You Only Propagate Once: Accelerate Adversarial Training via Maximal Principle.

Dinghuai Zhang*, Tianyuan Zhang*, Yiping Lu*, Zhanxing Zhu, Bin Dong (*equal contribution)

Peking University and Stanford University

Back Ground And Aim

Adversarial Examples:

Deep networks are often sensitive to adversarial perturbations. To effectively defend the adversarial attacks, [1] proposed adversarial training, which can be formulated as a robust optimization

$$\min_{\theta} \mathbb{E}_{(x,y)\sim p} \max_{|\eta|\le\epsilon} l(\theta; x+\eta, y)$$

parameter, adversary, label

Adversarial Defense is expansive:

Multi-step gradient decent is applied to achieve the "optimal" adversary

Aim: Faster adversarial defense method via decoupling adversary updating and gradient calculation.

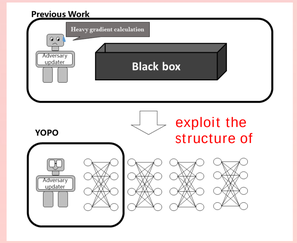
Contribution

A differential game formulation for defensing adversarial example for deep neural networks.

To the best of our knowledge, it is the first attempt to **design NN–specific algorithm for adversarial defense.**

Derive the an optimality condition, i.e. the Pontryagin's Maximum Principle

The PMP motivates a new adversarial training algorithm, YOPO. We finally **achieve about 4– 5 times speed up** than the original PGD training with comparable results on MNIST/CIFAR10.



A New Formulation: Differential Game

$$\min_{\theta} \max_{\|\eta_t\|_\infty \le \epsilon} J(\theta,\eta) := \frac{1}{N}\sum_{i=1}^{N} \ell_i(x_{i,T}) + \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} R_t(x_{i,t};\theta_t)$$

$$\text{Subject to. } x_{i,1} = f_0(x_{i,0}+\eta_i, \theta_0), i = 1,2,\cdots,N$$

$$x_{i,t+1} = f_t(x_{i,t},\theta_t), t = 1,2,\cdots,T-1$$

Loss function — Regularization

Here

- $\{(x_i, y_i)\}_{i=1}^{N}$ denotes the dataset
- the dynamics $\{f_t(x_t,\theta_t), t=0,\cdots,T\}$ represent a deep neural network, T denote the number of the layer, f_t denotes a nonlinear transformation for one layer of neural network
- θ_t denotes the parameters in layer
- η_i denotes the adversary associated with data x_i

Algorithm: Gradient Based YOPO

Let us first rewrite the original robust optimization problem (in a mini-batch form) as

$$\min_{\theta}\max_{|\eta_i|\le\epsilon} \sum_{i=1}^{B} l(g_{\tilde\theta}(f_0(x_i+\eta_i,\theta_0)), y_i)$$

Here f_0 denotes the first layer and $g_{\tilde\theta} = f_{T-1}^{\theta_{T-1}} \circ f_{T-2}^{\theta_{T-2}} \circ \cdots \circ f_1^{\theta_1}$ denotes the network without the first layer.

Original PGD:

- For $s = 0,1,\ldots,r-1$, perform

$$\eta_i^{s+1} = \eta_i^s + \alpha_1 \nabla_{\eta_i} \ell(g_{\tilde\theta}(f_0(x_i+\eta_i^s,\theta_0)), y_i),\ i = 1,\cdots,B,$$

Where by the chain rule

$$\nabla_{\eta_i}\ell(g_{\tilde\theta}(f_0(x_i+\eta_i^s,\theta_0)), y_i) = \nabla_{g_{\tilde\theta}}\left(\ell(g_{\tilde\theta}(f_0(x_i+\eta_i^s,\theta_0)), y_i)\right)\cdot\nabla_{f_0}\left(g_{\tilde\theta}(f_0(x_i+\eta_i^s,\theta_0))\right)\cdot\nabla_{\eta_i} f_0(x_i+\eta_i^s,\theta_0)$$

- Perform the SGD weight update (momentum SGD can also be used here)

$$\theta \leftarrow \theta - \alpha_2 \nabla_\theta \left(\sum_{i=1}^{B} \ell(g_{\tilde\theta}(f_0(x_i+\eta_i^m,\theta_0)), y_i)\right)$$

Gradient based YOPO:

- Initialize $\{\eta_i^{1,0}\}$ for each input x_i. For $j = 1,2,\cdots,m$
  - Calculate the slack variable p

$$p = \nabla_{g_{\tilde\theta}}\left(\ell(g_{\tilde\theta}(f_0(x_i+\eta_i^{j,0},\theta_0)), y_i)\right)\cdot\nabla_{f_0}\left(g_{\tilde\theta}(f_0(x_i+\eta_i^{j,0},\theta_0))\right)$$

  - Update the adversary for $s = 0,1,\ldots,n-1$ for fixed p

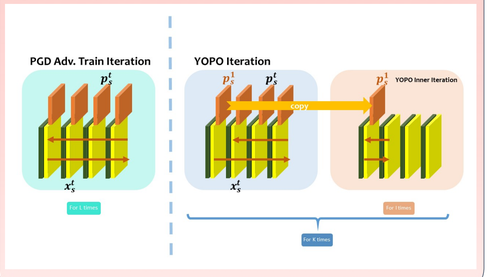
$$\eta_i^{j,s+1} = \eta_i^{j,s} + \alpha\left(p\cdot\nabla_{\eta_i} f_0(x_i+\eta_i^{j,s},\theta_0), i = 1,\cdots,B\right.$$

  - Let $\eta_i^{j+1,0} = \eta_i^{j,n}$.
- Calculate the weight update

$$U = \sum_{j=1}^{m}\nabla_\theta\left(\sum_{i=1}^{B}\ell(g_{\tilde\theta}(f_0(x_i+\eta_i^{j,n},\theta_0)), y_i)\right)$$

and update the weight $\theta \leftarrow \theta - \alpha_2 U$. (Momentum SGD can also be used here.)

Pipeline of YOPO-m-n. The yellow and olive blocks represent feature maps while the orange blocks represent the gradients of the loss w.r.t. feature maps of each layer.



Motivation: The Pontryagin's Maximum Principle

Then we derive the an optimality condition, i.e. the Pontryagin's Maximum Principle, for the differential game.

First, we design a layer-wise Hamiltonian function:

$$H_t(x,p,\theta_t) = p\cdot f_t(x_t,\theta_t) - \frac{1}{B}R_t(x,\theta_t)$$

(PMP for adversarial training) Assume ℓ_i is twice continuous differentiable, $f_t(\cdot,\theta), R_t(\cdot,\theta)$ are twice continuously differentiable with respect to x; $f_t(\cdot,\theta), R_t(\cdot,\theta)$ together with their x partial derivatives are uniformly bounded in t and θ, and the sets $\{f_t(x,\theta):\theta\in\Theta_t\}$ and $\{R_t(x,\theta):\theta\in\Theta_t\}$ are convex for every t and $x\in\mathbb{R}^d$. Denote θ^* as the solution of the differential game, then there exists co-state processes $p_t^* := \{p_{i,t}^* : t\in[T]\}$ such that the following holds for all $t\in[T]$ and $i\in[B]$:

1. $x_{i,t+1}^* = \nabla_p H_t(x_{i,t}^*, p_{i,t+1}^*, \theta_t^*), x_{i,0}^* = x_{i,0} + \eta_i^*$
2. $p_{i,t}^* = \nabla_x H_t(x_{i,t}^*, p_{i,t+1}^*, \theta_t^*), p_{i,T}^* = -\frac{1}{B}\nabla\ell_i(x_{i,T}^*)$

At the same time, the parameters of the first layer $\theta_0^*\in\Theta_0$ and the optimal adversarial perturbation η_i^* satisfy **Adversary only couples with the first layer!**

$$\sum_{i=1}^{B} H_0(x_{i,0}^* + \eta_i, p_{i,1}^*, \theta_0^*) \ge \sum_{i=1}^{B} H_0(x_{i,0}^* + \eta_i^*, p_{i,1}^*, \theta_0^*) \ge \sum_{i=1}^{B} H_0(x_{i,0}^* + \eta_i^*, p_{i,1}^*, \theta_0)\ \forall\theta_0\in\Theta_0, \|\eta_i\|_\infty \le \epsilon$$

and the parameters of the other layers $\theta_t^*\in\Theta_t, t\in[T]$ maximize the Hamiltonian functions

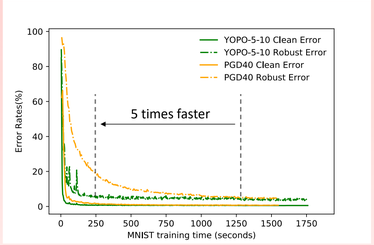
$$\sum_{i=1}^{B} H_t(x_{i,t}^*, p_{i,t+1}^*, \theta_t^*) \ge \sum_{i=1}^{B} H_t(x_{i,t}^*, p_{i,t+1}^*, \theta_t),\ \forall\theta_t\in\Theta_t$$

Forward process

Back Propagation for feature: $p_t^* = -\nabla_{x_t}(l(x_T))$ — Layer-wise maximal principle

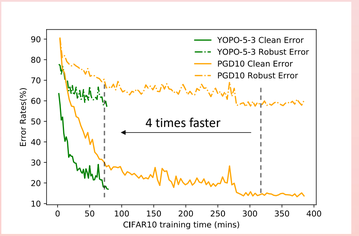
Results: MNIST

	Clean Data	PGD40	CW attack
PGD5	99.43	42.39	77.04
PGD10	99.53	77.00	82.00
PGD 40	99.49	96.56	93.52
YOPO-5-10	99.46	96.27	93.56



Results: CIFAR

	Clean Data	PGD20	Training Time
Natural Training	95.03	0.00	233
PGD3	90.07	39.18	1134
PGD5	89.65	43.85	1574
PGD10	***87.30***	***47.04***	***2713***
Free-8(Neurips19)[4]	***86.29***	***47.00***	***667***
YOPO-3-5	87.27	43.04	299
YOPO-5-3	***86.70***	***47.98***	***476***

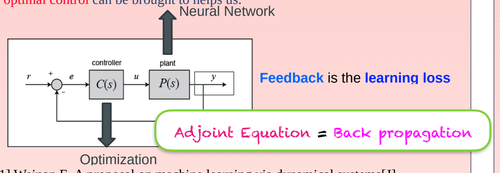


Result for TRADES:

	Clean Data	PGD20	Training Time
TRADES	***86.14***	***44.50***	***633***
YOPO-3-4	***87.82***	***46.13***	***259***
YOPO-2-5	88.15	42.48	218

Relation To Neural ODE

Understanding neural network as an ODE [1-3], training a neural network is equivalent to solving an optimal control problem. Thus theory(PMP) and algorithms in optimal control can be brought to helps us.



Feedback is the learning loss

Adjoint Equation = Back propagation

[1] Weinan E. A proposal on machine learning via dynamical systems[J]. Communications in Mathematics and Statistics, 2017, 5(1): 1-11.

[2] Yiping Lu, Aoxiao Zhong, Quanzheng Li, Bin Dong. "Beyond Finite Layer Neural Network:Bridging Deep Architects and Numerical Differential Equations" Thirty-fifth International Conference on Machine Learning (ICML), 2018

[3] Chen T Q, Rubanova Y, Bettencourt J, et al. Neural ordinary differential equations[C]//Advances in neural information processing systems. 2018: 6571-6583.

Concolusions

In this work, we have developed an efficient strategy for accelerating adversarial training. We recast the adversarial training of deep neural networks as a discrete time differential game and derive a Pontryagin's Maximum Principle (PMP) for it. Based on this maximum principle, we discover that the adversary is only coupled with the weights of the first layer. This motivates us to **split the adversary updates from the back-propagation gradient calculation.** The proposed algorithm, called YOPO, avoids computing full forward and backward propagation for too many times, thus effectively reducing the computational time as supported by our experiments.

-Consider NN as ODE, training becomes control problem [2,3]
-Write down P.M.P, adversary just couple with the first layer
-3-5 times faster because You Only Propagate Once!
-Equivalent to decouple the gradient!

The first two authors are looking for Ph.D. Positions, feel free to contact them if you are interested!

Reference

[1] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations, 2018.

[2] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for backpropagation. In Proceedings of the 1988 connectionist models summer school, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

[3] Li Q, Chen L, Tai C, et al. Maximum principle based algorithms for deep learning[J]. The Journal of Machine Learning Research, 2017, 18(1): 5998-6026.

[4] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Xu Zeng, John Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! arXiv preprint arXiv:1904.12843, 2019.

Github

Code is available: https://github.com/a1600012888/YOPO-You-Only-Propagate-Once