ℓ_∞-Robustness and Beyond: Unleashing Efficient Adversarial Training

Hadi M. Dolatabadi, Sarah Erfani, and Christopher Leckie
School of Computing and Information Systems, The University of Melbourne

ECCV TEL AVIV 2022

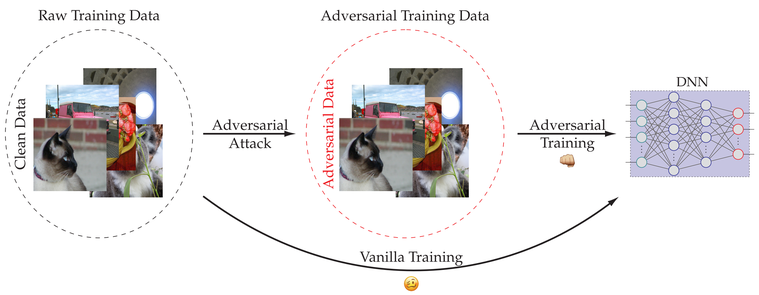
The University of Melbourne

Abstract

- **Motivation**: adversarial training (AT) is one of the most successful approaches to defend deep neural networks (DNN) against adversarial attacks. Unfortunately, AT is time-consuming as we need to generate adversarial attacks for the entire training data in each iteration.
- **Proposal**: we propose *adversarial coreset selection* to decrease the training data population and run AT on this smaller subset of data.
- **Key Features of Adversarial Coreset Selection**:
  1. A principled method that can be used along various AT objectives.
  2. Compatible with existing efficient AT methods.
  3. Speeding up AT by 2-3 times while experiencing a slight reduction in the clean and robust accuracy.

Background: Adversarial Training

- In contrast to vanilla training that uses the raw data samples to train a DNN, we use adversarial examples to perform adversarial training:



- We can have different AT methods depending on how adversarial examples are generated. Since the network state changes during the training, each adversarial example needs to be generated using the DNN at that particular step. As such, one of the main disadvantages of AT is its speed.

Background: Coreset Selection

- Coreset selection aims at finding a *weighted subset* of the data that can approximate certain behaviors of the entire data samples.
- In particular, let us denote the behavior of interest as a function $\mathcal{B}(\cdot,\cdot)$ that receives a set and its associated weights.
- The goal of coreset selection is to move from the original data $\mathcal{V}$ with uniform wights $\mathbf{1}$ to a weighted subset $\mathcal{S}^* \subseteq \mathcal{V}$ with weights γ^* such that:

$$\mathcal{B}(\mathcal{V},\mathbf{1}) \approx \mathcal{B}(\mathcal{S}^*,\gamma^*).$$

Our Method: Adversarial Coreset Selection

- To train a DNN $f_{\boldsymbol{\theta}}$ over the dataset $\mathcal{V}$, we first define an objective:

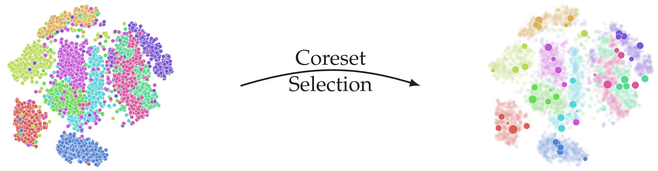
$$\mathcal{L}(\boldsymbol{\theta}) \coloneqq \sum_{i\in\mathcal{V}} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right).$$

- Different choices of $\boldsymbol{\Phi}$ amount to different training objectives:
  1. **Vanilla Training**: $\boldsymbol{\Phi} \coloneqq \mathcal{L}_{\text{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right)$
  2. **Adversarial Training**: $\boldsymbol{\Phi} \coloneqq \max_{\tilde{\boldsymbol{x}}:\mathrm{d}(\tilde{\boldsymbol{x}},\boldsymbol{x})\leq\varepsilon} \mathcal{L}_{\text{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), y\right)$
- Since the loss gradient determines the learning procedure, if we can find a good approximation to the gradient using a subset of the entire data, we can replace the training data with that subset.
- In particular, we set our behavior of interest $\mathcal{B}(\mathcal{A},\boldsymbol{w})$ to:

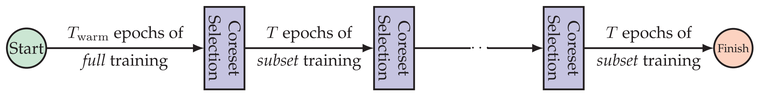
$$\mathcal{B}(\mathcal{A},\boldsymbol{w}) \coloneqq \sum_{i\in\mathcal{A}} w_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right).$$

- We want to find a coreset $\mathcal{S}^* \subseteq \mathcal{V}$ with weights γ^* such that:

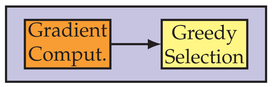
$$\mathcal{B}(\mathcal{V},\mathbf{1}) \approx \mathcal{B}(\mathcal{S}^*,\gamma^*)$$



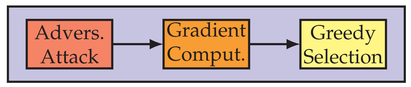
- To this end, we first take the DNN gradient and then use existing greedy selection algorithms to find the coreset $\mathcal{S}^*$.
- For vanilla training, the first step is equal to finding the DNN's gradient. For adversarial training, however, we resort to Danskin's theorem and take the gradient of a maximization objective, which requires finding an adversarial example and then computing the gradient for this point.
- Our final approach:



(a) Selection is done every T epochs. During the next episodes, the network is only trained on this subset.



(b) Coreset selection module for vanilla training.



(c) Coreset selection module for adversarial training.

Overview of DNN training using coreset selection.

Experimental Results

1. **Accelerating Adversarial Training**:

Objec.	Data	Training Scheme	↑ Clean Acc. (%)	↑ Robust Acc. (%)	↓ Time (mins)
TRADES	CIFAR-10	Adv. CRAIG (Ours)	83.03 (−2.38)	41.45 (−2.74)	179.20 (−165.09)
		Adv. GRADMATCH (Ours)	83.07 (−2.34)	41.52 (−2.67)	178.73 (−165.56)
		Full Adv. Training	85.41	44.19	344.29
ℓ_∞-PGD	CIFAR-10	Adv. CRAIG (Ours)	80.37 (−2.77)	45.07 (+3.68)	148.01 (−144.86)
		Adv. GRADMATCH (Ours)	80.67 (−2.47)	45.23 (+3.84)	148.03 (−144.84)
		Full Adv. Training	83.14	41.39	292.87

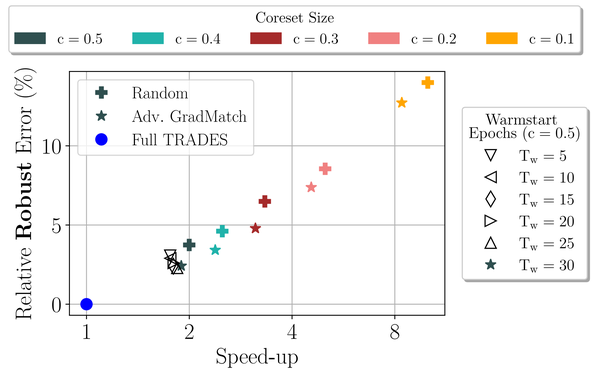
- **Takeaway 1**: Our approach results in 2-3 times faster training while decreasing the clean and robust accuracy slightly.
- **Takeaway 2**: Our approach is compatible with different objectives and greedy coreset selection algorithms.

2. **Fast Adversarial Training using ACS**:

Training Scheme	↑ Clean Acc. (%)	↑ Robust Acc. (%)	↓ Speed (min/epoch)
Fast Adv. Training	86.20	47.54	0.5178
+ Adv. GRADMATCH (Ours)	82.53 (−3.67)	47.88 (+0.34)	0.2737

- **Takeaway 3**: Our approach is complementary to existing methods that accelerate adversarial training.

3. **Ablation Study (Robust Error vs. Speed-up)**:



- **Takeaway 4**: Our method performs better than a random selection of data!

Code and Contact Information

Twitter hmdolatabadi
Website hmdolatabadi.github.io
Repo. github.com/hmdolatabadi/ACS