Amata: An Annealing Mechanism for Adversarial Training Acceleration

Nanyang Ye,[1] **Qianxiao Li ,**[2, 5] **Xiao-Yun Zhou,** [3] **Zhanxing Zhu** * [4]

[1] Shanghai Jiao Tong University, Shanghai, China
[2] National University of Singapore, Singapore
[3] The Hamlyn Centre for Robotic Surgery, Imperial College, London, United Kingdom
[4] Peking University, Beijing, China
[5] Institute of High Performance Computing, A*STAR, Singapore
ynylincoln@sjtu.edu.cn, qianxiao@nus.edu.sg, xiaoyun.zhou27@gmail.com, zhanxing.zhu@pku.edu.cn.

Abstract

Despite the empirical success in various domains, it has been revealed that deep neural networks are vulnerable to maliciously perturbed input data that much degrade their performance. This is known as adversarial attacks. To counter adversarial attacks, adversarial training formulated as a form of robust optimization has been demonstrated to be effective. However, conducting adversarial training brings much computational overhead compared with standard training. In order to reduce the computational cost, we propose an annealing mechanism, Amata, to reduce the overhead associated with adversarial training. The proposed Amata is provably convergent, well-motivated from the lens of optimal control theory and can be combined with existing acceleration methods to further enhance performance. It is demonstrated that on standard datasets, Amata can achieve similar or better robustness with around 1/3 to 1/2 the computational time compared with traditional methods. In addition, Amata can be incorporated into other adversarial training acceleration algorithms (e.g. YOPO, Free, Fast, and ATTA), which leads to further reduction in computational time on large-scale problems.

Introduction

Deep neural networks were found to be vulnerable to malicious perturbations on the original input data. While the perturbations remain almost imperceptible to humans, they can lead to wrong predictions over the perturbed examples (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014; Akhtar and Mian 2018). These maliciously crafted examples are known as adversarial examples, which have caused serious concerns over the reliability and security of deep learning systems, particularly when deployed in life-critical scenarios, such as autonomous driving systems and medical domains.

Several defense mechanisms have been proposed, such as input reconstruction (Meng and Chen 2017; Song et al. 2018), input encoding (Buckman et al. 2018), and adversarial training (Goodfellow, Shlens, and Szegedy 2014; Tramèr et al. 2017; He et al. 2017; Madry et al. 2017). Among these methods, adversarial training is one of the most effective defense methods so far. It can be posed as a robust optimization problem (Ben-Tal and Nemirovski 1998), where a min-max optimization problem is solved (Madry et al. 2017; Kolter and Wong 2017). For example, given a C-class dataset $S = \{(\mathbf{x}_i^0, y_i)\}_{i=1}^n$ with $\mathbf{x}_i^0 \in \mathcal{R}^d$ as a normal or clean example in the d-dimensional input space and $y_i \in \mathcal{R}^C$ as its associated one-hot label, the objective of adversarial training is to solve the following *min-max optimization* problem:

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \max_{\|\mathbf{x}_i - \mathbf{x}_i^0\| \leq \epsilon} \ell(h_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) \tag{1}$$

where $h_{\boldsymbol{\theta}} : \mathcal{R}^d \to \mathcal{R}^C$ is the deep neural network (DNN) function, ℓ is the loss function and ϵ controls the maximum perturbation magnitude. The *inner maximization* problem is to find an adversarial example $\mathbf{x}_i$, within the ϵ-ball around a given normal example $\mathbf{x}_i^0$ that maximizes the surrogate loss ℓ for classification error. The *outer minimization* problem is to find model parameters that minimizes the loss ℓ on the adversarial examples $\{\mathbf{x}_i\}_{i=1}^n$ that are generated from the inner maximization. Compared with a rich body of non-convex optimization algorithms for neural networks (Goodfellow, Bengio, and Courville 2016; Kingma and Ba 2014; Pan and Jiang 2015), designing efficient algorithm to solve the min-max problem to achieve robustness is relatively less studied.

The inner maximization problem is typically solved by projected gradient descent (PGD). PGD perturbs a normal example $\mathbf{x}^0$ by iteratively updating it in approximately the steepest ascent direction for a total of K times. Each ascent step is modulated by a small step size and a projection step back onto the ϵ-ball of $\mathbf{x}^0$ to prevent the updated value from falling outside the ϵ-ball of $\mathbf{x}^0$ (Madry et al. 2017):

$$\mathbf{x}^k = \prod \left(\mathbf{x}^{k-1} + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}} \ell(h_{\boldsymbol{\theta}}(\mathbf{x}^{k-1}), y)\right) \tag{2}$$

where α is the step size, $\prod(\cdot)$ is the orthogonal projection function onto $\{\mathbf{x}' : \|\mathbf{x}^0 - \mathbf{x}'\| \leq \epsilon\}$, and $\mathbf{x}^k$ is the adversarial example at k-th step [1].

A major issue limiting the practical applicability of adversarial training is the huge computational burden associated

*Corresponding author. The full appendix is available on Arxiv.

[1] Note that our methodology can also be applied to single step methods, such as Fast. We take PGD as the first example for clarity.

with the inner maximization steps: we need to iteratively solve the inner maximization problem to find good adversarial examples for DNN to be robust. Recently, to accelerate adversarial training, a few methods have been proposed. For example, YOPO estimated the gradient on the input by only propagating the first layer (Zhang et al. 2019), parallel adversarial training utilized multiple graphics processing units (GPUs) for acceleration (Bhat and Tsipras 2019) and running PGD-1 for multiple steps to reuse gradients (Shafahi et al. 2019).

Orthogonal to these approaches, in this paper we consider accelerating adversarial training by adjusting the number of inner maximization steps as training proceeds. This is in line with an empirical observation made by (Wang et al. 2019), indicating that we might not need to find "good" solutions to the inner maximization at the initial stages of adversarial training to achieve better robustness. Hence, by varying the extent the inner maximization problem is solved, we may reduce the amount of wasted computation. This forms the basis of our proposed Annealing Mechanism for Adversarial Training Acceleration, named as **Amata**. Compared with traditional methods, Amata takes 1/3 to 1/2 the time to achieve comparable or slightly better robustness. Moreover, the general applicability of annealing procedures allows for effective combination of Amata with existing acceleration approaches. On the theoretical side, Amata is shown to converge. Furthermore, as adaptive training algorithms can often be interpretted as optimal control problems (Li, Tai, and E 2017, 2019), we also develop a control theoretic viewpoint of general adversarial training. Under this framework, we can motivate the qualitative form of the Amata's annealing scheme based on loss landscapes. Furthermore, a new criterion based on the Pontryagin's maximum principle can also be derived to quantify the approximate optimality of annealing schedules.

In summary, our contributions are as follows:

1. We propose an adversarial training algorithm, Amata, based on annealing the inner maximization steps to reduce computation. The method is shown to be effective on benchmarks, including MNIST, CIFAR10, Caltech256, and the large-scale ImageNet dataset.

2. As Amata is largely orthogonal to existing acceleration methods for adversarial training, it can be easily combined with them to further decrease computation. The combination of Amata with YOPO (Zhang et al. 2019), with adversarial training for free (Shafahi et al. 2019), with fast adversarial training (Wong, Rice, and Kolter 2020), and with adversarial training with transferable adversarial examples (Zheng et al. 2020) is demonstrated.

3. On the theoretical side, we prove the convergence of Amata. Moreover, we develop a general optimal control framework for annealed adversarial training, from which we can use the optimal control to qualitatively and quantitatively justify the proposed annealing schedule. This framework is also potentially useful as a basic formulation for future work on adaptive adversarial training methods.

Accelerating Adversarial Training by Annealing

In this section, we first introduce the proposed Amata, which aims to balance the computational cost and the accuracy of solving the inner maximization problem. A proof of the convergence of the algorithm can be found in the Appendix. Moreover, we introduce an optimal control formulation of general annealed adversarial training, from which one can elucidate the motivation and working principles of Amata, both qualitatively and quantitatively.

Proposed Annealing Adversarial Training Algorithm

Algorithm 1 An instantatiation of Amata for PGD

Input: T: training epochs; $K_{\min}/K_{\max}$: the minimal/maximal number of adversarial perturbations; $\boldsymbol{\theta}$: parameter of neural network to be adversarially trained; $\mathcal{B}$: mini-batch; α: step size for adversarial training; η: learning rate of neural network parameters. τ: constant, maximum perturbation:ϵ.
Initialization $\boldsymbol{\theta} = \boldsymbol{\theta}_0$
for $t = 0$ to $T-1$ **do**
  Compute the annealing number of adversarial perturbations: $K_t = K_{\min} + (K_{\max} - K_{\min}) \cdot \frac{t}{T}$
  Compute adversarial perturbation step size: $\alpha_t = \frac{\tau}{K_t}$
  for each mini-batch $\mathbf{x}_{\mathcal{B}}^0$ **do**
    for $k = 1$ to K_t **do**
      Compute adversarial perturbations:

$$\mathbf{x}_{\mathcal{B}}^k = \mathbf{x}_{\mathcal{B}}^{k-1} + \alpha_t \cdot \text{sign}(\nabla_{\mathbf{x}} \ell(h_{\boldsymbol{\theta}}(\mathbf{x}_{\mathcal{B}}^k), y),$$
$$\mathbf{x}_{\mathcal{B}}^k = \text{clip}(\mathbf{x}_{\mathcal{B}}^k, \mathbf{x}_{\mathcal{B}}^0 - \epsilon, \mathbf{x}_{\mathcal{B}}^0 + \epsilon)$$

    end for
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \ell(h_{\boldsymbol{\theta}_t}(\mathbf{x}_{\mathcal{B}}^{K_t}), y)$
  end for
end for
Collect $\boldsymbol{\theta}_T$ as the parameter of adversarially-trained neural network.

To set the stage we first summarize the proposed algorithm (Amata) in Algorithm 1. The intuition behind Amata is that, at the initial stage, the neural network focuses on learning features, which might not require very accurate adversarial examples. Therefore, we only need a coarse approximation of the inner maximization problem solutions. With this consideration, a small number of update steps K with a large step size α is used for inner maximization at the beginning, and then gradually increase K and decrease α to improve the quality of inner maximization solutions. This adaptive annealing mechanism would reduce the computational cost in the early iterations while still maintaining reasonable accuracy for the entire optimization. *Note that this algorithm is only an instantiation of this mechanism on PGD and the mechanism can also be seamlessly incorporated into other acceleration algorithms.* This will be demonstrated later[2].

[2]Amata can be also applied to other algorithms, such as YOPO,

Next, we will show the sketch for proving the convergence of the algorithm with details in the Appendix. We denote $\mathbf{x}_i^*(\boldsymbol{\theta}) = \arg\max_{\mathbf{x}_i \in \mathcal{X}^i} \ell(\boldsymbol{\theta}, \mathbf{x}_i)$ where $\ell(\boldsymbol{\theta}, \mathbf{x}_i)$ is a short hand notation for the classification loss function $\ell(h_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$, and $\mathcal{X}^i$ is the permitted perturbation range for $\mathbf{x}_i$. Before we prove the convergence of the algorithm, we have the following assumptions which are commonly used in literature for studying convergence of deep learning algorithms (Gao et al. 2019; Wang et al. 2019).

Assumption 1. *The function $\ell(\boldsymbol{\theta}, \mathbf{x})$ satisfies the gradient Lipschitz conditions:*

$$sup_{\mathbf{x}} \|\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \mathbf{x}) - \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}^*, \mathbf{x})\|_2 \leq L_{\boldsymbol{\theta}\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2$$
$$sup_{\boldsymbol{\theta}} \|\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \mathbf{x}) - \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \mathbf{x}^*)\|_2 \leq L_{\boldsymbol{\theta}\mathbf{x}} \|\mathbf{x} - \mathbf{x}^*\|_2$$
$$sup_{\mathbf{x}} \|\nabla_{\mathbf{x}} \ell(\boldsymbol{\theta}, \mathbf{x}) - \nabla_{\mathbf{x}} \ell(\boldsymbol{\theta}^*, \mathbf{x})\|_2 \leq L_{\mathbf{x}\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2$$

where $L_{\boldsymbol{\theta}\boldsymbol{\theta}}$, $L_{\boldsymbol{\theta}\mathbf{x}}$, and $L_{\mathbf{x}\boldsymbol{\theta}}$ are positive constants. Assumption 1 was used in (Sinha, Namkoong, and Duchi 2018; Wang et al. 2019).

Assumption 2. *The function $\ell(\boldsymbol{\theta}, \mathbf{x})$ is locally μ-strongly concave in $\mathcal{X} = \{\mathbf{x} : \left\|\mathbf{x} - \mathbf{x}_i^0\right\|_\infty \leq \epsilon\}$ for all $i \in [n]$, i.e., for any $\mathbf{x}_1$, $\mathbf{x}_2 \in \mathcal{X}_i$:*

$$\ell(\boldsymbol{\theta}, \mathbf{x}_1) \leq \ell(\boldsymbol{\theta}, \mathbf{x}_2) + \langle \nabla_{\mathbf{x}} \ell(\boldsymbol{\theta}, \mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2 \rangle - \frac{\mu}{2} \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2$$

where μ is a positive constant which measures the curvature of the loss function. This assumption was used for analyzing distributional robust optimization problems (Sinha, Namkoong, and Duchi 2018).

Assumption 3. *The variance of the stochastic gradient $g(\boldsymbol{\theta})$ is bounded by a constant $\sigma^2 > 0$:*

$$\mathbb{E}[\|g(\boldsymbol{\theta}) - \nabla L(\boldsymbol{\theta})\|_2^2] \leq \sigma^2$$

where $\nabla L(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \left(\sum_{i=1}^N \ell(\boldsymbol{\theta}, \mathbf{x}_i^*) \right)$ is the full gradient.

The Assumption 3 is commonly used for analyzing stochastic gradient optimization algorithms.

We denote the objective function in Equation 1 as $L(\boldsymbol{\theta})$, its gradient by $\nabla L(\boldsymbol{\theta})$, the optimality gap between the initial neural network parameters and the optimal neural network parameters $\Delta = L(\boldsymbol{\theta}_0) - \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$, the maximum distance between the output adversarial example generated by Amata and the original example as δ, and T as the number of iterations. Then, we have the following theorem for convergence of the algorithm:

Theorem 1 (Convergence of Amata)**.** *, If the step size of outer minimization is $\eta_t = \min(1/\beta, \sqrt{\frac{\Delta}{TL\sigma^2}})$. Then, after T iterations, we have:*

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla L(\boldsymbol{\theta}_t)\|_2^2] \leq 4\sigma \sqrt{\frac{\beta \Delta}{T}} + 5 L_{\boldsymbol{\theta}\mathbf{x}}^2 \delta^2$$

where σ is the bound for variance between the batch gradient and the stochastic gradient and $\beta = L_{\boldsymbol{\theta}\mathbf{x}} L_{\mathbf{x}\boldsymbol{\theta}} / \mu + L_{\boldsymbol{\theta}\boldsymbol{\theta}}$ is a constant.

Free, and Fast, we use PGD as an example for clarity.

The detailed notations, assumptions and proof are shown in the Appendix due to limited space. This theorem proves that our algorithm can converge under a suitable selection of learning rate.

The remainder of this section serves to motivate and justify, both qualitatively and quantitatively, the annealing method in Amata from an optimal control viewpoint. We will try to answer this question in the following sections.

How good is the annealing scheduling in terms of adversarial training acceleration?

We start with a general formulation of annealed adversarial training as an optimal control problem.

Optimal Control Formulation of Annealed Adversarial Training

In essence, the PGD-based adversarial training algorithm (Madry et al. 2017) is a result of a number of relaxations of the original min-max problem in Eq. (1), which we will now describe. For simplicity of presentation, let us consider just one fixed input-label pair $(\mathbf{x}^0, y)$, since the N-sample case is similar. The original min-max adversarial training problem is given in (1) with $N = 1$. The first relaxation is to replace the outer minimization with gradient descent so that we obtain the iteration

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \max_{\{\mathbf{x}: \|\mathbf{x} - \mathbf{x}^0\| \leq \epsilon\}} \ell(h_{\boldsymbol{\theta}_t}(\mathbf{x}), y). \tag{3}$$

Then, the remaining maximization in each outer iteration step is replaced by an abstract algorithm $\mathcal{A}_{\mathbf{u},\boldsymbol{\theta}} : \mathcal{R}^d \to \mathcal{R}^d$ which solves the inner maximization approximately. Here, we assume that the algorithm depends on the current parameters of our neural network $\boldsymbol{\theta}$, as well as hyper-parameters $\mathbf{u}$ which takes values in a closed subset G of an Euclidean space. No further assumptions are placed on G, which may be a continuum, a countable set, or even a finite set.

This relaxation leads to the following iterations[3]

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \ell(h_{\boldsymbol{\theta}_t}(\mathcal{A}_{\boldsymbol{\theta}_t, \mathbf{u}_t}), y). \tag{4}$$

Eq. (4) represents a general formulation of annealed adversarial, of which Algorithm 1 is an example with $\mathcal{A}_{\theta_t, u_t}$ being the inner PGD loop and $\mathbf{u}_t = \{\alpha_t, K_t\}$ are the hyper-parameters we pick at each t step. The function $t \mapsto \mathbf{u}_t$ is an *annealing schedule* for the hyper-parameters. How to pick an optimal schedule can be phrased as an optimal control problem.

To make analysis simple, we will take a continuum approximation assuming that the outer loop learning rate η is small. This allows us to replace (4) by an ordinary differential equation or gradient flow with the identification $s \approx t\eta$:

$$\dot{\boldsymbol{\theta}}_s = -\nabla_{\boldsymbol{\theta}} \ell(h_{\boldsymbol{\theta}_s}(\mathcal{A}_{\boldsymbol{\theta}_s, \mathbf{u}_s}), y). \tag{5}$$

Here, the time s is a continuum idealization of the outer loop iterations on the trainable parameters in the model. We consider two objectives in designing the annealing algorithm:

[3]Here we assume that the gradient with respect to θ is the partial derivative with respect to the parameters of the network h_θ and θ_t in $\mathcal{A}_{\theta_t, \mathbf{u}_t}$ is held constant. This is the case for the PGD algorithm. Alternatively, we can also take the total derivative, but this leads to different algorithms.

on a training interval $[T_1, T_2]$ in the outer loop, we want to minimize the loss under adversarial training measured by a real-valued function $\Phi(\boldsymbol{\theta})$ while also minimizing the training cost associated with each inner algorithm loop under the hyper-parameter $\mathbf{u}$, which is measured by another real-valued function $R(\mathbf{u})$. An optimal annealing algorithm can then be defined as a solution to the following problem:

$$\begin{aligned} &\min_{\mathbf{u}_{T_1:T_2}} \Phi(\boldsymbol{\theta}_T) + \int_{T_1}^{T_2} R(\mathbf{u}_s) ds \\ &\text{subject to:} \quad \dot{\boldsymbol{\theta}}_t = F(\boldsymbol{\theta}_s, \mathbf{u}_s) \\ &\quad \text{and} \quad F(\boldsymbol{\theta}_s, \mathbf{u}_s) := -\nabla_{\boldsymbol{\theta}} \ell(h_{\boldsymbol{\theta}_s}(\mathcal{A}_{\boldsymbol{\theta}_s, \mathbf{u}_s}), y), \end{aligned} \tag{6}$$

where we have defined the shorthand $\mathbf{u}_{T_1:T_2} = \{\mathbf{u}_s : s \in [T_1, T_2]\}$. In this paper, we take Φ to be the DNN's prediction loss given an adversarial example (adversarial robustness), and set $R(\mathbf{u}_s) = \gamma K_s$, where K_s is the number of inner PGD steps at outer iteration number s, and γ is the coefficient for trade-off between adversarial robustness and training time. This is to account for the fact that when K_s increases, the cost of the inner loop training increases accordingly. The integral over s of R is taken so as to account for the total computational cost corresponding to a choice of hyper-parameters $\{\mathbf{u}_s\}$. The objective function taken as a sum serves to balance the adversarial robustness and computational cost, with γ as a balancing coefficient.

Problem (6) belongs to the class of Bolza problems in optimal control, and its necessary and sufficient conditions for optimality are well-studied. In this paper, we will use a necessary condition, namely the Pontryagin's maximum principle, in order to motivate our annealing algorithm and derive a criterion to test its approximate optimality. For more background on the theory of calculus of variations and optimal control, we refer the reader to (Boltyanskii, Gamkrelidze, and Pontryagin 1960; Bertsekas et al. 1995).

Theorem 2 (Pontryagin's Maximum Principle (PMP))**.** *Let* $\mathbf{u}^*_{T_1:T_2}$ *be a solution to* (6)*. Suppose* $F(\boldsymbol{\theta}, \mathbf{u})$ *is Lipschitz in* $\boldsymbol{\theta}$ *and measurable in* $\mathbf{u}$*. Define the Hamiltonian function*

$$H(\boldsymbol{\theta}, \mathbf{p}, \mathbf{u}) = \mathbf{p}^\top F(\boldsymbol{\theta}, \mathbf{u}) - R(\mathbf{u}) \tag{7}$$

Then, there exists an absolutely continuous co-state process $\mathbf{p}^*_{T_1:T_2}$ *such that*

$$\dot{\boldsymbol{\theta}}^*_s = F(\boldsymbol{\theta}^*_s, \mathbf{u}^*_s) \qquad \boldsymbol{\theta}^*_{T_1} = \boldsymbol{\theta}_{T_1} \tag{8}$$

$$\dot{\mathbf{p}}^*_s = -\nabla_{\boldsymbol{\theta}} H(\boldsymbol{\theta}^*_s, \mathbf{p}^*_s, \mathbf{u}^*_s) \qquad \mathbf{p}^*_{T_2} = -\nabla_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}^*_{T_2}) \tag{9}$$

$$H(\boldsymbol{\theta}^*_s, \mathbf{p}^*_s, \mathbf{u}^*_s) \geq H(\boldsymbol{\theta}^*_s, \mathbf{p}^*_s, \mathbf{v}) \quad \forall \mathbf{v} \in G, s \in [T_1, T_2] \tag{10}$$

In short, the maximum principle says that a set of optimal hyper-parameter choices $\{\mathbf{u}^*_s\}$ (in our specific application, these are the optimal choices of inner-loop hyper-parameters as the training proceeds) must globally maximize the Hamiltonian defined above for *each* outer iteration. The value of the Hamiltonian at each layer depends in turn on coupled ODEs involving the states and co-states. This statement is especially appealing for our application because unlike first-order gradient conditions, the PMP holds even when our hyper-parameters can only take a discrete set of values, or when there are non-trivial constraints amongst them.

We now show how the optimal control formulation and the PMP motivates and justifies the proposed Amata both qualitatively and quantitatively.

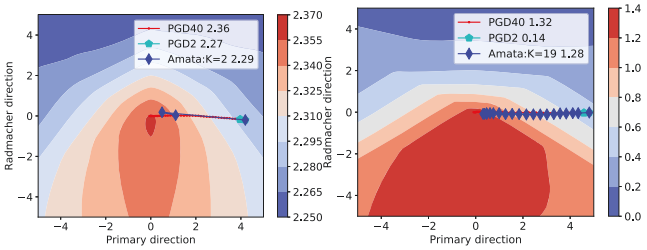

Figure 1: (Better viewed in the zoom-in mode) Visualization of inner maximizations landscape and PGD-40, PGD-2, and Amata's trajectories at Epoch 1 (Left) and Epoch 10 (Right). K is the Amata's number of steps and the numerics indicate the adversarial loss obtained by different methods (the higher the better for adversarial perturbations). Applying standard PGD-2 for acceleration cannot find strong adversarial example at Epoch 10. However, Amata can adaptively balance the number of steps and the step size to achieve good trade-off between time costs and robustness as justified before by theoretic analysis. More details are in the Appendix.

Qualitative Justification of Amata via Landscape Analysis

We now show that the form of an effective annealing schedule depends on how the loss landscape with respect to the data (i.e. landscape for the inner loop maximization) changes as training proceeds. Let us first visualize the change in landscape for actual neural network training using a similar method as (Shafahi et al. 2019). In Figure 1, we observe that as the outer training proceeds, the loss landscape with respect to the data becomes steeper, and thus we require smaller step sizes and/or more steps in the inner maximization to find good quality adversarial examples.

Let us now make this observation more precise using the maximum principle. In general, it is not possible to solve the PMP Eq. (8-10) in closed form. However, we can consider representative loss landscapes qualitatively similar to Fig. 1 where such equations are solvable in order to motivate our proposed algorithm. To this end, consider a 1D optimization problem with loss function

$$\ell(\theta, x) = \frac{\theta^2}{2} - \frac{(x-\theta)^2}{\theta^2 + 1}, \tag{11}$$

where x plays the role of data and θ plays the role of trainable parameters. We will assume that the data point $x = 0$ so that the non-robust loss is $l(\theta, 0) = \frac{\theta^2}{2} + \frac{1}{\theta^2+1} - 1$, which has two minima at $\theta = \pm\sqrt{\sqrt{2} - 1}$. However, the robust loss is $\tilde{\ell}(\theta) = \max_{x \in \mathcal{R}} l(\theta, x) = \frac{1}{2}\theta^2$, which has a unique minimum at $\theta = 0$. The key motivation behind this example is the fact that the loss landscape for x becomes more steep (i.e. $|d\ell/dx|$ increases) as training in θ proceeds towards the unique minimum of the robust loss at 0, just like in actual

adversarial training in Figure 1. Effectively, this means that as training proceeds, it becomes more important to ensure the stability/convergence of the inner loop training. Given a limited computational budget, one should then allocate more of them towards the later part of training, hence giving rise to an annealing schedule that gradually increases the complexity of the inner loop iterations. Indeed, this can be made precise for this example - if we apply two-loop adversarial training with K_t inner loop steps at step size α_t (with $\tau = \alpha_t K_t$ fixed, c.f. Algorithm 1) to ℓ, we can solve the PMP explicitly in this case to obtain the optimal schedule for K_t.

$$K_t^* = \frac{\tau}{\alpha_t^*} = \frac{2\tau}{\theta_0^2 e^{-2t} + 1} = \frac{2\tau}{\theta_0^2 + 1} + \left(\frac{4\theta_0^2 \tau}{(\theta_0^2 + 1)^2} \right) t + \mathcal{O}(t^2), \tag{12}$$

where the last step is valid for small t. This is a schedule that gradually increases the number of inner maximization steps as the outer loop t proceeds, and motivates our annealing choice in Amata. Note that this example is qualitative, so we do not adopt an identical schedule in Amata, but a linear approximation of it (right hand side in (12)) that also increases in time. It is clear from the derivation that the origin of this increasing schedule is that the curvature $l_{xx}(\theta, x)$ increases as $\theta \to \theta^* = 0$, hence this motivates the Amata algorithm in view of the numerical observations in Fig 1.

Quantitative Justification of Amata via a Criterion of Approximate Optimality

Besides qualitative motivations, it is desirable to have a criterion to test the approximate optimality of any chosen annealing schedule. In this subsection, we develop a quantitative criterion for this purpose, based on the PMP. Given any hyper-parameter choice $\mathbf{u}_{T_1:T_2}$ over the training interval, let us define its "distance" from optimality as

$$C(\mathbf{u}_{T_1:T_2}) = \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} \max_{\mathbf{v} \in G} H(\boldsymbol{\theta}_s^{\mathbf{u}}, \mathbf{p}_s^{\mathbf{u}}, \mathbf{v}) - H(\boldsymbol{\theta}_s^{\mathbf{u}}, \mathbf{p}_s^{\mathbf{u}}, \mathbf{u}_s) ds \tag{13}$$

where $\{\boldsymbol{\theta}_s^{\mathbf{u}}, \mathbf{p}_s^{\mathbf{u}} : s \in [T_1, T_2]\}$ represents the solution of the Eq. (8) and (9) with $\mathbf{u}_s$ in place of $\mathbf{u}_s^*$. Observe that $C(\mathbf{u}_{T_1:T_2}) \geq 0$ for any $\mathbf{u}_{T_1:T_2}$ with equality if and only if $\mathbf{u}_{T_1:T_2}$ satisfies the PMP for almost every $s \in [T_1, T_2]$. Hence, C can be used as a measure of deviation from optimality. When C is small, our annealing strategy $\{\mathbf{u}_s\}$ is close to at least a locally optimal strategy, where as when it is large, our annealing strategy is far from an optimal one. For ease of calculation, we can further simplify C by Taylor expansions, assuming $T_2 - T_1 = \eta$ is small, yielding (See Appendix)

$$C(\mathbf{u}_{t:t+\eta}) \approx \max_{\alpha, K} \left\{ \|\nabla_{\boldsymbol{\theta}} \ell(h_{\boldsymbol{\theta}_t}[\mathcal{A}_{\boldsymbol{\theta}_t, \alpha, K}(x)], y)\|^2 - \gamma K \right\} - \left(\|\nabla_{\boldsymbol{\theta}} \ell(h_{\boldsymbol{\theta}_t}[\mathcal{A}_{\boldsymbol{\theta}_t, \alpha_t, K_t}(x)], y)\|^2 - \gamma K_t \right) \tag{14}$$

with $\mathcal{A}_{\boldsymbol{\theta}_t, \alpha, K}$ denoting the inner PGD loop starting from x with K steps and step size α. Criterion (14) is a greedy version of the general criterion derived from the maximum principle. It can be used to either evaluate the near-term optimality of some choice of hyper-parameters $\mathbf{u}$, or to find an approximately optimal hyperparameter greedily by solving $C(\mathbf{u}, t) = 0$ for $\mathbf{u}$, which amounts to maximizing the first term. In this paper, we use Bayesian optimization[4] to perform the maximization in (14) to evaluate strategies from the controllable space G.

Comparison with FOSC Criterion: (Wang et al. 2019) proposed an empirical criterion to measure the convergence of inner maximization:

$$\text{FOSC}(\mathbf{x}) = \epsilon \|\nabla_{\mathbf{x}} \ell(h_{\boldsymbol{\theta}}(\mathbf{x}), y)\| - \langle \mathbf{x} - \mathbf{x}^0, \nabla_{\mathbf{x}} \ell(h_{\boldsymbol{\theta}}(\mathbf{x}), y) \rangle \tag{15}$$

There are some similarities between the proposed criterion and FOSC when the computational cost term R is not considered. For example, when the stationary saddle point is achieved, both the proposed criterion and FOSC reach the minimum. However, the proposed criterion is different from FOSC in the following aspects: 1) The proposed criterion is derived from the optimal control theory, whereas FOSC is concluded from empirical observations. 2) The proposed criterion takes computation costs into consideration, whereas FOSC only considers the convergence of adversarial training. 3) The proposed criterion is based on the gradient of DNN parameters, whereas FOSC is based on the gradient of the input. Measuring the gradient of DNN parameters is arguably more suitable for considering robustness-training time trade-off as the variance of the DNN parameters is much larger than the input during training.

Evaluation of Amata Using C.

We now use the numerical form of the optimal control criterion (14) to analyze Amata for robustness and computational efficiency trade-off. We use the LeNet architecture [5] for MNIST classification as an example. We set the γ as 0.04 for this criterion, and show the result in Table 1. From this Table, we observe that C and performance (robustness, time) are correlated in the expected way, and that Amata has lower C values and better performances. Furthermore, C takes into account both robustness and computational cost as seen in the first two rows, where a lower C value is associated with similar robustness but lower time cost. Hence, C can help us choose a good annealing schedule.

Although computing the exact optimal control strategy for DNN adversarial training is expensive for real-time tasks, with the criterion derived from the PMP, we are able to numerically compare the optimality of different adversarial training strategies. From this numerical evaluation, we have demonstrated that the proposed Amata algorithm is close to an optimal adversarial training strategy, or at least one that satisfies the maximum principle. We will show that our algorithm can achieve similar or even better adversarial accuracy much faster with empirical experiments on popular DNN models later in Experiments section.

[4]Implementations can be found in https://github.com/hyperopt/hyperopt

[5]Implementations can be found in https://github.com/pytorch/examples/blob/master/mnist/main.py

Strategy	C	Robustness	Time
Amata(Setting 1)	*0.54*	*91.47%*	*697.73s*
Amata(Setting 2)	*0.68*	*91.46%*	*760.16s*
PGD-10	7.82	68.07%	307.57s
PGD-20	1.52	85.23%	567.11s
PGD-40	1.20	90.56%	1086.31s

Table 1: Comparison of adversarial training strategies. Amata setting 1: $K_{min} = 5, K_{max} = 40$, Amata setting 2: $K_{min} = 10, K_{max} = 40$.

Experiments

To demonstrate the effectiveness of Amata mechanism, experiments are conducted on MNIST(in the appendix), CIFAR10, Caltech256 and the large-scale ImageNet dataset. With less computational cost, the proposed Amata method trained models achieve comparable or slightly better performance than the models trained by other methods, such as PGD. In our experiment, PyTorch 1.0.0 and a single GTX 1080 Ti GPU were used for MNIST, CIFAR10, and Caltech256 experiment, while PyTorch 1.3.0 and four V100 GPUs were used for the ImageNet experiment. Note that different from research focus on improving adversarial accuracy, for efficient adversarial training research, it is important to run algorithms in the same hardware and driver setting for fair comparison. We evaluate the adversarial trained networks against PGD and Carlini-Wagner (CW) attack (Carlini and Wagner 2017). In addition, Amata can also be seamlessly incorporated into existing adversarial training acceleration algorithms, such as you only propogate once(YOPO, (Zhang et al. 2019)), adversarial training for free (Free, (Shafahi et al. 2019)), and fast adversarial training (Fast, (Wong, Rice, and Kolter 2020)). As an ablation study, results with other annealing schemes, such as exponential one, are shown in the Appendix. We first evaluate Amata on standard datasets and then incorporate Amata into other adversarial training acceleration algorithms.

Evaluation of Amata on Standard Datasets

CIFAR10 Classification For this task, we use the PreAct-Res-18 network (Madry et al. 2017). PGD, FOSC, FAT, and Amata are tested. τ is set as 20/255 for Amata. The clean and robust error of PGD-10 and Amata are shown in Figure 2 Left. The proposed Amata method takes 3045 seconds to achieve less than 55% robust error while for PGD-10, it takes 6944 seconds. FOSC takes 8385 seconds to achieve similar accuracy. During the experiment, FAT cannot achieve 55% robust error. This is because FAT always generate adversarial examples near the decision boundaries and the adversarial loss might be too small to make the adversarial training effective. Furthermore, we run the PGD, FOSC and Amata experiment for 100 epochs until full convergence, with showing the clean accuracy, PGD-20 attack accuracy, CW attack accuracy, and the consumed time in Table 2. We can see that Amata outperforms PGD-10 with reducing the consumed time to 61.9% and achieves comparable accuracy to FOSC with reducing the consumed time to 54.8%.

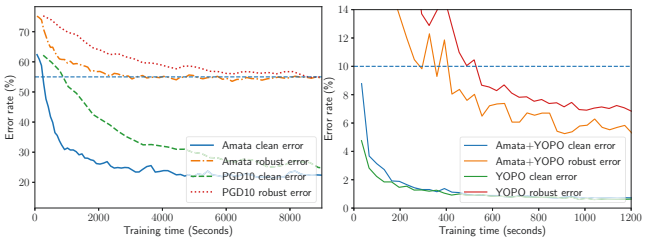

Figure 2: (Better viewed in the zoom-in mode) Left: The clean and robust error of Amata and PGD-10 in the CIFAR10 validation for achieving less than 55% robust error. We use Amata with the setting $K_{min} = 2$ and $K_{max} = 10$. Right: The clean and robust error of training with YOPO and Amata+YOPO on MNIST. We use Amata with the setting $K_{min} = 2$ and $K_{max} = 5$.

Caltech256 Classification Experiment is also conducted on Caltech256 dataset (Griffin, Holub, and Perona 2007) with ResNet-18 network. It consists of 30,607 images spanning 257 object categories. Object categories are diverse, including animals, buildings and musical instruments. We use the same experiment setting as in (Zhang and Zhu 2019). As it is already known that FOSC cannot lead to acceleration, we do not include FOSC in this experiment. For achieving the adversarial accuracy of 28%, Amata takes 3403 seconds while PGD-20 takes 4956 seconds. Furthermore, we run the PGD-20 and Amata training for 21 epochs until full convergence. The accuracies under clean, PGD-5, PGD-20, and PGD-100 attack data are shown in Table 3. We can see that, with saving around 30% computational time, the proposed Amata can achieve similar accuracy to PGD-20 under up to 100 iterations of PGD attacks.

Amata+

The proposed Amata mechanism is largely orthogonal to existing acceleration approaches to adversarial training acceleration, and hence can be readily incorporated into them. We now demonstrate this for YOPO (Zhang et al. 2019), adversarial training for free (Free) (Shafahi et al. 2019), and fast adversarial training (Fast) (Wong, Rice, and Kolter 2020). We name this kind of jointly implemented Amata as Amata+.

Amata+YOPO YOPO's MNIST classification experiment[6] is demonstrated as an example. For Amata incorporation, we gradually increase the K and decrease the σ in the codes that is similar to the case of modifying the PGD algorithm. The clean and robust error of YOPO and Amata+YOPO are shown in Figure 2 Right. We can see that Amata+YOPO takes 294 seconds to reach the adversarial accuracy of 94%, which is around the half of the time consumed by YOPO. It is also worth noting that Amata+YOPO achieves better adversarial accuracy when converged. This

[6]https://github.com/a1600012888/YOPO-You-Only-Propagate-Once/tree/82c5b902508224c642c8d0173e61435795c0ac42/experiments/MNIST/YOPO-5-10

Training methods	Clean accuracy	PGD-20 Attack	CW Attack	Time (Seconds)
ERM	94.75%	0.0%	0.23%	2099.58
PGD-2	90.16%	31.70%	13.36%	6913.36
PGD-10	85.27%	47.31%	51.73%	23108.10
FAT(Zhang et al. 2020)	89.30%	41.34%	41.16%	14586.08
FOSC(Wang et al. 2019)	85.29%	47.75%	47.70%	26126.98
Amata($K_{min} = 2$, $K_{max} = 10$)	*85.52%*	*47.62%*	*52.94%*	*14308.96*

Table 2: CIFAR10 adversarial training convergence results. We run all algorithms on the same computation platform for fair comparison.

Caltech256 results.				
Training methods	Clean accuracy	PGD-5	PGD-20	PGD-100
ERM	83.1%	0.0%	0.0%	0.0%
PGD-20	65.7%	29.7%	28.5%	28.5
Amata($K_{min} = 10$, $K_{max} = 20$)	*66.1%*	*29.6%*	*28.3%*	*28.3*
ImageNet results.				
Training methods	Clean accuracy	PGD-10	PGD-20	PGD-50
Free	60.57%	32.1%	31.5%	31.3%
Amata($K_{min} = 2$, $K_{max} = 4$)+Free	*59.7%*	*31.8%*	*31.2%*	*31.0*

Table 3: Caltech256 and ImageNet results. Amata and Amata+ achieve *almost the same* robustness under various strengths of attacks after convergence.

phenomenon corresponds to the finding in FOSC (Wang et al. 2019) that too strong adversarial example is not needed at the beginning. From this example, we can see that Amata can be easily incorporated in other adversarial training acceleration algorithm to provide further acceleration and to improve adversarial accuracy.

Amata+Free ImageNet is a large-scale image classification dataset consisting of 1000 classes and more than one million images (Russakovsky et al. 2015). Adversarial training on ImageNet is considered to be challenging due to the high computation cost (Kannan, Kurakin, and Goodfellow 2018; Xie et al. 2018). Recently, Ali *et al.*proposed the Free algorithm for adversarial training acceleration by alternatively running PGD-1 to solve the inner maximization and the outer minimization. This process is run $m = 4$ times for each input data batch, with four V100 GPUs. For Amata incorporation, similarly, we increase m from 2 to 4 and decrease the PGD step size also by two times in the code. In the experiment, ResNet-50 is used for adversarial training. We find that it takes Amata+Free 948 minutes to achieve 30% adversarial accuracy which saves around 1/3 computational time compared with 1318 minutes by the Free algorithm. We further run Free and Free+Amata 22 epochs for full convergence and test the obtained models with various iterations of PGD attacks. The results are shown in Table 3. We can see that Amata can still help reducing the computational cost almost without performance degradation even when combined with the state-of-the art adversarial training acceleration algorithm on the large-scale dataset.

Amata+Fast We also incorporate Amata into the fast adversarial training algorithm by using a weaker version of the fast adversarial training algorithm at the initial stage and then using the original version of the fast adversarial training algorithm later. The weaker version of the adversarial training algorithm is constructed by using a fixed non-zero initialization at the start. With the same setting, Amata+Fast can achieve 72% PGD-20 accuracy two times faster than fast adversarial training on CIFAR10. This further shows that Amata is a mechanism that can be readily incorporated into many existing algorithms.

Amata+ATTA We find that Amata can be seamlessly combined with a recently proposed adversarial training method—adversarial training with transferable adversarial examples (ATTA). We follow the same setting as in (Zheng et al. 2020). ATTA achieves efficient adversarial training by reusing a number of adversarial perturbations calculated in previous epochs, which is controled by a hyper-parameter reset in [7]. To implement Amata, we reduce reset to be 2 in the first five epochs to reduce the strength of adversarial examples. Compared with Amata, Amata+ATTA can achieve 58% PGD-20 accuracy around 1.5 times faster than ATTA.

Conclusion

We proposed a novel annealing mechanism for accelerating adversarial training that achieves comparable or better robustness with 1/3 to 1/2 the computational cost over a variety of benchmarks. Moreover, a convergence proof and a general optimal control formulation of annealed adversarial training is developed to justify its validity and performance. Our approach can also be seamlessly incorporated into existing adversarial training acceleration algorithms to achieve acceleration and improve performance. As a point of future work, we will explore adaptive methods for adversarial training based on the optimal control formulation (e.g. the MSA algorithm (Chernousko and Lyubushin 1982; Li et al. 2017; Li and Hao 2018)).

[7] https://github.com/hzzheng93/ATTA

Acknowledgements

Nanyang Ye was supported in part by National Key R&D Program of China 2017YFB1003000, in part by National Natural Science Foundation of China under Grant (No. 61672342, 61671478, 61532012, 61822206, 61832013, 61960206002, 62041205), in part by Tencent AI Lab Rhino Bird Focused Research Program JR202034, in part by the Science and Technology Innovation Program of Shanghai (Grant 18XD1401800, 18510761200), in part by Shanghai Key Laboratory of Scalable Computing and Systems.

Zhanxing Zhu was supported by Beijing Nova Program (No. 202072) from Beijing Municipal Science & Technology Commission, and National Natural Science Foundation of China (No.61806009 and 61932001), PKU-Baidu Funding 2019BD005.

References

Akhtar, N.; and Mian, A. 2018. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *arXiv preprint arXiv:1801.00553* .

Ben-Tal, A.; and Nemirovski, A. 1998. Robust Convex Optimization. *Mathematics of Operations Research* 23(4): 769–805.

Bertsekas, D. P.; Bertsekas, D. P.; Bertsekas, D. P.; and Bertsekas, D. P. 1995. *Dynamic Programming and Optimal Control*, volume 1. Athena scientific Belmont, MA.

Bhat, S.; and Tsipras, D. 2019. Towards Efficient Methods for Training Robust Deep Neural Networks. URL https://math.mit.edu/research/highschool/primes/materials/2018/Bhat.pdf. [Online].

Boltyanskii, V. G.; Gamkrelidze, R. V.; and Pontryagin, L. S. 1960. The Theory of Optimal Processes. The maximum principle. Technical report, TRW Space Tochnology Labs, Los Angeles, California.

Buckman, J.; Roy, A.; Raffel, C.; and Goodfellow, I. 2018. Thermometer Encoding: One Hot Way To Resist Adversarial Examples. In *International Conference on Learning Representations*.

Carlini, N.; and Wagner, D. 2017. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy*, 39–57.

Chernousko, F. L.; and Lyubushin, A. A. 1982. Method of Successive Approximations for Solution of Optimal Control Problems. *Optimal Control Applications and Methods* 3(2): 101–114. ISSN 0143-2087.

Gao, R.; Cai, T.; Li, H.; Wang, L.; Hsieh, C.; and Lee, J. D. 2019. Convergence of Adversarial Training in Overparametrized Networks. *arXiv preprint arXiv:1906.07916* .

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. The MIT Press. ISBN 0262035618.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572* .

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 Object Category Dataset. Technical Report 7694, California Institute of Technology. URL http://authors.library.caltech.edu/7694.

He, W.; Wei, J.; Chen, X.; Carlini, N.; and Song, D. 2017. Adversarial Example Defenses: Ensembles of Weak Defenses Are Not Strong. *arXiv preprint arXiv:1706.04701* .

Kannan, H.; Kurakin, A.; and Goodfellow, I. J. 2018. Adversarial Logit Pairing. *arXiv preprint arXiv:1803.06373* .

Kingma, D. P.; and Ba, J. 2014. Adam: a Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* .

Kolter, J. Z.; and Wong, E. 2017. Provable Defenses Against Adversarial Examples via The Convex Outer Adversarial Polytope. *arXiv preprint arXiv:1711.00851* .

Li, Q.; Chen, L.; Tai, C.; and E, W. 2017. Maximum Principle Based Algorithms for Deep Learning. *The Journal of Machine Learning Research* 18(1): 5998–6026. ISSN 1532-4435.

Li, Q.; and Hao, S. 2018. An Optimal Control Approach to Deep Learning and Applications to Discrete-Weight Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 2985–2994. Stockholmsmässan, Stockholm Sweden: PMLR.

Li, Q.; Tai, C.; and E, W. 2017. Stochastic Modified Equations and Adaptive Stochastic Gradient Algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 2101–2110. International Convention Centre, Sydney, Australia: PMLR.

Li, Q.; Tai, C.; and E, W. 2019. Stochastic Modified Equations and Dynamics of Stochastic Gradient Algorithms I: Mathematical Foundations. *Journal of Machine Learning Research* 20(40): 1–47. URL http://jmlr.org/papers/v20/17-526.html.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv preprint arXiv:1706.06083* .

Meng, D.; and Chen, H. 2017. MagNet: A Two-Pronged Defense against Adversarial Examples. In *ACM Conference on Computer and Communications Security*.

Pan, H.; and Jiang, H. 2015. Annealed Gradient Descent for Deep Learning. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI'15, 652–661. Arlington, Virginia, USA: AUAI Press. ISBN 9780996643108.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3): 211–252. doi:10.1007/s11263-015-0816-y.

Shafahi, A.; Najibi, M.; Ghiasi, M. A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L. S.; Taylor, G.; and Goldstein, T. 2019.

Adversarial Training for Free! In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Sinha, A.; Namkoong, H.; and Duchi, J. 2018. Certifiable Distributional Robustness With Principled Adversarial Training. In *International Conference on Learning Representations*.

Song, Y.; Kim, T.; Nowozin, S.; Ermon, S.; and Kushman, N. 2018. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. In *International Conference on Learning Representations*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing Properties of Neural Networks. *arXiv preprint arXiv:1312.6199* .

Tramèr, F.; Kurakin, A.; Papernot, N.; Boneh, D.; and McDaniel, P. 2017. Ensemble Adversarial Training: Attacks and Defenses. *arXiv preprint arXiv:1705.07204* .

Wang, Y.; Ma, X.; Bailey, J.; Yi, J.; Zhou, B.; and Gu, Q. 2019. On the Convergence and Robustness of Adversarial Training. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, 6586–6595. Long Beach, California, USA: PMLR.

Wong, E.; Rice, L.; and Kolter, J. Z. 2020. Fast Is Better Than Free: Revisiting Adversarial Training. In *International Conference on Learning Representations*.

Xie, C.; Wu, Y.; van der Maaten, L.; Yuille, A. L.; and He, K. 2018. Feature Denoising for Improving Adversarial Robustness. *arXiv preprint arXiv:1812.03411* .

Zhang, D.; Zhang, T.; Lu, Y.; Zhu, Z.; and Dong, B. 2019. You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhang, J.; Xu, X.; Han, B.; Niu, G.; Cui, L.; Sugiyama, M.; and Kankanhalli, M. 2020. Attacks Which Do Not Kill Training Make Adversarial Learning Stronger. In *Proceedings of Machine Learning Research*, volume 119, 11278–11287. PMLR.

Zhang, T.; and Zhu, Z. 2019. Interpreting Adversarially Trained Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, 7502–7511. PMLR.

Zheng, H.; Zhang, Z.; Gu, J.; Lee, H.; and Prakash, A. 2020. Efficient Adversarial Training with Transferable Adversarial Examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.