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

Poster Session: Hall E #528, Wednesday (Tonight)

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

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

Sample-wise 'adversarial attack' generation

Assumption 1 (formulation level):

Attack type restriction: Must be the opposite of training objective.

Assumption 2 (computation level):

Each training step needs multiple gradient back-propagations to generate attacks.

(Min-Max) Adversarial Training: Existing Principled Solution

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

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

- Answer: Bi-level Optimization

- A properly designed formulation and solver will help scale up AT!

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 $\min_{\theta} \ \ell_{\mathrm{tr}}(\theta, \boldsymbol{\delta}^*(\theta))$

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

- **Decouple Attack objective ℓ_{atk} 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**

Implicit Gradient --- The Tricky Part of BLO

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

  The upper-level gradient calculation:

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

  IG: Gradient flow from lower-level to upper-level.
  $\boldsymbol{\delta}^*(\boldsymbol{\theta})$ is an implicit function of $\boldsymbol{\theta}$

- The lower-level constraint makes BLO even harder!
- Properly designed lower-level objective makes it tractable!

BLO w/ Lower-Level Linearization

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

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

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

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

  - **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 (No SIGN)!

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 <u>linear</u>

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

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

$1_{p<\delta<q}$ 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)**

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

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

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

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

(Single-step perturbation)

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

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

(IG-involved model updating)

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

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

(Theorem 1)

Fast BAT vs. Linearization Type

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

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

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

- **Why gradient sign?**

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

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

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

Fast AT
[Wong et al., 2020]

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

Numerical Experiments of Fast BAT on CIFAR-10

Metrics: Standard Accuracy, Robust Accuracy (PGD/AutoAttack)
Baselines: Fast-AT, Fast-AT with Gradient Alignment, 2-step AT

CIFAR-10, PARN-18 trained with $\epsilon = 8/255$					
Method	SA (%)	RA-PGD (%)		RA-AA (%)	
		$\epsilon = 8$	$\epsilon = 16$	$\epsilon = 8$	$\epsilon = 16$
FAST-AT	**82.39**±0.44	45.49 ±0.41	9.56 ±0.26	41.87 ±0.15	7.91 ±0.06
FAST-AT-GA	79.71±0.44	47.27 ±0.42	11.57±0.32	43.24 ±0.27	9.48 ±0.15
PGD-2-AT	81.97±0.41	44.62 ±0.39	9.39 ±0.32	41.73 ±0.20	7.54±0.25
FAST-BAT	79.97 ±0.12	**48.83** ±0.17	**14.00** ±0.21	**45.19** ±0.12	**11.51** ±0.20
CIFAR-10, PARN-18 trained with $\epsilon = 16/255$					
FAST-AT	44.15±7.27	37.17 ±0.74	21.83 ±1.32	31.66 ±0.27	12.49 ±0.33
FAST-AT-GA	58.29 ±1.32	43.86 ±0.67	26.01 ±0.16	38.69 ±0.56	17.97 ±0.33
PGD-2-AT	68.04 ±0.30	48.79 ±0.31	24.30 ±0.46	41.59 ±0.22	15.40 ±0.29
FAST-BAT	**68.16** ±0.25	**49.05** ±0.12	**27.69** ±0.16	**43.64** ±0.26	**18.79** ±0.24

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

PAPER

CODE

Poster Session: Hall E #528, Wednesday (Tonight)