Revisiting and Advancing Fast Adversarial Training through the Lens of Bi-level Optimization

Yihua Zhang[1,*] ,Guanhua Zhang[2,*], Prashant Khanduri[3], Mingyi Hong[3], Shiyu Chang[2], Sijia Liu[1,4]

[1]Michigan State University, [2]UC Santa Barbara, [3]University of Minnesota, [4]MIT-IBM Watson Lab

PAPER

CODE

(Min-Max) Adversarial Training: Existing Principled Solution

- Nearly all existing work adopted the **Adversarial Training (AT)** framework [Madry et al. 2017], formulated as **min-max optimization**

Training over adversarially perturbed dataset

$$\overbrace{\text{minimize}_{\boldsymbol{\theta}}\ \mathrm{E}_{(\boldsymbol{x},t)\sim D}\underbrace{\left[\max_{\|\boldsymbol{\delta}\|_\infty\leq\epsilon}\ell_{\text{tr}}(\boldsymbol{\theta};\boldsymbol{x}+\boldsymbol{\delta},t)\right]}}$$

Sample-wise ‘adversarial attack’ generation

Limitation 1 (formulation level):

Attack type restriction: Must be the opposite of training objective

Limitation 2 (computation level):

Each training step needs multiple gradient back-propagations for attack generation

(Min-Max) Adversarial Training: Existing Principled Solution

- In our paper, we focus on the following question:

How to advance the **algorithmic foundation** to scale up Adversarial Training?

- Answer: Bi-level Optimization

Bi-Level Optimization (BLO) Enables General AT Formulation

- **Standard min-max formulation for adversarial training:**

$$\min_{\theta} \mathrm{E}_{(\boldsymbol{x},t)\sim D} \left[\max_{|\boldsymbol{\delta}|_\infty \leq \epsilon} \ell_{\mathrm{tr}}(\boldsymbol{\theta}; \boldsymbol{x} + \boldsymbol{\delta}, t) \right]$$

- **BLO-oriented adversarial training (AT)**

Upper-level optimization $\min_{\theta} \quad \ell_{\mathrm{tr}}(\theta, \boldsymbol{\delta}^*(\theta))$

Lower-level optimization $\mathrm{s.t.} \quad \boldsymbol{\delta}^*(\boldsymbol{\theta}) = \mathrm{argmin}_{|\boldsymbol{\delta}|_\infty \leq \epsilon} \ \ell_{\mathrm{atk}}(\theta, \delta)$

- **Attack objective ℓ_{atk} will be set different from training objective ℓ_{tr}**
- **Why BLO?** A possible framework of **attack-agnostic robust training**

A careful design of ℓ_{atk} can **scale up adversarial training**

(Our focus)

Implicit Gradient --- The Tricky Part of BLO

- **Presence of implicit gradient (IG): the 'fingerprint' of BLO**

The upper-level gradient calculation:

$$\frac{\mathrm{d}\ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))}{\mathrm{d}\theta} = \frac{\partial \ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))}{\partial \theta} + \underbrace{\frac{\mathrm{d}\boldsymbol{\delta}^*(\boldsymbol{\theta})^T}{\mathrm{d}\theta}}_{\text{IG}} \frac{\partial \ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))}{\partial \delta}$$

IG: $\boldsymbol{\delta}^*(\boldsymbol{\theta})$ is an implicit function of $\boldsymbol{\theta}$

- BLO is hard to solve, while proper lower-level objective makes it tractable!

BLO w/ Lower-Level Linearization

- **BLO-oriented adversarial training (AT)**

$$\text{minimize}_{\boldsymbol{\theta}} \; \mathrm{E}_{\boldsymbol{x}\sim D}[\ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))]$$
$$\text{subject to} \quad \boldsymbol{\delta}^*(\boldsymbol{\theta}) = \text{argmin}_{\boldsymbol{\delta}\in C} \; \ell_{\mathrm{atk}}(\boldsymbol{\theta}, \boldsymbol{\delta})$$

- **BLO with customized lower-level attack objective**
  - **Linearization at z with quadratic regularization:**

$$\ell_{\mathrm{atk}}(\boldsymbol{\theta}, \boldsymbol{\delta}) = < \nabla_{\boldsymbol{\delta}=\boldsymbol{z}} \; \ell_{\mathrm{atk}}(\boldsymbol{\theta}, \boldsymbol{\delta}), \boldsymbol{\delta} - \boldsymbol{z} > + \left(\frac{\lambda}{2}\right) ||\boldsymbol{\delta} - \boldsymbol{z}||_2^2$$

  - **Benefit:** Unique, computation-efficient, closed-form lower-level minimizer

$$\boldsymbol{\delta}^*(\boldsymbol{\theta}) = \mathrm{Proj}_C(\boldsymbol{z} - (1/\lambda) \, \nabla_{\boldsymbol{\delta}} \ell_{\mathrm{atk}}(\boldsymbol{\theta}, \boldsymbol{z}))$$

Lower-level linearization leads to one-step PGD attack

Fast BAT

- **Fast Bi-level Adversarial Training (Fast BAT)**

$$\text{minimize}_{\boldsymbol{\theta}}\ \mathrm{E}_{\boldsymbol{x}\sim D}[\ell_{\mathrm{tr}}(\boldsymbol{\theta},\boldsymbol{\delta}^*(\boldsymbol{\theta}))]$$
$$\text{subject to}\ \ \boldsymbol{\delta}^*(\boldsymbol{\theta}) = \text{argmin}_{\boldsymbol{\delta}\in C} < \nabla_{\boldsymbol{\delta}}\, \ell_{\mathrm{atk}}(\boldsymbol{\theta},\boldsymbol{z}), \boldsymbol{\delta}-\boldsymbol{z} > + \left(\frac{\lambda}{2}\right) ||\boldsymbol{\delta}-\boldsymbol{z}||_2^2$$

- **Fast BAT algorithm: Alternating optimization**
  - ❖ Fix $\boldsymbol{\theta}$, obtain lower-level solution $\boldsymbol{\delta}^*(\boldsymbol{\theta})$

$$\boldsymbol{\delta}^*(\boldsymbol{\theta}) = \mathrm{Proj}_C(\boldsymbol{z} - (1/\lambda)\, \nabla_{\boldsymbol{\delta}}\ell_{\mathrm{atk}}(\boldsymbol{\theta},\boldsymbol{z}))$$

  - ❖ Fix $\boldsymbol{\delta}$, obtain upper-level model update by SGD

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \underbrace{\frac{\mathrm{d}\ell_{\mathrm{tr}}(\boldsymbol{\theta},\boldsymbol{\delta}^*(\boldsymbol{\theta}))}{\mathrm{d}\boldsymbol{\theta}}}$$

➢ **Non-trivial**: *Chain rule* cannot be applied since projection operation is not smooth

$$\frac{d\ell_{\mathrm{tr}}(\boldsymbol{\theta},\boldsymbol{\delta}^*(\boldsymbol{\theta}))}{d\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}}\ell_{\mathrm{tr}}(\boldsymbol{\theta},\boldsymbol{\delta}^*(\boldsymbol{\theta})) + \underbrace{\frac{d\boldsymbol{\delta}^*(\boldsymbol{\theta})^\top}{d\boldsymbol{\theta}}}_{\mathrm{IG}} \nabla_{\boldsymbol{\delta}}\ell_{\mathrm{tr}}(\boldsymbol{\theta},\boldsymbol{\delta}^*(\boldsymbol{\theta}))$$

Fast BAT

- **Derivation of implicit gradient (IG)** $\frac{\mathrm{d}\boldsymbol{\delta}^*(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}}$
  - **Key idea:** Extract implicit functions that involves IG from KKT conditions of lower-level problem
  - **Why is KKT tractable?** In robust training, the lower-level constraint $\boldsymbol{\delta} \in \mathcal{C}$ is linear

Theorem 1 [Zhang et al., 2021]**:** With Hessian-free assumption, $\nabla_{\boldsymbol{\delta}\boldsymbol{\delta}} \ell_{\text{atk}}(\boldsymbol{\theta}, \boldsymbol{\delta}) = \mathbf{0}$

$$\frac{d\boldsymbol{\delta}^*(\boldsymbol{\theta})^\top}{d\boldsymbol{\theta}} = -(1/\lambda)\nabla_{\boldsymbol{\theta}\boldsymbol{\delta}} \ell_{\text{atk}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*)\mathbf{H}_{\mathcal{C}}, \quad \textit{with } \mathbf{H}_{\mathcal{C}} := \begin{bmatrix} 1_{p_1 < \delta_1^* < q_1} \mathbf{e}_1 & \cdots & 1_{p_1 < \delta_d^* < q_d} \mathbf{e}_d \end{bmatrix}$$

$1_{p<\delta<q}$ is an indicator function, $p_i = \max\{-\epsilon, -x_i\}, q_i = \{\epsilon, 1 - x_i\}$

Fast BAT

- **Fast Bi-level Adversarial Training (Fast BAT)**

$$\text{minimize}_{\boldsymbol{\theta}}\ \mathrm{E}_{\boldsymbol{x}\sim D}[\ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))]$$

$$\text{subject to}\ \ \boldsymbol{\delta}^*(\boldsymbol{\theta}) = \mathrm{argmin}_{\boldsymbol{\delta}\in C} < \nabla_{\boldsymbol{\delta}}\, \ell_{\mathrm{atk}}(\boldsymbol{\theta}, \boldsymbol{z}), \boldsymbol{\delta} - \boldsymbol{z} > + \left(\frac{\lambda}{2}\right) ||\boldsymbol{\delta} - \boldsymbol{z}||_2^2$$

- **Fast BAT algorithm:**
  - ❖ Fix $\boldsymbol{\theta}$, obtain lower-level solution $\boldsymbol{\delta}^*(\boldsymbol{\theta})$

$$\boldsymbol{\delta}^*(\boldsymbol{\theta}) = \mathrm{Proj}_C(\boldsymbol{z} - (1/\lambda)\, \nabla_{\boldsymbol{\delta}}\ell_{\mathrm{atk}}(\boldsymbol{\theta}, \boldsymbol{z}))$$

(Single-step perturbation)

  - ❖ Fix $\boldsymbol{\delta}$, obtain upper-level model update by SGD

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{\mathrm{d}\ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))}{\mathrm{d}\boldsymbol{\theta}}$$

(IG-involved model updating)

$$\frac{d\ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))}{d\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}}\ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta})) + \underbrace{\frac{d\boldsymbol{\delta}^*(\boldsymbol{\theta})^\top}{d\boldsymbol{\theta}}}_{\mathrm{IG}} \nabla_{\boldsymbol{\delta}}\ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))$$

$$\frac{d\boldsymbol{\delta}^*(\boldsymbol{\theta})^\top}{d\boldsymbol{\theta}} = -(1/\lambda)\nabla_{\boldsymbol{\theta}\boldsymbol{\delta}}\ell_{\mathrm{atk}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*)\mathbf{H}_C$$

(Theorem 1)

Fast BAT vs. Linearization Type

- **Fast BAT with gradient sign-based linearization**

$$\text{minimize}_{\boldsymbol{\theta}} \ \mathrm{E}_{\boldsymbol{x}\sim D}[\ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))]$$

$$\text{subject to } \ \boldsymbol{\delta}^*(\boldsymbol{\theta}) = \operatorname{argmin}_{\boldsymbol{\delta}\in C} < \operatorname{sign}(\nabla_{\boldsymbol{\delta}} \, \ell_{\mathrm{atk}}(\boldsymbol{\theta}, \boldsymbol{z})), \boldsymbol{\delta} - \boldsymbol{z} > + \left(\frac{\lambda}{2}\right) ||\boldsymbol{\delta} - \boldsymbol{z}||_2^2$$

- **Why gradient sign?**

Theorem 2: With sign-based linearization, Fast BAT simplifies to alternating optimization (without involving computation of implicit gradients)

Lower-level: $\boldsymbol{\delta}^*(\boldsymbol{\theta}) = \operatorname{Proj}_C(\boldsymbol{z} - (1/\lambda)\operatorname{sign}(\nabla_{\boldsymbol{\delta}}\ell_{\mathrm{atk}}(\boldsymbol{\theta}, \boldsymbol{z})))$

Upper-level: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\left(\frac{\partial \ell_{\mathrm{tr}}(\boldsymbol{\theta}, \boldsymbol{\delta}^*(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} + \mathbf{0}\right)$

} Fast AT [Wong et al., 2020]

Fast BAT + gradient sign-based linearization => Fast AT

Numerical Experiments of Fast BAT on CIFAR-10

Train-time and test-time perturbation strength

Model	Method	SA(%) ($\epsilon = 8/255$)	RA-PGD(%) ($\epsilon = 8/255$)	SA(%) ($\epsilon = 16/255$)	RA-PGD(%) ($\epsilon = 16/255$)
PARN-50	FAST-AT	73.15±6.10	41.03±2.99	43.86±4.31	22.08±0.27
	FAST-AT-GA	77.40±0.81	46.16±0.98	42.28±6.69	22.87±1.25
	PGD-2-AT	**83.53**±0.17	46.17±0.59	68.88±0.39	22.37±0.41
	FAST-BAT	78.91±0.68	**49.18**±0.35	**69.01**±0.19	**24.55**±0.06
WRN-16-8	FAST-AT	84.39±0.46	45.80±0.57	49.39±2.17	21.99±0.41
	FAST-AT-GA	81.51±0.38	48.29±0.20	45.95±13.65	23.10±3.90
	PGD-2-AT	**85.52**±0.14	45.47±0.14	**72.11**±0.33	23.61±0.16
	FAST-BAT	81.66±0.54	**49.93**±0.36	68.12±0.47	**25.63**±0.44

- Fast BAT improves baselines in both SA and RA
- Improvement becomes more significant when facing stronger attack ($\epsilon = 16/255$)

Fast-BAT Does not Suffer from Catastrophic Overfitting

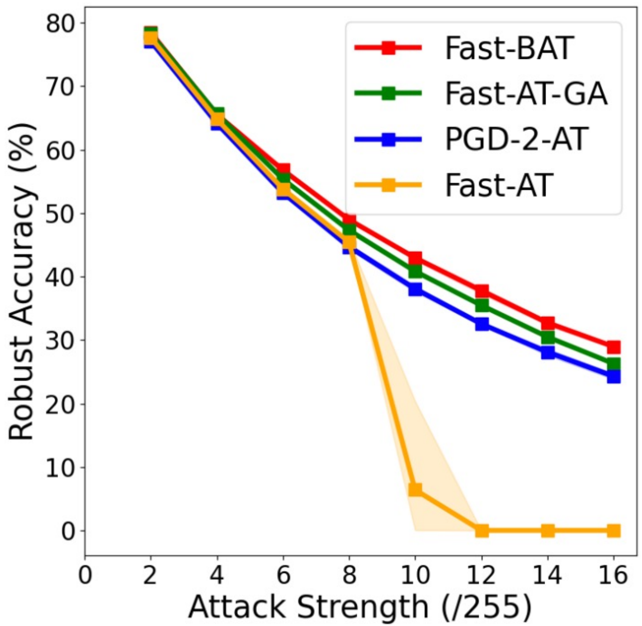

Figure. Robustness of different methods against different training attack strengths.

PAPER

CODE

thank you
tusind tak
谢谢
dakujem vám
ありがとう
ngiyabonga
dziękuję
merci
baie dankie
धन्यवाद
molte grazie
suksema
danke
gracias
obrigada
obrigado
takk
gràcies
dank u
teşekkür edire
mahalo
teşekkür ederim
شكرا
tänan
tack så mycket