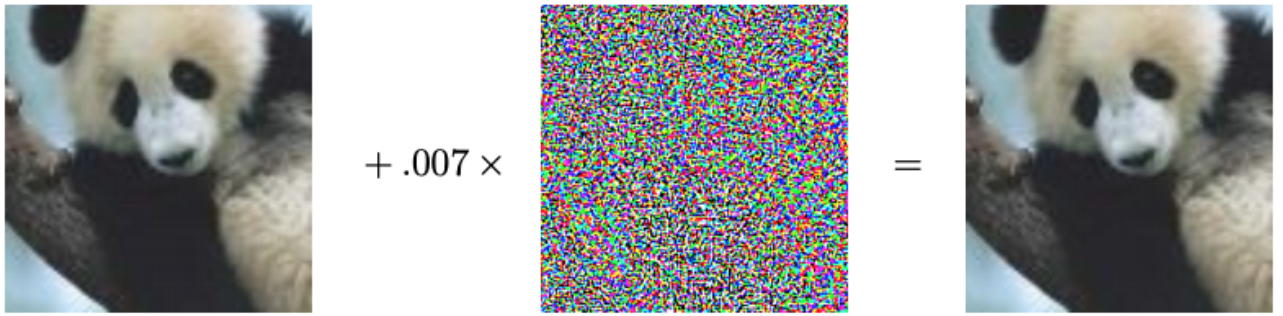


Figure: An attack on an image using the FGSM method

Explaining and harnessing adversarial examples (Goodfellow et al.)

One Pixel Attack



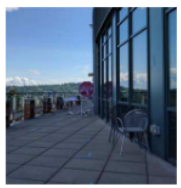
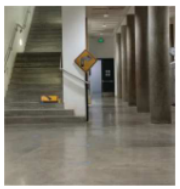
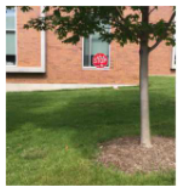
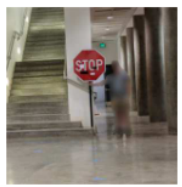
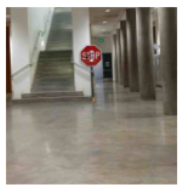
One Pixel Attack for Fooling Deep Neural Networks (Su et al.)

3D adversarial objects



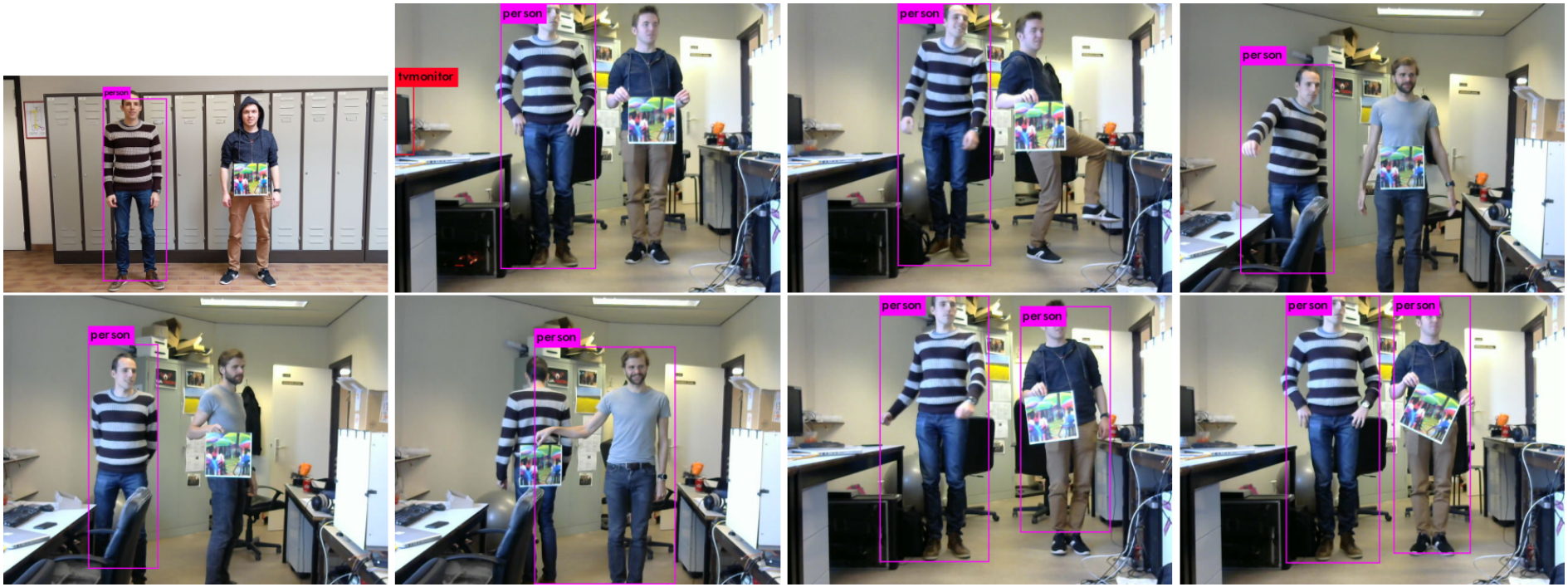
Synthesizing Robust Adversarial Examples (Athalye et al.)

Physical attacks on traffic signs

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%

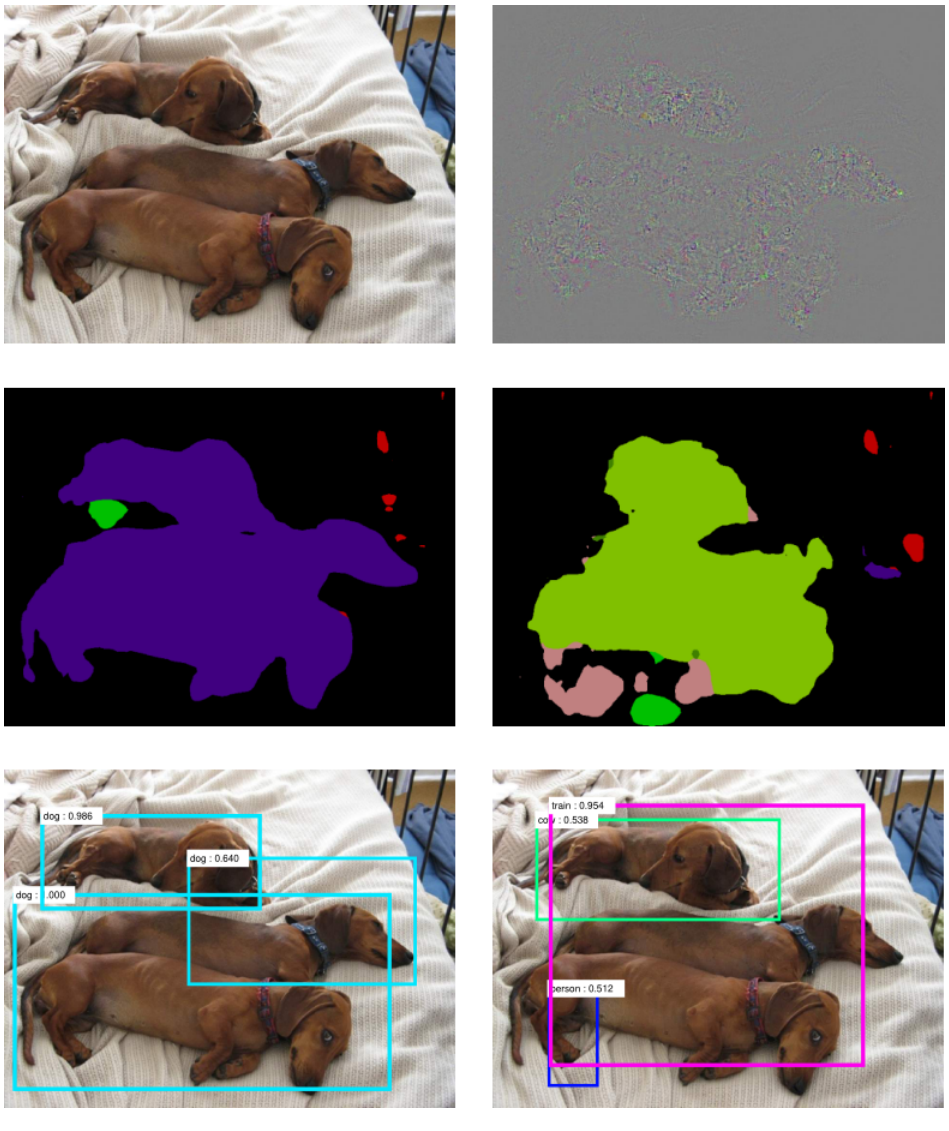
Robust Physical-World Attacks on Deep Learning Visual Classification (Eykholt et al.)

Adversarial Patches to Attack Person Detection



Adversarial Patches to Attack Person Detection (Thys et al.)

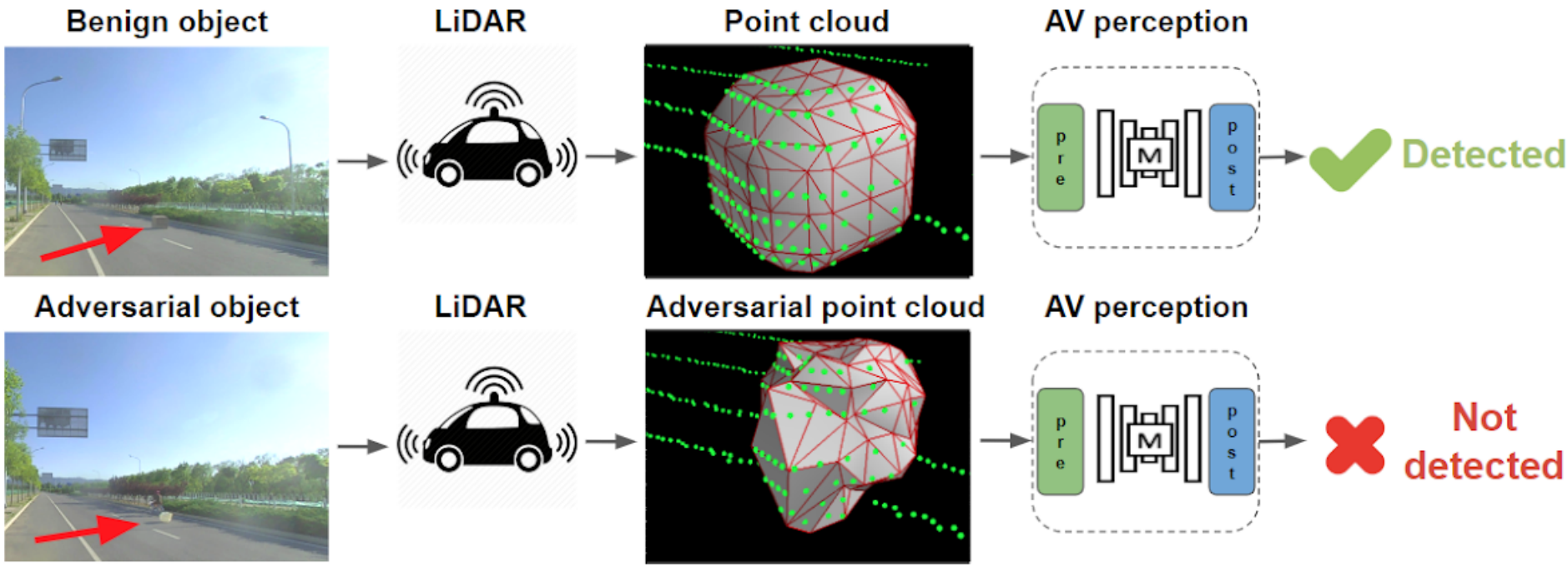
Semantic Segmentation and Object Detection





Adversarial Examples for Semantic Segmentation and Object Detection (Xie et al.)

LIDAR attack



Adversarial Objects Against LiDAR-Based Autonomous Driving Systems (Cao et al.)

Table of Contents

1. Introduction
2. Background
3. Contributions
4. Trust Region Optimization
5. Proposed Method
6. Results
7. Conclusion

Definitions

- *Untargeted attacks* are attacks that change the model classification to a wrong label. *Targeted attacks* change the model classification to a specific class.
- *One-shot / one-step* attack require one computational step to generate an adversary. Methods that require an iterative loop are called *iterative attacks*.
- *White-box* attacks need complete information of the target network (network architecture, gradients, parameters, etc.), while *black-box* attacks do not need such information.

Problem Formulation

- Szegedy et al. first formalized the problem of finding adversarial examples as the following optimization problem

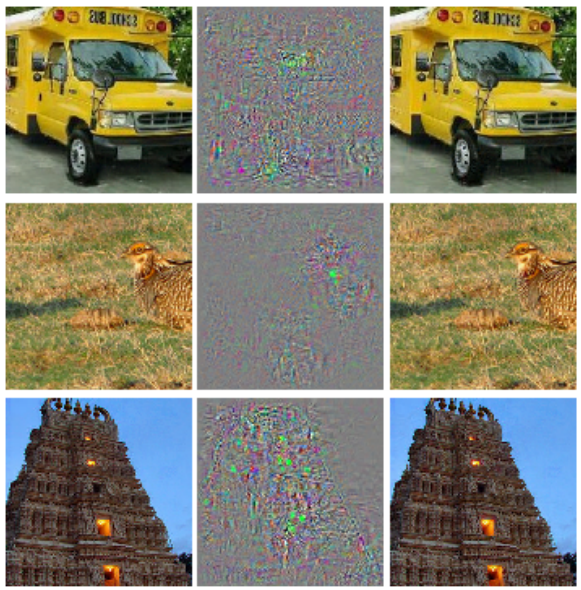
$$\begin{aligned} \min \quad & ||r||_2 \\ \text{subject to} \quad & f(x+r) \neq l \\ & x + r \in [0,1]^m \end{aligned}$$

  where l is the target label.
- However, this is often computationally infeasible to solve. A common approach is to approximate it.

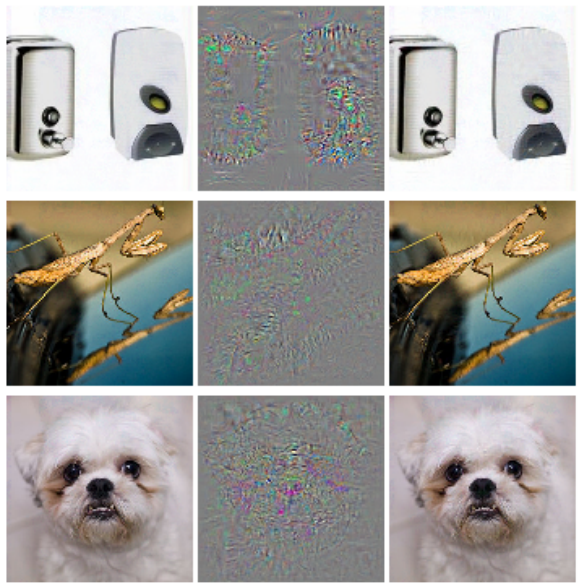
L-BFGS

- Szegedy et al. solved the following alternative optimization problem using a box-constrained L-BFGS

$$\begin{aligned} \min \quad & c|r| + \text{loss}_f(x+r, l) \\ \text{subject to} \quad & x + r \in [0,1]^m \end{aligned}$$



(a)



(b)

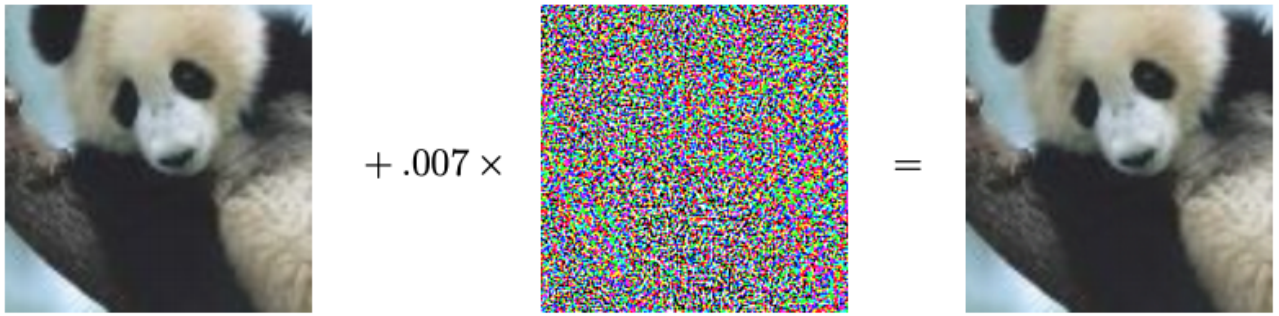
Intriguing properties of neural networks (Szegedy et al.)

Fast Gradient Sign Method

- Goodfellow et al. proposed the Fast Gradient Sign Method (FGSM), a single shot attack L_∞ attack that only requires one backpropagation call. The adversarial example is given by

$$\mathbf{x}^{adv} = \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} J(\mathbf{x}, y))$$

where J is the loss function. The adversarial example is then clipped to a specified range.



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

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

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

Explaining and harnessing adversarial examples (Goodfellow et al.)

Basic Iterative Method

- Kurakin et al. extended the FGSM into an iterative method.

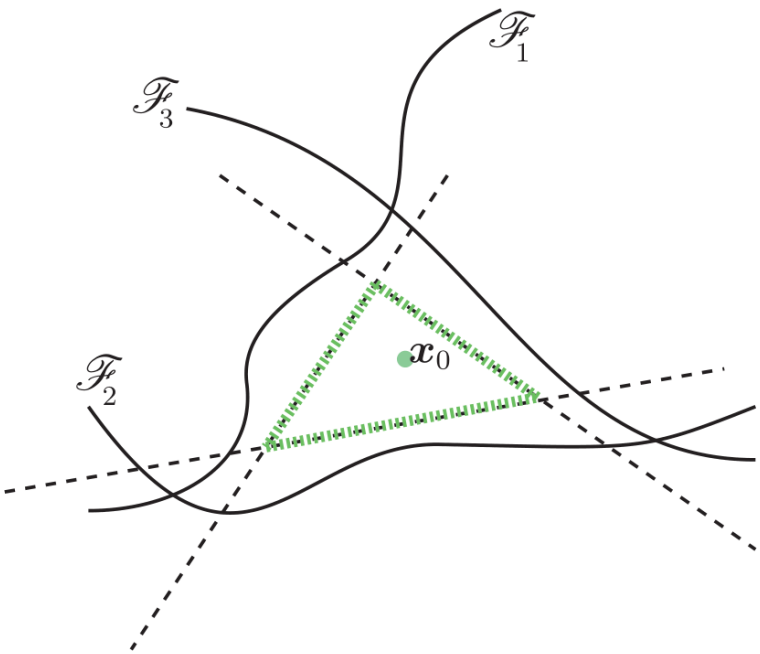
$$\mathbf{x}_0^{adv} = \mathbf{x}$$
$$\mathbf{x}_{n+1}^{adv} = \text{clip}(\mathbf{x}_n^{adv} + \alpha \text{sign}(\nabla_{\mathbf{x}} J(\mathbf{x}_n^{adv}, y)))$$

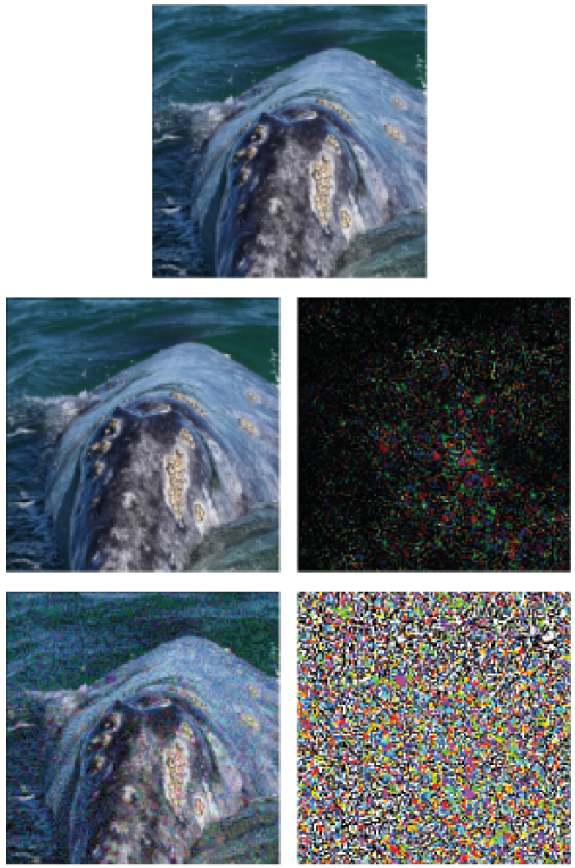
- Often more effective than FGSM

Adversarial examples in the physical world (Kurakin et al.)

DeepFool

- Fast iterative untargeted attack that produces adversarial examples by linearizing the network to an affine multiclass classifier.





DeepFool: a simple and accurate method to fool deep neural networks (Moosavi-Dezfooli et al.)

Carlini-Wagner Attack

- Strong iterative targeted L_2 attack that use gradient descent to minimize

$$||\delta||_2 + cf(x+\delta)$$

where

$$\delta = \frac{1}{2}(\tanh(w)+1) - x_i$$

$$f(x) = \max(\max_{i \neq t}\{Z(x)_i\} - Z(x)_t, -\kappa)$$

- Has been shown to beat defensive distillation, which was believed to be a robust defense against adversarial examples.
- Very sensitive to hyper-parameter tuning.

Towards Evaluating the Robustness of Neural Networks (Carlini and Wagner)

Problems and Challenges

- Understanding adversarial examples
  - Why are DNNs brittle?
- Stronger attacks
  - Physical attacks
  - Black box attacks
- Defense
  - Denoising, affine tranformations
  - Training a classifier to predict clean and adversarial examples
  - Training / finetuning on adversarial examples (adversarial training)
- Detection
  - Classifying images as clean or adversarial
  - Locating adversarial patches

Other Problems

- Adversarial training can be slow with stronger attacks
- Iterative methods do not adjust the step size
- In an attack setting, queries to a real world classifier may be limited or costly

Table of Contents

1. Introduction
2. Background
3. Contributions
4. Trust Region Optimization
5. Proposed Method
6. Results
7. Conclusion

Contributions

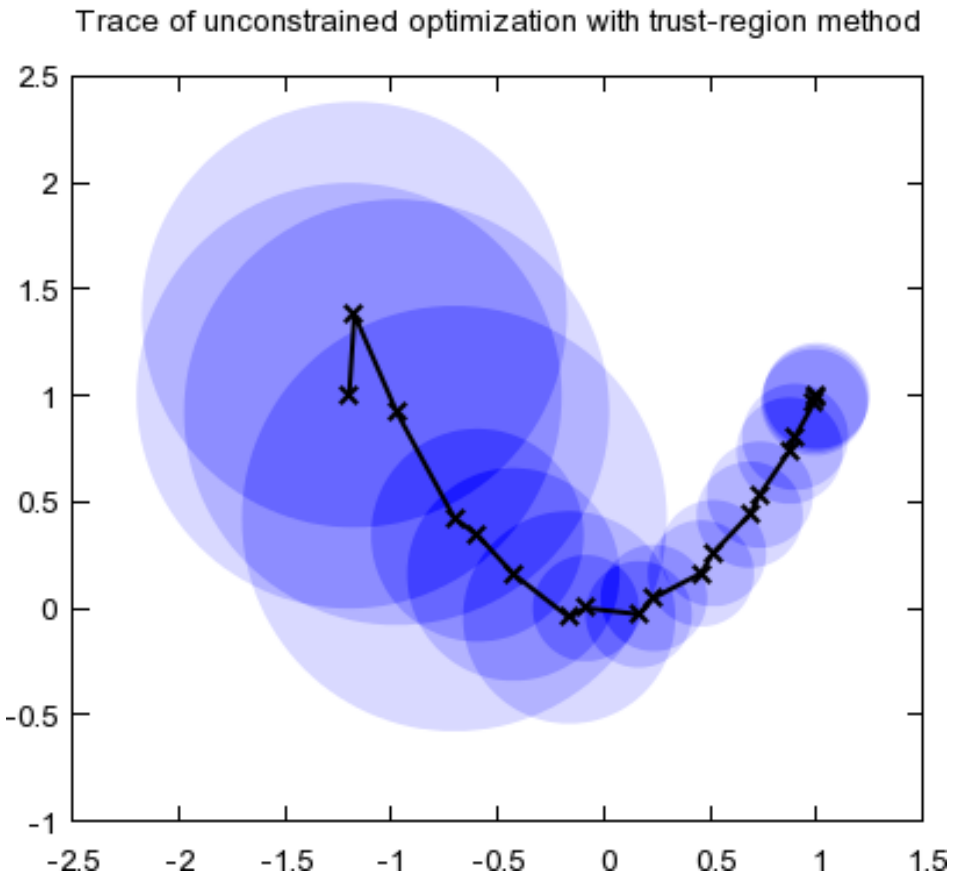
- The authors propose a white box targeted attack based on trust region (TR) optimization.
- Their method can adaptively choose the perturbation magnitude at each iteration, which removes the need for expensive hyperparameter tuning.
- TR attack can produce perturbations faster than CW (up to 37.5$\times$), and smaller in magnitude compared to DeepFool.
- Their method can easily be extended to second-order TR attacks, which could be useful for nonlinear activation functions.

Table of Contents

1. Introduction
2. Background
3. Contributions
4. Trust Region Optimization
5. Proposed Method
6. Results
7. Conclusion

Trust Region Optimization

- *Trust region* methods are a class of iterative nonlinear optimization algorithms
- They are based around trust regions, which are (usually) balls around the current point in which a quadratic model approximation is used to find a step direction



https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

Trust Region Subproblem

- Optimizing a function using a trust region method involves solving trust region subproblems

$$\min \; m_k(p) = f_k + {g_k}^T p + \frac{1}{2} p^T B_k p$$
$$s.t. \; ||p|| \leq \Delta_k$$

  where Δ_k is the trust region radius, g_k is the gradient at the current point, and B_k is the Hessian.
- The optimal p to the subproblem is called the *Cauchy point*.

Updating the Trust Region

- To update the size of the trust region at each iteration, we compute the following ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$$

- Based on the value of ρ_k and p_k, we may choose to increase the radius, keep it the same, or decrease it

Trust region example - Initial Start

- We will find the minimum of the Branin function. The Branin function has 3 global minima.
- $\min \ f(x_1, x_2) = (x_2 - 0.129{x_1}^2 + 1.6x_1 - 6)^2 + 6.07\cos(x_1) + 10$
- The initial variables we define are
  - $x = (6, 14)$
  - $\Delta_0 = 2, \Delta_M = 5$
  - $t_1 = 0.25, t_2 = 2$
  - $\eta_1 = 0.2, \eta_2 = 0.75$

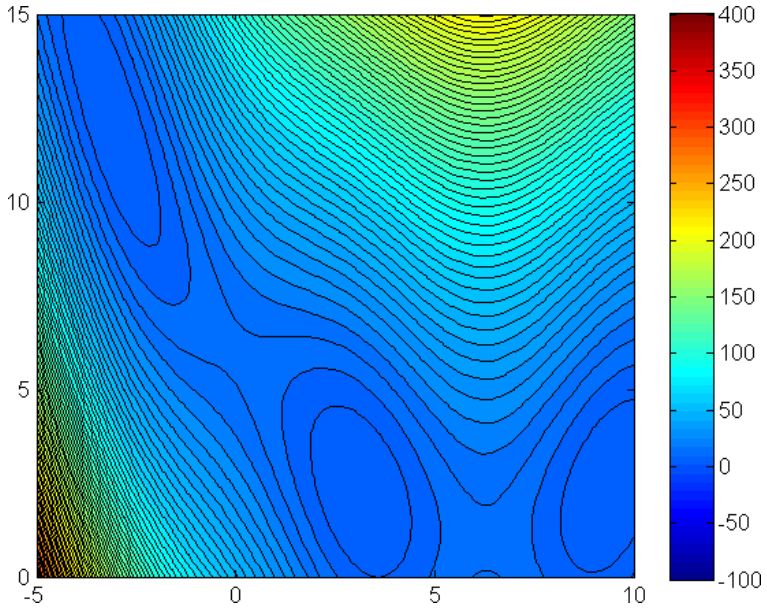
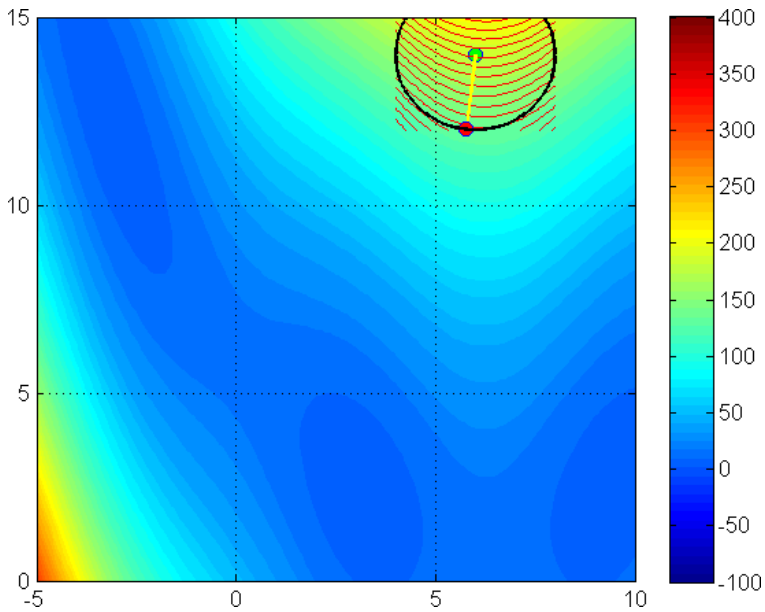


Figure: Contour of Branin function

https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

Iteration 1

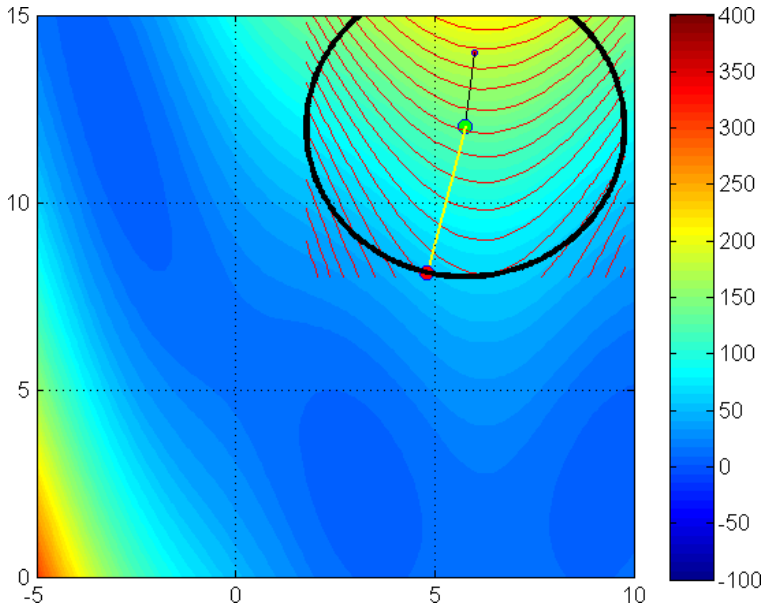
- The algorithm starts at the green point, and the trust region is defined as the area inside the circle centered at the starting point.
- After computing the Cauchy point ρ_k, we evaluate the ratio ρ_k, which comet out to be 0.99. Since $\rho_k > \eta_2$, we take a full step $x_{k+1} = x_k + p_k = (5.767, 12.014)$, and we increase the radius of the trust region to $\Delta_k = \min(t_2\Delta_k, \Delta_M)$.



https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

Iteration 2

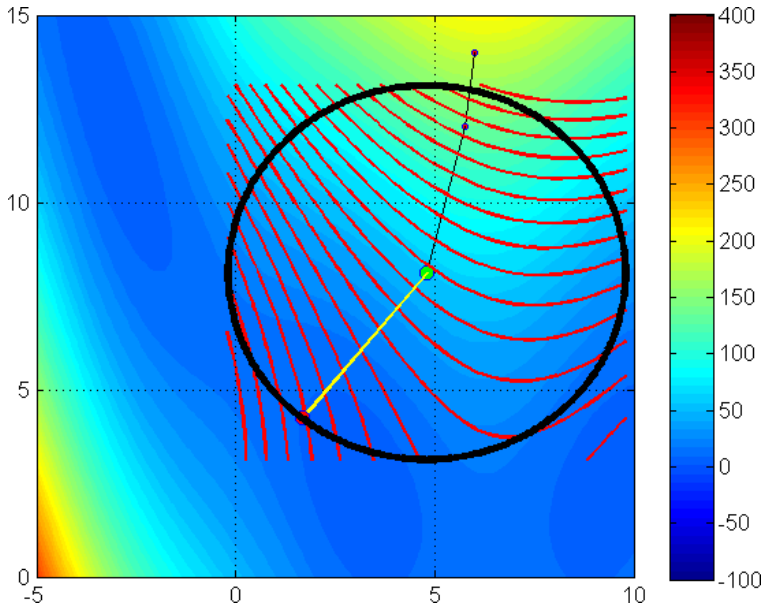
- Starting with the new point and the larger trust region, compute the Cauchy point and ρ_k again to get $\rho_k = 0.98$. Thus, we take a full step, and the trust region radius increases again.



https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

Iteration 3

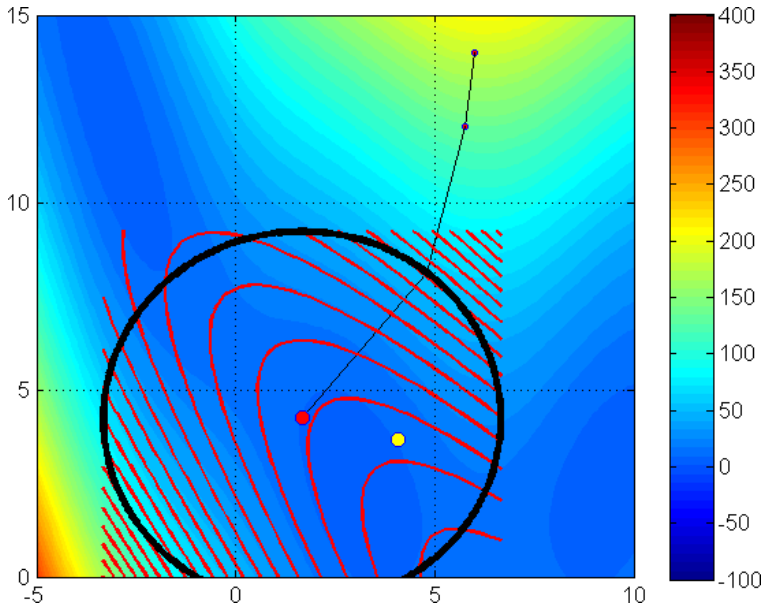
- This time, $\rho_k = 0.578$, which is not large enough to be trusted again. So we step again, but keep the radius the same.



https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

Iteration 4

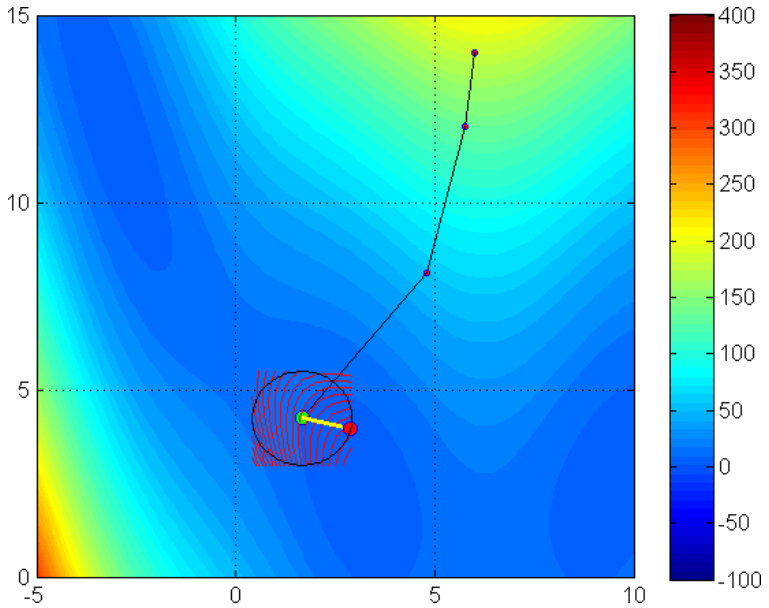
- The new point is not a good prediction, and gives us $\rho_k = -0.16$. We do not step, and we decrease the radius of the trust region by a factor of $t_1 = 0.25$.



https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

Iteration 5

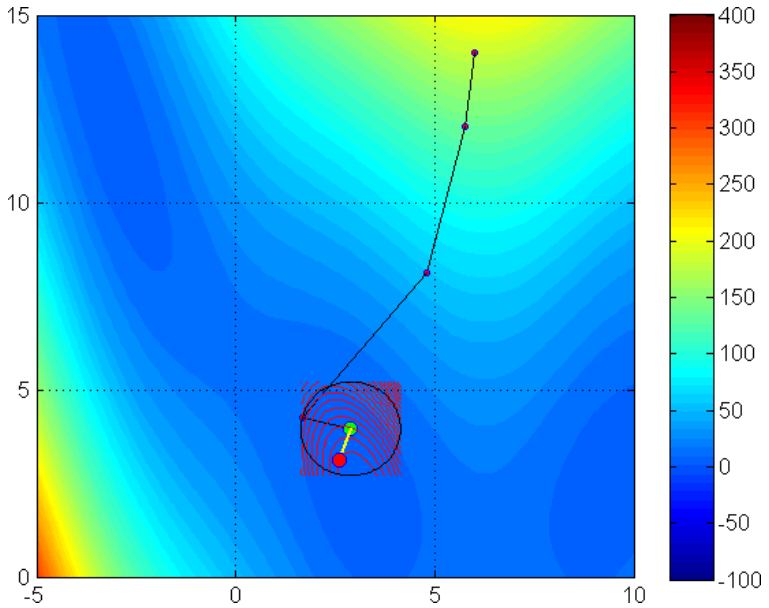
- We get $\rho_k = 0.729$, which is not large enough. So we step forward and keep the radius size the same.



https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

Iteration 6

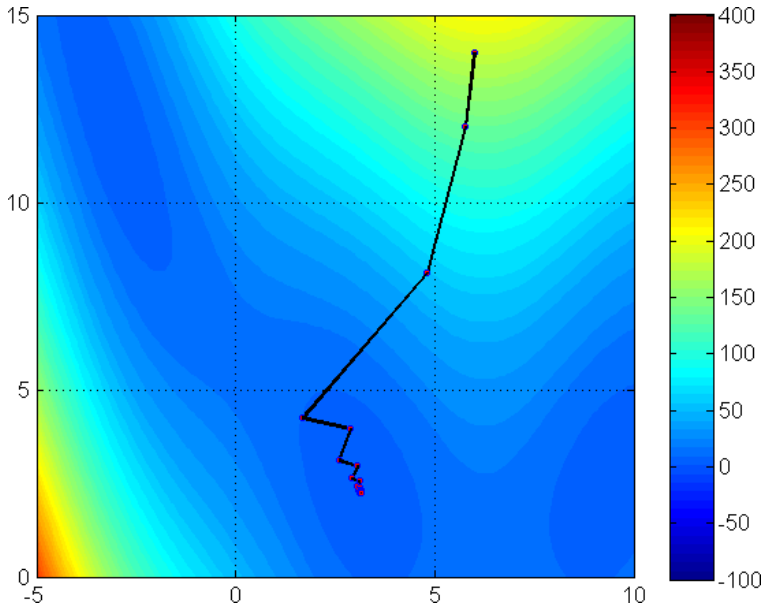
- In this case, $\rho_k = 0.989$. While this is high enough to update the radius, $||p_k|| \neq \Delta_k$, so we step, but do not make a full step, and we keep the radius unchanged.



https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

Final Trajectory after 20 iterations

- The algorithm terminates when the norm of the gradient is close to 0, or when the difference between successful points is close to 0.



https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

Table of Contents

1. Introduction
2. Background
3. Contributions
4. Trust Region Optimization
5. Proposed Method
6. Results
7. Conclusion

Proposed Method

- To find adversarial perturbations within a trust region, the authors solve

$$\min_{||\Delta \mathbf{x}^j||_p} m^j(\Delta \mathbf{x}^j) = \langle \Delta \mathbf{x}^j, \mathbf{g}_{t,i}^j \rangle + \frac{1}{2} \langle \Delta \mathbf{x}^j, \mathbf{H}_{t,i}^j \Delta \mathbf{x}^j \rangle$$

  where ϵ^j is the TR radius at the jth iteration, m^j is the approximation of the kernel function of $f(\mathbf{x}^{j-1}) = \mathbf{z}_t^{j-1} - \mathbf{z}_i^{j-1}$, with $\mathbf{g}_{t,i}^j$ and $\mathbf{H}_{t,i}^j$ denoting the corresponding gradient and Hessian.
- If a ReLU activation is used, the Hessian is zero almost everywhere, so they can omit the Hessian, and do a first order approximation instead.

Table of Contents

1. Introduction
2. Background
3. Contributions
4. Trust Region Optimization
5. Proposed Method
6. Results
7. Conclusion

Metrics

- The authors evaluated the speed of their method by recording the number of seconds it takes to find an adversarial image.
- To measure perturbation, the authors use relative perturbation. The relative perturbation of an image is defined as

$$\rho_p = \frac{||\Delta \mathbf{x}||_p}{||\mathbf{x}||_p}.$$

Types of attacks used

- Two types of attacks are performed: *best class* attack and *hardest class*. *Best class* attack means we target the class with

$$\arg\min_j \frac{z_t - z_j}{||\nabla_{\mathbf{x}}(x_t - z_j)||}.$$

- Similarly for *hardest class* attack, we attack the class with

$$\arg\max_j \frac{z_t - z_j}{||\nabla_{\mathbf{x}}(x_t - z_j)||}.$$

Summary of setup

- Datasets
  - CIFAR10
  - ImageNet
- Attacks
  - Iterative FGSM (only L_∞)
  - DeepFool
  - Carlini-Wagner (only L_2)
  - TR Non-Adapt
  - TR Adap
- Networks
  - CIFAR10
    - ★ AlexLike
    - ★ AlexLike-S (AlexLike with swiss activation)
    - ★ ResNet
    - ★ Wide ResNet
  - ImageNet
    - ★ AlexNet
    - ★ VGG16
    - ★ ResNet50
    - ★ DenseNet121

Time Performance on ImageNet

- TR and TR Adap produce similar perturbations as CW, but with significantly less time (up to 37.5×).

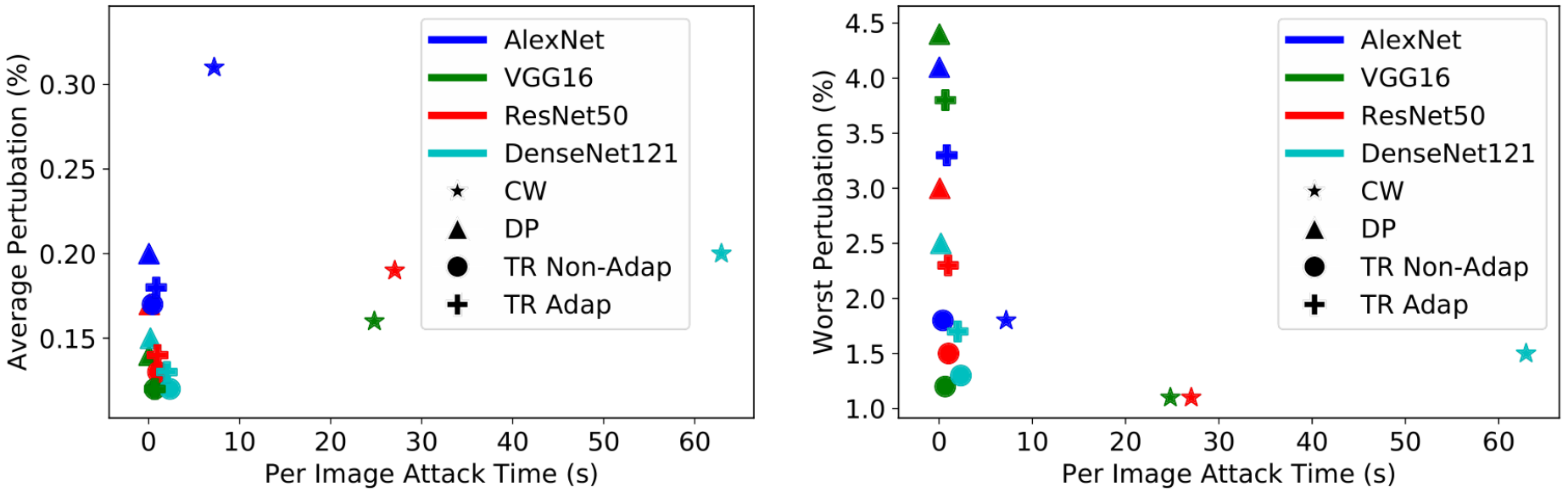


Figure 4: Perturbation magnitude vs. time

Qualitative Results

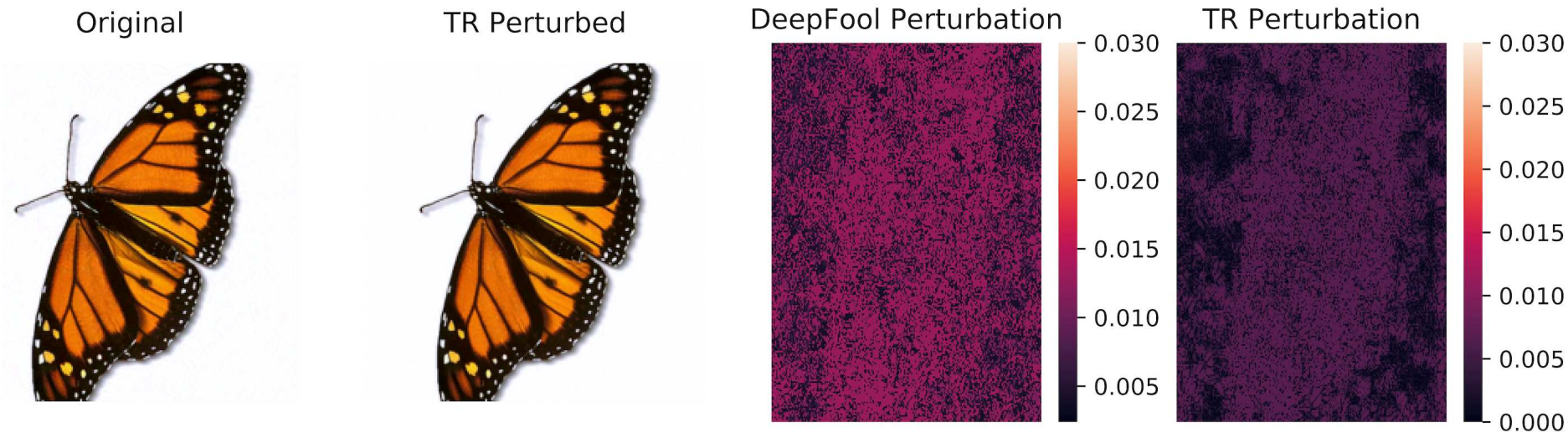


Figure 2: Attacks on VGG-16 with L_∞ norm. TR perturbation is smaller than DF (1.9$\times$ smaller)

CIFAR10 Results

For all tables and models, the perturbation is chosen such that the accuracy of the target model is reduced to less than 0.1%

		DeepFool	CW	TR Non-Adap	TR Adap
Model	Accuracy	ρ_2	ρ_2	ρ_2	ρ_2
AlexLike	85.78	1.67% / 11.5%	1.47% / 9.70%	1.49% / 9.13%	1.49% / 9.09%
AlexLike-S	86.53	1.74% / 11.0%	1.57% / 8.59%	1.57% / 9.48%	1.57% / 9.46%
ResNet	92.10	0.80% / 5.60%	0.62% / 3.12%	0.66% / 3.97%	0.66% / 3.96%
WResNet	94.77	0.89% / 5.79%	0.66% / 4.51%	0.73% / 4.43%	0.72% / 4.34%
		DeepFool	FGSM	TR Non-Adap	TR Adap
Model	Accuracy	ρ_∞	ρ_∞	ρ_∞	ρ_∞
AlexLike	85.78	1.15% / 6.85%	1.40% / 16.44%	1.05% / 5.45%	1.03% / 5.45%
AlexLike-S	86.53	1.18% / 6.01%	1.45% / 14.88%	1.09% / 4.76%	1.07% / 4.73%
ResNet	92.10	0.60% / 3.98%	0.85% / 4.35%	0.56% / 3.18%	0.50% / 3.35%
WResNet	94.77	0.66% / 3.34%	0.85% / 3.30%	0.56% / 2.67%	0.54% / 2.69%

Table 1: Best class attack.

CIFAR10 Results

	DeepFool	TR Non-Adap	TR Adap
Model	ρ_2	ρ_2	ρ_2
AlexLike	4.36% /18.9%	2.47% /13.4%	2.47% /13.4%
AlexLike-S	4.70% /17.7%	2.63% /14.4%	2.62% /14.2%
ResNet	1.71% /8.01%	0.99% /4.76%	0.99% /4.90%
WResNet	1.80% /8.74%	1.05% /6.23%	1.08% /6.23%
Model	ρ_∞	ρ_∞	ρ_∞
AlexLike	2.96% /12.6%	1.92% /9.99%	1.86% /10.0%
AlexLike-S	3.12% /12.2%	1.98% /8.19%	1.92% /8.17%
ResNet	1.34% /9.65%	0.77% /4.70%	0.85% /5.44%
WResNet	1.35% /6.49%	0.81% /3.77%	0.89% /3.90%

Table 2: Hardest class attack

ImageNet Results

Model	Accuracy	DeepFool	CW	TR Non-Adap	TR Adap
		ρ_2	ρ_2	ρ_2	ρ_2
AlexNet	56.5	0.20% / 4.1%	0.31% / 1.8%	0.17% / 2.5%	0.18% / 3.3%
VGG16	71.6	0.14% / 4.4%	0.16% / 1.1%	0.12% / 1.2%	0.12% / 3.8%
ResNet50	76.1	0.17% / 3.0%	0.19% / 1.1%	0.13% / 1.5%	0.14% / 2.3%
DenseNet121	74.4	0.15% / 2.5%	0.20% / 1.5%	0.12% / 1.3%	0.13% / 1.7%
Model	Accuracy	DeepFool	FGSM	TR Non-Adap	TR Adap
		ρ_∞	ρ_∞	ρ_∞	ρ_∞
AlexNet	56.5	0.14% / 4.3%	0.16% / 4.7%	0.13% / 1.4%	0.13% / 3.6%
VGG16	71.5	0.11% / 4.0%	0.18% / 5.1%	0.10% / 1.4%	0.10% / 3.4%
ResNet50	76.1	0.13% / 3.2%	0.18% / 3.7%	0.11% / 1.3%	0.11% / 2.7%
DenseNet121	74.4	0.11% / 2.3%	0.15% / 4.1%	0.10% / 1.1%	0.10% / 1.8%

Table 3: Best class attack

ImageNet Results

	DeepFool	TR Non-Adap	TR Adap
Model	ρ_2	ρ_2	ρ_2
AlexNet	0.74% /8.7%	0.39% /5.0%	0.39% /5.0%
VGG16	0.45% /5.4%	0.27% /3.6%	0.27% /3.8%
ResNet50	0.52% /5.8%	0.31% /4.2%	0.31% /4.2%
DenseNet	0.48% /5.7%	0.29% /3.8%	0.29% /3.8%
Model	ρ_∞	ρ_∞	ρ_∞
AlexNet	0.53% /9.9%	0.31% /7.5%	0.33% /9.1%
VGG16	0.36% /11.6%	0.25% /5.1%	0.26% /6.8%
ResNet50	0.43% /6.6%	0.28% /3.7%	0.30% /4.6%
DenseNet	0.38% /6.4%	0.24% /4.5%	0.27% /5.7%

Table 4: Hardest class attack

Second order attack results

Iter		1	2	3	4	5	6	7	8	9	10
MLP	TR First	47.63	33.7	22.24	13.76	8.13	4.59	2.41	1.31	0.63	0.27
MLP	TR Second	47.84	33.37	21.49	13.3	7.39	4.16	2.17	1.09	0.49	0.20
AlexNet	TR First	51.51	28.17	12.45	5.53	2.61	1.33	0.82	0.66	0.51	0.46
AlexNet	TR Second	50.96	26.97	10.73	4.11	1.79	0.91	0.67	0.54	0.47	0.44

Table of Contents

1. Introduction
2. Background
3. Contributions
4. Trust Region Optimization
5. Proposed Method
6. Results
7. Conclusion

Conclusion

- The authors propose a white box targeted attack called TR attack, which is based on trust region methods.
- Their method can produce smaller adversarial perturbations very quickly
- TR attack can choose the perturbations at each iteration, and can be easily extended to a second order attack.

References I


Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li.
Adversarial objects against lidar-based autonomous driving systems.
arXiv preprint arXiv:1907.05418, 2019.


Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song.
Robust physical-world attacks on deep learning visual classification.
In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.


Ian Goodfellow, Jonathon Shlens, and Christian Szegedy.
Explaining and harnessing adversarial examples.
In *International Conference on Learning Representations*, 2015.


Alexey Kurakin, Ian Goodfellow, and Samy Bengio.
Adversarial examples in the physical world.
arXiv preprint arXiv:1607.02533, 2016.


Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami.
Practical black-box attacks against machine learning.
Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security - ASIA CCS '17, 2017.

References II


Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai.
One pixel attack for fooling deep neural networks.
IEEE Transactions on Evolutionary Computation, 23(5):828–841, Oct 2019.


Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus.
Intriguing properties of neural networks.
In *International Conference on Learning Representations*, 2014.


Simen Thys, Wiebe Van Ranst, and Toon Goedeme.
Fooling automated surveillance cameras: Adversarial patches to attack person detection.
In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.


Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille.
Adversarial examples for semantic segmentation and object detection.
2017 IEEE International Conference on Computer Vision (ICCV), Oct 2017.