A. Adversarial Polytope

A.1. LP Formulation

Recall (4), which uses a convex outer bound of the adversarial polytope.

$$\underset{\hat{z}_k}{\text{minimize}}\, c^T\hat{z}_k, \;\; \text{subject to}\; \hat{z}_k \in \tilde{\mathcal{Z}}_\epsilon(x) \tag{18}$$

With the convex outer bound on the ReLU constraint and the adversarial perturbation on the input, this minimization problem is the following linear program

$$\begin{aligned}
&\underset{\hat{z}_k}{\text{minimize}}\, c^T\hat{z}_k, \;\; \text{subject to}\\
&\hat{z}_{i+1} = W_i z_i + b_i,\; i = 1,\ldots,k-1\\
&z_1 \le x + \epsilon\\
&z_1 \ge x - \epsilon\\
&z_{i,j} = 0,\; i = 2,\ldots,k-1, j \in \mathcal{I}_i^-\\
&z_{i,j} = \hat{z}_{i,j},\; i = 2,\ldots,k-1, j \in \mathcal{I}_i^+\\
&\left.\begin{aligned}
&z_{i,j} \ge 0,\\
&z_{i,j} \ge \hat{z}_{i,j},\\
&\Big((u_{i,j}-\ell_{i,j})z_{i,j}\\
&\quad - u_{i,j}\hat{z}_{i,j}\Big) \le -u_{i,j}\ell_{i,j}
\end{aligned}\right\}\; i = 2,\ldots,k-1, j \in \mathcal{I}_i
\end{aligned} \tag{19}$$

A.2. Proof of Theorem 1

In this section we derive the dual of the LP in (19), in order to prove Theorem 1, reproduced below:

Theorem. *The dual of* (4) *is of the form*

$$\begin{aligned}
&\underset{\alpha}{\text{maximize}} \;\; J_\epsilon(x, g_\theta(c,\alpha))\\
&\text{subject to} \;\; \alpha_{i,j} \in [0,1],\; \forall i,j
\end{aligned} \tag{20}$$

where $J_\epsilon(x,\nu) =$

$$-\sum_{i=1}^{k-1}\nu_{i+1}^T b_i - x^T\hat{\nu}_1 - \epsilon\|\hat{\nu}_1\|_1 + \sum_{i=2}^{k-1}\sum_{j\in\mathcal{I}_i}\ell_{i,j}[\nu_{i,j}]_+ \tag{21}$$

and $g_\theta(c,\alpha)$ *is a* k *layer feedforward neural network given by the equations*

$$\begin{aligned}
&\nu_k = -c\\
&\hat{\nu}_i = W_i^T\nu_{i+1}, \;\; \text{for}\; i = k-1,\ldots,1\\
&\nu_{i,j} = \begin{cases} 0 & j \in \mathcal{I}_i^-\\ \hat{\nu}_{i,j} & j \in \mathcal{I}_i^+\\ \frac{u_{i,j}}{u_{i,j}-\ell_{i,j}}[\hat{\nu}_{i,j}]_+ - \alpha_{i,j}[\hat{\nu}_{i,j}]_- & j \in \mathcal{I}_i, \end{cases}\\
&\qquad\qquad \text{for}\; i = k-1,\ldots,2
\end{aligned} \tag{22}$$

where ν *is shorthand for* $(\nu_i, \hat{\nu}_i)$ *for all* i *(needed because the objective* J *depends on* all ν *terms, not just the first), and where* $\mathcal{I}_i^-$, $\mathcal{I}_i^+$, *and* $\mathcal{I}_i$ *denote the sets of activations in layer* i *where the lower and upper bounds are both negative, both positive, or span zero respectively.*

Proof. In detail, we associate the following dual variables with each of the constraints

$$\begin{aligned}
\hat{z}_{i+1} = W_i z_i + b_i &\Rightarrow \nu_{i+1} \in \mathbb{R}^{|\hat{z}_{i+1}|}\\
z_1 \le x + \epsilon &\Rightarrow \xi^+ \in \mathbb{R}^{|x|}\\
-z_1 \le -x + \epsilon &\Rightarrow \xi^- \in \mathbb{R}^{|x|}\\
-z_{i,j} \le 0 &\Rightarrow \mu_{i,j} \in \mathbb{R}\\
\hat{z}_{i,j} - z_{i,j} \le 0 &\Rightarrow \tau_{i,j} \in \mathbb{R}\\
-u_{i,j}\hat{z}_{i,j} + (u_{i,j}-\ell_{i,j})z_{i,j} \le -u_{i,j}\ell_{i,j} &\Rightarrow \lambda_{i,j} \in \mathbb{R}
\end{aligned} \tag{23}$$

where we note that can easily eliminate the dual variables corresponding to the $z_{i,j} = 0$ and $z_{i,j} = \hat{z}_{i,j}$ from the optimization problem, so we don't define explicit dual variables for these; we also note that $\mu_{i,j}$, $\tau_{i,j}$, and $\lambda_{i,j}$ are only defined for i,j such that $j \in \mathcal{I}_i$, but we keep the notation as above for simplicity. With these definitions, the dual problem becomes

$$\begin{aligned}
&\text{maximize}\Big(-(x+\epsilon)^T\xi^+ + (x-\epsilon)^T\xi^-\\
&\qquad -\sum_{i=1}^{k-1}\nu_{i+1}^T b_i + \sum_{i=2}^{k-1}\lambda_i^T(u_i\ell_i)\Big)\\
&\text{subject to}\\
&\qquad \nu_k = -c\\
&\qquad \nu_{i,j} = 0,\; j \in \mathcal{I}_i^-\\
&\qquad \nu_{i,j} = (W_i^T\nu_{i+1})_j, \;\; j \in \mathcal{I}_i^+\\
&\left.\begin{aligned}
&\Big((u_{i,j}-\ell_{i,j})\lambda_{i,j}\\
&\quad -\mu_{i,j} - \tau_{i,j}\Big) = (W_i^T\nu_{i+1})_j\\
&\quad \nu_{i,j} = u_{i,j}\lambda_{i,j} - \mu_i
\end{aligned}\right\}\; \begin{aligned}&i = 2,\ldots,k-1\\ &j \in \mathcal{I}_i\end{aligned}\\
&\qquad W_1^T\nu_2 = \xi^+ - \xi^-\\
&\qquad \lambda,\tau,\mu,\xi^+,\xi^- \ge 0
\end{aligned} \tag{24}$$

The key insight we highlight here is that *the dual problem can also be written in the form of a deep network*, which provides a trivial way to find feasible solutions to the dual problem, which can then be optimized over. Specifically, consider the constraints

$$\begin{aligned}
(u_{i,j}-\ell_{i,j})\lambda_{i,j} - \mu_{i,j} - \tau_{i,j} &= (W_i^T\nu_{i+1})_j\\
\nu_{i,j} &= u_{i,j}\lambda_{i,j} - \mu_i.
\end{aligned} \tag{25}$$

Note that the dual variable λ corresponds to the upper bounds in the convex ReLU relaxation, while μ and τ correspond to the lower bounds $z \geq 0$ and $z \geq \hat{z}$ respectively; by the complementarity property, we know that at the optimal solution, these variables will be zero if the ReLU constraint is non-tight, or non-zero if the ReLU constraint is tight. Because we cannot have the upper and lower bounds be simultaneously tight (this would imply that the ReLU input $\hat{z}$ would exceed its upper or lower bound otherwise), we know that either λ or $\mu + \tau$ must be zero. This means that at the optimal solution to the dual problem

$$\begin{aligned}(u_{i,j} - \ell_{i,j})\lambda_{i,j} &= [(W_i^T \nu_{i+1})_j]_+ \\ \tau_{i,j} + \mu_{i,j} &= [(W_i^T \nu_{i+1})_j]_-\end{aligned} \tag{26}$$

i.e., the dual variables capture the positive and negative portions of $(W_i^T \nu_{i+1})_j$ respectively. Combining this with the constraint that

$$\nu_{i,j} = u_{i,j}\lambda_{i,j} - \mu_i \tag{27}$$

means that

$$\nu_{i,j} = \frac{u_{i,j}}{u_{i,j} - \ell_{i,j}}[(W_i^T \nu_{i+1})_j]_+ - \alpha[(W_i^T \nu_{i+1})_j]_- \tag{28}$$

for $j \in \mathcal{I}_i$ and for some $0 \leq \alpha \leq 1$ (this accounts for the fact that we can either put the "weight" of $[(W_i^T \nu_{i+1})_j]_-$ into μ or τ, which will or will not be passed to the next ν_i). This is exactly a type of leaky ReLU operation, with a slope in the positive portion of $u_{i,j}/(u_{i,j} - \ell_{i,j})$ (a term between 0 and 1), and a negative slope anywhere between 0 and 1. Similarly, and more simply, note that ξ^+ and ξ^- denote the positive and negative portions of $W_1^T \nu_2$, so we can replace these terms with an absolute value in the objective. Finally, we note that although it is possible to have $\mu_{i,j} > 0$ and $\tau_{i,j} > 0$ simultaneously, this corresponds to an activation that is identically zero pre-ReLU (both constraints being tight), and so is expected to be relatively rare. Putting this all together, and using $\hat{\nu}$ to denote "pre-activation" variables in the dual network, we can write the dual problem in terms of the network

$$\begin{aligned}&\nu_k = -c \\ &\hat{\nu}_i = W_i^T \nu_{i+1}, i = k-1, \ldots, 1 \\ &\nu_{i,j} = \begin{cases} 0 & j \in \mathcal{I}_i^- \\ \hat{\nu}_{i,j} & j \in \mathcal{I}_i^+ \\ \frac{u_{i,j}}{u_{i,j}-\ell_{i,j}}[\hat{\nu}_{i,j}]_+ - \alpha_{i,j}[\hat{\nu}_{i,j}]_- & j \in \mathcal{I}_i, \end{cases} \\ &\qquad\qquad\qquad\qquad \text{for } i = k-1, \ldots, 2\end{aligned} \tag{29}$$

which we will abbreviate as $\nu = g_\theta(c, \alpha)$ to emphasize the fact that $-c$ acts as the "input" to the network and α are per-layer inputs we can also specify (for only those activations in $\mathcal{I}_i$), where ν in this case is shorthand for all the ν_i and $\hat{\nu}_i$ activations.

The final objective we are seeking to optimize can also be written

$$\begin{aligned}J_\epsilon(x, \nu) = &-\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - (x+\epsilon)^T[\hat{\nu}_1]_+ + (x-\epsilon)^T[\hat{\nu}_1]_- \\ &+ \sum_{i=2}^{k-1}\sum_{j\in\mathcal{I}_i} \frac{u_{i,j}\ell_{i,j}}{u_{i,j} - \ell_{i,j}}[\hat{\nu}_{i,j}]_+ \\ = &-\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T\hat{\nu}_1 - \epsilon\|\hat{\nu}_1\|_1 \\ &+ \sum_{i=2}^{k-1}\sum_{j\in\mathcal{I}_i} \ell_{i,j}[\nu_{i,j}]_+\end{aligned} \tag{30}$$

□

A.3. Justification for Choice in α

While any choice of α results in a lower bound via the dual problem, the specific choice of $\alpha = \frac{u_{i,j}}{u_{i,j}-\ell_{i,j}}$ is also motivated by an alternate derivation of the dual problem from the perspective of general conjugate functions. We can represent the adversarial problem from (2) in the following, general formulation

$$\begin{aligned}\text{minimize } & c^T\hat{z}_k + f_1(z_1) + \sum_{i=2}^{k-1} f_i(\hat{z}_i, z_i) \\ \text{subject to } & \hat{z}_{i+1} = W_i z_i + b_i, \ i = 1, \ldots, k-1\end{aligned} \tag{31}$$

where f_1 represents some input condition and f_i represents some non-linear connection between layers. For example, we can take $f_i(\hat{z}_i, z_i) = I(\max(\hat{z}_i, 0) = z_i)$ to get ReLU activations, and take f_1 to be the indicator function for an ℓ_∞ ball with radius ϵ to get the adversarial problem in an ℓ_∞ ball for a ReLU network.

Forming the Lagrangian, we get

$$\begin{aligned}\mathcal{L}(z, \nu, \xi) = &\, c^T\hat{z}_k + \nu_k^T\hat{z}_k + f_1(z_1) - \nu_2^T W_1 z_1 \\ &+ \sum_{i=2}^{k-1}\left(f_i(\hat{z}_i, z_i) - \nu_{i+1}^T W_i z_i + \nu_i^T\hat{z}_i\right) \\ &- \sum_{i=1}^{k-1}\nu_{i+1}^T b_i\end{aligned} \tag{32}$$

Conjugate functions We can re-express this using conjugate functions defined as

$$f^*(y) = \max_x y^T x - f(x)$$

but specifically used as

$$-f^*(y) = \min_x f(x) - y^T x$$

Plugging this in, we can minimize over each $\hat{z}_i, z_i$ pair independently

$$\begin{aligned}
&\min_{z_1} f_1(z_1) - \nu_2^T W_1 z_1 = -f_1^*(W_1^T \nu_2) \\
&\min_{\hat{z}_i, z_i} f_i(\hat{z}_i, z_i) - \nu_{i+1}^T W_i z_i + \nu_i^T \hat{z}_i \\
&\qquad = -f_i^*(-\nu_i, W_i^T \nu_{i+1}),\ i = 2, \ldots, k-1 \\
&\min_{\hat{z}_k} c^T \hat{z}_k + \nu_k^T \hat{z}_k = I(\nu_k = -c)
\end{aligned} \tag{33}$$

Substituting the conjugate functions into the Lagrangian, and letting $\hat{\nu}_i = W_i^T \nu_{i+1}$, we get

$$\begin{aligned}
\underset{\nu}{\text{maximize}} \quad & -f_1^*(\hat{\nu}_1) - \sum_{i=2}^{k-1} f_i^*(-\nu_i, \hat{\nu}_i) - \sum_{i=1}^{k-1} \nu_{i+1}^T b_i \\
\text{subject to} \quad & \nu_k = -c \\
& \hat{\nu}_i = W_i^T \nu_{i+1},\ i = 1, \ldots, k-1
\end{aligned} \tag{34}$$

This is almost the form of the dual network. The last step is to plug in the indicator function for the outer bound of the ReLU activation (we denote the ReLU polytope) for f_i and derive f_i^*.

ReLU polytope Suppose we have a ReLU polytope

$$\begin{aligned}
\mathcal{S}_i = \{(\hat{z}_i, z_i) : \hat{z}_{i,j} &\geq 0, \\
z_{i,j} &\geq \hat{z}_{i,j}, \\
-u_{i,j}\hat{z}_{i,j} + (u_{i,j} - \ell_{i,j}) z_{i,j} &\leq -u_{i,j}\ell_{i,j}\}
\end{aligned} \tag{35}$$

So $I_\mathcal{S}$ is the indicator for this set, and $I_\mathcal{S}^*$ is its conjugate. We will omit subscripts (i, j) for brevity, but we can do this case by case elementwise.

1. If $u \leq 0$ then $\mathcal{S} \subset \{(\hat{z}, z) : z = 0\}$.
   Then, $I_\mathcal{S}^*(\hat{y}, y) \leq \max_{\hat{z}} \hat{y} \cdot \hat{z} = I(\hat{y} = 0)$.

2. If $\ell \geq 0$ then $\mathcal{S} \subset \{(\hat{z}, z) : \hat{z} = z\}$.
   Then, $I_\mathcal{S}^*(\hat{y}, y) \leq \max_z \hat{y} \cdot z + y \cdot z = (\hat{y} + y)z = I(\hat{y} + y = 0)$.

3. Otherwise $\mathcal{S} = \{(\hat{z}, z) : \hat{z} \geq 0, z \geq \hat{z}, -u\hat{z} + (u - \ell)z = -u\ell\}$. The maximum must occur either on the line $-u\hat{z} + (u - \ell)z = -u\ell$ over the interval $[0, u]$, or at the point $(\hat{z}, z) = (0, 0)$ (so the maximum must have value at least 0). We proceed to examine this last case.

Let $\mathcal{S}$ be the set of the third case. Then:

$$\begin{aligned}
&I_\mathcal{S}^*(\hat{y}, y) \\
&= \left[\max_{0 < \hat{z} < u} y \cdot \frac{u}{u - \ell}(\hat{z} - \ell) + \hat{y} \cdot \hat{z}\right]_+ \\
&= \left[\max_{0 < \hat{z} < u} \left(\frac{u}{u - \ell} y + \hat{y}\right)\hat{z} - \frac{u\ell}{u - \ell} y\right]_+ \\
&= \left[\max_{0 < \hat{z} < u} y \cdot \frac{u}{u - \ell}(\hat{z} - \ell) + \hat{y} \cdot \hat{z} = g(\hat{y}, y)\right]_+ \\
&= \begin{cases} \left[-\dfrac{u\ell}{u - \ell} y\right]_+ & \text{if } \dfrac{u}{u - \ell} y + \hat{y} \leq 0 \\ \left[\left(\dfrac{u}{u - \ell} y + \hat{y}\right) u - \dfrac{u\ell}{u - \ell} y\right]_+ & \text{if } \dfrac{u}{u - \ell} y + \hat{y} > 0 \end{cases}
\end{aligned} \tag{36}$$

Observe that the second case is always larger than first, so we get a tighter upper bound when $\frac{u}{u-\ell} y + \hat{y} \leq 0$. If we plug in $\hat{y} = -\nu$ and $y = \hat{\nu}$, this condition is equivalent to

$$\frac{u}{u - \ell} \hat{\nu} \leq \nu$$

Recall that in the LP form, the forward pass in this case was defined by

$$\nu = \frac{u}{u - \ell} [\hat{\nu}]_+ + \alpha [\hat{\nu}]_-$$

Then, $\alpha = \frac{u}{u-l}$ can be interpreted as the *largest choice of* α *which does not increase the bound* (because if α was any larger, we would enter the second case and add an additional $\left(\frac{u}{u-\ell}\hat{\nu} - \nu\right) u$ term to the bound).

We can verify that using $\alpha = \frac{u}{u-\ell}$ results in the same dual problem by first simplifying the above to

$$I_\mathbb{S}^*(\nu, \hat{\nu}) = -l[\nu]_+$$

Combining this with the earlier two cases and plugging into (34) using $f_i^* = I_\mathcal{S}^*$ results in

$$\begin{aligned}
\underset{\nu}{\text{maximize}} \quad & -x^T \hat{\nu}_1 - f_1^*(\hat{\nu}_1) - \sum_{i=1}^{k-1} \nu_{i+1}^T b_i \\
& + \sum_{i=2}^{k-1} \left(\sum_{j \in \mathcal{I}} l_{i,j} [\nu_{i,j}]_+\right) \\
\text{subject to} \quad & \\
& \nu_k = -c \\
& \hat{\nu}_i = W_i^T \nu_{i+1},\ i = 1, \ldots, k-1 \\
& \nu_{i,j} = 0,\ i = 2, \ldots, k-1,\ j \in \mathcal{I}_i^- \\
& \nu_{i,j} = \hat{\nu}_{i,j},\ , i = 2, \ldots, k-1,\ j \in \mathcal{I}_i^+ \\
& \nu_{i,j} = \frac{u_{i,j}}{u_{i,j} - l_{i,j}} \hat{\nu}_{i,j}, i = 2, \ldots, k-1,\ j \in \mathcal{I}_i
\end{aligned} \tag{37}$$

where the dual network here matches the one from (7) exactly when $\alpha = \frac{u_{i,j}}{u_{i,j} - l_{i,j}}$.

A.4. Proof of Theorem 2

In this section, we prove Theorem 2, reproduced below:

Theorem. *Let L be a monotonic loss function that satisfies Property 1. For any data point (x, y), and $\epsilon > 0$, the worst case adversarial loss from (11) can be upper bounded with*

$$\max_{\|\Delta\|_\infty \le \epsilon} L(f_\theta(x+\Delta), y) \le L(-J_\epsilon(x, g_\theta(\mathbf{e}_y 1^T - I)), y)$$

where J_ϵ is as defined in (6) for a given x and ϵ, and g_θ is as defined in (7) for the given model parameters θ.

Proof. First, we rewrite the problem using the adversarial polytope $\mathcal{Z}_\epsilon(x)$.

$$\max_{\|\Delta\|_\infty \le \epsilon} L(f_\theta(x+\Delta), y) = \max_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} L(\hat{z}_k, y)$$

Since $L(x, y) \le L(x - a1, y)$ for all a, we have

$$\begin{aligned}\max_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} L(\hat{z}_k, y) &\le \max_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} L(\hat{z}_k - (\hat{z}_k)_y 1, y) \\ &= \max_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} L((I - \mathbf{e}_y 1^T)\hat{z}_k, y) \\ &= \max_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} L(C\hat{z}_k, y)\end{aligned} \quad (38)$$

where $C = (I - \mathbf{e}_y 1^T)$. Since L is a monotone loss function, we can upper bound this further by using the elementwise maximum over $[C\hat{z}_k]_i$ for $i \ne y$, and elementwise-minimum for $i = y$ (note, however, that for $i = y$, $[C\hat{z}_k]_i = 0$). Specifically, we bound it as

$$\max_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} L(C\hat{z}_k, y) \le L(h(\hat{z}_k))$$

where, if C_i is the ith row of C, $h(z_k)$ is defined elementwise as

$$h(z_k)_i = \max_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} C_i \hat{z}_k$$

This is exactly the adversarial problem from (2) (in its maximization form instead of a minimization). Recall that J from (6) is a lower bound on (2) (using $c = -C_i$).

$$J_\epsilon(x, g_\theta(-C_i)) \le \min_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} -C_i^T \hat{z}_k \quad (39)$$

Multiplying both sides by -1 gives us the following upper bound

$$-J_\epsilon(x, g_\theta(-C_i)) \ge \max_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} C_i^T \hat{z}_k$$

Applying this upper bound to $h(z_k)_i$, we conclude

$$h(z_k)_i \le -J_\epsilon(x, g_\theta(-C_i))$$

Applying this to all elements of h gives the final upper bound on the adversarial loss.

$$\max_{\|\Delta\|_\infty \le \epsilon} L(f_\theta(x+\Delta), y) \le L(-J_\epsilon(x, g_\theta(\mathbf{e}_y 1^T - I)), y)$$

□

A.5. Proof of Corollary 1

In this section, we prove Corollary 1, reproduced below:

Theorem. *For a data point x and $\epsilon > 0$, if*

$$\min_{y \ne f(x)} [J_\epsilon(x, g_\theta(\mathbf{e}_{f(x)} 1^T - I, \alpha))]_y \ge 0 \quad (40)$$

then the model is guaranteed to be robust around this data point. Specifically, there does not exist an adversarial example $\tilde{x}$ such that $|\tilde{x} - x|_\infty \le \epsilon$ and $f_\theta(\tilde{x}) \ne f_\theta(x)$.

Proof. Recall that J from (6) is a lower bound on (2). Combining this fact with the certificate in (40), we get that for all $y \ne f(x)$,

$$\min_{\hat{z}_k \in \mathcal{Z}_\epsilon(x)} (\hat{z}_k)_{f(x)} - (\hat{z}_k)_y \ge 0$$

Crucially, this means that for every point in the adversarial polytope and for any alternative label y, $(\hat{z}_k)_{f(x)} \ge (\hat{z}_k)_y$, so the classifier cannot change its output within the adversarial polytope and is robust around x. □

B. Experimental Details

B.1. 2D Example

Problem Generation We incrementally randomly sample 12 points within the $[0, 1]$ xy-plane, at each point waiting until we find a sample that is at least 0.16 away from other points via ℓ_∞ distance, and assign each point a random label. We then attempt to learn a robust classifier that will correctly classify all points with an ℓ_∞ ball of $\epsilon = 0.08$.

Parameters We use the Adam optimizer (Kingma & Ba, 2015) (over the entire batch of samples) with a learning rate of 0.001.

Visualizations of the Convex Outer Adversarial Polytope We consider some simple cases of visualizing the outer approximation to the adversarial polytope for random networks in Figure 6. Because the output space is two-dimensional we can easily visualize the polytopes in the output layer, and because the input space is two dimensional, we can easily cover the entire input space densely to enumerate the true adversarial polytope. In this experiment, we initialized the weights of the all layers to be normal $\mathcal{N}(0, 1/\sqrt{n_{\text{in}}})$ and biases normal $\mathcal{N}(0, 1)$ (due to scaling, the actual absolute value of weights is not particularly important except as it relates to ϵ). Although obviously not too much should be read into these experiments with random networks, the main takeaways are that 1) for "small" ϵ, the outer bound is an extremely good approximation to the adversarial polytope; 2) as ϵ increases, the bound gets substantially weaker. This is to be expected: for small ϵ, the number of elements in $\mathcal{I}$ will also be relatively small, and

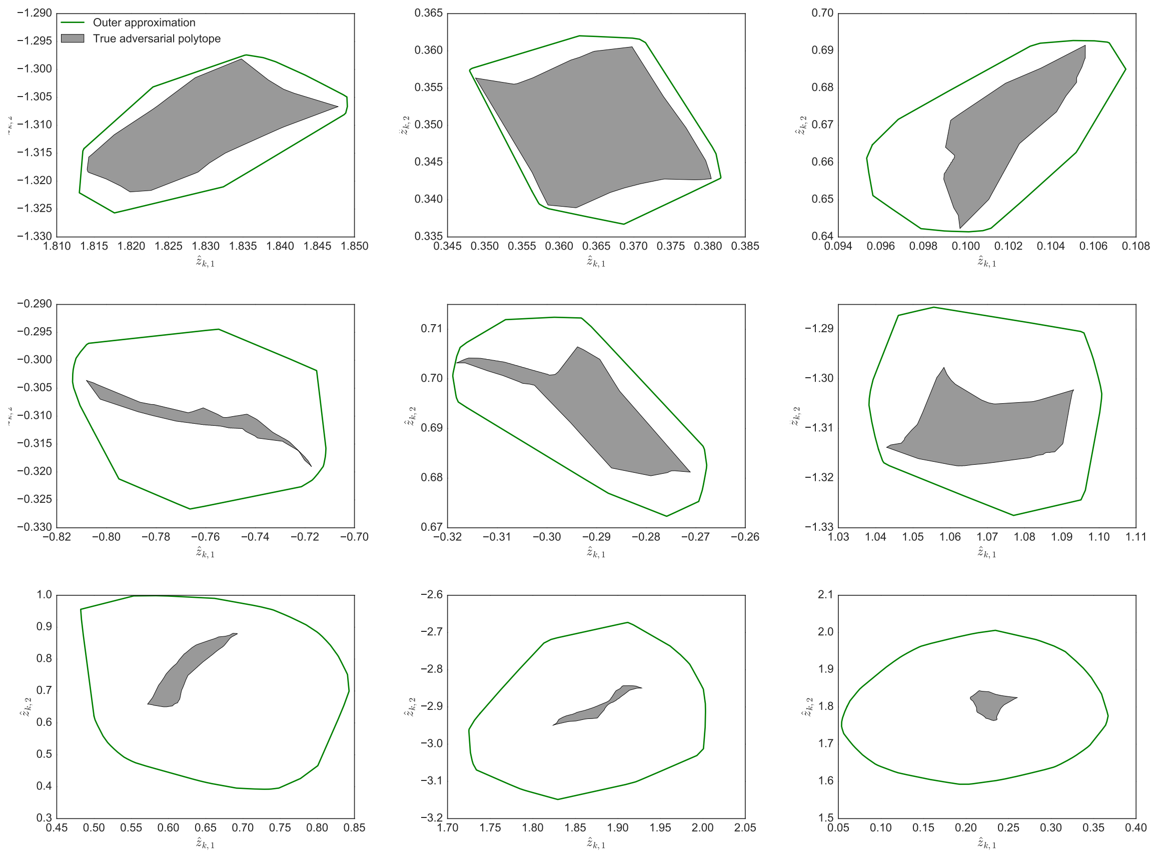

Figure 6. Illustrations of the true adversarial polytope (gray) and our convex outer approximation (green) for a random 2-100-100-100-100-2 network with $\mathcal{N}(0, 1/\sqrt{n})$ weight initialization. Polytopes are shown for $\epsilon = 0.05$ (top row), $\epsilon = 0.1$ (middle row), and $\epsilon = 0.25$ (bottom row).

thus additional terms that make the bound lose are expected to be relatively small (in the extreme, when no activation can change, the bound will be exact, and the adversarial polytope will be a convex set). However, as ϵ gets larger, more activations enter the set $\mathcal{I}$, and the available freedom in the convex relaxation of each ReLU increases substantially, making the bound looser. Naturally, the question of interest is how tight this bound is for networks that are actually trained to minimize the robust loss, which we will look at shortly.

Comparison to Naive Layerwise Bounds One additional point is worth making in regards to the bounds we propose. It would also be possible to achieve a naive "layerwise" bound by iteratively determining absolute allowable ranges for each activation in a network (via a simple norm bound), then for future layers, assuming each activation can vary arbitrarily within this range. This provides a simple iterative formula for computing layer-by-layer absolute bounds on the coefficients, and similar techniques have been used e.g. in Parseval Networks (Cisse et al., 2017) to produce more robust classifiers (albeit there considering ℓ_2 perturbations instead of ℓ_∞ perturbations, which likely are better suited for such an approach). Unfortunately, these naive bounds are extremely loose for multi-layer networks (in the first hidden layer, they naturally match our bounds exactly). For instance, for the adversarial polytope shown in Figure 6 (top left), the actual adversarial polytope is contained within the range

$$\hat{z}_{k,1} \in [1.81, 1.85], \quad \hat{z}_{k,2} \in [-1.33, -1.29] \tag{41}$$

with the convex outer approximation mirroring it rather closely. In contrast, the layerwise bounds produce the bound:

$$\hat{z}_{k,1} \in [-11.68, 13.47], \quad \hat{z}_{k,2} \in [-16.36, 11.48]. \tag{42}$$

Such bounds are essentially vacuous in our case, which makes sense intuitively. The naive bound has no way to exploit the "tightness" of activations that lie entirely in the positive space, and effectively replaces the convex ReLU approximation with a (larger) box covering the entire space. Thus, such bounds are not of particular use when considering robust classification.

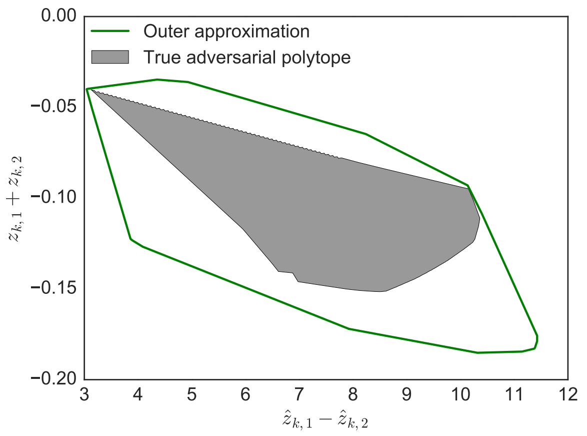

Figure 7. Illustration of the actual adversarial polytope and the convex outer approximation for one of the training points after the robust optimization procedure.

Outer Bound after Training It is of some interest to see what the true adversarial polytope for the examples in this data set looks like versus the convex approximation, evaluated at the solution of the robust optimization problem. Figure 7 shows one of these figures, highlighting the fact that for the final network weights and choice of epsilon, the outer bound is empirically quite tight in this case. In Appendix B.2 we calculate exactly the gap between the primal problem and the dual bound on the MNIST convolutional model. In Appendix B.4, we will see that when training on the HAR dataset, even for larger ϵ, the bound is empirically tight.

B.2. MNIST

Parameters We use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001 (the default option) with no additional hyperparameter selection. We use minibatches of size 50 and train for 100 epochs.

ϵ scheduling Depending on the random weight initialization of the network, the optimization process for training a robust MNIST classifier may get stuck and not converge. To improve convergence, it is helpful to start with a smaller value of ϵ and slowly increment it over epochs. For MNIST, all random seeds that we observed to not converge for $\epsilon = 0.1$ were able to converge when started with $\epsilon = 0.05$ and taking uniform steps to $\epsilon = 0.1$ in the first half of all epochs (so in this case, 50 epochs).

MNIST convolutional filters Random filters from the two convolutional layers of the MNIST classifier after robust training are plotted in Figure 9. We see a similar story

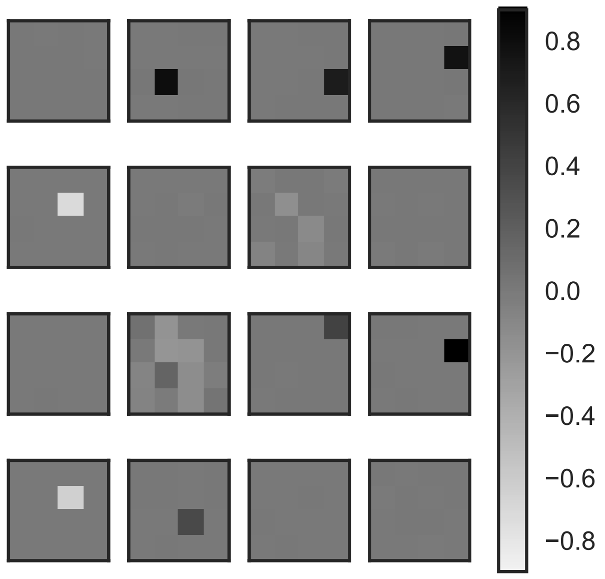

Figure 8. Learned convolutional filters for MNIST of the first layer of a trained robust convolutional network, which are quite sparse due to the ℓ_1 term in (6).

in both layers: they are highly sparse, and some filters have all zero weights.

Activation index counts We plot histograms to visualize the distributions of pre-activation bounds over examples in Figure 10. We see that in the first layer, examples have on average more than half of all their activations in the $\mathcal{I}_1^-$ set, with a relatively small number of activations in the $\mathcal{I}_1$ set. The second layer has significantly more values in the $\mathcal{I}_2^+$ set than in the $\mathcal{I}_2^-$ set, with a comparably small number of activations in the $\mathcal{I}_2$ set. The third layer has extremely few activations in the $\mathcal{I}_3$ set, with 90% all of the activations in the $\mathcal{I}_3^-$ set. Crucially, we see that in all three layers, the number of activations in the $\mathcal{I}_i$ set is small, which benefits the method in two ways: a) it makes the bound tighter (since the bound is tight for activations through the $\mathcal{I}_i^+$ and $\mathcal{I}_i^-$ sets) and b) it makes the bound more computationally efficient to compute (since the last term of (6) is only summed over activations in the $\mathcal{I}_i$ set).

Tightness of bound We empirically evaluate the tightness of the bound by exactly computing the primal LP and comparing it to the lower bound computed from the dual problem via our method. We find that the bounds, when computed on the robustly trained classifier, are extremely tight, especially when compared to bounds computed for random networks and networks that have been trained under standard training, as can be seen in Figure 11.

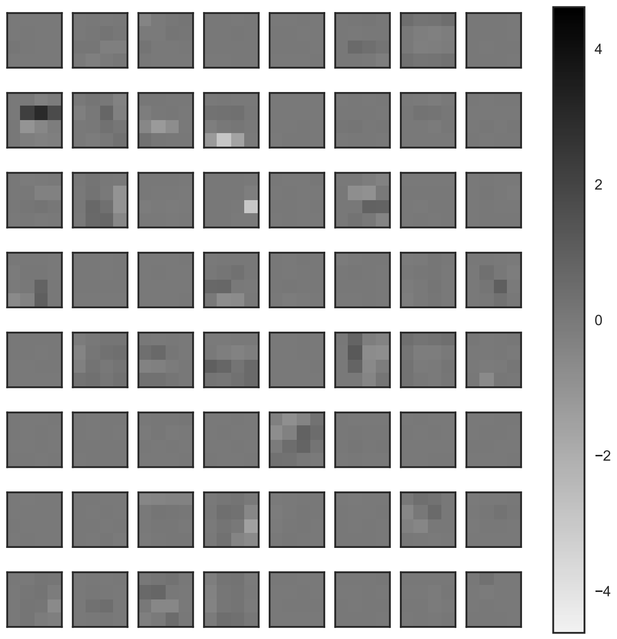

Figure 9. Learned convolutional filters for MNIST of the second layer of a trained robust convolutional network, which are quite sparse due to the ℓ_1 term in (6).

B.3. Fashion-MNIST

Parameters We use exactly the same parameters as for MNIST: Adam optimizer with the default learning rate 0.001, minibatches of size 50, and trained for 100 epochs.

Learning curves Figure 12 plots the error and loss curves (and their robust variants) of the model over epochs. We observe no overfitting, and suspect that the performance on this problem is limited by model capacity.

B.4. HAR

Parameters We use the Adam optimizer with a learning rate 0.0001, minibatches of size 50, and trained for 100 epochs.

Learning Curves Figure 13 plots the error and loss curves (and their robust variants) of the model over epochs. The bottleneck here is likely due to the simplicity of the problem and the difficulty level implied by the value of ϵ, as we observed that scaling to more more layers in this setting did not help.

Tightness of bound with increasing ϵ Earlier, we observed that on random networks, the bound gets progressively looser with increasing ϵ in Figure 6. In contrast, we find that even if we vary the value of ϵ, after robust training on the HAR dataset with a single hidden layer, the bound *still* stays quite tight, as seen in Table 2. As expected, training a robust model with larger ϵ results in a less accurate model since the adversarial problem is more difficult (and potentially impossible to solve for some data points), however the key point is that the robust bounds are extremely close to the achievable error rate by FGSM, implying that in this case, the bound is tight.

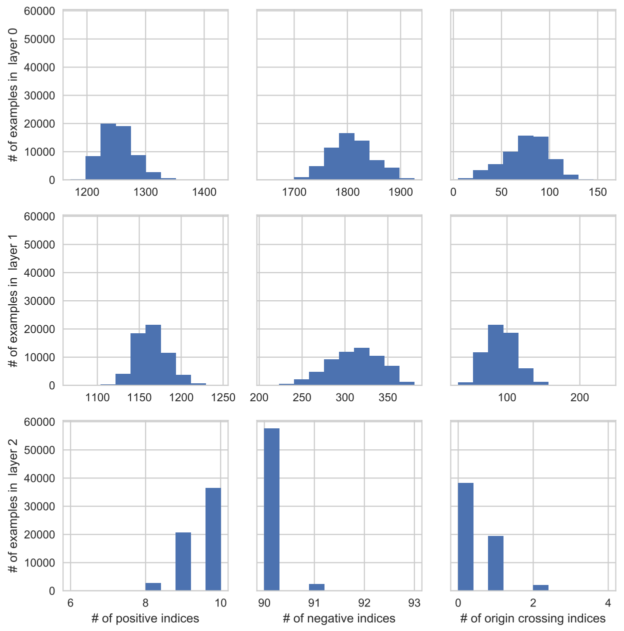

Figure 10. Histograms of the portion of each type of index set (as defined in 10 when passing training examples through the network.

Table 2. Tightness of the bound on a single layer neural network with 500 hidden units after training on the HAR dataset with various values of ϵ. We observe that regardless of how large ϵ is, after training, the bound matches the error achievable by FGSM, implying that in this case the robust bound is tight.

ϵ	TEST ERROR	FGSM ERROR	ROBUST BOUND
0.05	9.20%	22.20%	22.80%
0.1	15.74%	36.62%	37.09%
0.25	47.66%	64.24%	64.47%
0.5	47.08%	67.32%	67.86%
1	81.80%	81.80%	81.80%

B.5. SVHN

Parameters We use the Adam optimizer with the default learning rate 0.001, minibatches of size 20, and trained for 100 epochs. We used an ϵ schedule which took uniform steps from $\epsilon = 0.001$ to $\epsilon = 0.01$ over the first 50 epochs.

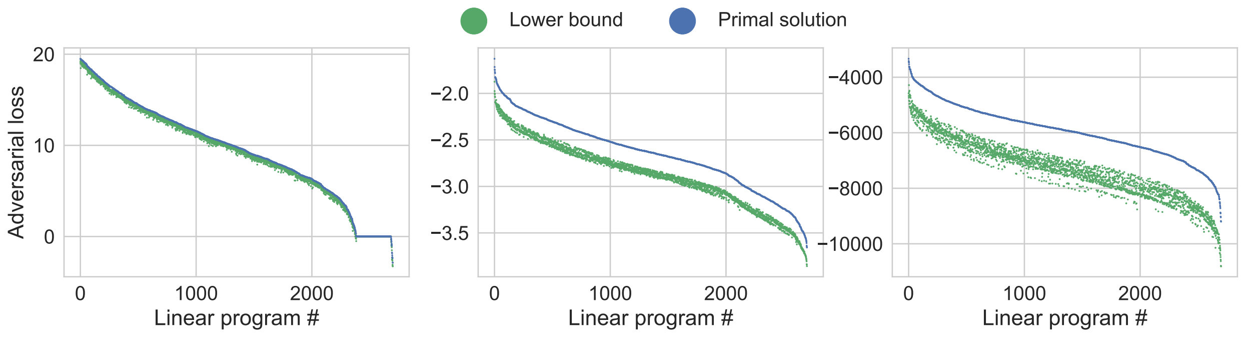

Figure 11. Plots of the exact solution of the primal linear program and the corresponding lower bound from the dual problem for a (left) robustly trained model, (middle) randomly intialized model, and (right) model with standard training.

Learning Curves Note that the robust testing curve is the only curve calculated with $\epsilon = 0.01$ throughout all 100 epochs. The robust training curve was computed with the scheduled value of ϵ at each epoch. We see that all metrics calculated with the scheduled ϵ value steadily increase after the first few epochs until the desired ϵ is reached. On the other hand, the robust testing metrics for $\epsilon = 0.01$ steadily decrease until the desired ϵ is reached. Since the error rate here increases with ϵ, it suggests that for the given model capacity, the robust training cannot achieve better performance on SVHN, and a larger model is needed.

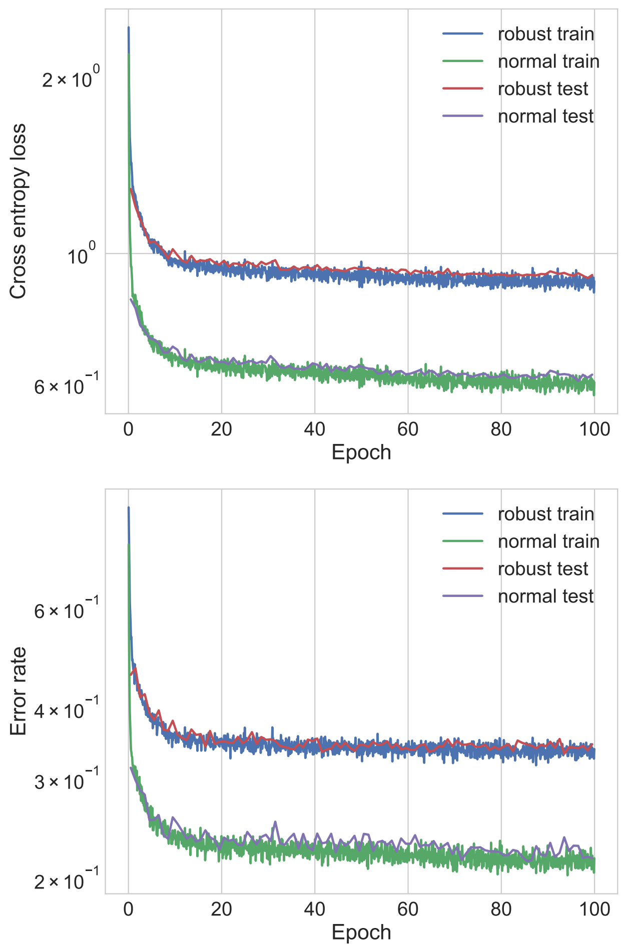

Figure 12. Loss (top) and error rate (bottom) when training a robust convolutional network on the Fashion-MNIST dataset.

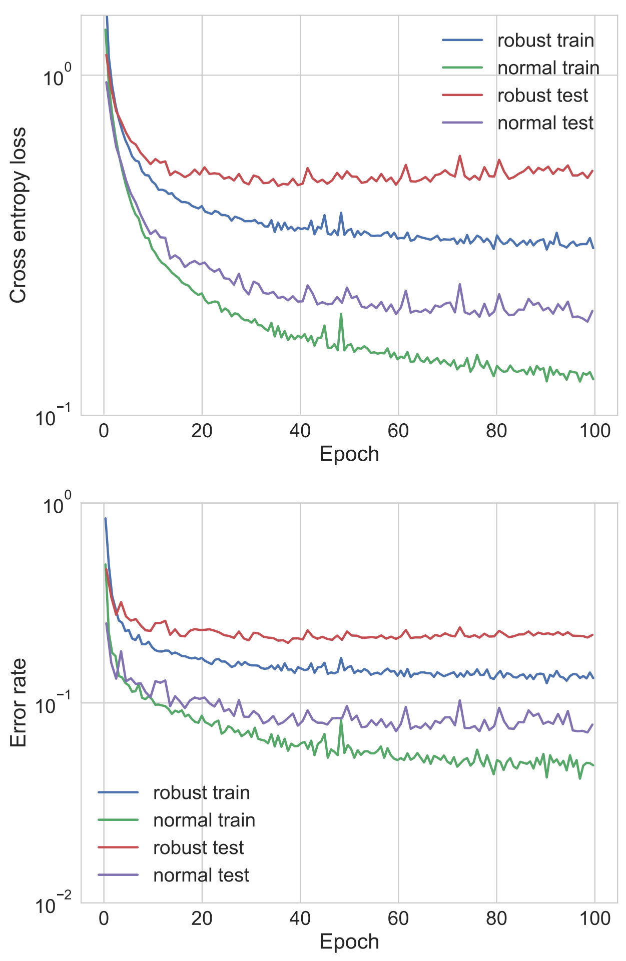

Figure 13. Loss (top) and error rate (bottom) when training a robust fully connected network on the HAR dataset with one hidden layer of 500 units.

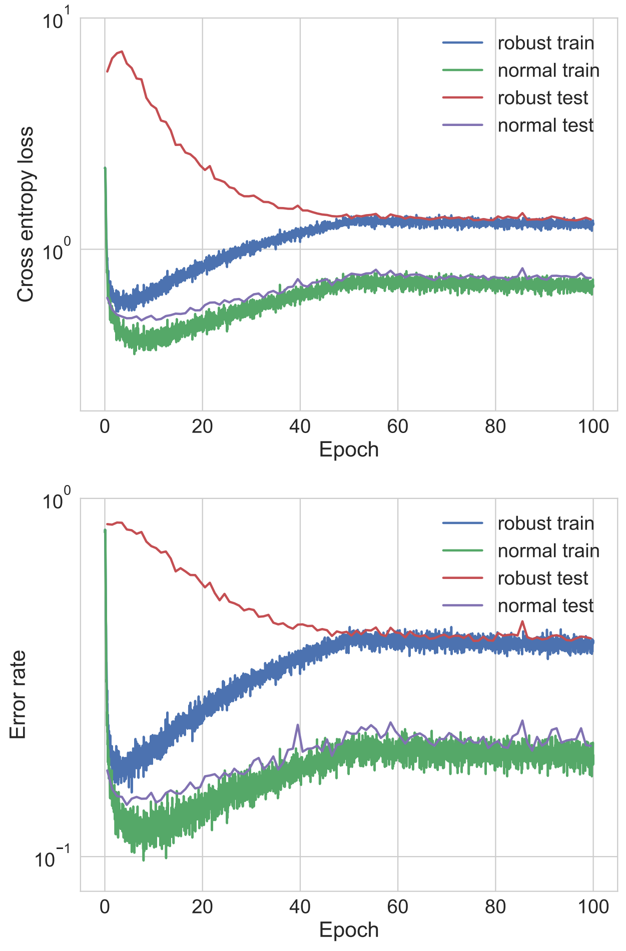

Figure 14. Loss (top) and error rate (bottom) when training a robust convolutional network on the SVHN dataset. The robust test curve is the only curve calculated with $\epsilon = 0.01$ throughout; the other curves are calculated with the scheduled ϵ value.