Stronger and Faster Wasserstein Adversarial Attacks

Kaiwen Wu [1 2] **Allen Houze Wang** [1 2] **Yaoliang Yu** [1 2]

Abstract

Deep models, while being extremely flexible and accurate, are surprisingly vulnerable to "small, imperceptible" perturbations known as adversarial attacks. While the majority of existing attacks focus on measuring perturbations under the ℓ_p metric, Wasserstein distance, which takes geometry in pixel space into account, has long been known to be a suitable metric for measuring image quality and has recently risen as a compelling alternative to the ℓ_p metric in adversarial attacks. However, constructing an effective attack under the Wasserstein metric is computationally much more challenging and calls for better optimization algorithms. We address this gap in two ways: (a) we develop an exact yet efficient projection operator to enable a stronger projected gradient attack; (b) we show that the Frank-Wolfe method equipped with a suitable linear minimization oracle works extremely fast under Wasserstein constraints. Our algorithms not only converge faster but also generate much stronger attacks. For instance, we decrease the accuracy of a residual network on CIFAR-10 to 3.4% within a Wasserstein perturbation ball of radius 0.005, in contrast to 65.6% using the previous Wasserstein attack based on an *approximate* projection operator. Furthermore, employing our stronger attacks in adversarial training significantly improves the robustness of adversarially trained models.

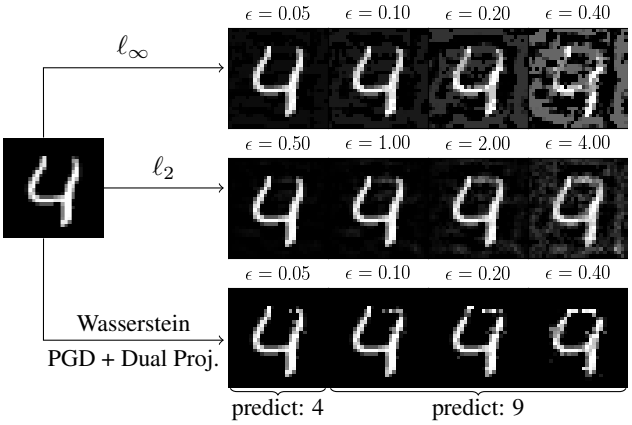


Figure 1: ℓ_∞, ℓ_2 and Wasserstein adversarial examples generated by projected gradient descent (PGD) with dual projection. While l_∞ and l_2 norm adversarial examples tend to perturb the background, Wasserstein adversarial examples *redistribute* the pixel mass.

1. Introduction

Deep models are surprisingly vulnerable to adversarial attacks, namely small or even imperceptible perturbations that completely change the prediction (Szegedy et al., 2014). The existence of adversarial examples has raised a lot of security concerns on deep models, and a substantial amount of work has devoted to this emerging field (Goodfellow et al., 2018), including various attacks as well as defences (*e.g.* Goodfellow et al., 2015; Papernot et al., 2016; Carlini & Wagner, 2017; Moosavi-Dezfooli et al., 2016; Kurakin et al., 2017; Madry et al., 2018). Some empirical defences are shown ineffective later under stronger attacks (Athalye et al., 2018), which has motivated a line of research on certified defences with *provable* guarantees (*e.g.* Wong & Kolter, 2018; Tjeng et al., 2019; Gowal et al., 2019; Raghunathan et al., 2018; Cohen et al., 2019).

The majority of existing work on adversarial robustness focused on the ℓ_p threat model where the perturbation is measured using the ℓ_p norm. However, the ℓ_p norm, despite being computationally convenient, is long known to be a poor proxy for measuring image similarity: two semantically similar images for human perception are not necessarily close under ℓ_p norm, see (*e.g.* Wang & Bovik, 2009) for some astonishing examples. To this end, threat models beyond ℓ_p norms have been proposed, *e.g.* Engstrom et al. (2019) explore geometric transformation to fool deep networks; Laidlaw & Feizi (2019) use point-wise functions on pixel values to flip predictions; Tramer & Boneh (2019) study robustness against multiple (ℓ_p) perturbations.

[1]David R. Cheriton School of Computer Science, University of Waterloo [2]Vector Institute. Correspondence to: Kaiwen Wu <kaiwen.wu@uwaterloo.ca>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Table 1: A summary of projected Sinkhorn and our proposed algorithms. **3rd and 4th column:** Computational complexity for a single iteration of each algorithm (without or with local transportation constraint). n is the dimension of inputs, and k is the local transportation region size (see §5.1). **5th column:** The exact convergence rate of projected Sinkhorn is not known yet. **Last column:** Whether the method is an exact or approximate algorithm.

method	optimization space	cost/iter	cost/iter (local)	convergence rate	exact?
projected Sinkhorn (Wong et al., 2019)	image space	$O(n^2)$	$O(nk^2)$	?	✗
dual projection (ours)	coupling matrix	$O(n^2 \log n)$	$O(nk^2 \log k)$	linear	✓
dual linear minimization oracle (ours)	coupling matrix	$O(n^2)$	$O(nk^2)$	linear	✗

In the same spirit, Wong et al. (2019) recently proposed the Wasserstein threat model, *i.e.*, adversarial examples are subject to a perturbation budget measured by the Wasserstein distance (a.k.a. earth mover's distance, see *e.g.* Peyré & Cuturi, 2019). The idea is to *redistribute* pixel mass instead of adjusting each pixel value as in previous ℓ_p threat models. Examples of Wasserstein adversarial attacks (generated by our algorithm) and comparison to ℓ_2 and ℓ_∞ adversarial attacks are shown in Figure 1. A key advantage of the former is that it explicitly captures geometric information in the image space (*i.e.* how mass moves around matters). For example, a slight translation or rotation of an image usually induces small change in the Wasserstein distance, but may change the l_p distance drastically. In addition, Wasserstein distance has played a pivotal role in generative adversarial networks (Arjovsky et al., 2017), computer vision (Rubner et al., 1997), and much beyond (Peyré & Cuturi, 2019).

Contributions Generating Wasserstein adversarial examples requires solving a Wasserstein constrained optimization problem. Wong et al. (2019) developed a projected gradient attack using *approximate* projection (projected Sinkhorn), which we find sometimes too crude and generating suboptimal attacks. In this work, we develop two stronger and faster attacks, based on reformulating the optimization problem (§2) and applying projected gradient descent (PGD) and Frank-Wolfe (FW), respectively. For the PGD attack, we design a specialized algorithm to compute the projection *exactly*, which significantly improves the attack quality (§3). For FW, we develop a *faster* algorithm to solve the linear minimization step with entropic smoothing (§4). Both subroutines enjoy fast linear convergence rates. Synthetic experiments on simple convex functions (Table 2) show that both algorithms are able to converge to high precision solutions. Extensive experiments on large scale datasets (§6) confirm the improved quality and speed of our attacks. In particular, for the first time we successfully construct Wasserstein adversarial examples on the ImageNet dataset. A quick comparison of projected Sinkhorn and our algorithms is shown in Table 1. Finally, we show that employing our stronger and faster attacks in adversarial training can significantly improve the robustness of adversarially trained models. Our implementation is available at `https://github.com/watml/fast-wasserstein-adversarial`.

2. Formulation

Wasserstein distance (a.k.a. earth mover's distance) is a metric defined on the space of finite measures with equal total mass (Peyré & Cuturi, 2019). For images, we view them as discrete measures supported on pixel locations. Let $\mathbf{x}, \mathbf{z} \in [0,1]^n$ be two vectorized images such that $\mathbf{1}^\top \mathbf{x} = \mathbf{1}^\top \mathbf{z}$ (equal mass). Their Wasserstein distance is defined as:

$$\mathcal{W}(\mathbf{x}, \mathbf{z}) = \min_{\Pi \geq 0} \langle \Pi, C \rangle \ \text{s.t.}\ \Pi \mathbf{1} = \mathbf{x}, \Pi^\top \mathbf{1} = \mathbf{z}, \quad (1)$$

where $C \in \mathbb{R}^{n \times n}$ is a cost matrix, with C_{ij} representing the cost of transportation from the i-th to the j-th pixel; and $\Pi \in \mathbb{R}^{n \times n}$ is a transportation/coupling matrix, with Π_{ij} representing the amount of mass transported from the i-th to the j-th pixel. Intuitively, Wasserstein distance measures the minimum cost to move mass from $\mathbf{x}$ to $\mathbf{z}$. Unlike usual statistical divergences (*e.g.* KL), Wasserstein distance takes the distance between pixels into account hence able to capture the underlying geometry. It has been widely used in statistics, image processing, graphics, machine learning, *etc.*, see the excellent monograph (Peyré & Cuturi, 2019).

Throughout the paper, w.l.o.g. we assume that all entries in the cost matrix C are nonnegative and $C_{ij} = 0 \Leftrightarrow i = j$. All common cost matrices satisfy this assumption.

2.1. PGD with projected Sinkhorn

Given a deep model that already minimizes some training loss $\mathbb{E}\ell(X, Y; \boldsymbol{\theta})$, we fix (hence also suppress in notation) the model parameter $\boldsymbol{\theta}$ and aim to generate adversarial examples by *maximizing* the loss ℓ subject to some perturbation budget on an input image $\mathbf{x}$. Following Wong et al. (2019), we use the Wasserstein distance to measure perturbations:

$$\underset{\mathbf{z} \in [0,1]^n}{\text{maximize}}\ \ell(\mathbf{z}, y) \ \text{s.t.}\ \mathcal{W}(\mathbf{x}, \mathbf{z}) \leq \delta = \epsilon \mathbf{1}^\top \mathbf{x}, \quad (2)$$

where the perturbation budget δ is proportional to the total mass in the input image $\mathbf{x}$ and ϵ indicates the "proportion". We focus on untargeted attack throughout, where y is the true label of the input image $\mathbf{x}$. All techniques in this paper can be easily adapted for targeted attacks as well.

To optimize (2), Wong et al. (2019) developed an *approximate* projection operator to the Wasserstein ball constraint,

Table 2: (Exact) Wasserstein distances $\mathcal{W}(\mathbf{a}, \hat{\mathbf{p}})$ and number of dual iterations in projected Sinkhorn in the first four columns. γ is the entropic regularization constant. Projected Sinkhorn encountered numerical issues for small $\gamma = 5 \cdot 10^{-5}$.

γ	10^{-3}		$2 \cdot 10^{-4}$		10^{-4}		$5 \cdot 10^{-5}$		ours		ground truth
	$\mathcal{W}$	iter	$\mathcal{W}$	iter	$\mathcal{W}$	iter	$\mathcal{W}$	iter	PGD	FW	
$\epsilon = 0.5$	0.267	28	0.402	44	0.437	205	–	–	0.500	0.500	0.500
$\epsilon = 1.0$	0.356	21	0.498	111	0.555	197	–	–	0.797	0.797	0.797

called projected Sinkhorn, to enable the projected gradient (PGD) adversarial attack (Madry et al., 2018). The approximate projection is based on solving an entropic regularized quadratic program (see Appendix A). However, we observed that this approximation is not always accurate in practice. To test this, we randomly generate two vectors $\mathbf{a}, \mathbf{b} \in [0, 1]^{400}$ with unit total mass. The initial Wasserstein distance between $\mathbf{a}$ and $\mathbf{b}$ is 0.797. Next, we project $\mathbf{b}$ using projected Sinkhorn onto Wasserstein balls centered at $\mathbf{a}$ with two different radii $\epsilon = 0.5$ and $\epsilon = 1.0$, respectively. We report in Table 2 the number of iterations for projected Sinkhorn to output an approximate projection $\hat{\mathbf{p}}$, and we compute the (exact) Wasserstein distance between $\mathbf{a}$ and $\hat{\mathbf{p}}$ using a linear programming solver for (1).

In the first row of Table 2, we expect $\mathcal{W}(\mathbf{a}, \hat{\mathbf{p}}) = 0.5$, since for an exterior point the exact projection should be at the boundary of the Wasserstein ball. In the second row, we expect $\mathcal{W}(\mathbf{a}, \hat{\mathbf{p}}) = 0.797$, since an exact projection of an interior point should be itself. However, in both cases, the actual Wasserstein distances of projected Sinkhorn are always much smaller than the ground truth. Although $\mathcal{W}(\mathbf{a}, \hat{\mathbf{p}})$ gets closer to the ground truth as γ (the entropic regularization constant) decreases, non-negligible gaps remain. Further decreasing γ may potentially reduce the approximation error, but (a) overly small γ causes numerical issues easily and (b) the number of iterations increases as γ decreases. As we will confirm in §6, this approximation error in projected Sinkhorn leads to a substantially suboptimal attack. In comparison, we minimize the quadratic objective in (Euclidean) projection iteratively by PGD with dual projection (§3) and by FW with dual LMO (§4). Their outputs are exact in the first three digits after the decimal point, which serves as a simple sanity check of our algorithms in the convex setting.

2.2. Our Reformulation

The large approximation error in projected Sinkhorn motivates us to develop more accurate algorithms for stronger attacks. First, we slightly reformulate (2) to simplify the constraint. We expand the constraint in (2), and jointly maximize the objective over $\mathbf{z}$ and Π:

$$\begin{aligned} &\underset{\mathbf{z}, \Pi \geq 0}{\text{maximize}} \;\; \ell(\mathbf{z}, y) \\ &\text{subject to } \Pi \mathbf{1} = \mathbf{x},\; \Pi^\top \mathbf{1} = \mathbf{z},\; \langle \Pi, C \rangle \leq \delta. \end{aligned}$$

Note that we have dropped the domain constraint $\mathbf{z} \in [0, 1]^n$, which we will revisit in §5.3. We plug in the constraint $\Pi^\top \mathbf{1} = \mathbf{z}$ into the objective to eliminate $\mathbf{z}$:

$$\underset{\Pi \geq 0}{\text{maximize}} \;\; \ell(\Pi^\top \mathbf{1}, y) \;\; \text{s.t. } \Pi \mathbf{1} = \mathbf{x},\; \langle \Pi, C \rangle \leq \delta, \quad (3)$$

arriving at a constrained optimization problem w.r.t. Π alone with two linear constraints on it. Yet, problem (2) and (3) are clearly equivalent. Moreover, problem (3) has its own interpretation: instead of maximizing the loss in the image space, it maximizes the loss in the transportation space, searching for a feasible transportation plan Π with cost no greater than δ, to transport $\mathbf{x}$ to an adversarial example. Given a solution Π to (3), we generate adversarial examples by summing over the columns of Π, *i.e.*, $\mathbf{x}_{adv} = \Pi^\top \mathbf{1}$. We note that $\nabla_\Pi \ell = \mathbf{1}(\nabla_{\mathbf{x}_{adv}} \ell)^\top$, where $\nabla_{\mathbf{x}_{adv}} \ell$ can be computed efficiently using backpropagation.

We propose PGD and FW to optimize (3). While both PGD and FW have been previously used to generate adversarial examples (Madry et al., 2018; Chen et al., 2020), they are based on the ℓ_p threat model that measures image perturbations under the ℓ_p distance. Instead, for the Wasserstein problem in (3), applying PGD and FW requires specialized algorithms for the projection operator and the linear minimization oracle, which are the main goals of §3 and §4.

3. Projected Gradient with Dual Projection

We apply PGD to maximize (3) for generating Wasserstein adversarial examples, and arrive at the following update rule on the coupling matrix Π:

$$\begin{aligned} \Pi^{(t+1)} &= \text{Proj}_{\mathcal{C}}\left(G\right), \;\; \text{where} \\ G &= \Pi^{(t)} + \eta_t \nabla_\Pi \ell\big((\Pi^{(t)})^\top \mathbf{1}, y\big) \end{aligned}$$

and $\text{Proj}_{\mathcal{C}}\left(\cdot\right)$ denotes the (Euclidean) projection operator onto the convex set $\mathcal{C}$, represented by the constraints in (3). Namely, we take a gradient step and then project it back to the feasible set. The projection operator $\text{Proj}_{\mathcal{C}}\left(G\right)$ is given by the following quadratic program:

$$\underset{\Pi \geq 0}{\text{minimize}} \;\; \tfrac{1}{2} \|\Pi - G\|_{\text{F}}^2 \;\; \text{s.t. } \Pi \mathbf{1} = \mathbf{x},\; \langle \Pi, C \rangle \leq \delta. \quad (4)$$

While any quadratic programming solver (*e.g.* interior point method) could be used to solve (4), they do not scale well

in high dimensions. For high resolution images, (4) could involve millions of variables. Since this projection is called in each iteration of PGD, it needs to be solved by a highly efficient algorithm. Below, we exploit the structure of this problem to design a fast specialized projection operator.

3.1. A First Attempt: Dykstra's Projection

A simple observation is that the constraint in (4) is precisely the intersection of the following two convex sets:

$$\mathcal{C}_s = \left\{\Pi \in \mathbb{R}^{n\times n} : \Pi \geq 0,\ \Pi\mathbf{1} = \mathbf{x}\right\}, \text{ and}$$
$$\mathcal{C}_h = \left\{\Pi \in \mathbb{R}^{n\times n} : \langle \Pi, C\rangle \leq \delta\right\},$$

where each row of $\mathcal{C}_s$ is a simplex constraint, requiring Wassserstein adversarial examples to preserve total mass; and $\mathcal{C}_h$ is a half space constraint, restricting the perturbation budget of Wasserstein adversarial examples. It is known that projection to the intersection of convex sets can be computed by Dykstra's algorithm (Boyle & Dykstra, 1986; Dykstra, 1983), provided that the projection to each convex set can be computed easily. Hence, our first attempt is to apply Dykstra's algorithm to solve (4) (see Appendix C for a description of Dykstra's algorithm and methods for projection to each convex set). However, the convergence rate of Dykstra's algorithm highly relies on the geometry of these two convex sets $\mathcal{C}_s$ and $\mathcal{C}_h$ (Deutsch & Hundal, 1994). In fact, we observe that Dykstra's algorithm converges very slowly in some cases (see Appendix C).

3.2. Dual Projection

Instead, we develop a dual projection method which converges much faster than Dykstra's algorithm. By the method of Lagrange multipliers, we derive the dual problem:

Proposition 1. *The dual of* (4) *is*

$$\underset{\lambda\geq 0}{\text{maximize}}\ g(\lambda), \quad \textit{where} \tag{5}$$
$$g(\lambda) = \min_{\Pi\mathbf{1}=\mathbf{x},\Pi\geq 0} \tfrac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 + \lambda\left(\langle \Pi, C\rangle - \delta\right). \tag{6}$$

In addition, the derivative of $g(\lambda)$ at a point $\lambda = \tilde{\lambda}$ is

$$g'(\tilde{\lambda}) = \langle \tilde{\Pi}, C\rangle - \delta, \quad \textit{where} \tag{7}$$
$$\tilde{\Pi} = \operatorname*{argmin}_{\Pi\mathbf{1}=\mathbf{x},\Pi\geq 0} \|\Pi - G + \tilde{\lambda}C\|_{\mathrm{F}}^2. \tag{8}$$

Both $g(\lambda)$ and $g'(\lambda)$ can be evaluated in $O(n^2\log n)$ time deterministically for any given λ.

The derivation of Proposition 1 is based on the observation that (6) is equivalent to computing a projection operator $\mathrm{Proj}_{\mathcal{C}_s}(G - \lambda C)$. Since the constraint $\mathcal{C}_s$ is independent for each row, it can be further reduced to projecting each row of $G - \lambda C$ to a simplex. The well-known simplex projection algorithm (*e.g.* Duchi et al., 2008) takes $O(n\log n)$ time, thus the projection to $\mathcal{C}_s$ takes $O(n^2\log n)$ time.

Algorithm 1: Dual Projection

Input: $G, C \in \mathbb{R}^{n\times n}$, $\mathbf{x}\in\mathbb{R}^n$, $\delta > 0$, $l = 0$, $u > 0$
Output: $\tilde{\Pi}\in\mathbb{R}^{n\times n}$

1 **while** *not converged* **do**
2 $\tilde{\lambda} = \frac{1}{2}(l+u)$
3 $\tilde{\Pi} = \operatorname{argmin}_{\Pi\mathbf{1}=\mathbf{x},\Pi\geq 0} \|\Pi - G + \tilde{\lambda}C\|_{\mathrm{F}}^2$
4 **if** $\langle\tilde{\Pi}, C\rangle > \delta$ **then** $l = \tilde{\lambda}$
5 **else** $u = \tilde{\lambda}$

Although this dual problem does not have a closed form expression, Proposition 1 provides a method to evaluate its objective and gradient, which is sufficient to use first order optimization algorithms. Here, we choose a simple algorithm with linear convergence rate, by exploiting the fact that the dual objective (6) is a univariate function. First, we derive an upper bound of the dual solution.

Proposition 2. *The dual solution $\lambda^\star$ of* (5) *satisfies*

$$0 \leq \lambda^\star \leq \frac{2\left\|\mathrm{vec}(G)\right\|_\infty + \left\|\mathbf{x}\right\|_\infty}{\min_{i\neq j}\{C_{ij}\}}. \tag{9}$$

Since $g(\lambda)$ is concave and differentiable, we make the following simple observation: (a) any point $l > 0$ with positive derivative is a lower bound of $\lambda^\star$; (b) any point $u > 0$ with negative derivative is an upper bound of $\lambda^\star$. Thus, we start with the lower bound and upper bound in Proposition 2, and use bisection method to search for $\lambda^\star$, by iteratively testing the sign of the derivative. Eventually, the bisection method converges to either a stationary point or the boundary $\lambda = 0$, which are exactly maximizers in both cases. Since the bisection method halves the gap between lower and upper bounds in each iteration, it converges linearly. Once we solve the dual, we can recover the primal solution by the following.

Proposition 3. *The primal solution $\Pi^\star$ and the dual solution $\lambda^\star$ satisfies*

$$\Pi^\star = \operatorname*{argmin}_{\Pi\mathbf{1}=\mathbf{x},\Pi\geq 0} \|\Pi - G + \lambda^\star C\|_{\mathrm{F}}^2,$$

thus $\Pi^\star$ can be computed in $O(n^2\log n)$ time given $\lambda^\star$.

The full dual projection is presented in Algorithm 1, where u is initialized as the upper bound (9). A discussion of the stopping criterion in Line 1 is deferred to Appendix B.

Finally, when the loss ℓ is convex (concave) in $\mathbf{x}$, it is also convex (concave) in Π after the reformulation in §2.2. In this case, projected gradient with dual projection is *guaranteed* to converge to a global optimum. When ℓ is nonconvex, projected gradient still converges to a stationary point.

4. Frank-Wolfe with Dual LMO

In this section, we apply the Frank-Wolfe algorithm (Frank & Wolfe, 1956) to maximize (3). During each iteration, FW first solves the following linear minimization problem:

$$\hat{\Pi} = \operatorname*{argmin}_{\Pi \in \mathcal{C}} \langle \Pi, H \rangle, \quad \text{where} \tag{10}$$

$$H = -\nabla_{\Pi} \ell\big((\Pi^{(t)})^\top \mathbf{1}, y\big) \tag{11}$$

and $\mathcal{C}$ is the convex set represented by constraints in (3). Then, we take a convex combination of $\Pi^{(t)}$ and $\hat{\Pi}$:

$$\Pi^{(t+1)} = (1-\eta_t)\Pi^{(t)} + \eta_t \hat{\Pi}.$$

Step (10) is referred as the linear minimization oracle (LMO) in the literature, and is reduced to solving the following linear program in each iteration:

$$\operatorname*{minimize}_{\Pi \geq 0} \ \langle \Pi, H \rangle \ \text{ s.t. } \ \Pi \mathbf{1} = \mathbf{x}, \ \langle \Pi, C \rangle \leq \delta. \tag{12}$$

Standard linear programming solvers do not scale well in high dimensions. Instead, we exploit the problem structure again to design a fast, specialized algorithm for (12).

4.1. A First Attempt: Optimizing the Dual

Our fist attempt is to extend the idea in §3 to the linear minimization step. We first derive an equivalent dual problem via the method of Lagrange multipliers:

Proposition 4. *The dual problem of* (12) *is*

$$\operatorname*{maximize}_{\lambda \geq 0} \ -\lambda\delta + \sum_{i=1}^{n} x_i \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij} \Big). \tag{13}$$

In addition, we provide an upper bound of the maximizer.

Proposition 5. *The dual solution* $\lambda^\star$ *of* (13) *satisfies*

$$0 \leq \lambda^\star \leq \frac{2 \left\| \mathrm{vec}(H) \right\|_\infty}{\min_{i \neq j} \{C_{ij}\}}. \tag{14}$$

Unlike dual projection, the dual objective (13) here is not differentiable. In fact, it is piecewise linear. Nevertheless, one can still solve it using derivative-free methods such as bisection method on the supergradient, or golden section search on the objective, both of which converge linearly.

However, after obtaining the dual solution, one cannot recover the primal solution easily. Consider the following recovery rule by minimizing the Lagrangian:

$$\Pi^\star \in \operatorname*{argmin}_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} \ \langle \Pi, H + \lambda^\star C \rangle - \lambda^\star \delta, \tag{15}$$

where $\Pi^\star$ and $\lambda^\star$ are primal and dual solutions respectively. There are two issues: (a) there might be infinitely many solutions to (15) and it is not easy to determine $\Pi^\star$ among them; (b) even if the solution to (15) is unique, a slight numerical perturbation could change the minimizer drastically. In practice, such instability may even result in an infeasible Π, generating *invalid* adversarial examples outside the Wasserstein perturbation budget. We direct readers to Appendix D for a concrete example and further discussions.

4.2. Dual LMO via Entropic Regularization

To address the above issues, we instead solve an entropic regularized version of (12) as an approximation:

$$\begin{aligned} \operatorname*{minimize}_{\Pi \geq 0} \ & \langle \Pi, H \rangle + \gamma \sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij} \log \Pi_{ij} \\ \text{subject to } & \Pi \mathbf{1} = \mathbf{x}, \ \langle \Pi, C \rangle \leq \delta, \end{aligned} \tag{16}$$

where γ is a regularization parameter. The new objective in (16) is strongly convex after the entropic regularization. For any dual variable, the corresponding primal variable minimizing the Lagrangian is always unique, which allows us to recover the primal solution from dual easily.

Proposition 6. *The dual problem of* (16) *is*

$$\begin{aligned} \operatorname*{maximize}_{\lambda \geq 0} \ & -\lambda\delta + \gamma \sum_{i : x_i > 0} x_i \log x_i \\ & - \gamma \sum_{i=1}^{n} x_i \log \sum_{j=1}^{n} \exp\left(-\frac{H_{ij} + \lambda C_{ij}}{\gamma} \right). \end{aligned} \tag{17}$$

The third term in (17) is essentially a softmin operator along each row of $H + \lambda C$, by observing that

$$\lim_{\gamma \to 0} -\gamma \log \sum_{j=1}^{n} \exp\left(-\tfrac{H_{ij} + \lambda C_{ij}}{\gamma} \right) = \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij} \Big).$$

Thus, from the point view of dual, (17) is a smooth approximation of (13), recovering (13) precisely as $\gamma \to 0$. In our implementation, we use the usual log-sum-exp trick to enhance the numerical stability of the softmin operator.

With entropic regularization, we have the following recovery rule for primal solution and upper bound on dual solution.

Proposition 7. *The primal solution* $\Pi^\star$ *and the dual solution* $\lambda^\star$ *satisfy*

$$\Pi^\star_{ij} = x_i \cdot \frac{\exp\left(-\frac{H_{ij} + \lambda^\star C_{ij}}{\gamma} \right)}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij} + \lambda^\star C_{ij}}{\gamma} \right)}. \tag{18}$$

Proposition 8. *The dual solution* $\lambda^\star$ *of* (17) *satisfies*

$$0 \leq \lambda^\star \leq \frac{\left[2 \left\| \mathrm{vec}(H) \right\|_\infty + \gamma \log\left(\frac{1}{\delta} \mathbf{x}^\top C \mathbf{1} \right) \right]_+}{\min_{i \neq j} \{C_{ij}\}}. \tag{19}$$

Note that the recovery rule (18) is essentially applying the softmin activation along each row of $H + \lambda^\star C$. With the upper bound (19), we can apply the same technique in §3 to solve the dual. In particular, the bisection method on the dual objective results in an algorithm almost identical to Algorithm 1 with only two exceptions: (a) we replace the upper bound (9) with (19) and (b) we replace Line 3 with the primal recover rule (18).

Unlike dual projection, the second order derivative of (17) can be computed in a closed form, which enables the usage of second order methods, *e.g.*, Newton's method, for further acceleration. However, we observed that Newton's method fails to converge in some cases. The smooth dual (17), as an approximation of (13), still behaves similar to a piecewise linear function after the regularization. Pure Newton's method might easily overshoot in this case. Thus, we still choose bisection method due to its stability and relatively fast convergence rate. In other applications where the convergence speed of the dual is a concern, it is possible to consider second order methods for acceleration.

Although both projected Sinkhorn (Wong et al., 2019) and dual LMO use entropic approximation, we emphasize that there are two key differences. First, the entropic regularization in dual LMO does not affect the convergence rate. In contrast, the convergence rate of projected Sinkhorn highly depends on γ. In particular, small γ often slows down the convergence rate empirically. Second, unlike the entropic regularized quadratic program used in projected Sinkhorn, entropic regularized linear program like (16) has been well studied. Applications in optimal transport (Cuturi, 2013) have demonstrated its empirical success; theoretical guarantees on the exponential decay of approximation error have been established (Cominetti & Martín, 1994; Weed, 2018).

For a thorough discussion on the convergence properties of FW on convex or nonconvex functions and beyond, we direct readers to (Yu et al., 2017).

5. Practical Considerations

In this section, we comment on some practical considerations for implementations of algorithms in §3 and §4.

5.1. Acceleration via Exploiting Sparsity

Both algorithms in §3 and §4 require computation on a full transportation matrix $\Pi \in \mathbb{R}^{n\times n}$, which is computationally expensive and memory consuming, especially in high dimensions. For example, for ImageNet where $n = 224 \times 224 \times 3$, the transportation matrix Π has billions of variables. The memory storage for it (using single precision floating numbers) exceeds the limit of most GPUs.

To accelerate computation and reduce memory usage, we enforce a local transportation constraint to impose a *structured sparsity* in the transportation matrix (Wong et al., 2019). Specifically, we only allow moving pixels in a $k \times k$ neighborhood (see Figure 2). For images with multiple channels, we only allow transportation within each channel. Adversarial examples generated with these restrictions are still valid under the original Wasserstein threat models.

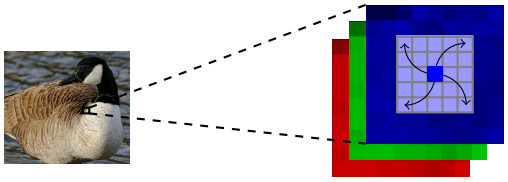


Figure 2: An illustration of 5×5 local transportation.

With local transportation, each pixel can be only transported to at most k^2 possible locations, thus Π is a highly sparse matrix with at most k^2 nonzero entries in each row. Such sparsity reduces the per iteration cost of dual projection from $O(n^2 \log n)$ to $O(nk^2 \log k)$, and reduces that of dual LMO from $O(n^2)$ to $O(nk^2)$, both of which are linear w.r.t. n, treating k (typically much smaller than n) as a constant.

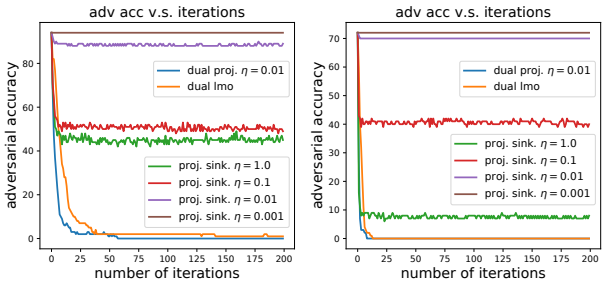
Operations on sparse matrices in general are not easy to parallelize on GPUs. Nevertheless, for dual projection and dual LMO, we do have simple customized strategies to support the sparse operations on GPUs by exploring the sparsity pattern in Π (see Appendix F.4 for a discussion).

5.2. Gradient Normalization

In both PGD and FW, we normalize the gradient in each iteration by its largest absolute value of entries. For PGD, gradient normalization is a standard practice, yielding consistent scale of gradient and easing step size tuning. For FW, normalization in the linear minimization step (12) does not change the minimizer, but it does affect the scale of the entropic regularization γ in (16). In this case, normalization keeps the scale of entropic regularization consistent. In addition, gradient normalization leads to more consistent upper bounds on the dual solutions in (9) and (19).

5.3. Hypercube Constraint in Image Domain

For image domain adversarial examples, there is an additional hypercube constraint, *e.g.*, $\mathbf{x}_{adv} \in [0, 1]^n$ if pixels are represented by real numbers in $[0, 1]$. In practice, we observe that solving problem (3) often generates adversarial examples that violate the hypercube constraint, *e.g.*, images with pixel values exceeding 1. Although simply clipping pixels can enforce the hypercube constraint, certain amount of pixel mass is lost during clipping, leading to undefined Wasserstein distance. This is in sharp contrast to ℓ_p threat models, where clipping is the typical practice that still retains validness of generated adversarial examples.



(a) CIFAR-10 (b) ImageNet

Figure 3: Adversarial accuracy of models w.r.t. different iterations of attacks using $\epsilon = 0.005$. Projected Sinkhorn uses $\gamma = 5 \cdot 10^{-5}$ on CIFAR-10 and $5 \cdot 10^{-6}$ on ImageNet. Dual LMO uses $\gamma = 10^{-3}$ and decay schedule $\frac{2}{t+1}$.

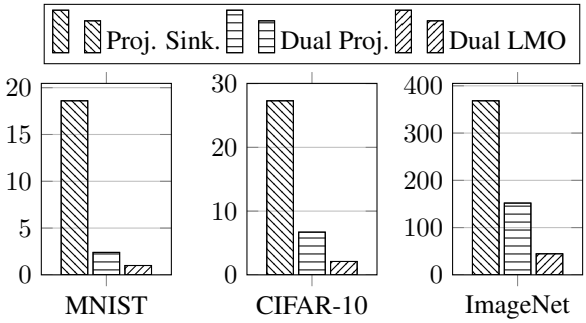


Figure 4: Per iteration running time (in milliseconds) of different algorithms measured on a single P100 GPU.

To address this issue, we develop another specialized quadratic programming solver to project a transportation matrix Π to the intersection of both constraints. To the best of our knowledge, the hypercube constraint has not been addressed in previous study of Wasserstein adversarial attacks. For instance, Wong et al. (2019) in their implementation simply applied clipping regardless. This new algorithm, however, is not as efficient as dual projection nor dual LMO. Thus, we recommend using it as a post-processing procedure on $\Pi^\star$ at the very end (see Appendix G). Adversarial attacks tend to be weaker after the post-processing. However, it ensures that the generated adversarial images satisfy both the Wasserstein constraint and the hypercube constraint simultaneously hence are genuinely valid.

6. Experiments

Datasets and models We conduct experiments on MNIST (LeCun, 1998), CIFAR-10 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009). On MNIST and CIFAR-10, we attack two deep networks used by Wong et al. (2019). On ImageNet, we attack a 50-layer residual network (He et al., 2016). ImageNet experiments are run on the first 100 samples in the validation set. See model details in appendix F.

Choice of cost Throughout the experiment, the cost matrix C is defined as $C_{(i_1,j_1)(i_2,j_2)} = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2}$, namely, the Euclidean distance between pixel indices. We use 5×5 local transportation plan (see §5.1) to accelerate computation of all algorithms.

Choice of γ We follow all parameter settings reported by Wong et al. (2019) for projected Sinkhorn, except that we try a few more γ. For dual LMO, we use a fixed $\gamma = 10^{-3}$, which we find to work well across different datasets.

Optimization parameters Stopping criteria of projected Sinkhorn, dual projection and dual LMO, as well as the choice of step sizes of PGD, are deferred to Appendix F. FW uses a fixed decay schedule $\eta_t = \frac{2}{t+1}$. Step sizes of PGD are tuned in $\{1, 10^{-1}, 10^{-2}, 10^{-3}\}$. Some experiments for different step sizes are presented in §6.1.

6.1. Convergence Speed of Outer Maximization

Our method depends on a different but equivalent formulation (3) that simplifies the constraint. It is reasonble to ask: could the reformulation make the outer maximization in PGD and FW harder? Intuitively, it does not, since the formulation simply embeds a linear transformation before the input layer (summation over columns of Π). To verify this, we plot adversarial accuracy of models w.r.t. number of iterations for different attack algorithms in Figure 3 to compare their convergence rate of outer maximization. More thorough results are deferred to Appendix F.5. [1]

We observe that FW with the default decay schedule converges very fast, especially at the initial stage. Meanwhile, PGD with dual projection converges slightly faster than FW with carefully tuned step sizes. In contrast, PGD with projected Sinkhorn barely decreases the accuracy when using small step sizes. The output of projected Sinkhorn is only a feasible point in the Wasserstein ball, rather than an accurate projection, due to the crude approximation. If using small step sizes, projected Sinkhorn brings the iterates of PGD closer to the center of Wasserstein ball in every iteration. Thus, the iterates always stay around the center of Wasserstein ball during the optimization, hence cannot decrease the accuracy. To make progress in optimization, it is required to use aggressively large step sizes (*e.g.* $\eta = 1.0$ and $\eta = 0.1$).

6.2. Attack Strength and Dual Convergence Speed

In Table 3, we compare (a) strength of different attacks by adversarial accuracy, *i.e.* model accuracy under attacks

[1] The numbers in Figure 3 are meant for comparison of convergence speed of Wasserstein constrained optimization problem. Thus they are shown without the post-processing algorithm discussed in §5.3 and may differ slightly from those in Table 3.

Table 3: Comparison of adversarial accuracy and average number of dual iterations. $\gamma = \frac{1}{1000}$ on MNIST and $\gamma = \frac{1}{3000}$ on CIFAR-10 are the parameters used by Wong et al. (2019). "–" indicates numerical issues during computation.

	method	$\epsilon = 0.1$		$\epsilon = 0.2$		$\epsilon = 0.3$		$\epsilon = 0.4$		$\epsilon = 0.5$	
		acc	iter	acc	iter	acc	iter	acc	iter	acc	iter
MNIST	PGD + Proj. Sink. ($\gamma = 1/1000$)	96.5	92	91.2	88	78.0	85	59.1	82	40.1	80
	PGD + Proj. Sink. ($\gamma = 1/1500$)	95.2	110	82.3	116	58.2	112	–	–	–	–
	PGD + Proj. Sink. ($\gamma = 1/2000$)	–	–	–	–	–	–	–	–	–	–
	PGD + Dual Proj.	63.4	15	13.3	15	1.4	15	0.1	15	0.0	15
	FW + Dual LMO ($\gamma = 10^{-3}$)	67.5	15	16.9	15	2.2	15	0.4	15	0.1	15
	PGD + Dual Proj. (w/o post-processing)	42.6	15	4.2	15	0.4	15	0.0	15	0.0	15
	FW + Dual LMO (w/o post-processing)	48.3	15	6.7	15	1.0	15	0.3	15	0.1	15
	method	$\epsilon = 0.001$		$\epsilon = 0.002$		$\epsilon = 0.003$		$\epsilon = 0.004$		$\epsilon = 0.005$	
		acc	iter	acc	iter	acc	iter	acc	iter	acc	iter
CIFAR-10	PGD + Proj. Sink. ($\gamma = 1/3000$)	93.0	33	91.3	33	89.5	33	87.6	33	85.7	33
	PGD + Proj. Sink. ($\gamma = 1/10000$)	89.9	79	84.5	79	78.3	79	71.9	79	65.6	79
	PGD + Proj. Sink. ($\gamma = 1/20000$)	–	–	–	–	–	–	–	–	–	–
	PGD + Dual Proj.	30.3	15	10.5	15	5.6	15	4.0	15	3.4	15
	FW + Dual LMO ($\gamma = 10^{-3}$)	33.5	15	13.6	15	7.2	15	4.7	15	3.7	15
	PGD + Dual Proj. (w/o post-processing)	25.9	15	6.0	15	1.7	15	0.5	15	0.2	15
	FW + Dual LMO (w/o post-processing)	29.6	15	9.1	15	3.2	15	1.1	15	0.6	15
	method	$\epsilon = 0.001$		$\epsilon = 0.002$		$\epsilon = 0.003$		$\epsilon = 0.004$		$\epsilon = 0.005$	
		acc	iter	acc	iter	acc	iter	acc	iter	acc	iter
ImageNet	PGD + Proj. Sink. ($\gamma = 1/100000$)	68.0	42	61.2	43	59.2	43	55.8	43	52.4	43
	PGD + Proj. Sink. ($\gamma = 1/200000$)	61.2	72	56.5	72	48.3	72	42.9	71	38.1	71
	PGD + Proj. Sink. ($\gamma = 1/1000000$)	–	–	–	–	–	–	–	–	–	–
	PGD + Dual Proj.	9.0	15	9.0	15	8.0	15	8.0	15	7.0	15
	FW + Dual LMO	10.0	15	9.0	15	8.0	15	8.0	15	7.0	15
	PGD + Dual Proj. (w/o post-processing)	0.0	15	0.0	15	0.0	15	0.0	15	0.0	15
	FW + Dual LMO (w/o post-processing)	1.0	15	0.0	15	0.0	15	0.0	15	0.0	15

and (b) the running speed by the average number of dual iterations. We observe that PGD with dual projection attack and FW with dual LMO attack are generally stronger than PGD with projected Sinkhorn, since the latter is only an *approximate* projection hence it does not solve (2) adequately. As γ decreases, PGD with projected Sinkhorn gradually becomes stronger due to better approximation, but at a cost of an increasing number of iterations to converge. However, projected Sinkhorn is still weaker than PGD with dual projection and FW with dual LMO, even after tuning γ. Unfortunately, further decreasing γ runs into numerical overflow. We notice that PGD with dual projection is often slightly stronger than FW with dual LMO for two possible reasons: the projection step is solved exactly without any approximation error; we use the default decay schedule in FW. Tuning the decay schedule for specific problems might improve the attack strength and convergence speed of FW.

For completeness, we also report the results of dual projection and dual LMO without post-processing in Table 3. After post-processing (see §5.3), the adversarial accuracy is increased, sometimes by a lot. This is especially the case on MNIST (*e.g.*, $\epsilon = 0.1$) where there are many white pixels, thus it is very easy to violate the hypercube constraint. Note that PGD with projected Sinkhorn might be even weaker than what is indicated by the statistics in Table 3, if we post-process its adversarial examples appropriately so that they are genuinely valid. However, we do not have an efficient algorithm for post-processing projected Sinkhorn, thus we simply let it ignore the hypercube constraint. Even so, our attacks are still much stronger than it.

Thanks to the bisection method and the tight upper bounds (9) and (19), dual projection and dual LMO converge very fast to high precision solutions. In practice, they often terminate exactly in 15 iterations due to the consistent scales of the upper bounds (see Appendix F.2 for a discussion). On the other hand, projected Sinkhorn typically requires more dual iterations. Besides convergence speed, we also compare the real running time of a single dual iteration of all

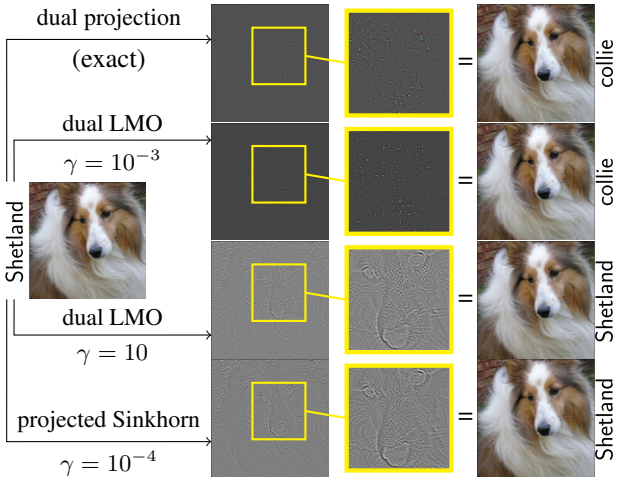


Figure 5: Wasserstein adversarial examples ($\epsilon = 0.005$) generated by different algorithms on ImageNet. Perturbations are normalized to $[0, 1]$ for visualization. Dog faces can be observed after zooming in.

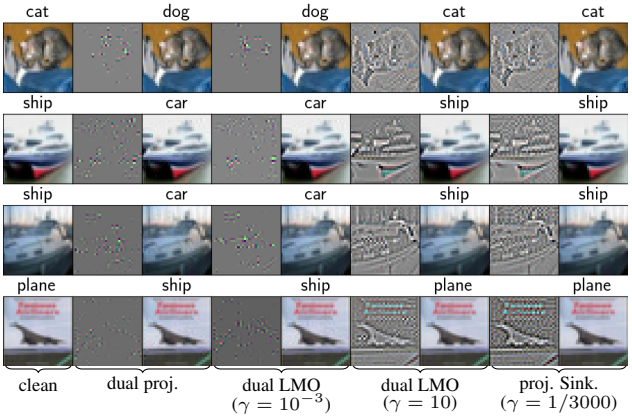


Figure 6: Wasserstein adversarial examples ($\epsilon = 0.002$) generated by different algorithms on CIFAR-10. Perturbations reflect shapes in images only when using large entropic regularization (the 6th and 8th columns).

three methods in Figure 4. On MNIST and CIFAR-10, dual projection is $5 \sim 7$ times faster than projected Sinkhorn; while on ImageNet, dual projection is roughly twice faster than projected Sinkhorn. Meanwhile, dual LMO is $2 \sim 3$ times faster than dual projection due to the absence of the extra logarithm factor. See Appendix F.9 for more details.

6.3. Entropic Regularization Reflects Shapes in Images

Wong et al. (2019) noted that Wasserstein perturbations reflect shapes in original images. Instead, we argue that it is the large entropic regularization that causes the phenomenon. We visualize adversarial examples and perturbations generated by different attacks in Figure 5 and Figure 6. Perturbations generated by PGD with dual projection and FW with dual LMO using small γ tend to be very sparse, *i.e.*, only moving a few pixels in the images. In comparison, we gradually increase the entropic regularization in dual LMO and eventually are able to reproduce the shape reflection phenomenon observed by Wong et al. (2019). The connection between entropic regularization and shape reflection phenomenon is deferred to Appendix F.8. The fact that projected Sinkhorn generates adversarial perturbations reflecting shapes in clean images could be another evidence that the entropic regularization is too large. Notice that large entropic regularization causes large approximation error, thus potentially requires larger ϵ in order to successfully generate adversarial examples.

6.4. Improved Adversarial Training

Since we have developed stronger and faster attacks, it is natural to apply them in adversarial training (Madry et al., 2018) to improve the robustness of adversarially trained models. Indeed, models adversarially trained by our stronger attack have much higher adversarial accuracy compared with models trained by projected Sinkhorn. We apply FW with dual LMO in adversarial training due to its fast convergence speed and fast per iteration running speed. On MNIST, we produce a robust model with 59.7% accuracy against all three attacks with perturbation budget $\epsilon = 0.5$, compared with 0.6% using projected Sinkhorn. On CIFAR-10, we produce a robust model with 56.8% accuracy against all three attacks with perturbation budget $\epsilon = 0.005$, compared with 41.2% using projected Sinkhorn. See Appendix F.11 for a thorough evaluation of adversarially trained models.

7. Conclusion

We have demonstrated that the previous Wasserstein adversarial attack based on *approximate* projection is suboptimal due to inaccurate projection. To generate stronger Wasserstein adversarial attacks, we introduce two *faster* and more *accurate* algorithms for Wasserstein constrained optimization. Each algorithm has its own advantage thus they complement each other nicely: PGD with dual projection employs exact projection and generates the strongest attack. On the other hand, with minimal entropic smoothing, FW with dual LMO is extremely fast in terms of both outer maximization and linear minimization step without much tuning of hyperparameters. Extensive experiments confirm the effectiveness of our algorithms in two ways: (a) properly evaluating Wasserstein adversarial robustness and (b) improving robustness through adversarial training. Finally, our algorithms impose minimal assumptions on the cost matrix in Wasserstein distance, thus they can be directly applied to other applications involving Wasserstein constrained optimization problems on discrete domains.

Acknowledgement

We thank the reviewers for their constructive comments. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. We gratefully acknowledge funding support from NSERC, the Canada CIFAR AI Chairs Program, and Waterloo-Huawei Joint Innovation Lab. We thank NVIDIA Corporation (the data science grant) for donating two Titan V GPUs that enabled in part the computation in this work.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 214–223, 2017.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 274–283, 2018.

Boyle, J. P. and Dykstra, R. L. A Method for Finding Projections onto the Intersection of Convex Sets in Hilbert Spaces. In *Advances in order restricted statistical inference*, pp. 28–47, 1986.

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

Chen, J., Zhou, D., Yi, J., and Gu, Q. A Frank-Wolfe Framework for Efficient and Effective Adversarial Attacks. In *AAAI*, 2020.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified Adversarial Robustness via Randomized Smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1310–1320, 2019.

Cominetti, R. and Martín, J. S. Asymptotic Analysis of the Exponential Penalty Trajectory in Linear Programming. *Mathematical Programming*, 67:169–187, 1994.

Cuturi, M. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems 26*, pp. 2292–2300, 2013.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Li, F.-F. ImageNet: A Large-scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Deutsch, F. and Hundal, H. The Rate of Convergence of Dykstra's Cyclic Projections Algorithm: The Polyhedral Case. *Numerical Functional Analysis and Optimization*, 15:537–565, 1994.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient Projections onto the ℓ_1-ball for Learning in High Dimensions. In *Proceedings of the 25th international conference on Machine learning*, pp. 272–279, 2008.

Dykstra, R. L. An Algorithm for Restricted Least Squares Regression. *Journal of the American Statistical Association*, 78:837–842, 1983.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. Exploring the Landscape of Spatial Robustness. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1802–1811, 2019.

Frank, M. and Wolfe, P. An Algorithm for Quadratic Programming. *Naval Research Logistics (NRL)*, 3:95–110, 1956.

Goodfellow, I., McDaniel, P., and Papernot, N. Making Machine Learning Robust Against Adversarial Inputs. *Communications of the ACM*, 61:56–66, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representation*, 2015.

Gowal, S., Dvijotham, K. D., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. Scalable Verified Training for Provably Robust Image Classification. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Technical Report, 2009.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial Machine Learning at Scale. In *International Conference on Learning Representations*, 2017.

Laidlaw, C. and Feizi, S. Functional Adversarial Attacks. In *Advances in Neural Information Processing Systems 32*, pp. 10408–10418, 2019.

LeCun, Y. The MNIST Dataset, 1998.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*, 2018.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, 2016.

Peyré, G. and Cuturi, M. *Computational Optimal Transport*. Foundations and Trends® in Machine Learning, 2019.

Raghunathan, A., Steinhardt, J., and Liang, P. S. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems 31*, pp. 10877–10887, 2018.

Rubner, Y., Guibas, L. J., and Tomasi, C. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. In *Proceedings of the ARPA image understanding workshop*, pp. 661–668, 1997.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *International Conference on Learning Representations*, 2019.

Tramer, F. and Boneh, D. Adversarial Training and Robustness for Multiple Perturbations. In *Advances in Neural Information Processing Systems 32*, pp. 5858–5868, 2019.

Wang, Z. and Bovik, A. C. Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures. *IEEE Signal Processing Magazine*, 26:98–117, 2009.

Weed, J. An explicit analysis of the entropic penalty in linear programming. In *Proceedings of the 31st Conference On Learning Theory*, pp. 1841–1855, 2018.

Wong, E. and Kolter, Z. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5286–5295, 2018.

Wong, E., Schmidt, F., and Kolter, Z. Wasserstein Adversarial Examples via Projected Sinkhorn Iterations. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 6808–6817, 2019.

Yu, Y., Zhang, X., and Schuurmans, D. Generalized Conditional Gradient for Sparse Estimation. *Journal of Machine Learning Research*, 18:1–46, 2017.