Distributionally Adversarial Attack

Tianhang Zheng,[1] **Changyou Chen,**[1] **Kui Ren**[1,2]
[1]State University of New York at Buffalo
[2]Zhejiang University
{tzheng4, changyou, kuiren}@buffalo.edu

Abstract

Recent work on adversarial attack has shown that Projected Gradient Descent (PGD) Adversary is a universal first-order adversary, and the classifier adversarially trained by PGD is robust against a wide range of first-order attacks. It is worth noting that the original objective of an attack/defense model relies on a data distribution $p(\mathbf{x})$, typically in the form of risk maximization/minimization, *e.g.*, $\max/\min \mathbb{E}_{p(\mathbf{x})}\mathcal{L}(\mathbf{x})$ with $p(\mathbf{x})$ some unknown data distribution and $\mathcal{L}(\cdot)$ a loss function. However, since PGD generates attack samples independently for each data sample based on $\mathcal{L}(\cdot)$, the procedure does not necessarily lead to good generalization in terms of risk optimization. In this paper, we achieve the goal by proposing distributionally adversarial attack (DAA), a framework to solve an optimal *adversarial-data distribution*, a perturbed distribution that satisfies the L_∞ constraint but deviates from the original data distribution to increase the generalization risk maximally. Algorithmically, DAA performs optimization on the space of potential data distributions, which introduces direct dependency between all data points when generating adversarial samples. DAA is evaluated by attacking state-of-the-art defense models, including the adversarially-trained models provided by *MIT MadryLab*. Notably, DAA ranks *the first place* on MadryLab's white-box leaderboards, reducing the accuracy of their secret MNIST model to 88.56% (with l_∞ perturbations of $\epsilon = 0.3$) and the accuracy of their secret CIFAR model to 44.71% (with l_∞ perturbations of $\epsilon = 8.0$). Code for the experiments is released on https://github.com/tianzheng4/Distributionally-Adversarial-Attack.

Introduction

Recent years have witnessed widespread use of deep neural networks (DNNs), achieving remarkable performance on different machine-learning tasks, such as object detection and recognition (Krizhevsky, Sutskever, and Hinton 2012), strategy optimization (Silver et al. 2016), and natural language processing (Cho et al. 2014). At the same time, DNNs also have been proved to be vulnerable to adversarial samples – data that are indistinguishable from natural samples by human but endow additional maliciously-embedded perturbations. Those maliciously perturbed samples can cause DNNs to make predictions different from the ground truth with high confidence. Various first-order algorithms have been proposed to generate adversarial samples, such as Fast Gradient Sign Method (FGSM) (Szegedy et al. 2013), Projected Gradient Descent (PGD) (Kurakin, Goodfellow, and Bengio 2016b), and Carlini & Wagner Attacks (CW) (Carlini and Wagner 2017).

Among all those first-order attacks, Madry et al. (2017) suggest that PGD is a universal attack algorithm, and the classifier adversarially trained by PGD is robust against a wide range of first-order attacks. Carlini et al. (2017) strengthen the hypothesis by demonstrating that PGD-adversarial training provably succeeds at increasing the distortion required to construct adversarial examples by a factor of 4.2. Moreover, among all the white-box defenses that appeared in ICLR-2018 and CVPR-2018, PGD-adversarial training is the only empirical defense that has not been further attacked (Athalye, Carlini, and Wagner 2018; Athalye and Carlini 2018).

Despite the success of PGD, one notable limitation is that the adversarial samples are not globally optimal, in the sense that adversarial samples are generated independently for each data sample. From a machine-learning perspective, this lacks a statistical interpretation in terms of risk maximization, *i.e.*, PGD is not a training procedure, thus the underlying optimization problem is not mathematically clear. In this paper, we provide a distribution-optimization view of PGD, and propose distributionally adversarial attack (DAA), a new concept of adversarial attack that is performed on the space of probability measures (*e.g.*, unknown data distributions). In DAA, the problem is formulated as optimizing an adversarial data distribution (from which adversarial samples are drawn from) such that the *generalization risk* increases maximumly. This generalizes PGD by lifting the optimization onto the space of probability measures, and can be interpreted as Wasserstein gradient flows (WGFs), a framework for distribution optimization which always decreases an "energy functional" over time. The energy functional reflexes the data manifold in adversarial attack, and is designed in correspondence with the original objective of a DNN. When using testing data to approximate the unknown data manifold, DAA leads to a variant of the standard PGD where all adversarial samples are explicitly dependent.

DAA is extensively evaluated on four datasets, including MNIST, Fashion MNIST (FMNIST), CIFAR10, and Imagenet, by attacking their state-of-the-art defense models. We

show that a single run of DAA with $0.3/1.0$ l_∞ perturbations can reduce the accuracy of MadryLab's MNIST model (Madry et al. 2017) to approximately 90.5%, outperforming a single run of PGD that reduces the accuracy to approximately 92.5%. Furthermore, a single run of DAA with $0.2/1.0$ l_∞ perturbations reduces the accuracy of our adversarially trained FMNIST model to 67.6%, outperforming a single run of PGD that reduces the accuracy to 71.5%. Similarly, a single run of DAA reduces the accuracy of MadryLab's CIFAR10 model to 44.98% ($8/255$ l_∞ perturbations) and the accuracy of the ensemble adversarial trained Imagenet model (Kurakin, Goodfellow, and Bengio 2016b) to 16.43% (only $2/255$ l_∞ perturbations). For DAA with 50 random restarts, it reduces the accuracy of MadryLab's public and secret MNIST model to 88.7% and 88.56%, respectively; whereas DAA with 10 random restarts reduces the accuracy of MadryLab's secret CIFAR10 model to 44.71%. *Both settings outperform other attack algorithms listed in MIT MadryLab's white-box leaderboards.*[1]

Preliminaries

We introduce necessary background in this section, including Wasserstein gradient flows and adversarial attack/defense methods.

Wasserstein Gradient Flows

Wasserstein Metric Space Wasserstein metric is a distance metric defined between probability measures (distributions) on the Wasserstein metric space. Formally, let $P_2(\Omega)$ denote the collection of all probability measures on $\Omega \subset R^r$ with finite 2nd moment. The 2nd-order Wasserstein distance between two probability measures in $P_2(\Omega)$ is defined as:

$$W_2^2(\mu,\nu) \triangleq \inf_{\gamma}\{\int_{\Omega\times\Omega} || \mathbf{x}-\mathbf{x}' ||_2^2 d\gamma(\mathbf{x},\mathbf{x}') : \quad \gamma \in \Omega(\mu,\nu)\}, \tag{1}$$

where $\Gamma(\mu,\nu)$ denotes the collection of all joint probability measures on $\Omega\times\Omega$ with two marginals equal to μ and ν. One way to understand the motivation of the above definition is to consider the optimal transport problem, where one wants to transform elements in the domain of μ to ν with minimum cost. The cost to transport $\mathbf{x}$ in μ to $\mathbf{x}'$ in ν is quantified by $|| \mathbf{x}-\mathbf{x}' ||_2^2$. If μ is absolutely continuous w.r.t. the Lebesgue measure, there exists a unique optimal transport plan, i.e., a mapping $T : R^r \to R^r$, to transform elements in μ to elements in ν. The Wasserstein distance can be equivalently reformulated as: $W_2^2(\mu,\nu) \triangleq \inf_T\{\int_\Omega || \mathbf{x}-T(\mathbf{x})||_2^2 d\mu(\mathbf{x})\}$.

Wasserstein Gradient Flows The 2nd-order Wasserstein metric endows $P_2(\Omega)$ with a Riemannian geometry. Let $\{\mu_t\}_{t\in[0,1]}$ be an absolutely continuous curve on this geometry, with change between μ_t and μ_{t+h} measured by $W^2(\mu_t,\mu_{t+h})$. The change can be reflected by a vector field: $v_t(\mathbf{x}) \triangleq \lim_{h\to 0}\frac{T(\mathbf{x})-\mathbf{x}}{h}$. This vector field is regarded as the velocity field of the elements $\mathbf{x}$. Based on the above description, a gradient flow can be defined on $P_2(\Omega)$ in Lemma 1.

Lemma 1 *Let $\{\mu_t\}_{t\in[0,1]}$ be an absolutely continuous curve in $P_2(\Omega)$. Then for a.e. $t \in [0,1]$, the above vector field $\mathbf{v}_t$ defines a gradient flow on $P_2(\Omega)$ as: $\partial_t\mu_t + \nabla\cdot(\mathbf{v}_t\mu_t) = 0$.*

Actually, the velocity field $\mathbf{v}$ can be derived for optimization on an *energy functional* $E : P_2(\Omega) \to R$. In this case, it can be shown that $\mathbf{v}_t$ has the form $\mathbf{v}_t = -\nabla\frac{\delta E}{\delta \mu_t}$ (Ambrosio, Gigli, and Savaré 2008), where $\frac{\delta E}{\delta \mu_t}$ is called the first variation of E at μ_t. Thus the gradient flow on $P_2(\Omega)$ can be rewritten as:

$$\partial_t\mu_t = -\nabla\cdot(\mathbf{v}_t\mu_t) = \nabla\cdot(\mu_t\nabla\frac{\delta E}{\delta\mu_t}) \tag{2}$$

Adversarial Sample

Definition and Notation In this paper, we only study adversarial samples on neural networks used for classification, with final layers as softmax activation functions. We represent such a network as a vector function $\{F_i(\mathbf{x})\}_i$. Given an input $\mathbf{x}$, the network predicts its label as $\tilde{y} = \arg_i\max F_i(x)$. A sample $\mathbf{x}'$ is called an adversarial sample if $\arg_i\max F(\mathbf{x}') \neq y$, where y is the true label and $\mathbf{x}'$ is close to the original $\mathbf{x}$ under a certain distance metric.

Fast Gradient Sign Method (FGSM) Fast Gradient Sign Method (FGSM) is a single-step adversarial attack proposed by Szegedy et al. (2013). FGSM performs a single step update on the original sample $\mathbf{x}$ along the direction of the gradient of a loss function $\mathcal{L}(\mathbf{x},y;\boldsymbol{\theta})$. The loss function is usually defined as the cross-entropy between the output of a network and the true label y. Formally, FGSM adversarial samples are generated as

$$\mathbf{x}' = \mathsf{clip}_{[0,1]}\{\mathbf{x} + \epsilon\cdot\mathrm{sign}(\nabla_x\mathcal{L}(\mathbf{x},y;\boldsymbol{\theta})\}\ , \tag{3}$$

where ϵ controls the maximum l_∞ perturbation of the adversarial samples, and the $\mathsf{clip}_{[a,b]}(\mathbf{x})$ function forces $\mathbf{x}$ to reside in the range of $[a,b]$.

Projected Gradient Descent (PGD) Projected Gradient Descent (PGD) is an iterative variant of FGSM. In each iteration, PGD follows the update rule:

$$\mathbf{x}'_{l+1} = \mathrm{II}_{\mathsf{clip}}\{\mathsf{FGSM}(\mathbf{x}'_l)\}\ , \tag{4}$$

where $\mathsf{FGSM}(\mathbf{x}'_l)$ represents an FSGM update of $\mathbf{x}'_l$ as in (3), and the outer clip function $\mathrm{II}_{\mathsf{clip}}$ keeps $\mathbf{x}'_{l+1}$ within a predefined perturbation range. PGD can also be interpreted as an iterative algorithm to solve the following problem:

$$\max_{\mathbf{x}':||\mathbf{x}'-\mathbf{x}||_\infty<\alpha}\mathcal{L}(\mathbf{x}',y;\boldsymbol{\theta}). \tag{5}$$

Madry et al. (2017) observe that the local maxima of the cross-entropy loss found by PGD with 10^5 random starts are distinctive, but all have similar loss values, for both normally- and adversarially-trained networks. Inspired by this concentration phenomena, they propose that PGD is a universal adversary among all the first-order adversaries, *i.e.*, attacks only rely on first-order information.

[1] https://github.com/MadryLab/mnist_challenge; https://github.com/MadryLab/cifar10_challenge

Momentum-based Iterative Fast Gradient Sign Method (MI-FGSM) MI-FGSM is derived from the Iterative FGSM (Kurakin, Goodfellow, and Bengio 2016a), which integrates the momentum term into an iterative process to generate adversarial samples (Dong et al. 2018). Given $g_0 = 0$ and $g_{l+1} = \mu \cdot g_l + \frac{\nabla_x \mathcal{L}(\mathbf{x}'_l, y; \boldsymbol{\theta})}{||\nabla_x \mathcal{L}(\mathbf{x}'_l, y; \boldsymbol{\theta})||_1}$, the iterative version of MI-FGSM can be expressed as:

$$\mathbf{x}'_{l+1} = \mathbf{x}'_l + \epsilon \cdot \text{sign}(g_{l+1}). \tag{6}$$

Based on MI-FGSM, we further derive its PGD variant, called Momentum PGD, with iterative updates

$$\mathbf{x}'_{l+1} = \mathbf{x}'_l + \cdot \Pi_{\mathsf{clip}}\{\epsilon \cdot \text{sign}(g_{l+1})\}. \tag{7}$$

Remark 2 *Momentum PGD is a stronger attack than MI-FGSM, since it can proceed for more steps with an appropriate step size ϵ (ϵ can not be too small, otherwise the adversarial samples are very likely to get trapped in bad local maxima). The clip function Π_{clip} ensures the adversarial samples to have the predefined perturbation size after the extra iterations.*

Adversarial Training

Definition and Notation Adversarial training is a defense method against adversarial samples first proposed by Goodfellow, Shlens, and Szegedy (2014). The approach attempts to improve the robustness of a network by training it together with adversarial samples. Formally, adversarial training solves the following min-max problem:

$$\min_{\theta} \max_{x': D(\mathbf{x}, \mathbf{x}') < \alpha} \mathcal{L}(\mathbf{x}', y; \boldsymbol{\theta}) \,, \tag{8}$$

where $D(\mathbf{x}, \mathbf{x}')$ represents certain distance metric between x and x'. The inner maximization problem is equivalent to constructing the strongest adversarial samples. If l_∞ distance is employed as the distance metric $D(\mathbf{x}, \mathbf{x}')$, the inner maximization problem is equivalent to the adversarial problem solved by PGD, i.e., (5). The outer minimization is the standard training procedure to minimize the loss of a DNN. Recent work shows that this straightforward method is one of the most effective defenses against adversarial samples (Madry et al. 2017; Tramèr et al. 2017; Cai et al. 2018).

PGD Adversarial Training The fact that PGD adversary is a first-order universal adversary implies that robustness against PGD should yield robustness against all first-order adversaries (Madry et al. 2017). Hence, Madry et al. (2017) propose to adversarially train a robust classifier using PGD attack. Specifically, in each training iteration, PGD is applied to generate a minibatch of adversarial samples to update the current network. In the training process, a steady decrease of the training loss is usually observed, indicating the effectiveness of this training paradigm. Experiment results show that PGD adversarial-trained models are robust against PGD attack as well as another strong attacks such as the CW_∞ attack (Carlini and Wagner 2017). Empirically we also found that their MNIST and CIFAR-10 models are indeed robust to a wide range of existing first-order attacks, including DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2016), and Jacobian-based Saliency Map Attack (JSMA) (Papernot et al. 2016), as long as the adversarial perturbations are l_∞- bounded.

Ensemble Adversarial Training To scale up adversarial training to ImageNet-scale datasets, Kurakin, Goodfellow, and Bengio (2016b) adversarially train a model using a fast single-step attack method. However, their adversarially-trained model is vulnerable to multi-step white-box attacks (Kurakin, Goodfellow, and Bengio 2016b). Tramèr et al. (2017) further demonstrate that the model of Kurakin, Goodfellow, and Bengio (2016b) is vulnerable to black-box adversaries (Tramèr et al. 2017). To tackle this problem, Tramèr et al. (2017) propose a training methodology that incorporates adversarial samples transferred from other pre-trained models, called Ensemble Adversarial Training (EAT) (Tramèr et al. 2017). Intuitively, this approach increases the diversity of adversarial samples used for adversarial training. In their experiments, the models trained by EAT exhibit robustness against adversarial samples transferred from other holdout models, using various single-step and multi-step attacks.

Distributionally Adversarial Attack

We first interpret distributionally adversarial attack as WGFs, and then propose a specific energy functionals to construct a WGF for better adversarial-sample generation, and finally propose particle-approximation methods to solve the DAA problem, leading to a variant of the standard PGD.

Adversarial Attack as WGFs

For a given DNN, the landscape of a loss function $\mathcal{L}(\mathbf{x}, y; \boldsymbol{\theta})$ constitutes a geometry structure indexed by input images $\mathbf{x}$. From a probability perspective, under regularized conditions, it is natural to define a probability distribution for each input $\mathbf{x}$ based on the loss-function landscape, *i.e.*,

$$p(\mathbf{x}, y; \boldsymbol{\theta}) \propto \exp\{-\mathcal{L}(\mathbf{x}, y; \boldsymbol{\theta})\} \,. \tag{9}$$

Since y is deterministic given $\mathbf{x}$, we would omit y, and write $p(\mathbf{x}, y; \boldsymbol{\theta})$ as $p(\mathbf{x}; \boldsymbol{\theta})$ in the following for simplicity. Based on this, we explain our intuitions on generalizing adversarial attack on the space of data distributions in the following. First note that the objective of an adversarial attack (5) is equivalently rewritten as:

$$\mathbf{x}' = \arg \max_{x': D(x, x') < \alpha} \{\mathcal{L}(\mathbf{x}', y; \boldsymbol{\theta}) - \mathcal{L}(\mathbf{x}, y; \boldsymbol{\theta})\} \,, \tag{10}$$

which describes the increase of a loss with an adversarial sample. On the space of probability measure, the loss is instead described by an energy functional $E(\mu)$, assuming the minimum being reached at $p(\mathbf{x}; \boldsymbol{\theta})$. Consequently, instead of finding an optimal adversarial sample $\mathbf{x}'$ for each $\mathbf{x}$, DAA tries to find an optimal *adversarial-data distribution*, μ^*, such that μ^* is close to $p(\mathbf{x}; \boldsymbol{\theta})$ but increases $E(\cdot)$ maximumly, *i.e.*,

$$\begin{aligned} \mu^* &= \arg \max_{\mu: W_2(\mu, p(\mathbf{x}; \boldsymbol{\theta})) < \alpha} \{E(\mu) - E(p)\} \\ &= \arg \max_{\mu: W_2(\mu, p(\mathbf{x}; \boldsymbol{\theta})) < \alpha} E(\mu) \,. \end{aligned} \tag{11}$$

Theorem 3 *The solution μ^* of* (11) *is equivalently described by the following PDE:*

$$\partial_t \mu_t = -\nabla_{\mathbf{x}} \cdot \left(\mu_t \nabla_{\mathbf{x}} \left(\frac{\delta E}{\delta \mu_t}(\mu_t) \right) \right) , \qquad (12)$$

and $\mu^* = \mu_t$ where $t = \{\inf\{t'\} : \mu_0(\mathbf{x}) = p(\mathbf{x};\boldsymbol{\theta}), W_2(\mu_{t'}(\mathbf{x}), p(\mathbf{x};\boldsymbol{\theta})) < \alpha\}$.

Energy Functional

It is crucial to define an energy functional as it directly affects adversarial-sample behaviors. Recent studies (Song et al. 2017; Ma et al. 2018) show that adversarial samples/subspaces mainly lie in the low probability regions of the original data distribution, thus we expect adversarial distribution to derivate from the original distribution by optimization over the energy functional. Besides, the energy functional should also be simple enough to possess a unique solution, *i.e.*, it should be convex w.r.t. μ on the space of probability measures.Therefore, we define a new energy functional as:

$$E(\mu) = \int_{\mu} \mathcal{L}(\mathbf{x}, y; \boldsymbol{\theta}) d\mu + c \cdot \mathsf{KL}(\mu || p) , \qquad (13)$$

where $\mathcal{L}(\mathbf{x}, y; \boldsymbol{\theta})$ is the loss of the system; $KL(\mu||p)$ is the KL-divergence between the adversarial distribution μ and the optimal data distribution p; and c is a hyperparameter balancing those two terms. Intuitively, maximizing (13) will increase the individual losses in addition to the deviation between the adversarial and original distributions. Note the energy functional (13) is still convex w.r.t. μ, maintaining the optimality condition and making the problem easier to solve.

Adversarial-Distribution Optimization and Adversarial-Sample Generation

Note a closed-formed solution of (12) is infeasible given the energy functional defined by (13). Following standard methods such as those in (Chen et al. 2018), we adopt particle approximation to solve (12). The idea is to approximate μ with a set of M particles $\{\mathbf{x}^{(i)}\}_{i=1}^{M}$ as $\mu \approx \frac{1}{M}\sum_{i=1}^{M} \delta_{\mathbf{x}^{(i)}}$, where $\delta_{\mathbf{x}}$ is a delta function with a spike at $\mathbf{x}$. Consequently, solving for the optimal μ corresponds to optimizing the particles as *adversarial samples* from the adversarial distribution. In the following, based on (Chen et al. 2018), we investigate two methods for particle approximation: the Lagrangian blob method and the discrete-gradient-flow method.

Lagrangian Blob Method The idea is to use particle approximations directly in the original problem (12). Specifically, define $\mathbf{v}_t \triangleq \nabla_{\mathbf{x}} \left(\frac{\delta E}{\delta \mu_t}(\mu_t) \right)$. According to (Carrillo, Craig, and Patacchini 2017), $\mathbf{v}_t$ is interpreted as the velocity function of a particle in the gradient flow. Consequently, Lagrangian blob methods evolve particles on a grid with a time-spacing h following the velocity $\mathbf{v}_t$ (Carrillo, Craig, and Patacchini 2017). Thus solving the WGF (12) is equivalent to evolving the particles along their velocities as

$$d\,\mathbf{x}^{(i)}\,/dt = \mathbf{v}_t(\mathbf{x}^{(i)}) .$$

To calculate $\mathbf{v}_t$, we substitute the form of $E(\mu)$ in (13) into $\mathbf{v}_t$. First note that under the $\mathcal{H}$-Wasserstein distance metric defined by (Liu et al. 2017), we have

$$\nabla_{\mathbf{x}} \left(\frac{\delta KL(\mu_t|p)}{\delta \mu_t} \right) = \mathbb{E}_{\tilde{\mathbf{x}}\sim\mu_t}[\nabla_{\tilde{\mathbf{x}}}[p(\tilde{\mathbf{x}})K(\mathbf{x},\tilde{\mathbf{x}})]/p(\tilde{\mathbf{x}})]$$
$$= \mathbb{E}_{\tilde{\mathbf{x}}\sim\mu_t}[K(\mathbf{x},\tilde{\mathbf{x}})\nabla_{\tilde{\mathbf{x}}} \log p(\tilde{\mathbf{x}}) + \nabla_{\tilde{\mathbf{x}}} K(\mathbf{x},\tilde{\mathbf{x}})], \qquad (14)$$

where $K(\cdot,\cdot)$ is a kernel function such as the RBF kernel. For the first term on the right of (13), we have

$$\nabla_{\mathbf{x}} \left(\frac{\delta \int_{\mu} \mathcal{L}(\mathbf{x}, y; \boldsymbol{\theta}) d\mu}{\delta \mu_t} \right) = \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y; \boldsymbol{\theta}). \qquad (15)$$

Combining Eq. 14 and 15, one ends up solving the following ordinary differential equation:

$$\frac{d\,\mathbf{x}}{dt} = \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y; \boldsymbol{\theta}) + \qquad (16)$$
$$c \cdot \mathbb{E}_{\tilde{\mathbf{x}}\sim\mu_t}[K(\mathbf{x},\tilde{\mathbf{x}})\nabla_{\tilde{\mathbf{x}}} \log p(\tilde{\mathbf{x}}) + \nabla_{\tilde{\mathbf{x}}} K(\mathbf{x},\tilde{\mathbf{x}})].$$

Considering a discrete approximation of μ_t with particles, (16) can be solved numerically as

$$\mathbf{x}_{\ell+1}^{(i)} = \mathbf{x}_{\ell}^{(i)} + \epsilon_l \cdot \{\nabla_{\mathbf{x}_\ell^{(i)}} \mathcal{L}(\mathbf{x}_\ell^{(i)}, y^{(i)}; \boldsymbol{\theta}) + \qquad (17)$$
$$\frac{c}{M} \sum_{j=1}^{M} [K(\mathbf{x}_\ell^{(i)}, \mathbf{x}_\ell^{(j)}) \nabla_{\mathbf{x}_\ell^{(j)}} \log p(\mathbf{x}_\ell^{(j)}) +$$
$$\nabla_{\mathbf{x}_\ell^{(j)}} K(\mathbf{x}_\ell^{(i)}, \mathbf{x}_\ell^{(j)})]\},$$

where, in contrast to the continuous case, we use ℓ to index the number of steps for the discretized particles.

Discrete Gradient Flows Discrete-gradient-flow (DGFs) approximation for (12) consists of a sequence of sub-optimization problems whose composition approximates μ_t, *i.e.*, $\mu_t \approx \tilde{\mu}_L \circ \cdots \circ \tilde{\mu}_1$, where $L = t/h$, and $\tilde{\mu}_\ell$ is the solution of the following functional optimization problem [2]:

$$\tilde{\mu}_\ell = \arg \max_{\mu \in P_2(R^r)} \{E(\mu) - \frac{1}{2h} W_2^2(\tilde{\mu}_{\ell-1}, \mu)\}. \qquad (18)$$

Again, (18) is solved by particle approximation, with gradient ascent on the particles. To this end, we need gradients for the two terms on the RHS of (18). Inspired by (Chen et al. 2018), we decompose the two terms and re-organize terms:

$$E_1 \triangleq \sum_{i=1}^{M} \left(\mathcal{L}(\mathbf{x}_\ell^{(i)}, y^{(i)}; \boldsymbol{\theta}) - c \log p(\mathbf{x}_\ell^{(i)}) \right)$$
$$E_2 \triangleq \mathbb{E}_{\mu}[\log \mu] + \frac{1}{2h} W_2^2(\mu, \tilde{\mu}_{\ell-1}) .$$

The gradient of the first term can be easily calculated as

$$\frac{\partial E_1}{\partial\, \mathbf{x}_\ell^{(i)}} = \nabla_{\mathbf{x}_\ell^{(i)}} \mathcal{L}(\mathbf{x}_\ell^{(i)}, y^{(i)}; \boldsymbol{\theta}) + c \nabla_{\mathbf{x}_\ell^{(i)}} \log p(\mathbf{x}_\ell^{(i)})$$
$$= (1 + c) \nabla_{\mathbf{x}_\ell^{(i)}} \mathcal{L}(\mathbf{x}_\ell^{(i)}, y^{(i)}; \boldsymbol{\theta}) . \qquad (19)$$

[2] Note the difference between (18) and the original DGF formula is to replace the original $\min$ to $\max$ because the flow direction is reversed in adversarial-distribution optimization.

For the E_2 term, we apply similar idea as (Chen et al. 2018) by introducing Lagrangian multipliers, resulting in

$$\frac{\partial E_2}{\partial \mathbf{x}_\ell^{(i)}} \approx c \cdot [\sum_j 2u_i v_j (\frac{d_{ij}}{\lambda} - 1) e^{-\frac{d_{ij}}{\lambda}} (\mathbf{x}^{(i)} - \mathbf{x}_{k-1}^{(j)})], \quad (20)$$

where λ, u_i and v_j are Lagrangian multipliers, and $d_{ij} = ||\mathbf{x}^{(i)} - \mathbf{x}_{k-1}^{(j)}||_2^2$. For the sake of simplicity, we do not update u_i and v_j, but instead use a fixed scaling factor γ to approximate the product $u_i v_j$.

Adversarial-Sample Generation Once an *adversarial distribution* is learned, adversarial samples, *e.g.* $\mathbf{x}_\ell^{(i)}$, can be obtained by drawing samples from it. However, adversarial samples typically follow certain constraints, *e.g.*, l_∞ bounded. We propose two adversarial-sample generation methods based on the particle-optimization formula above, named DAA-BLOB and DAA-DGF. DAA-BLOB substitutes the gradient used in PGD with the gradient derived in Eq. 17. Formally, in each iteration, $\mathbf{x}^i$ is updated by

$$\mathbf{x}_{l+1}^{(i)} = \Pi_{\mathsf{clip}}\{\mathbf{x}_l^{(i)} + \epsilon \cdot \mathrm{sign}(\nabla_{\mathbf{x}_l^i} \mathcal{L}(\mathbf{x}_l^{(i)}, y^{(i)}; \boldsymbol{\theta}) + \frac{c}{M}[\sum_{j=1}^{M} K(\mathbf{x}_l^{(i)}, \mathbf{x}_l^{(j)}) \nabla_{\mathbf{x}_l^{(j)}} \mathcal{L}(\mathbf{x}_l^{(j)}, y^{(j)}; \boldsymbol{\theta}) + \nabla_{\mathbf{x}_t^{(j)}} K(\mathbf{x}_l^{(i)}, \mathbf{x}_l^{(j)})])\}. \quad (21)$$

By contrast, DAA-DGF substitutes the gradient used in PGD with a combination of Eq. 19 and Eq. 20. Specifically, in each iteration, $\mathbf{x}^i$ is updated by

$$\mathbf{x}_{l+1}^{(i)} = \Pi_{\mathsf{clip}}\{\mathbf{x}_l^{(i)} + \epsilon \cdot \mathrm{sign}(\nabla_{\mathbf{x}_l^{(i)}} \mathcal{L}(\mathbf{x}_l^{(i)}, y^{(i)}; \boldsymbol{\theta}) - \frac{2\gamma c}{1+c} \cdot [\sum_{j=1}^{M} (\frac{d_{ij}}{\lambda} - 1) e^{-\frac{d_{ij}}{\lambda}} (\mathbf{x}_l^{(i)} - \mathbf{x}_l^{(j)})])\}. \quad (22)$$

Optimization by Data Subsampling In theory, the adversarial distribution μ to be optimized corresponds to a data manifold. Thus a good discrete approximation to μ is to use all the testing samples. In practice, however, it is computationally infeasible to update the particles following (21) or (22), as the complexity is $\mathcal{O}(M^2)$ for each particle update. To mitigate this issue, we propose a subsampling method to update testing samples in an unbiased and computationally feasible way: First, testing samples are randomly permuted and divided into finite number of minibatches; then samples in each minibatch are updated sequentially for a certain number of steps. This procedure is iterated for multiple rounds. The full algorithm is shown in Algorithm 1.

Connection with PGD There are two situations where the proposed DAA framework reduces to PGD. The first situation is when $c = 0$, where the second terms of both (21) and (22) become 0, making the two gradients degraded to the gradient used in PGD. The second case is when $M = 1$, meaning that DAA is equivalent to PGD when the data-manifold is approximated by only one sample. In this case, the second term of the gradient used in DAA-DGF becomes 0, and the second term of the gradient used in DAA-BLOB reduces to $c \cdot [K(\mathbf{x}_t^i, \mathbf{x}_t^i) \nabla_{\mathbf{x}_t^i} \mathcal{L}(\mathbf{x}_t^i, y^i; \boldsymbol{\theta}) + \nabla_{\mathbf{x}_t^j} K(\mathbf{x}_t^i, \mathbf{x}_t^i)] = c \cdot \nabla_{\mathbf{x}_t^i} \mathcal{L}(\mathbf{x}_t^i, y^i; \boldsymbol{\theta})$.

Algorithm 1 DAA algorithm (untargeted attack)

Require: A classifier with loss function $\mathcal{L}(\mathbf{x}^i, y^i; \boldsymbol{\theta})$; testing dataset $\{\mathbf{x}^i, y^i\}_{i=1}^N$; minibatch size M; step size ϵ; predefined final perturbation size α; total iterations L; rounds R; hyperparameter c or $\frac{2\gamma c}{1+c}$;
Random Start: $\mathbf{x}_0^i = \mathbf{x}^i + \boldsymbol{\gamma}^i$ ($\boldsymbol{\gamma}^i \sim \mathcal{U}(-\alpha, \alpha)$)
No Random Start: $\mathbf{x}_0^i = \mathbf{x}^i$
for r = 0 to $R - 1$ **do**
  Randomly permutate the testing samples
  for k = 0 to $L/R - 1$ **do**
    $l = rL/R + k$
    for j = 0 to $N/M - 1$ **do**
      Follow Eq. 21 or 22 to update the minibatch $\{\mathbf{x}_l^i, y_l^i\}$ ($i = jM + 1 \sim (j+1)M$), where $\Pi_{\mathsf{clip}}(\cdot) = \mathsf{clip}_{[0,1]}(\mathsf{clip}_{[\mathbf{x}^i - \alpha, \mathbf{x}^i + \alpha]}(\cdot))$ ($[0, 1]$ is the pixel value range, maybe $[-1, 1]$ or $[0, 255]$)
    end for
  end for
end for

Experiment

Setup

Datasets and Related Models The proposed DAA together with state-of-the-art methods, PGD and Momentum PGD, are evaluated and compared on four standard datasets, including MINST, Fashion MNIST (FMNIST), CIFAR10 and ImageNet. For MINST, the attack target is the state-of-the-art PGD-adversarially-trained MINST model provided by MIT MadryLab (Madry et al. 2017). The defense architecture contains a convolutional neural network (CNN) with two convolutional layers and a fully connected layer. For FMNIST, we adversarially train a model by PGD as the target model. The network architecture consists of four convolutional layers and a fully connected layer with batch normalization. For CIFAR10, MadryLab's PGD adversarially trained CIFAR10 model is adopted as the target model. The network architecture is a residual CNN consisting of five residual units and a fully connected layer. For ImageNet, we adopt the target model in (Kurakin, Goodfellow, and Bengio 2016c), which is an adversarially trained Inception ResNet-v2 model. *In addition, we also evaluate DAA on a provable defense model (Wong and Kolter 2018), with code also provided in the Github link (in abstract)*

Implementation Details For all the methods related to kernel functions, an RBF kernel $K(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|_2^2 / h)$ is adopted. The bandwidth is set as $h = med^2 / \log M$, same as the kernel used in (Liu and Wang 2016) and (Chen et al. 2018). Here med is the

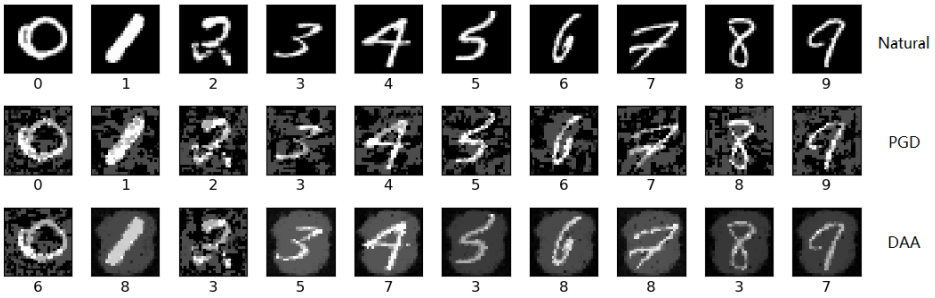

Figure 1: Comparison between PGD and DAA. DAA tends to generate more structured perturbations.

Table 1: Empirical worst-case accuracy of MIT MadryLab's secret **MNIST** model under 200-step attacks with 50 random starts ($0.3/1.0$ l_∞ perturbations). Loss 1: cross-entropy, Loss 2: CW_∞ loss

Rand+FGSM		PGD		Momentum PGD		DAA-BLOB		DAA-DGF	
Loss 1	Loss 2	Loss 1	Loss 2	Loss 1	Loss 2	Loss 1	Loss 2	Loss 1	Loss 2
93.48%	93.47%	89.49%	89.57%	89.29%	89.36%	**88.79%**	**88.85%**	**88.92%**	89.25%

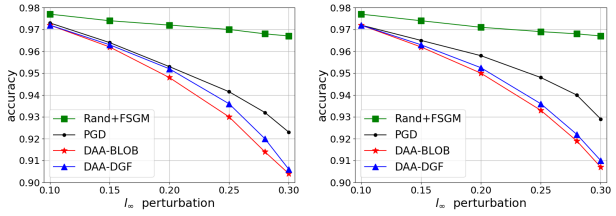

Figure 2: Averaged classification accuracy of MIT MadryLab's adversarially trained MNIST model under a single run of different attacks: cross-entropy (left), CW_∞ loss (right)

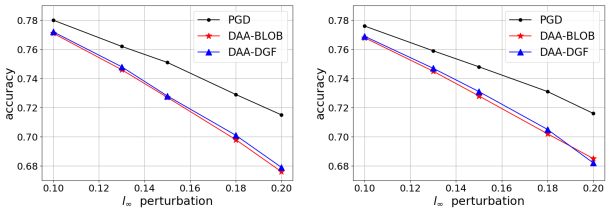

Figure 3: Averaged classification accuracy of adversarially trained FMNIST model under a single run of different attacks: cross-entropy (left), CW_∞ loss (right)

median of the pairwise distance between particles. The minibatch size (number of particles) is set to $100 \sim 200$ for computational feasibility. Our specific settings on hyper-parameters c and $\frac{2\gamma c}{1+c}$ can be found in our Github link (in abstract). It is worth noting that the discrepancy regarding the parameter choices on those datasets is caused by different pixel ranges and network structures used by their classifiers. All experiments are conducted on a single Titan V GPU under a white-box setting, where an adversary has full access to a target model including model weights.

Adversarial Perturbation Analysis

To intuitively understand the advantage of DAA over PGD, we plot ten natural samples and their PGD and DAA adversarial samples in Figure 1. For the ten samples, DAA successfully attacks the defense model with a $0.3/1.0$ l_∞ perturbation, whereas PGD with 50 random starts cannot. As shown in Figure 1, the perturbations generated by PGD tend to scatter throughout the images, whereas those of DAA are more structuredly focused around the target digits.

Empirical Results

MNIST We plot the averaged classification accuracy of MadryLab's adversarially trained MNIST model under a single run of different attack algorithms in Figure 2. It is observed that the proposed DAA consistently outperforms other methods under different levels of l_∞ perturbations and two different losses. To test its statistic significance, we also conduct paired t-tests between the accuracies reduced by PGD and DAA with random starts. For DAA-BLOB and DAA-DGF, the p-values are almost zeros, *i.e.*, 0.0 for DAA-BLOB and 5e-43 for DAA-DGF in the given decimal degree accuracy, suggesting that both methods outperform PGD in a statistical sense. In addition, we show the worst classification accuracy of Madry's adversarially trained model under PGD, Momentum PGD and DAA with 50 random restarts in Table 1. It is seen that DAA-BLOB is able to reduce the classification accuracy to approximately 88.79% (with $c = 1.1$ and minibatch size of 200), outperforming the attacks listed in MadryLab's white-box leaderboard.

FMNIST We next plot the classification accuracy of our adversarially trained FMNIST model under a single run of

Table 2: Empirical worst-case accuracy of adversarially trained **FMNIST** model under 100-step attacks with 10 random starts ($0.2/1.0$ l_∞ perturbations). Loss 1: cross-entropy, Loss 2: CW_∞ loss

Rand+FGSM		PGD		Momentum PGD		DAA-BLOB		DAA-DGF	
Loss 1	Loss 2	Loss 1	Loss 2	Loss 1	Loss 2	Loss 1	Loss 2	Loss 1	Loss 2
77.45%	77.21%	68.54%	68.94%	69.72%	69.51%	$\mathbf{65.70}\%$	66.64%	66.04%	66.60%

Table 3: Empirical worst-case accuracy of MIT MadryLab's adversarially trained **CIFAR10** model under a single run of 100-step attacks without random start (8, 16/255 l_∞ perturbations). Loss 1: cross-entropy, Loss 2: CW_∞ loss

l_∞	Rand+FGSM		PGD		Momentum PGD		DAA-BLOB		DAA-DGF	
	Loss 1	Loss 2	Loss 1	Loss 2	Loss 1	Loss 2	Loss 1	Loss 2	Loss 1	Loss 2
$8/255$	55.63%	55.05%	45.09%	46.27%	45.86%	46.76%	$\mathbf{44.98}\%$	46.30%	45.07%	46.31%
$16/255$	38.80%	37.78%	14.59%	16.06%	17.73%	18.70%	$\mathbf{14.43}\%$	16.05%	14.52%	16.06%

different attack algorithms in Figure 3. Similarly, the proposed DAA-based methods outperform the state-of-the-art PGD method. The two variants, DAA-BLOB and DAA-DGF, are comparable. Again, the p-values in the t-tests are also close to zero, *i.e.*, 0.0 for DAA-BLOB and 3e-27 for DAA-DGF, indicating significant performance differences between PGD and the proposed methods. The worst accuracy under white-box PGD, Momentum PGD and DAA with 10 random starts are listed in Table 2, suggesting the advance of the proposed DAA framework over both PGD and Momentum PGD.

CIFAR10 The classification accuracies of Madry's adversarially trained model under a single run of PGD, Momentum PGD and DAA without random start are shown in Table. 3. As can be seen, the adversarially-trained model only achieves weak robustness, *i.e.*, a 100-step PGD with $16/255$ l_∞ perturbations reduces the accuracy to 14.59%, while DAA only reduces it to 14.43%. We conjecture this is because the data are too complex and sparse, making the particle approximation with testing samples in DAA badly represent the true data manifold. As a result, testing results with the learned adversarial samples are similar to those of PGD. This argument is also validated by Recht et al. (2018), which shows that existing high-accuracy CIFAR10 classifiers does not generalize well to a truly unseen CIFAR10 testing set.

ImageNet We also evaluated the ensemble adversarial trained Inception ResNet-v2 under 50-step targeted Rand+FGSM and DAA attacks, using the least likely class as the target. Our experiments show that Rand+FGSM with $\mathbf{2/255}$ l_∞ *perturbations* (approximately 0.0157/2.0 l_∞ perturbations after normalization) can reduce the accuracy of the Inception model to approximately 70% (Kurakin, Goodfellow, and Bengio 2016b), while 50-step DAA can dramatically reduce it to 16.43%. This indicates the ensemble adversarial trained ImageNet model is still vulnerable to well-designed iterative attacks, *e.g.,* our DAA.

Discussion

To our knowledge, there was not a first-order l_∞ attack algorithm that can really outperform PGD under the white-box setting. Even for MI-FGSM, which won the NIP2017 competition under a black-box setting, we found that its PGD variant, which is stronger than MI-FGSM, cannot outperform standard PGD under the white-box setting, let alone MI-FGSM. In this paper, we generalize PGD on the space of data distributions. Our theoretical derivations and experiments validate the effectiveness of our framework. To the best of our knowledge, the proposed DAA framework is the first-and-only first-order l_∞ attack algorithm that can outperform PGD, especially on robust adversarially trained models. To further attack those adversarially-trained models with small l_∞ perturbations, we might have to involve higher-order information, which is usually very computationally expensive. In practice, those l_∞ adversarially trained models can also be further attacked by perturbing few pixels with large l_∞ perturbations, which still yields small l_1 or l_2 distance (Sharma and Chen 2018). However, such a change sometimes even leads to misclassification by human. Moreover, it is unfair to compare an l_1 or l_2 attack with an l_∞ attack (PGD) on l_∞ adversarially trained models.

Conclusion

We generalize PGD on the space of data distributions, by learning an adversarial data distribution that maximally increases the generalization risk of a model. To solve the adversarial-distribution problem, we define a new energy functional to better reflex the discriminative data manifold in the WGF framework. When adopting particle approximation, adversarial samples can be generated by solving the corresponding WGF problems, leading to an algorithm closely related to the standard PGD method. Extensive evaluations show that our distributionally-adversarial attack outperforms PGD and Momentum PGD, achieving state-of-the-art attack results on the adversarially trained defense and provable defense models that demonstrated notable robustness against first-order l_∞ attacks.

References

Ambrosio, L.; Gigli, N.; and Savaré, G. 2008. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media.

Athalye, A., and Carlini, N. 2018. On the robustness of the cvpr 2018 white-box adversarial example defenses. *arXiv preprint arXiv:1804.03286*.

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*.

Cai, Q.-Z.; Du, M.; Liu, C.; and Song, D. 2018. Curriculum adversarial training. *arXiv preprint arXiv:1805.04807*.

Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, 39–57. IEEE.

Carlini, N.; Katz, G.; Barrett, C.; and Dill, D. L. 2017. Ground-truth adversarial examples. *CoRR* abs/1709.10207.

Carrillo, J. A.; Craig, K.; and Patacchini, F. S. 2017. A blob method for diffusion. *arXiv preprint arXiv:1709.09195*.

Chen, C.; Zhang, R.; Wang, W.; Li, B.; and Chen, L. 2018. A unified particle-optimization framework for scalable bayesian sampling. *arXiv preprint arXiv:1805.11659*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. *arXiv preprint*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016a. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016b. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2016c. Adversarial machine learning at scale. *CoRR* abs/1611.01236.

Liu, Q., and Wang, D. 2016. Stein variational gradient descent: A general purpose bayesian inference algorithm. 2378–2386.

Liu, Y.; Ramachandran, P.; Liu, Q.; and Peng, J. 2017. Stein variational policy gradient. *CoRR* abs/1704.02399.

Ma, X.; Li, B.; Wang, Y.; Erfani, S. M.; Wijewickrema, S.; Houle, M. E.; Schoenebeck, G.; Song, D.; and Bailey, J. 2018. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*.

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*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2574–2582.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, 372–387. IEEE.

Recht, B.; Roelofs, R.; Schmidt, L.; and Shankar, V. 2018. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*.

Sharma, Y., and Chen, P.-Y. 2018. Attacking the madry defense model with l 1-based adversarial examples. In *Proc. of AAAI*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature* 529(7587):484–489.

Song, Y.; Kim, T.; Nowozin, S.; Ermon, S.; and Kushman, N. 2017. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*.

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.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

Wong, E., and Kolter, Z. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 5283–5292.