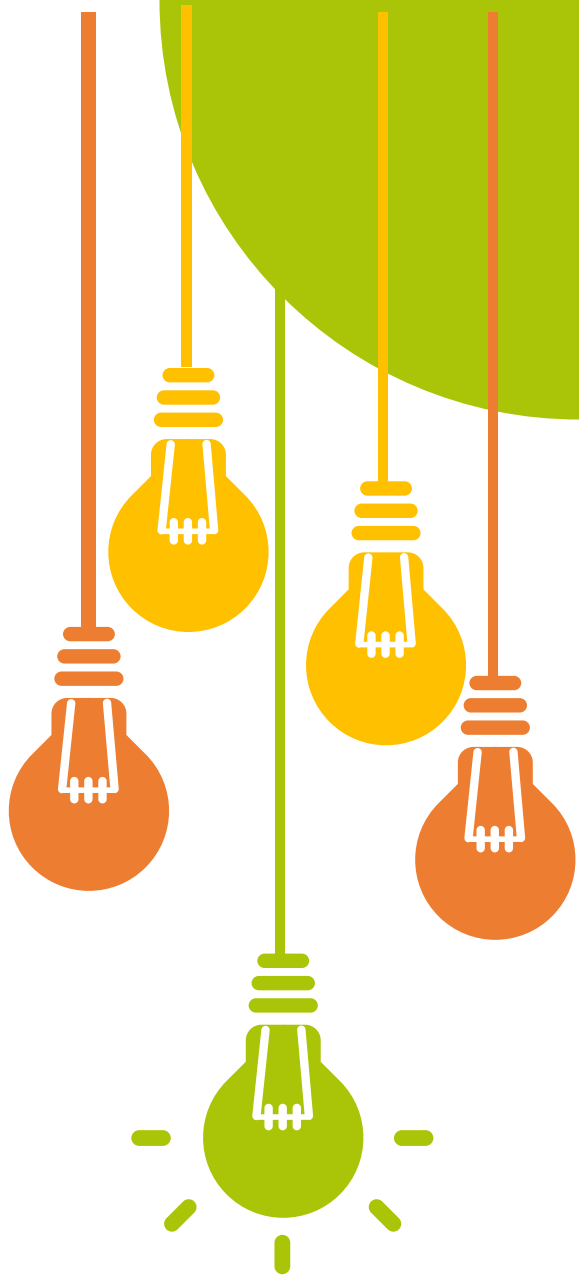
You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle

Student: *Razan Dibo*

May, 2021

Skoltech

Contents:



Skoltech

Problem statement

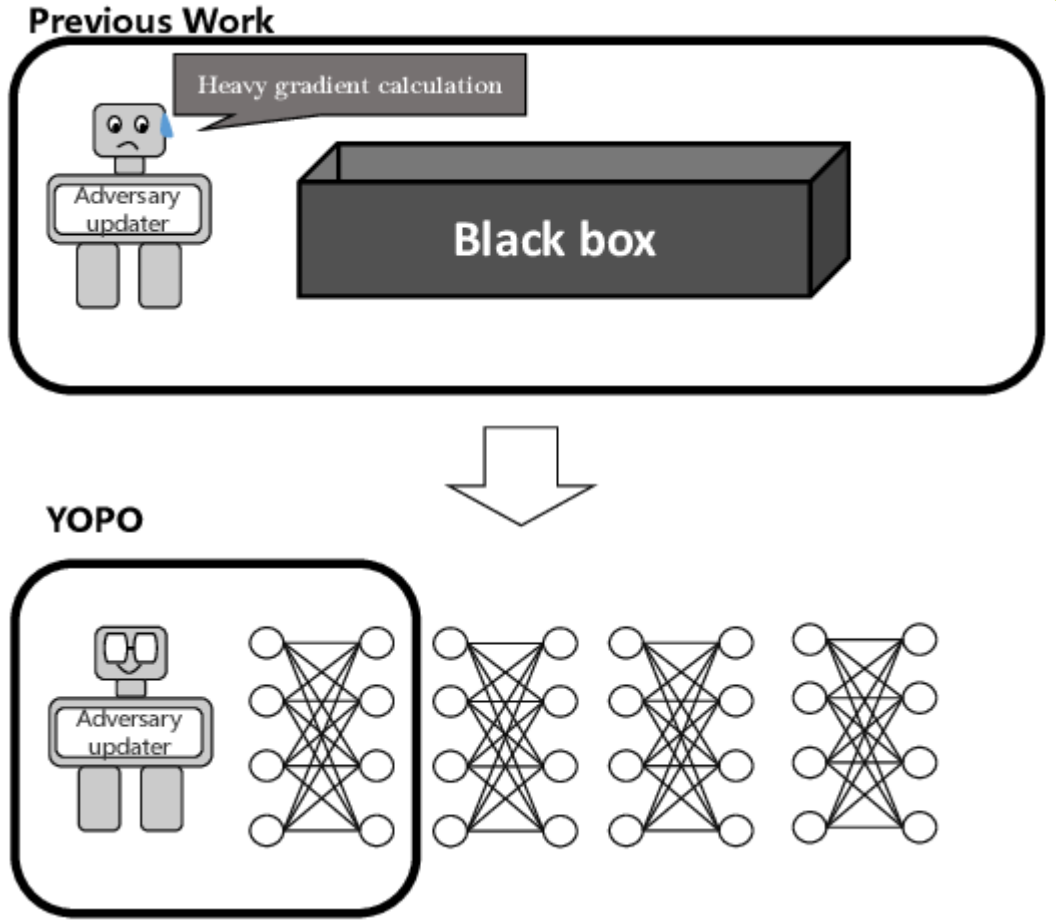
1- Deep networks are often sensitive to adversarial perturbations.

2- The computational overhead of the generation of adversarial examples

Skoltech

URGENCY

Reduction of the total number of full forward and backward propagation to only one for each group of adversary updates.



Skoltech

Related Theory

To effectively defend the adversarial attacks, proposed adversarial training, which can be formulated as a robust optimization:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \max_{\|\eta\|\leq\epsilon} \ell(\theta; x+\eta, y),$$

Skoltech

Related Theory

The robust optimization problem (1) can be written as a differential game as follows,

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

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

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

Code

Let us first rewrite the original robust optimization problem (1) (in a mini-batch form) as

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

- 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,

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

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

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

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

Code

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

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

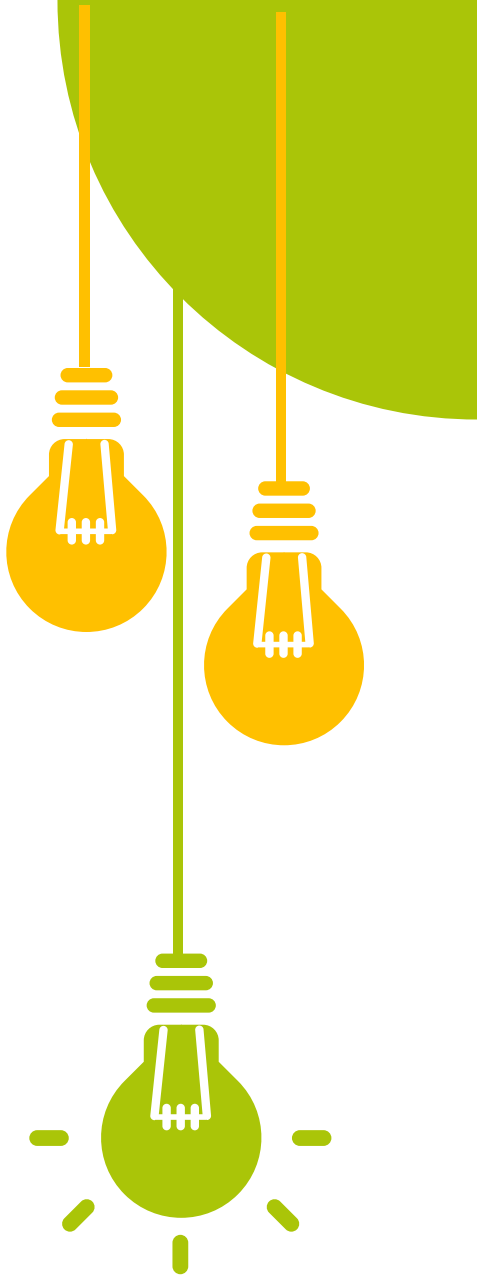
  - Update the adversary for $s = 0, 1, \ldots, n-1$ for fixed p

$$\eta_i^{j,s+1} = \eta_i^{j,s} + \alpha_1 p \cdot \nabla_{\eta_i} f_0(x_i + \eta_i^{j,s}, \theta_0), i = 1, \cdots, B$$

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

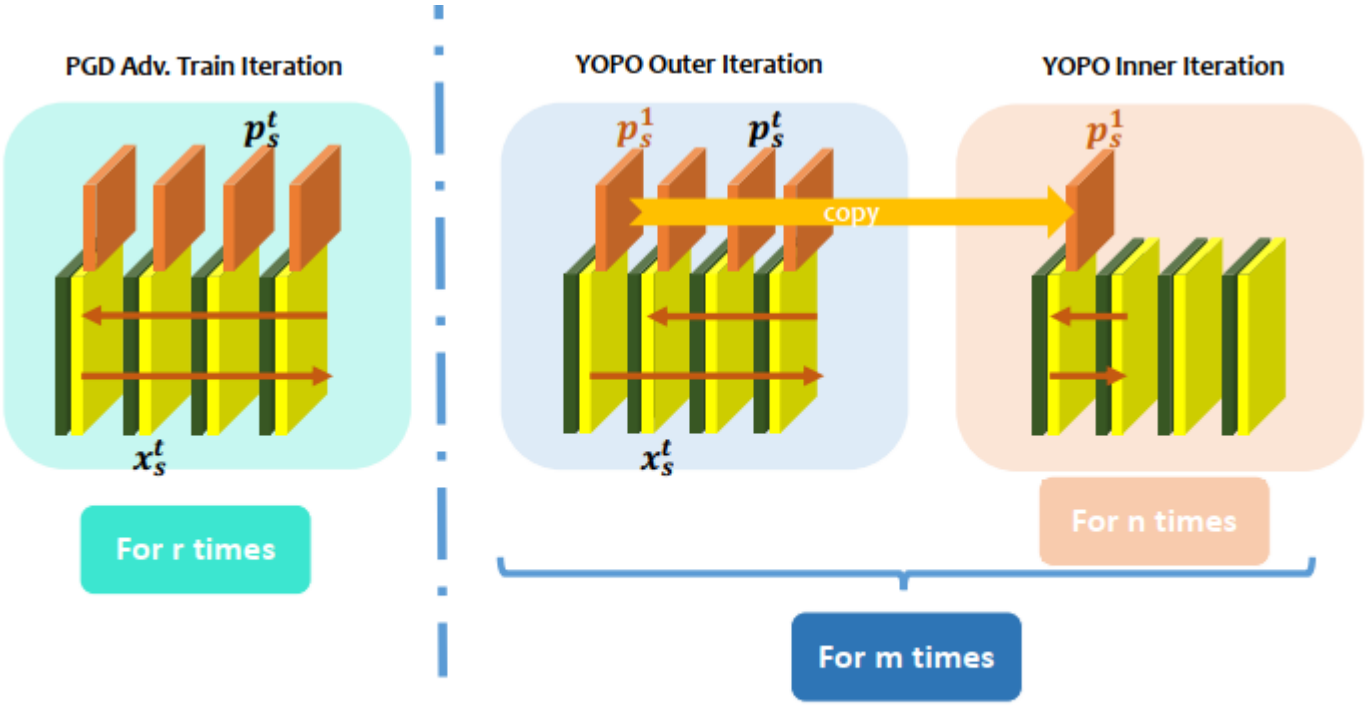
$$U = \sum_{j=1}^{m} \nabla_\theta \left(\sum_{i=1}^{B} \ell(g_{\tilde{\theta}}(f_0(x_i + \eta_i^{j,n}, \theta_0)), y_i) \right)$$

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



Skoltech

Code



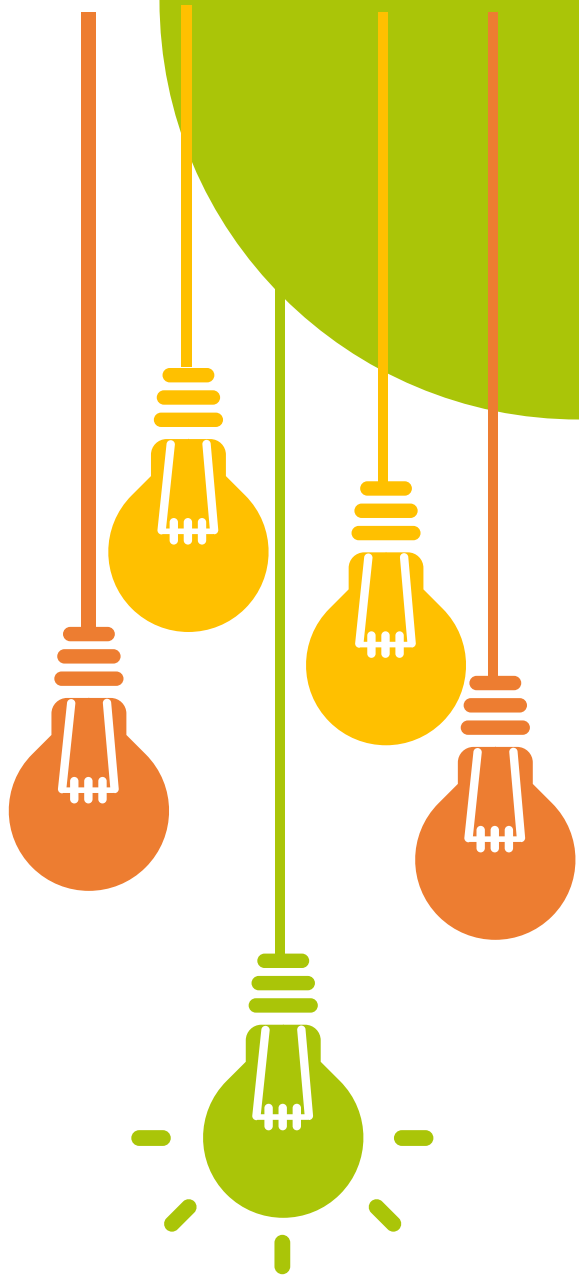
Pipeline of YOPO-m-n described in Algorithm 1. The yellow and olive blocks represent feature maps while the orange blocks represent the gradients of the loss w.r.t. feature maps of each layer.

Code

Training against PGD-40 is a common practice ,our results on MNIST. The network architectures: four convolutional layers followed by three fully connected layers

set the size of perturbation 0:3 in an infinite norm sense. Experiments

are taken on google colab GPUs. The model for 40 epochs with a batch size of 256, longer than what convergence needs for both training methods. The learning rate is set to 0.1 initially, and is lowered by 10 times at epoch 45.



Skoltech

Results

Training method	Clean Data	Adversarial attack	Training time
PGD40	98.83%	92.96%	**122 min**
YOPO_5_10	99.24%	92.41%	**58min**

Skoltech

Conclusion

YOPO is considered 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 algorithm avoids computing full forward and backward propagation for too many times, thus effectively reducing the computational time as supported by the experiment.

Skoltech

thx.

Skoltech