Improving Adversarial Robustness with Proximal Algorithms

Analiza adwersaryjna, bezpieczeństwo i wyjaśnialność systemów sztucznej inteligencji

June 2025

Plan for today

- Research question
- Types of optimizers
- Types of adversarial attacks
- Related works
- Methodology
- Empirical findings
- Conclusion

Research questions

- Q1 - Do models trained with proximal optimizers exhibit higher adversarial robustness than those trained with standard optimizers?
- Q2 - Are such models more robust across some types of attacks and different metrics ℓ_0, ℓ_2, ..., ℓ_∞?

Gradient Descent (GD)

Optimization Problem:

$$\min_x f(x)$$

Update Rule:

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

Optimization Problem:

$$\min_x f(x)$$

Update Rule:

$$y_{t+1} = x_t - \eta \nabla f(x_t)$$

$$x_{t+1} = y_{t+1} + \frac{k-1}{k+2}(y_{t+1} - y_t)$$

Optimization Problem:

$$\min_x f(x)$$

Update Rule:

$$\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla f(x_t) \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla f(x_t))^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \\
x_{t+1} &= x_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}
\end{aligned}$$

where β_1, β_2 are decay rates, and ϵ is a small constant for numerical stability.

Optimization Problem:

$$\min_x g(x) + h(x)$$

where $h(x)$ is a regularization term, e.g., $h(x) = \lambda\|x\|_1$.

Update Rule:

$$x_{t+1} = \text{prox}_{\lambda\eta h}(x_t - \eta\nabla g(x_t)) = T(x_t - \eta\nabla g(x_t))$$

Proximal Operator for L1 Regularization:

$$\text{prox}_{\lambda||x||_1}(v) := T(v) = \begin{cases} v - \lambda, & v > \lambda \\ 0, & |v| \leq \lambda \\ v + \lambda, & v < -\lambda \end{cases}$$

FISTA (Fast ISTA)

Optimization Problem:

$$\min_x f(x) + g(x)$$

where $g(x)$ is a regularization term.

Update Rule:

$$y_{t+1} = \mathrm{prox}_{\lambda\eta g}(x_t - \eta\nabla f(x_t))$$

$$x_{t+1} = y_{t+1} + \frac{t-1}{t+2}(y_{t+1} - y_t)$$

Modification: Adaptive step size $\eta \to \eta_k$:

$$\eta_k = \frac{1}{L} + k\mu$$

Modification: Step-size depends on support structure:

$$\eta_k = c \frac{\|s_k \nabla f(x_k)\|^2}{\|\Phi(s_k \nabla f(x_k))\|^2}$$

where:

- c is a scaling constant controlling the step size.
- s_k is a binary mask indicating which elements of x_k are nonzero (support of x_k).
- $\nabla f(x_k)$ is the gradient of the loss function with respect to the model parameters.
- $\Phi(\cdot)$ represents a transformation that captures structural information, such as a convolutional layer in a CNN.

What are budget adversarial attacks?

Goal: Create an adversarial example that fools the model within perturbation in some norm $||.||$

$$\text{Find } \delta \text{ such that } f(x+\delta) \neq y \quad \text{and} \quad \|\delta\| \leq \epsilon$$

- x: original input
- y: true label
- f: classifier
- δ: perturbation
- ϵ: maximum allowed perturbation (budget)

What are minimal adversarial attacks?

Goal: Find the smallest possible perturbation that causes the model to misclassify.

$$\text{Find } \min_{\delta} \|\delta\| \quad \text{subject to } f(x+\delta) \neq y$$

Related Work: Bridging the Gap Between Adversarial Robustness ...

Preprint

Bridging the Gap Between Adversarial Robustness and Optimization Bias

February 2021

DOI:10.48550/arXiv.2102.08868

Authors:

Fartash Faghri
University of Toronto

Cristina Nader Vasconcelos
Fluminense Federal University

David J. Fleet
University of Toronto

Fabian Pedregosa

- Analysis of multiple factors affecting robustness
- Overparametrized models using GD seems to bias toward solutions with minimal norm
- To avoid high overparametrization, one may introduce regularisation

Choice of:

- **Optimizer**
- Network architecture
- **Regularizer**

each has significant impact on adversarial robustness of resulting model, where bolded ones may benefit from the use of proximal methods

Unveiling and Mitigating Adversarial Vulnerabilities in Iterative Optimizers

Submitted on 26 Apr 2025

Elad Sofer, Tomer Shaked, Caroline Chaux, Nir Shlezinger

Deep unfolding of the optimizer:

- Iterations of iterative optimizer become layers
- Optimizer hyperparameters are made learnable
- This process may be directed toward e.g. adversarial robustness

Key takeaways:

- Proximal optimizers like ISTA are vulnerable to attacks due to improved differentiability
- Presented unfolding procedure that improves robustness of such optimizers

Attacks can use proximal algorithms too

Corpus ID: 235313522

PDPGD: Primal-Dual Proximal Gradient Descent Adversarial Attack

Alexander Matyasko, Lap-Pui Chau • Published in arXiv.org 3 June 2021 • Computer Science

TLDR A fast, general and accurate adversarial attack that optimises the original non-convex constrained minimisation problem and introduces primal-dual proximal gradient descent attack for non-smooth norm minimisation. Expand

PDPGD: Primal-Dual Proximal Gradient Descent Attack

$$\min_{r} \max_{\lambda \geq 0} \mathcal{L}(r, \lambda) = \|r\| + \lambda \cdot \mathbb{I}\left[\hat{k}(x + r) \neq y\right]$$

- **r**-player minimizes by **r**
- λ-player maximizes by λ

- **r**-player uses proximal operators, as regularisation is non-smooth

PDPGD: Primal-Dual Proximal Gradient Descent Attack

For each dataset, they use three types of models:

- Plain, ℓ_∞, ℓ_2

Attacks were also norm-based:

- ℓ_0, ℓ_1, ℓ_2 ℓ_∞

PDPGD: Key takeaways

- Robustness is norm-specific
- Model trained on given norm has better robustness against attacks of this norm
- … but may behave worse against other norms

Network Architecture

```python
# Network used
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1)
        self.bn1 = nn.BatchNorm2d(16)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1)
        self.bn2 = nn.BatchNorm2d(32)
        self.fc = nn.Linear(32 * 28 * 28, 10)

    def forward(self, x):
        x = torch.relu(self.bn1(self.conv1(x)))
        x = torch.relu(self.bn2(self.conv2(x)))
        x = x.view(x.size(0), -1)  # Flatten
        return self.fc(x)
```

Figure: Convolutional Neural Network trained on MNIST

Training

The models were trained for 30 epochs with following optimizers.

1. ADAM,
2. AGD,
3. FISTA,
4. GD,
5. ISTA,
6. Adaptive μ FISTA,
7. Adaptive support FISTA,

White-box attacks used[1]

Attack	Type	Norm	Params
C&W	Minimal	ℓ_2, ℓ_∞	$iters = 50$
PGD	Budget	ℓ_∞	$\epsilon = 0.1$
StrAttack	Minimal	ℓ_2	$iters = 50$
DDN	Minimal	ℓ_2	
Trust Region	Minimal	ℓ_2,	
FAB	Minimal	ℓ_2	
Auto-PGD	Budget	ℓ_2	$\epsilon = 0.1$
ALMA	Minimal	ℓ_2	
FGA	Minimal	ℓ_0	
FMN	Minimal	ℓ_2	
PDGD / PDPGD	Minimal	ℓ_2	
SuperDeepFool	Minimal	ℓ_2	
σ-zero	Minimal	ℓ_0	

Model training results

Algorithm	Test Accuracy (%)
AGD	97.27
GD	97.76
adaptive support FISTA	69.58
adaptive μ FISTA	97.67
FISTA	98.20
ISTA	98.25
ADAM	98.19

Table: Comparison of Algorithms: Test Accuracy

Algorithm	**Final sparsity (%)**
AGD	0.062
GD	0.064
Adaptive Support FISTA	46.868
Adaptive μ FISTA	19.695
FISTA	37.666
ISTA	43.787
ADAM	0.012

Table: Comparison of Algorithms: Final Sparsity for threshold $= 10^{-5}$

Fooling rates

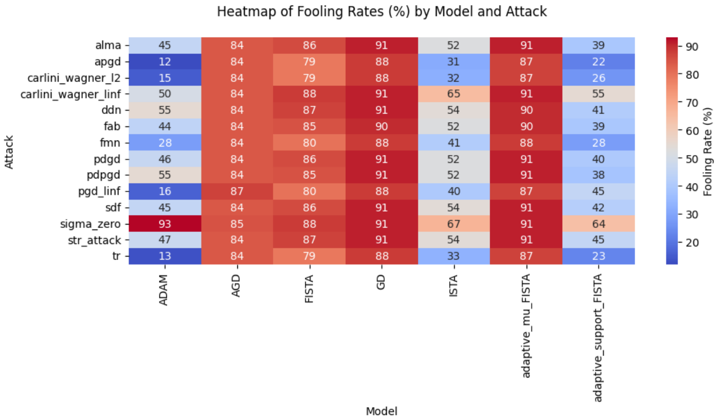

Figure: Table of fooling rates (%) achieved by different attacks for different models

Cross-examination

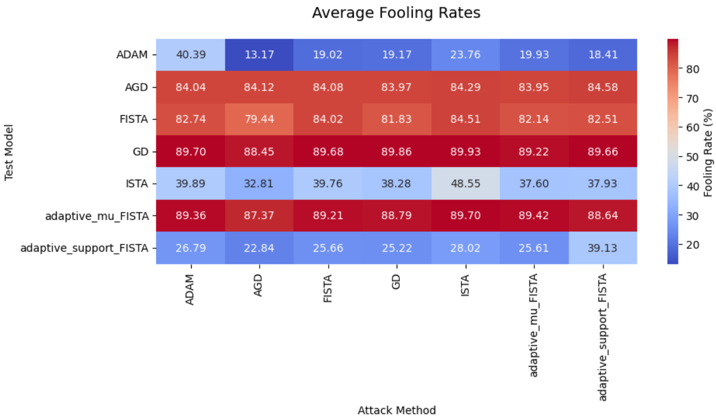

Figure: Cross-examination of adversarial examples generated for "test" models against other models

- Q1: *Do models trained with proximal optimizers exhibit higher adversarial robustness than those trained with standard optimizers?*
  - No, because Adam. Except σ_0. But better than GD and AGD.
- Q2: *Are such models more robust across some types of attacks and different metrics ℓ_0, ℓ_2, ..., ℓ_∞?*
  - Yeah, kinda. Especially ISTA and Adaptive Support FISTA.

References

- *Adversarial library on Github*.
- F. Faghri, C. N. Vasconcelos, D. J. Fleet, F. Pedregosa, and N. Le Roux, *Bridging the Gap Between Adversarial Robustness and Optimization Bias*, arXiv preprint arXiv:2102.08868, 2021.
- E. Sofer, T. Shaked, C. Chaux, and N. Shlezinger, *Unveiling and Mitigating Adversarial Vulnerabilities in Iterative Optimizers*, arXiv preprint arXiv:2504.19000, 2025.
- A. Matyasko and L.-P. Chau, *PDPGD: Primal-Dual Proximal Gradient Descent Adversarial Attack*, arXiv preprint arXiv:2106.01538, 2021.