A. Projected Sinkhorn derivation

A.1. Proof of Lemma 1

Lemma 1. *The dual of the entropy-regularized Wasserstein projection problem in Equation (6) is*

$$\underset{\alpha,\beta\in\mathbb{R}^n,\psi\in\mathbb{R}_+}{\text{maximize}}\ g(\alpha,\beta,\psi) \tag{7}$$

where

$$\begin{aligned} g(\alpha,\beta,\psi) = &-\frac{1}{2\lambda}\|\beta\|_2^2 - \psi\epsilon + \alpha^T x + \beta^T w \\ &- \sum_{ij}\exp(\alpha_i)\exp(-\psi C_{ij} - 1)\exp(\beta_j) \end{aligned} \tag{8}$$

Proof. For convenience, we multiply the objective by λ and solve this problem instead:

$$\begin{aligned} \underset{z\in\mathbb{R}_+^n,\Pi\in\mathbb{R}_+^{n\times n}}{\text{minimize}}\ & \frac{\lambda}{2}\|w-z\|_2^2 + \sum_{ij}\Pi_{ij}\log(\Pi_{ij}) \\ \text{subject to}\ & \Pi 1 = x \\ & \Pi^T 1 = z \\ & \langle \Pi, C\rangle \le \epsilon. \end{aligned} \tag{15}$$

Introducing dual variables (α,β,ψ) where $\psi\ge 0$, the Lagrangian is

$$\begin{aligned} &L(z,\Pi,\alpha,\beta,\psi) \\ =&\frac{\lambda}{2}\|w-z\|_2^2 + \sum_{ij}\Pi_{ij}\log(\Pi_{ij}) + \psi(\langle\Pi,C\rangle - \epsilon) \\ &+\alpha^T(x-\Pi 1) + \beta^T(z-\Pi^T 1). \end{aligned} \tag{16}$$

The KKT optimality conditions are now

$$\begin{aligned} \frac{\partial L}{\partial \Pi_{ij}} &= \psi C_{ij} + (1+\log(\Pi_{ij})) - \alpha_i - \beta_j = 0 \\ \frac{\partial L}{\partial z_j} &= \lambda(z_j - w_j) + \beta_j = 0 \end{aligned} \tag{17}$$

so at optimality, we must have

$$\begin{aligned} \Pi_{ij} &= \exp(\alpha_i)\exp(-\psi C_{ij} - 1)\exp(\beta_j) \\ z &= -\frac{\beta}{\lambda} + w \end{aligned} \tag{18}$$

Plugging in the optimality conditions, we get

$$\begin{aligned} &L(z^*,\Pi^*,\alpha,\beta,\psi) \\ =&-\frac{1}{2\lambda}\|\beta\|_2^2 - \psi\epsilon + \alpha^T x + \beta^T w \\ &-\sum_{ij}\exp(\alpha_i)\exp(-\psi C_{ij} - 1)\exp(\beta_j) \\ =&g(\alpha,\beta,\psi) \end{aligned} \tag{19}$$

so the dual problem is to maximize g over $\alpha,\beta,\psi\ge 0$. □

A.2. Proof of Lemma 2

Lemma 2. *Suppose α^*,β^*,ψ^* maximize the dual problem g in Equation (8). Then,*

$$\begin{aligned} z_i^* &= w_i - \beta_i/\lambda \\ \Pi_{ij}^* &= \exp(\alpha_i^*)\exp(-\psi^* C_{ij} - 1)\exp(\beta_j^*) \end{aligned} \tag{13}$$

are the corresponding solutions that minimize the primal problem in Equation (6).

Proof. These equations follow directly from the KKT optimality conditions from Equation (18). □

A.3. Algorithm derivation and interpretation

Derivation To derive the algorithm, note that since this is a strictly convex problem to get the α and β iterates we solve for setting the gradient to 0. The derivative with respect to α is

$$\frac{\partial g}{\partial \alpha_i} = x_i - \exp(\alpha_i)\sum_j \exp(-\psi C_{ij} - 1)\exp(\beta_j) \tag{20}$$

and so setting this to 0 and solving for α_i gives the α iterate. The derivative with respect to β is

$$\frac{\partial g}{\partial \beta_j} = -\frac{1}{\lambda}\beta + w - \exp(\beta_j)\sum_i \exp(\alpha_i)\exp(-\psi C_{ij} - 1) \tag{21}$$

and setting this to 0 and solving for β_j gives the β iterate (this step can be done using a symbolic solver, we used Mathematica). Lastly, the ψ updates are straightforward scalar calculations of the derivative and second derivative.

Interpretation The original Sinkhorn iteration has a natural interpretation as a matrix scaling algorithm, iteratively rescaling the rows and columns of a matrix to achieve the target distributions. The Projected Sinkhorn iteration has a similar interpretation: while the α step rescales the rows of $\exp(-\psi C - 1)$ to sum to x, the β step rescales the columns of $\exp(-\psi C - 1)$ to sum to $-\beta/\lambda + w$, which is the primal transformation of the projected variable z at optimality as described in Lemma 2. Lastly, the ψ step can be interpreted as correcting for the transport cost of the current scaling: the numerator of the Newton step is simply the difference between the transport cost of the current matrix scaling and the maximum constraint ϵ.

To see this, recall the transformation of dual to primal variables from Lemma 2. We can interpret these quantities before optimality as primal iterates. Namely, at each iteration t, let

$$\begin{aligned} z_i^{(t)} &= w_i - \beta_i^{(t)}/\lambda \\ \Pi_{ij}^{(t)} &= \exp(\alpha^{(t)})\exp(-\psi^{(t)} C_{ij} - 1)\exp(\beta^{(t)}) \end{aligned} \tag{22}$$

Then, since the α and β steps are equivalent to setting Equations (20) and (21) to 0, we know that after an update for $\alpha^{(t)}$, we have that

$$x_i = \sum_j \Pi_{ij}^{(t)} \tag{23}$$

so the α step rescales the transport matrix to sum to x. Similarly, after an update for $\beta^{(t)}$, we have that

$$z_i^{(t)} = \sum_i \Pi_{ij}^{(t)} \tag{24}$$

which is a rescaling of the transport matrix to sum to the projected value. Lastly, the numerator of the $\psi^{(t)}$ step can be rewritten as

$$\psi^{(t+1)} = \psi^{(t)} + t \cdot \frac{\langle \Pi^{(t)}, C \rangle - \epsilon}{\langle \Pi^{(t)}, C \cdot C \rangle} \tag{25}$$

as a simple adjustment based on whether the current transport plan $\Pi^{(t)}$ is above or below the maximum threshold ϵ.

B. Experimental setup

B.1. MNIST

Adaptive ϵ During adversarial training for MNIST, we adopt an adaptive ϵ scheme to avoid selecting a specific ϵ. Specifically, to find an adversarial example, we first let $\epsilon = 0.1$ on the first iteration of projected gradient descent, and increase it by a factor of 1.4 every 5 iterations. We terminate the projected gradient descent algorithm when either an adversarial example is found, or when 50 iterations have passed, allowing ϵ to take on values in the range $[0.1, 2.1]$

Optimizer hyperparameters To update the model weights during adversarial training, we use the SGD optimizer with 0.9 momentum and 0.0005 weight decay, and batch sizes of 128. We begin with a learning rate of 0.1, reduce it to 0.01 after 10 epochs.

B.2. CIFAR10

Adaptive ϵ We also use an adaptive ϵ scheme for adversarial training in CIFAR10. Specifically, we let $\epsilon = 0.01$ on the first iteration of projected gradient descent, and increase it by a factor of 1.5 every 5 iterations. Similar to MNIST, we terminate the projected gradient descent algorithm when either an adversarial example is found, or 50 iterations have passed, allowing ϵ to take on values in the range $[0.01, 0.38]$.

Optimizer hyperparameters Similar to MNIST, to update the model weights, we use the SGD optimizer with 0.9 momentum and 0.0005 weight decay, and batch sizes of 128. The learning rate is also the same as in MNIST, starting at 0.1, and reducing to 0.01 after 10 epochs.

B.3. Motivation for adaptive ϵ

A commonly asked question of models trained to be robust against adversarial examples is "what if the adversary has a perturbation budget of $\epsilon + \delta$ instead of ϵ?" This is referring to a "robustness cliff," where a model trained against an ϵ strong adversary has a sharp drop in robustness when attacked by an adversary with a slightly larger budget. To address this, we advocate for the slightly modified version of typical adversarial training used in this work: rather than picking a fixed ϵ and running projected gradient descent, we instead allow for an adversarial to have a range of $\epsilon \in [\epsilon_{min}, \epsilon_{max}]$. To do this, we begin with $\epsilon = \epsilon_{\min}$, and then gradually increase it by a multiplicative factor γ until either an adversarial example is found or until ϵ_{max} is reached. While similar ideas have been used before for evaluating model robustness, we specifically advocate for using this schema *during adversarial training*. This has the advantage of extending robustness of the classifier beyond a single ϵ threshold, allowing a model to achieve a potentially higher robustness threshold while not being significantly harmed by "impossible" adversarial examples.

C. Auxiliary experiments

In this section, we explore the space of possible parameters that we treated as fixed in the main paper. While this is not an exhaustive search, we hope to provide some intuition as to why we chose the parameters we did.

C.1. Effect of λ and C

We first study the effect of λ and the cost matrix C. First, note that λ could be any positive value. Furthermore, note that to construct C we used the 2-norm which reflects the 1-Wasserstein metric, but in theory we could use any p-Wasserstein metric, where the the cost of moving from pixel (i, j) to (k, l) is $\left(|i-j|^2 + |k-l|^2\right)^{p/2}$. Figure 8 shows the effects of λ and p on both the adversarial example and the radius at which it was found for varying values of $\lambda = [1, 10, 100, 500, 1000]$ and $p = [1, 2, 3, 4, 5]$.

We find that it is important to ensure that λ is large enough, otherwise the projection of the image is excessively blurred. In addition to qualitative changes, smaller λ seems to make it harder to find Wasserstein adversarial examples, making the ϵ radius go up as λ gets smaller. In fact, for $\lambda = (1, 10)$ and almost all of $\lambda = 100$, the blurring is so severe that no adversarial example can be found.

In contrast, we find that increasing p for the Wasserstein distance used in the cost matrix C seems to make the images more "blocky". Specifically, as p gets higher tested, more pixels seem to be moved in larger amounts. This seems to counteract the blurring observed for low λ to some degree.

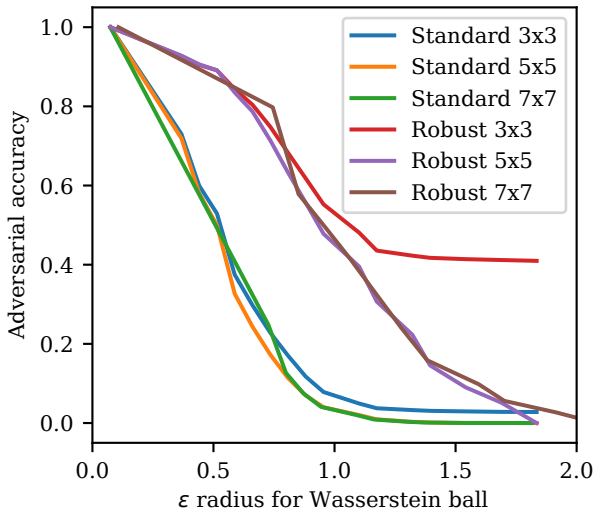

Figure 7. Adversarial accuracy of a standard model and a model trained to be provably robust against ℓ_∞ attacks for different sizes of transport plans. In most cases the size of the transport plan doesn't seem to matter, except for the 3×3 local transport plan. In this case, the adversary isn't quite able to reach 0% accuracy for the standard model, reaching 2.8% for for $\epsilon = 1.83$. The adversary is also unable to attack the robust MNIST model, bottoming out at 41% adversarial accuracy at $\epsilon = 1.83$.

Naturally, the ϵ radius also grows since the overall cost of the transport plan has gone up.

C.2. Size of local transport plan

In this section we explore the effects of different sized transport plans. In the main paper, we used a 5×5 local transport plan, but this could easily be something else, e.g. 3×3 or 7×7. We can see a comparison on the robustness of a standard and the ℓ_∞ robust model against these different sized transport plans in Figure 7, using $\lambda = 1000$. We observe that while 3×3 transport plans have difficulty attacking the robust MNIST model, all other plan sizes seem to have similar performance.

Once a transport plan is large enough to attack a model, allowing larger plans does not substantially change the attack success rate. Since the cost of transport grows as distance from the source pixel increases and optimal transport minimizes the transport cost, Wasserstein adversarial examples will tend to be local perturbations. While the size of the transport plans considered in the main paper are quite small, this is due to the weakness of networks to the adversarial attack. A 25×25 local transport plan takes 0.08 seconds per iteration on a 1080ti which can do 5×5 local transport plans in 0.02 seconds, and so local transport size is not the bottleneck.

D. Provable defense

In this section we show how a Sinkhorn-like algorithm can be derived for provable defenses, and that the resulting algorithm is actually just a simplified version of the Projected Sinkhorn iteration, which we call the Conjugate Sinkhorn iteration (since it solves the conjugate problem).

D.1. Conjugate Sinkhorn iteration

By subtracting the same entropy term to the conjugate objective from Equation (14), we can get a problem similar to that of projecting onto the Wasserstein ball.

$$\begin{aligned} \underset{z \in \mathbb{R}^n_+, \Pi \in \mathbb{R}^{n \times n}_+}{\text{minimize}} \quad & -\lambda z^T y + \sum_{ij} \Pi_{ij} \log(\Pi_{ij}) \\ \text{subject to} \quad & \Pi 1 = x \\ & \Pi^T 1 = z \\ & \langle \Pi, C \rangle \leq \epsilon. \end{aligned} \tag{26}$$

where again we've multiplied the objective by λ for convenience. Following the same framework as before, we introduce dual variables (α, β, ψ) where $\psi \geq 0$, to construct the Lagrangian as

$$\begin{aligned} & L(z, \Pi, \alpha, \beta, \psi) \\ =& -\lambda z^T y + \sum_{ij} \Pi_{ij} \log(\Pi_{ij}) + \psi(\langle \Pi, C \rangle - \epsilon) \\ & + \alpha^T (x - \Pi 1) + \beta^T (z - \Pi^T 1). \end{aligned} \tag{27}$$

Note that since all the terms with Π_{ij} are the same, the corresponding KKT optimality condition for Π_{ij} also remains the same. The only part that changes is the optimality condition for z, which becomes

$$\beta = \lambda y \tag{28}$$

Plugging the optimality conditions into the Lagrangian, we get the following dual problem:

$$\begin{aligned} & L(z^*, \Pi^*, \alpha, \beta, \psi) \\ =& -\psi\epsilon + \alpha^T x \\ & - \sum_{ij} \exp(\alpha_i) \exp(-\psi C_{ij} - 1) \exp(\beta_j) \\ =& g(\alpha, \psi) \end{aligned} \tag{29}$$

Finally, if we minimize this with respect to α and ψ we get exactly the same update steps as the Projected Sinkhorn iteration. Consequently, the Conjugate Sinkhorn iteration is identical to the Projected Sinkhorn iteration except that we replace the β step with the fixed value $\beta = \lambda y$.

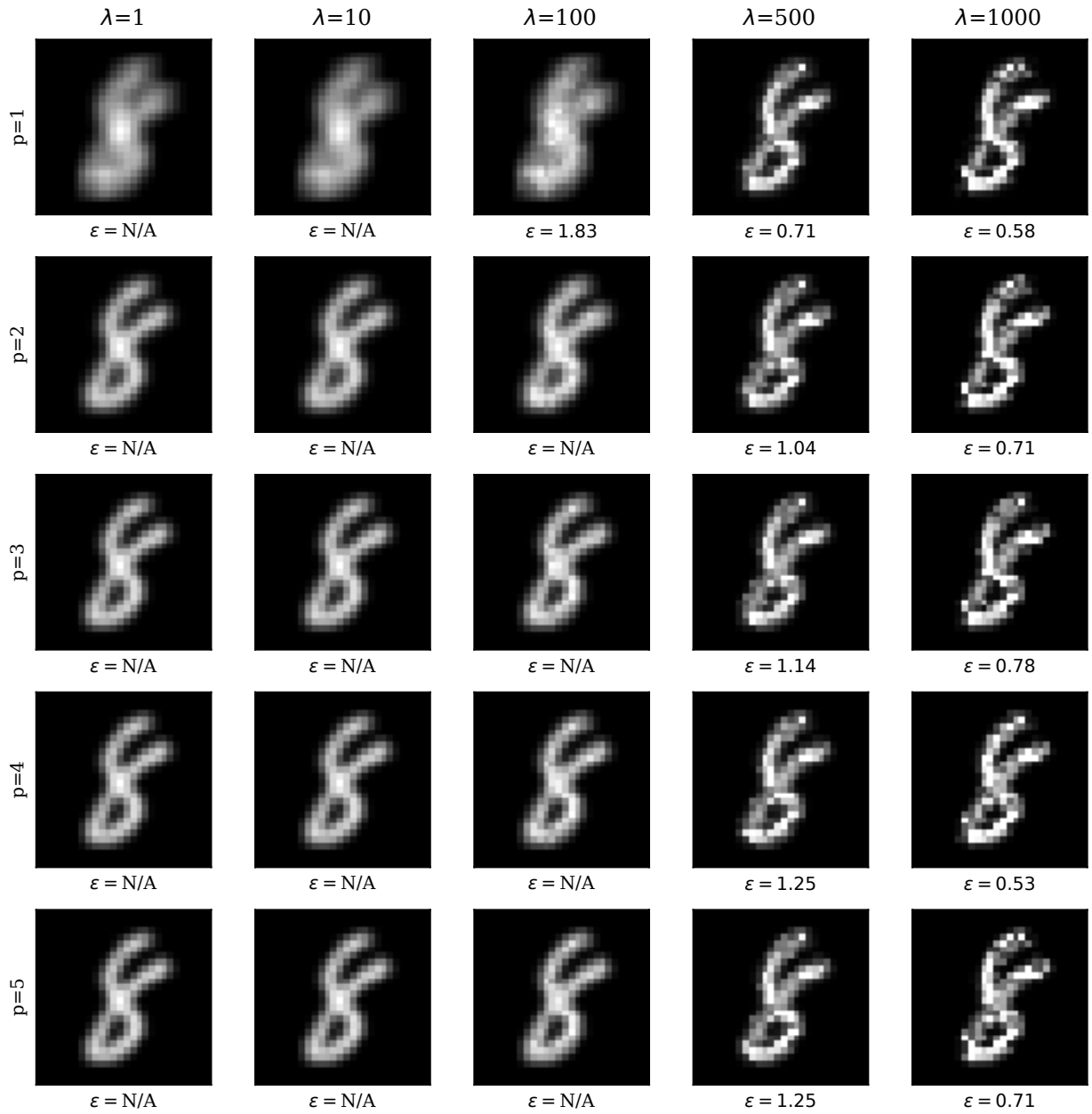

Figure 8. A plot of the adversarial examples generated with different p-Wasserstein metrics used for the cost matrix C and different regularization parameters λ. Note that when regularization is low, the image becomes blurred, and it is harder to find adversarial examples. In contrast, changing p does not seem to make any significant changes.