Fast is better than free: Revisiting adversarial training

By Daniel Silva, Blake Wyatt,
and Kesar Murthy

About the Paper

- Fast is better than free: Revisiting adversarial training
- Authors
    - Eric Wong
    - Leslie Rice
    - J. Zico Kolter
- International Conference on Learning Representations (ICLR) 2020
- 135 citations
- https://arxiv.org/abs/2001.03994
- https://github.com/locuslab/fast_adversarial

Outline

- Abstract
- Introduction
- Background and related work
  - Fast Gradient Sign Method (FGSM)
  - Projected Gradient Descent (PGD)
  - Adversarial Training
  - PGD Adversarial Training
  - Free Adversarial Training
- Fast Adversarial Training with FGSM
  - Algorithmic Comparison
  - Experiments
  - Results and comparison to PGD-based training
- Conclusion
  - Arguments for and against

Abstract

- Adversarial training comes with a large time and computational overhead
  - Strongest attacks require multiple forward passes and gradient computations
  - PGD, DeepFool, etc.
- Has slowed progress in neural network robustness research

Can we leverage one-step methods (such as FGSM) to the same effectiveness as multi-step for adversarial training?

Introduction

- FGSM adversarial training, combined with random initialization, can be just as effective as PGD-based training
  - Significantly more efficient than multi-step methods
  - Training time nearly equal to standard training
  - Previously thought to be ineffective
- Authors adopt general training techniques to further improve training time
  - Cyclic learning rate
  - Mixed-precision training
- Produces robust networks in state-of-the-art time
  - 45% adversarial accuracy on CIFAR10 in 6 training minutes (previous best of 10 hours)
  - 43% accuracy on ImageNet in 12 hours (previous best of 50)

Fast Gradient Sign Method (FGSM)

- Adversarial attack algorithm that requires a single gradient computation
  - White-box (needs model weights) untargeted attack
- R-FGSM (Tramer et al. 2017)
  - Enhance FGSM performance by randomly initializing perturbation, rather than 0-initialization
  - Tramer initializes with a non-uniform noise
- Early attempts at FGSM-based adversarial training did not succeed
  - Authors argue this is due to specific implementation details of the initialization or complete lack of random initialization

$$\delta^\star = \epsilon \cdot \mathrm{sign}(\nabla_x \ell(f(x), y)) + R$$

Projected Gradient Descent (PGD)

- Extends previous one-step methods with a more powerful multi-step gradient-based attack
  - Equivalent to Iterative FGSM (I-FGSM)
  - White box, capable of both targeted and untargeted attacks
  - Increase in time complexity by a factor of N (number of iterations)
- Many gradient steps produces significantly stronger perturbations
  - Has seen great success for adversarial training
  - Downside is the runtime and compute costs of multiple gradient calculation iterations

$\delta = 0$ // *or randomly initialized*
for $j = 1 \ldots N$ **do**
  $\delta = \delta + \alpha \cdot \mathrm{sign}(\nabla_\delta \ell(f_\theta(x_i + \delta), y_i))$
  $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
end for

Adversarial Training

- Simply involves training a network on adversarially perturbed images
  - Perturbations δ calculated in real time for the network
  - Perturbations are applied to training data images to create adversarial examples
  - The adversarial example losses are used to make the network more robust
- Calculating perturbations δ efficiently is important
- Methods frequently based on:
  - Projected Gradient Descent (PGD): common method and one of the best
  - Fast Gradient Sign Method (FGSM): prior work indicated it was not as effective until "Free" adversarial training

$$\min_{\theta} \sum_{i} \max_{\delta \in \Delta} \ell(f_{\theta}(x_i + \delta), y_i)$$

$$\Delta = \{\delta : \|\delta\|_{\infty} \leq \epsilon\}$$

PGD Adversarial Training

- PGD is used to calculate perturbations δ
- Gradient computations performed are proportional to O(M N) each epoch
- Standard training gradient computations are proportional to O(M) each epoch

Thus, PGD adversarial training is much more expensive than standard training

Free Adversarial Training

- FGSM-based perturbation calculations
- Gradient computations are proportional to O(M) each epoch (comparable to standard training!)
- Minibatch replays: model weights are updated alongside FGSM attack

Parameters:

- Step sizes where $\alpha = \epsilon$
- Epochs T divided by N

Fast Adversarial Training

- FGSM-based perturbation calculations
- Random initialization of perturbations (Tramer et al, 2017)
  - Uniform distribution used in this work proves more effective
- Empirical evidence indicates Fast has comparable performance to that of PGD
- FGSM step size
  - $\alpha = \epsilon$ and zero-initialization is too weak. It is not guaranteed to lie on the LInf ball boundary
  - $\alpha = 2\epsilon$: Leads to catastrophic overfitting (covered later)
  - Increasing step size by a factor of 1.25 instead, improves model robustness
- O(M) gradient computations

Adversarial Training Comparison

Standard PGD adversarial training

for $t = 1 \ldots T$ **do**
  for $i = 1 \ldots M$ **do**
    // Perform PGD adversarial attack
    $\delta = 0$ *// or randomly initialized*
    for $j = 1 \ldots N$ **do**
      $\delta = \delta + \alpha \cdot \text{sign}(\nabla_\delta \ell(f_\theta(x_i + \delta), y_i))$
      $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
    end for
    $\theta = \theta - \nabla_\theta \ell(f_\theta(x_i + \delta), y_i)$ *// Update model weights*
  end for
end for

Fast FGSM adversarial training

for $t = 1 \ldots T$ **do**
  for $i = 1 \ldots M$ **do**
    // Perform FGSM adversarial attack
    $\delta = \text{Uniform}(-\epsilon, \epsilon)$
    $\delta = \delta + \alpha \cdot \text{sign}(\nabla_\delta \ell(f_\theta(x_i + \delta), y_i))$
    $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
    $\theta = \theta - \nabla_\theta \ell(f_\theta(x_i + \delta), y_i)$ *// Update model weights*
  end for
end for

No inner loop required

"Free" FGSM adversarial training

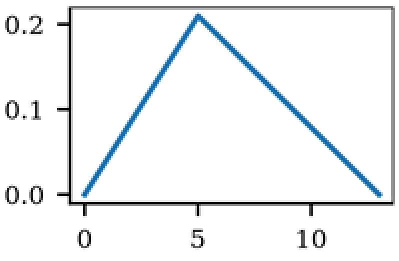
$\delta = 0$
// Iterate T/N times to account for minibatch replays and run for T total epochs
for $t = 1 \ldots T/N$ **do**
  for $i = 1 \ldots M$ **do**
    // Perform simultaneous FGSM adversarial attack and model weight updates T times
    for $j = 1 \ldots N$ **do**
      // Compute gradients for perturbation and model weights simultaneously
      $\nabla_\delta, \nabla_\theta = \nabla \ell(f_\theta(x_i + \delta), y_i)$
      $\delta = \delta + \epsilon \cdot \text{sign}(\nabla_\delta)$
      $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
      $\theta = \theta - \nabla_\theta$ *// Update model weights with some optimizer, e.g. SGD*
    end for
  end for
end for

Inner loop cancelled out by T/N

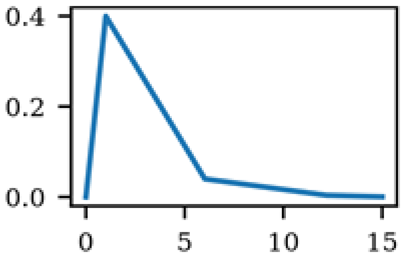
Fast Adversarial Training (DAWNBench)

- Authors also used standard techniques for improving training time:
  - Cyclic learning rate (Smith & Topin, 2018)
    - Schedules a learning rate to change linearly between 0 and a max rate
  - Mixed-precision arithmetic
    - Leverage GPU tensor core half-precision computation capability
    - Reduces memory utilization and runtime

Cyclic learning rate schedules



(a) CIFAR10



(b) ImageNet

Experiments

- Adversarial training experiments were conducted on MNIST, CIFAR10, and ImageNet
  - CIFAR10 - ResNet18
  - ImageNet - ResNet50
- FGSM training
  - Random initialization, cyclic LR, mixed precision with Apex amp package
  - Step size = 1.25 * *epsilon*
- Adversaries for testing generated with PGD (Shafahi et al. (2019))
  - 50 iterations with 10 random restarts, step size 2/255

MNIST Results

- Small CNN with 16 and 32 kernels and a 100-unit MLLP (Tjeng et al. 2017)
- Adversarial training performed with both PGD and FGSM
  - with *epsilon* = 0.1, 0.3, 40 PGD iterations, *step size* = 0.01 (Madry et al. 2017)
- FGSM results in equal robustness to PGD
  - Exact (verified) robustness of model calculated using mixed-integer linear programming

Method	Standard accuracy	PGD ($\epsilon = 0.1$)	PGD ($\epsilon = 0.3$)	Verified ($\epsilon = 0.1$)
PGD	99.20%	97.66%	89.90%	96.7%
FGSM	99.20%	97.53%	88.77%	96.8%

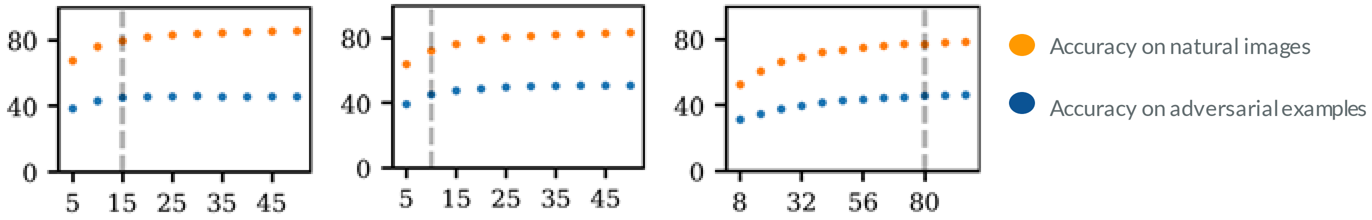
CIFAR10 Results

- Compare various adversarial training approaches on the CIFAR10 dataset
    - DAWNBench refers to SOTA training techniques: cyclic LR & mixed precision
- FGSM with random initialization results in high robustness and lowest time
    - "Free" adversarial training just as robust, but not as fast
    - PGD still produces most robust model - 50% accuracy vs 46% (R-FGSM)

Method	Standard accuracy	PGD ($\epsilon = 8/255$)	Time (min)
FGSM + DAWNBench			
+ zero init	85.18%	0.00%	12.37
+ early stopping	71.14%	38.86%	7.89
+ previous init	86.02%	42.37%	12.21
+ random init	85.32%	44.01%	12.33
+ $\alpha = 10/255$ step size	83.81%	46.06%	12.17
+ $\alpha = 16/255$ step size	86.05%	0.00%	12.06
+ early stopping	70.93%	40.38%	8.81
"Free" ($m = 8$) (Shafahi et al., 2019)[1]	85.96%	46.33%	785
+ DAWNBench	78.38%	46.18%	20.91
PGD-7 (Madry et al., 2017)[2]	87.30%	45.80%	4965.71
+ DAWNBench	82.46%	50.69%	68.8

CIFAR10 Results

Accuracy vs epochs of adversarial training required to reach 45% robust accuracy



(a) FGSM (b) PGD (c) "Free" ($m = 8$)

FGSM results in lowest total time

Method	Epochs	Seconds/epoch	Total time (minutes)
DAWNBench + PGD-7	10	104.94	17.49
DAWNBench + Free ($m = 8$)	80	13.08	17.44
DAWNBench + FGSM	15	25.36	6.34
PGD-7 (Madry et al., 2017)[5]	205	1456.22	4965.71
Free ($m = 8$) (Shafahi et al., 2019)[6]	205	197.77	674.39

ImageNet Results

Strongest FGSM training procedure compared to free adversarial training

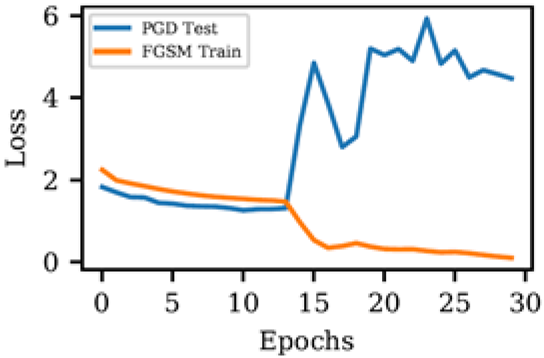
- FGSM significantly faster
- "Free" has improved std. accuracy
  - First time seeing this trend

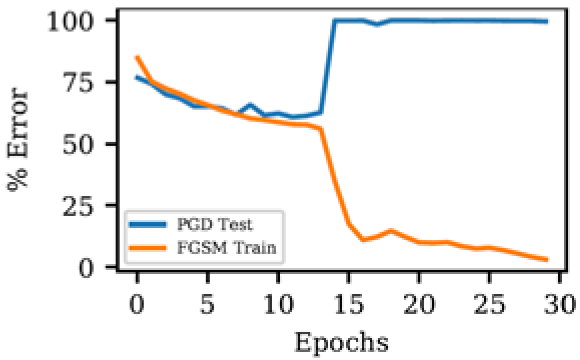
Method	Precision	Epochs	Min/epoch	Total time (hrs)
FGSM (phase 1)	single	6	22.65	2.27
FGSM (phase 2)	single	6	65.97	6.60
FGSM (phase 3)	single	3	114.45	5.72
FGSM	single	15	-	14.59
Free ($m = 4$)	single	92	34.04	52.20
FGSM (phase 1)	mixed	6	20.07	2.01
FGSM (phase 2)	mixed	6	53.39	5.34
FGSM (phase 3)	mixed	3	95.93	4.80
FGSM	mixed	15	-	12.14
Free ($m = 4$)	mixed	92	25.28	38.76

Method	ϵ	Standard acc.	PGD+1 restart	PGD+10 restarts	Total time (hrs)
FGSM	2/255	60.90%	43.46%	43.43%	12.14
Free ($m = 4$)	2/255	64.37%	43.31%	43.28%	52.20
FGSM	4/255	55.45%	30.28%	30.18%	12.14
Free ($m = 4$)	4/255	60.42%	31.22%	31.08%	52.20

Catastrophic Overfitting

- Bad design decisions which cause FGSM adversarial training to fail (0% racc)
- Common causes:
  - Zero initialization
  - Too large of a step size
  - Certain learning rate schedules
  - Certain numbers of epochs
- Early stopping saves the model from catastrophic overfitting before it happens





Conclusion

- FGSM with random initialization provides an efficient yet powerful approach to adversarial training
- Takeaways
  - Adversarial examples during training **need** to span the entire threat model
    - Lack of random initialization may have caused FGSM's weak performance thus far
  - Defenders don’t need strong adversaries during training
    - This work shows that rough approximations (FGSM) to inner optimization are sufficient
  - Standard training improvement strategies still work for adversarial training
    - Cyclic LR and mixed precision

Argument For

- Allows adversarial training to be just as fast as standard training
- Simple to implement and leverage
- Extensive experimentation and comparison to other approaches
  - Results are extremely strong
  - Vast improvement over previous baselines for adversarial training
- Discovered and analyzed “Catastrophic Overfitting”

Argument Against

- Not incredibly novel
  - Contribution and results are purely empirical
  - Random initialization previously introduced by R-FGSM (Tramer et al. (2017))
  - Cyclic learning rate and mixed precision already shown success for standard training
- R-FGSM and FGSM previously used for adversarial training
  - Ensemble Adversarial Training: Attacks and Defenses
    - https://arxiv.org/abs/1705.07204
  - Defensive Quantization: When Efficiency Meets Robustness
    - https://arxiv.org/abs/1904.08444
  - Better Generalization with Adaptive Adversarial Training
    - https://openreview.net/pdf?id=B1goj125pN
- Interested to see how this compares against more recent attacks
  - Ex: Do results hold vs skip connection based attacks?

Thank you