This Supplementary Material accompanies paper: *Characterizing the Optimal* $0-1$ *Loss for Multi-class Classification with a Test-time Attacker* to NeurIPS 2023.

We first present proofs for lemmas from the main body in §A. We then explain fractional packing and covering in §B. We present our algorithm for hyperedge finding in §C. §D contains further details about the experimental setup with all additional experimental results in §E.

A Proofs

Proof of Lemma 1. This follows immediately from Lemma 2 with $m = K$. □

Proof of Lemma 2. Recall that the definition of the correct-classification probabilities is

$$q_{m,N}(h)_{(x,y)} = \inf_{\tilde{x}\in N(x)} \min_{C\in\binom{[K]}{m}:y\in C} h(\tilde{x}, C)_y$$

and the set of achievable correct-classification probabilities is

$$\mathcal{P}_{\mathcal{V},N,\mathcal{H}} = \bigcup_{h\in\mathcal{H}} \prod_{v\in\mathcal{V}} [0, q_N(h)_v].$$

First, we will show $\mathcal{P}_{m,\mathcal{V},N,\mathcal{H}_{soft}} \subseteq \{q \in \mathbb{R}^{\mathcal{V}} : q \geq \mathbf{0},\ B^{\leq m}q \leq \mathbf{1}\}$. The constraint $h \in \mathcal{H}_{soft}$, $\mathbf{0} \leq q_{m,N}(h) \leq \mathbf{1}$ holds because classification probabilities $h(\tilde{x}, C)_y$ must lie in the range $[0, 1]$.

We will now demonstrate that the constraint $B^{\leq m}q \leq \mathbf{1}$ must also hold. Let $e = ((x_1, y_1), ..., (x_\ell, y_\ell))$ be a size-ℓ hyperedge in $\mathcal{E}^{(\leq m)}$. By construction of $\mathcal{E}^{(\leq m)}$, there exists some $\tilde{x} \in \bigcap_{i=1}^{\ell} N(x_i)$. Let $S \in \binom{[K]}{m}$ be some superset of $\{y_1, \ldots, y_\ell\}$, which exists because $\ell \leq m$. From the definition of $q_{m,N}(h)$, we have that $q_{m,N}(h)_{(x_i,y_i)} \leq h(\tilde{x}, S)_{y_i}$ for each $1 \leq i \leq \ell$. Thus,

$$\sum_{i=1}^{\ell} q_{m,N}(h)_{(x_i,y_i)} \leq \sum_{i=1}^{\ell} h(\tilde{x}, S)_{y_i} \leq \sum_{j\in\mathcal{Y}} h(\tilde{x}, S)_j = 1.$$

This gives $(B^{\leq m}q)_e \leq 1$.

Now we will show $\mathcal{P}_{m,\mathcal{V},N,\mathcal{H}_{soft}} \supseteq \{q \in \mathbb{R}^{\mathcal{V}} : q \geq \mathbf{0},\ B^{\leq m}q \leq \mathbf{1}\}$.

For any vector q in the polytope, we have a classifier $h : \mathcal{X} \times \binom{[K]}{m} \to \mathbb{R}^{[K]}$ that achieves at least those correct classification probabilities. This mean that h has the following properties. First, $h(\tilde{x}, L)_y \geq 0$ and $\sum_{y\in[K]} h(\tilde{x}, L)_y = 1$. Second, for all $(x, y) \in \mathcal{V}$, all $\tilde{x} \in N(x)$, and all $L \in \binom{[K]}{m}$ such that $y \in L$, we have $h(\tilde{x}, L)_y \geq q_{(x,y)}$.

To get h, first define the function $g : \mathcal{X} \times \binom{[K]}{m} \to \mathbb{R}^{[K]}$ so $g(\tilde{x}, L)_y = 0$ for $i \notin L$ and $g(\tilde{x}, L)_y = \max(0, \sup\{q_{(x_y,y)} : x_y \in \mathcal{V}_y, \tilde{x} \in N(x_y)\})$. Let $L' \subseteq L$ be the set of indices where $g(\tilde{x}, L)_y > 0$. Then any list of vertices $e = (x_y : y \in L', x_y \in \mathcal{V}_y, \tilde{x} \in N(x_y))$ forms a hyperedge of size $|L'| \leq m$. Thus

$$\sum_{y\in[K]} g(\tilde{x}, L)_y = \sum_{y\in L'} g(\tilde{x}, L)_y = \sup_e \sum_{y\in L'} q_{(x_y,y)} \leq \sup_e 1 = 1.$$

To produce h, allocate the remaining probability $(1 - \sum_y g(\tilde{x}, L)_y)$ to an arbitrary class. □

Proof of Lemma 3. The first part of this proof applies for any side-information size m. The adversarial strategy for selecting C is a specified by a conditional p.m.f. $p_{C|y}(C|y)$. Thus $p_{y|C}(y|C) = p_{C|y}(C|y)p_{\mathbf{y}}(y)/\sum_{y'} p_{C|y}(C|y')p_y(y')$.

The optimal loss of the classifier against a particular adversarial strategy is just a mixture of the optimal losses for each class list: $\sum_C p_{C|y}(C|y) \Pr[p_y(y)L^*(P|(y \in \{i, j\}), N, \mathcal{H})$.

If $p_{C|y}(C|y) = p_{C|y}(C|y')$ for all $y, y' \in C$, then $p_{y|C}(y|C) = p_{\mathbf{y}}(y)/\sum_{y'\in C} p_{\mathbf{y}}(y')$ and the adversary has not provided the classifier with extra information beyond the fact that $y \in C$. Thus $P_{x|y}P_{y|C} = P|(y \in C)$.

Now we can spcialize to the $m = 2$ case. Any stochastic matrix s with zeros on the diagonal specifies an adversarial strategy for selecting C with $p_{C|y}(\{i,j\}|i) = s_{i,j}$. Furthermore, if s is also symmetric, $p_{C|y}(\{i,j\}|i) = p_{C|y}(\{i,j\}|j)$ and $p_{y|C}(i|\{i,j\}) = p_{y|C}(j|\{i,j\})$. Then the optimal classifier for the side-information game uses the $\binom{K}{2}$ optimal classifiers for the two-class games and incurs loss $\sum_{i,j} Pr[Y = i]a_{i,j}s_{i,j}$ where $a_{i,j} = L^*(P|(y \in \{i,j\}), \mathcal{H}, N)$. Because the diagonal entries of a are all zero, there is always a maximizing choice of s with a zero diagonal. Thus it is not necessary to include that constraint on s when specifying the optimization. □

Proof of Lemma 4. If $w = 0$, then the lower bound is zero and holds trivially. Otherwise, $\frac{1}{\mathbf{1}^T w} w$ forms a probability distribution over the vertices. Let $X \in \mathcal{V}^{\mathbb{N}}$ be a sequence of i.i.d. random vertices with this distribution. From this sequence, we define a random independent set as follows. Include v in the set if it appears in the sequence X before any of its neighbors in $\mathcal{G}$. If v and v' are adjacent, at most one of them can be included, so this procedure does in fact construct an independent set. The probability that $X_i = v$ is $\frac{w_i}{\mathbf{1}^T w}$ and the probability that X_i is v or is adjacent to v is $\frac{((A+I)w)_v}{\mathbf{1}^T w}$. The first time that the latter event occurs, v is either included in the set or ruled out. If $w_i > 0$, the probability that v is included in the set is $\frac{w_v}{((A+I)w)_v}$ and otherwise it is zero. Thus the quantity $P(S$ in Lemma 4 is the expected size of the random independent set and $\mathcal{G}$ must contain some independent set at least that large. □

B Fractional packing and covering

In this section, we record some standard definition in fractional graph and hypergraph theory [8].

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be hypergraph and let $B \in \mathbb{R}^{\mathcal{E} \times \mathcal{V}}$ be its edge-vertex incidence matrix. That is, $B_{e,v} = 1$ if $v \in e$ and $B_{e,v} = 0$ otherwise. Let $p \in \mathbb{R}^{\mathcal{V}}$ be a vector of nonnegative vertex weights. Let $\mathbf{0}$ be the vector of all zeros and let $\mathbf{1}$ be the vector of all ones (of whichever dimensions are required).

A fractional vertex packing in $\mathcal{G}$ is a vector $q \in \mathbb{R}^{\mathcal{V}}$ in the polytope

$$q \geq \mathbf{0} \qquad Bq \leq \mathbf{1}.$$

The minimum weight fractional vertex packing linear program is

$$\min p^T q \quad \text{s.t.} \quad q \geq \mathbf{0} \qquad Bq \leq \mathbf{1}.$$

A fractional hyperedge covering in $\mathcal{G}$ for weights p is a vector $z \in \mathbb{R}^{\mathcal{V}}$ in the polytope

$$q \geq \mathbf{0} \qquad B^T z \geq p.$$

The minimum weight fractional hyperedge covering linear program is

$$\min_z \mathbf{1}^T z \quad \text{s.t.} \quad z \geq 0, \quad B^T z \geq p.$$

These linear programs form a dual pair.

The independent set polytope for $\mathcal{G}$ is the convex hull of the independent set indicator vectors.

Let $\mathcal{G}'$ be a graph on the vertex set $\mathcal{V}$. A clique in $\mathcal{G}'$ is a subset of vertices in which every pair are adjacent. Let $\mathcal{C}$ be the set of cliques of $\mathcal{G}'$ and let $C \in \mathbb{R}^{\mathcal{C} \times \mathcal{V}}$ be the clique vertex incidence matrix. (In fact this construction can be interpreted as another hypergraph on $\mathcal{V}$.) A fractional clique cover in $\mathcal{G}'$ for vertex weights p is a vector $z \in \mathbb{R}^{\mathcal{C}}$ in the polytope

$$q \geq \mathbf{0} \qquad C^T z \geq p.$$

Somewhat confusingly, the dual concept is called a fractional independent set in $\mathcal{G}'$. This is a vector $q \in \mathbb{R}^{\mathcal{V}}$ in the polytope

$$q \geq \mathbf{0} \qquad Cq \leq \mathbf{1}.$$

C Hyperedge Finding

One challenge in computing lower bounds for $0-1$ loss in the multi-class setting is that we need to find hyperedges in the conflict hypergraph. In this section, we will consider an ℓ_2 adversary: $N(x) = \{x' \in \mathcal{X} |\ ||x' - x||_2 \leq \epsilon\}$ and describe an algorithm for finding hyperedges within the conflict graph.

We first note that for an n-way hyperedge to exist between n inputs $\{x_i\}_{i=1}^n$, $\{x_i\}_{i=1}^n$ must all lie on the interior of an $n-1$-dimensional hypersphere of radius ϵ.

Given input x_1, ..., x_n where $x_i \in \mathbb{R}^d$, we first show that distance between any two points in the affine subspace spanned by the inputs can be represented by a distance matrix whose entries are the squared distance between inputs. This allows us to compute the circumradius using the distance information only, not requiring a coordinate system in high dimension. Then we find the circumradius using the properties that the center of the circumsphere is in the affine subspace spanned by the inputs and has equal distance to all inputs.

We construct matrix $X \in \mathbb{R}^{d \times n}$ whose i^{th} column is input x_i. Let $D \in \mathbb{R}^{n \times n}$ be the matrix of squared distances between the inputs, i.e., $D_{i,j} = \|x_i - x_j\|^2$.

We first notice that D can be represented by X and a vector in $\mathbb{R}^n$ whose i^{th} entry is the squared norm of x_i. Let $\Delta \in \mathbb{R}^n$ be such vector such that $\Delta_i = \|x_i\|^2 = (X^T X)_{i,i}$. Then given that $D_{i,j}$ is the squared distance between x_i and x_j, we have

$$D_{i,j} = \|x_i\|^2 + \|x_j\|^2 - 2\langle x_i, x_j \rangle,$$

which implies that

$$D = \Delta \mathbf{1}^T + \mathbf{1}\Delta^T - 2X^T X.$$

Let $\alpha, \beta \in \mathbb{R}^n$ be vectors of affine weights: $\mathbf{1}^T\alpha = \mathbf{1}^T\beta = 1$. Then $X\alpha$ and $X\beta$ are two points in the affine subspace spanned by the columns of X. The distance between $X\alpha$ and $X\beta$ is $\frac{-(\alpha-\beta)^T D(\alpha-\beta)}{2}$, shown as below:

$$\begin{aligned}
\frac{-(\alpha-\beta)^T D(\alpha-\beta)}{2} &= \frac{-(\alpha-\beta)^T(\Delta\mathbf{1}^T + \mathbf{1}\Delta^T - 2X^T X)(\alpha-\beta)}{2} \\
&= \frac{-(0+0-2(\alpha-\beta)^T X^T X(\alpha-\beta))}{2} \\
&= \|X\alpha - X\beta\|^2.
\end{aligned}$$

Now we compute the circumradius using the squared distance matrix D. The circumcenter is in the affine subspace spanned by the inputs so we let $X\alpha$ to be the circumcenter where $\mathbf{1}^T\alpha = 1$. Let $e^{(i)} \in \mathbb{R}^n$ be the i^{th} standard basis vector. The distance between the circumcenter and x_i is $\|X\alpha - Xe^{(i)}\|^2$. From previous computation, we know that $\|X\alpha - Xe^{(i)}\|^2 = \frac{-(\alpha-e^{(i)})^T D(\alpha-e^{(i)})}{2}$. Since the circumcenter has equal distance to all inputs, we have

$$(\alpha - e^{(1)})^T D(\alpha - e^{(1)}) = \ldots = (\alpha - e^{(n)})^T D(\alpha - e^{(n)}). \tag{1}$$

Note that the quadratic term in α is identical in each of these expressions. In addition, $e^{(i)T} D e^{(i)} = 0$ for all i. So equation 1 simplifies to the linear system

$$\begin{aligned}
e^{(i)T} D\alpha = c &\implies D\alpha = c\mathbf{1} \\
&\implies \alpha = cD^{-1}\mathbf{1}
\end{aligned}$$

for some constant c. Since $\mathbf{1}^T\alpha = 1$, we have

$$\begin{aligned}
1 = \mathbf{1}^T\alpha = c\mathbf{1}^T D^{-1}\mathbf{1} \\
\implies \frac{1}{c} = \mathbf{1}^T D^{-1}\mathbf{1}
\end{aligned}$$

assuming that D is invertible. The square of the circumradius, r^2, which is the squared distance between the circumcenter and x_1, is

$$\begin{aligned}
&\|X\alpha - Xe^{(1)}\|^2 \\
=&\frac{-(\alpha - e^{(1)})^T D(\alpha - e^{(1)})}{2} \\
=&e^{(1)T} D\alpha - \frac{\alpha^T D\alpha}{2} \\
=&c - \frac{c^2 \mathbf{1}^T D^{-1}\mathbf{1}}{2} \\
=&\frac{c}{2} \\
=&\frac{1}{2\mathbf{1}^T D^{-1}\mathbf{1}}.
\end{aligned}$$

Therefore, assuming matrix D is invertible, the circumradius is $\frac{1}{\sqrt{2\mathbf{1}^T D^{-1}\mathbf{1}}}$.

The inverse of D can be computed as $\frac{\det D}{\operatorname{adj} D}$. Since $\alpha = cD^{-1}\mathbf{1}$, we have $\alpha = c\frac{\det D}{\operatorname{adj} D}\mathbf{1}$. As $r^2 = \frac{c}{2}$, constant c is non-negative. Therefore, $\alpha \propto \frac{\det D}{\operatorname{adj} D}\mathbf{1}$.

When all entries of α are non-negative, the circumcenter is a convex combination of the all inputs and the circumsphere is the minimum sphere in $\mathbb{R}^{n-1}$ that contains all inputs. Otherwise, the circumsphere of $\{x_i | \alpha_i > 0\}$ is the minimum sphere contains all inputs.

After finding the radius of the minimum sphere that contains all inputs, we compare the radius with the budget ϵ. If the radius is no larger than ϵ, then there is a hyperedge of degree n among the inputs.

D Experimental Setup

In this section, we describe our experimental setup. Our code for computing bounds is also available at `https://github.com/inspire-group/multiclass_robust_lb`.

Datasets: We compute lower bounds for MNIST [6], CIFAR-10, and CIFAR-100 [5]. Since we do not know the true distribution of these datasets, we compute lower bounds based on the empirical distribution of the training set for each dataset.

Attacker: We will consider an ℓ_2 adversary: $N(x) = \{x' \in \mathcal{X} | \ ||x' - x||_2 \leq \epsilon\}$. This has been used in most prior work [3, 7, 9].

LP solver: For solving the LP in Equation (4), we primarily use the Mosek LP solver [2]. When the Mosek solver did not converge, we default to using CVXOpt's LP solver [1].

Computing infrastructure. In order to compute lower bounds, we perform computations across 10 2.4 GHz Intel Broadwell CPUs. For adversarial training, we train on a single A100 GPU.

Training Details For MNIST, we use 40-step optimization to find adversarial examples during training and use step size $\frac{\epsilon}{30}$ and train all models for 20 epochs. For CIFAR-10 and CIFAR-100, we use 10 step optimization to find adversarial examples and step size $\frac{\epsilon}{7}$ and train models for 100 epochs. For MNIST TRADES training, we use $\beta = 1$ and for CIFAR-10 and CIFAR-100, we use $\beta = 6$. Additionally, for CIFAR-10 and CIFAR-100, we optimize the model using SGD with learning rate and learning rate scheduling from [4]. For MNIST, we use learning rate 0.01.

Architectures used: For CIFAR-10 and CIFAR-100, we report results from training a WRN-28-10 architecture. For MNIST, we train a small CNN architecture consisting of 2 convolutional layers, each followed by batch normalization, ReLU, and 2 by 2 max pooling. The first convolutional layer uses a 5 by 5 convolutional kernel and has 20 output channels. The second convolutional layer also uses a 5 by 5 kernel and has 50 output channels. After the set of 2 convolutional layers with batch normalization, ReLU, and pooling, the network has a fully connected layer with 500 output channels followed by a fully connected classifier (10 output channels). In §E.8, we consider the impact of architecture on closing the gap to the optimal loss.

E Additional Experimental Results

In this Section, we provide additional experimental results to complement those from the main body of the paper. We organize it as follows:

- §E.1: We analyze a toy problem on 2D Gaussian data to get a better understanding of the impact of hyperedges on the optimal loss computation.
- §E.2: We investigate why higher degree hyperedges dp not have a large impact on lower bounds at lower values of ϵ.
- §E.3: We show the runtime explosion at higher values of ϵ that makes it challenging for us to report optimal loss values.
- §E.4: Classwise L2 distance statistics and heatmaps for pairwise losses used to compute class-only lower bounds in main paper.
- §E.5: We provide results with standard PGD-based adversarial training, and show it is outperformed by Trades.
- §E.6: We provide results on the CIFAR-100 dataset.
- §E.7: We show lower bounds for a different set of 3 classes than the one considered in the main body. Main takeaways remain the same.
- §E.8: We ablate across larger neural networks to check if increasing capacity reduces the gap to optimal.
- §E.9: We attempt dropping examples from the training set that even the optimal clasifier cannot classify correctly in order to improve convergence.
- §E.10: We compute lower bounds on the test set for MNIST 3-class classification

E.1 Results for Gaussian data

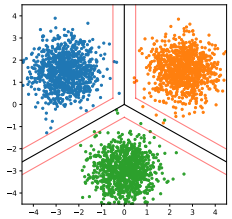
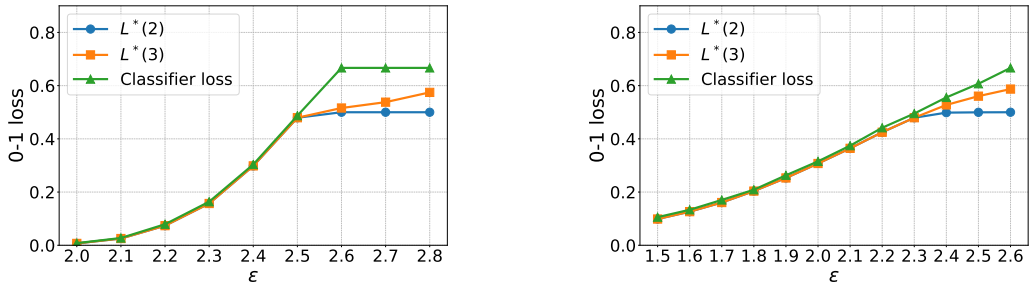


Figure 1: A sample 3-class Gaussian problem (each color pertains to a class) and a corresponding classifier for this problem shown in black. The classifier classifies a sample incorrectly when it lies over the edge of the ϵ margin (shown by the red lines) nearest the corresponding Gaussian center.

We begin with a 3-way classification problem on 2D Gaussian data. To generate our Gaussian dataset, we sample 1000 points per class from 3 spherical Gaussians with means at distance 3 away from from the origin (a sample is shown in Figure 1). We compute multiclass lower bounds via the LP in Lemma on robust accuracy at various ℓ_2 budget ϵ and display these values in Figure 2 as $L^*(3)$. Additionally, we compare to a deterministic 3 way classifier. This classifier is the best performing out of the 2 strategies: 1) constantly predict a single class (thus achieving $\frac{2}{3}$ loss) or 2) is the classifier in black in Figure 1 which classifies incorrectly when a sample lies over the edge of the nearest ϵ margin of the classifier.

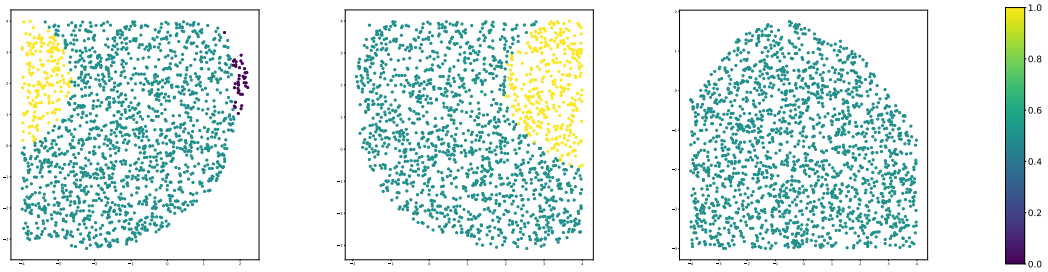
We observe that at smaller values of ϵ, the loss achieved by the 3-way classifier matches optimal loss ($L^*(3)$); however, after $\epsilon = 2.5$ for $\sigma^2 = 0.05$ and $\epsilon = 2.3$ for $\sigma^2 = 0.5$, we find the classifier no longer achieves optimal loss. This suggests that there is a more optimal classification strategy at these larger values of ϵ. In Figures 3 and 4, we visualize the distribution of correct classification probabilities obtained through solving the LP with and without considering hyperedges. These figures are generated by taking a fresh sample of 1000 points from each class and coloring the point based on the correct classification probability q_v assigned to its nearest neighbor that was used in the conflict



(a) $\sigma^2 = 0.05$ (b) $\sigma^2 = 0.5$

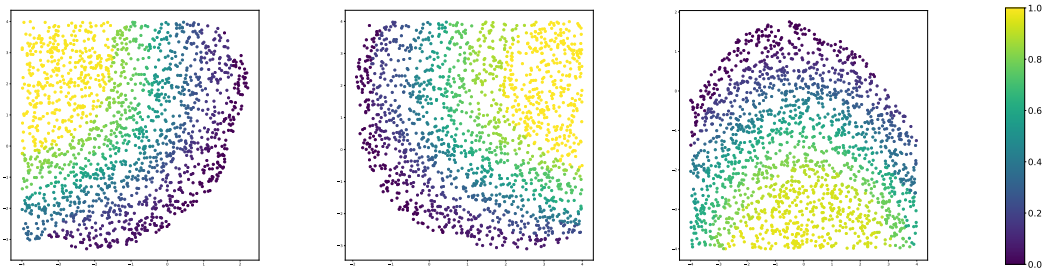
Figure 2: Lower bounds on error for the Gaussian 3-class problem ($\sigma^2 = 0.05$ and $\sigma^2 = 0.5$) computed using only constraints from edges ($L^*(2)$) and up to degree 3 hyperedges ($L^*(3)$) in comparison to the performance of the deterministic 3-way classifier depicted in Figure 1.

hypergraph when computing the lower bound. We observe from Figure 3, for all classes, the data are mostly assigned classification probabilities around 0.5. In Figure 4, we can see that when we consider hyperedges, we some of these 0.5 assignments are reassigned values close to $\frac{2}{3}$ and $\frac{1}{3}$. Interestingly, we notice that when we do not consider hyperedges, our solver finds an asymmetric solution to the problem (strategies for class 0, 1, and 2 differ) while when considering hyperedges this solution becomes symmetric.



(a) Class 0 (b) Class 1 (c) Class 2

Figure 3: Distribution of optimal classification probabilities across samples from each class of the Gaussian obtained as a solution when computing $L^*(2)$.



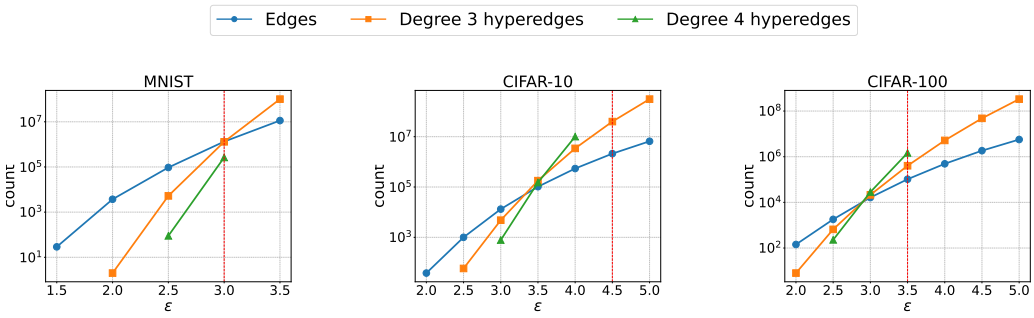
(a) Class 0 (b) Class 1 (c) Class 2

Figure 4: Distribution of optimal classification probabilities across samples from each class of the Gaussian obtained as a solution when computing $L^*(3)$.

E.2 Impact of hyperedges

In Figure 5, we show the count of edges, degree 3 hyperedges, and degree 4 hyperedges found in the conflict hypergraphs of the MNIST, CIFAR-10, and CIFAR-100 train sets. We note that we did

not observe any increase in loss when considering degree 4 hyperedges at the ϵ with a data point for number of degree 4 hyperedges in Figure 5. We find that the relative number of edges and hyperedges is not reflective of whether we expect to see an increase in loss after considering hyperedges. For example in CIFAR-10, at $\epsilon = 4.0$, we there are about 100 times more hyperedges than edges, but we see no noticeable increase in the $0-1$ loss lower bound when incorporating these hyperedge constraints.

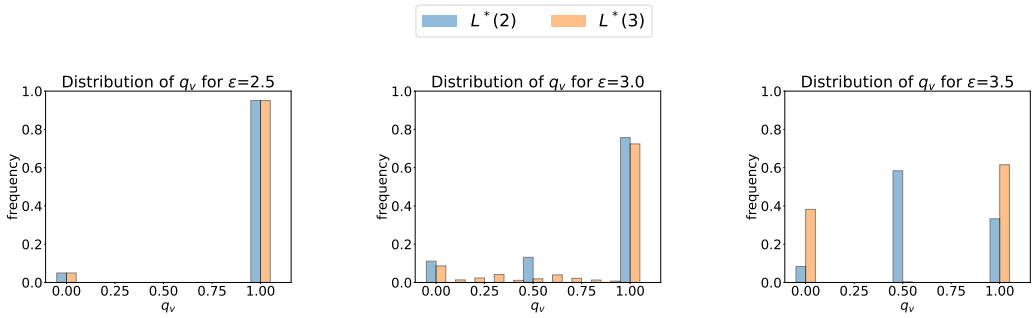


(a) MNIST (b) CIFAR-10 (c) CIFAR-100

Figure 5: Number of edges, degree 3 hyperedges, and degree 4 hyperedges found in the conflict hypergraphs of MNIST, CIFAR-10, and CIFAR-100 train sets. The red vertical line indicates the ϵ at which we noticed an increase in the $0-1$ loss lower bound when considering degree 3 hyperedges.

To understand why including more information about hyperedges does not influence the computed lower bound much, we examine the distribution of q_v obtained from solutions to the LP with ($L^*(3)$) and without degree 3 hyperedges ($L^*(2)$). Fig. 6 contains a histogram of the distributions of q_v for MNIST. For small ϵ, there is no change in the distribution of q_v, the distribution of q_v is almost identical between $L^*(2)$ and $L^*(3)$. At larger values of ϵ, in $L^*(2)$, a significant fraction of vertices are assigned q_v near 0.5. While these shift with the addition of hyperedges, very few of them were in triangles of $\mathcal{G}^{\leq 2}$ that were replaced by a hyperedge in $\mathcal{G}^{\leq 3}$. This leads to the loss value changing minimally.

Similar to Figure 6, we plot the distribution of vertex weights q_v obtained through solving the LP for $L^*(2)$ and $L^*(3)$ for CIFAR-10 in Figure 7. Similar to trends for MNIST, we find that the gap between $L^*(2)$ and $L^*(3)$ only occurs when the frequency of 0.5 weights is higher.

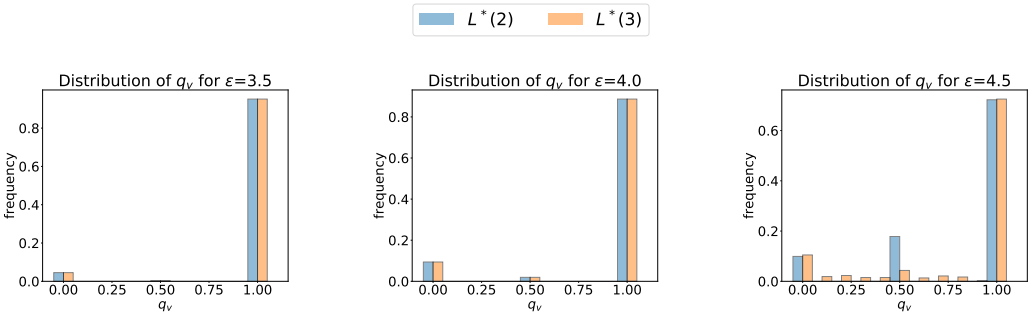


(a) $\epsilon = 2.5$ (b) $\epsilon = 3.0$ (c) $\epsilon = 3.5$

Figure 6: Distribution of optimal classification probabilities q obtained by solving the LP with up to degree 2 hyperedges ($m = 2$) and up to degree 3 hyperedges ($m = 3$) on the MNIST training set.

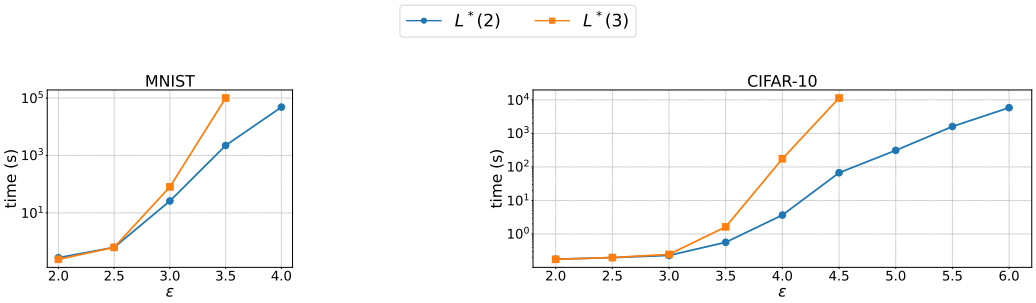
E.3 Computational complexity of computing lower bounds

Our experiments of $L^*(3)$ and $L^*(4)$ at higher ϵ are limited due to computation constraints. Figure 8 we see that the time taken to compute the $L^*(3)$ grows rapidly with ϵ. We also report timings for all bounds for CIFAR-10 at $\epsilon = 4$ in Table 1. In future work, we are seeking algorithmic optimization to achieve more results at high ϵ.



(a) $\epsilon = 3.5$ (b) $\epsilon = 4.0$ (c) $\epsilon = 4.5$

Figure 7: Distribution of optimal classification probabilities q obtained by solving the LP with up to degree 2 hyperedges ($L^*(2)$) and up to degree 3 hyperedges ($L^*(3)$) on the CIFAR-10 training set.



(a) MNIST (b) CIFAR-10

Figure 8: Time taken to compute $L^*(2)$ and $L^*(3)$ for MNIST and CIFAR-10.

E.4 Classwise statistics and pairwise losses

In order to have a better understanding of the difficulty of the classification task under the presence of an ℓ_2 bounded adversary, we report the average ℓ_2 to the nearest neighbor of another class for each class in the MNIST and CIFAR-10 datasets in Table 2.

Another way of understanding the relative difficulty of classifying between classes is by computing the optimal loss for all 1v1 binary classification tasks. We note that these values are used in Section 4.2 to compute a lower bound on the optimal loss in the 10-class case from maximum weight coupling over optimal losses for $1v1$ binary classification problems. In Figure 9, we show the heat maps for optimal losses for each pair of 1v1 classification problems for $\epsilon = 3$ on MNIST and $\epsilon = 4$ on CIFAR-10. We find that for both datasets only a few pairs of classes have high losses. Specifically, for MNIST, we find that the class pairs 4-9 and 7-9 have significantly higher loss than all other pairs of classes. For CIFAR-10, we find that 2-4 has the highest loss compared to other pairs, and 2-6 and 6-4 are also quite high.

E.5 Additional adversarial training results

In Figure 10, we also add the loss achieved by PGD adversarial training. We find that this approach generally performs worse on MNIST compared to TRADES and is also unable to fit to CIFAR-10 data at the ϵ values tested.

E.6 Truncated hypergraph lower bounds for CIFAR-100

We provide results for truncated hypergraph lower bounds for the CIFAR-100 train set. We observe that similar to MNIST and CIFAR-10, including more hyperedge constraints does not influence the computed lower bound.

Loss bound	Runtime (s)
$L^*(2)$	188.24
$L^*(3)$	10413.91
$L^*(4)$	>86400
$L^*_{co}(2)$	327.92
L_{CW}	192.27

Table 1: Runtimes for computing different bounds for CIFAR-10 dataset at $\epsilon = 4$. We note that the $L^*_{co}(2)$ reports the time for computing all pairwise losses sequentially and can be sped up by running these computations in parallel. We also note that $L^*(4)$ computation did not terminate within a day.

	0	1	2	3	4	5	6	7	8	9
MNIST	7.07	4.33	6.94	6.29	5.72	6.29	6.42	5.52	6.29	5.15
CIFAR-10	8.96	10.84	8.48	9.93	8.22	10.04	8.72	10.05	9.23	10.91

Table 2: Average ℓ_2 distance to nearest neighbor in another class for each class in MNIST and CIFAR-10 datasets.

E.7 Computed bounds for a different set of 3-classes

In Figure 12, we plot 3-class lower bounds via truncated hypergraphs ($L^*(2)$ and $L^*(3)$) for a different set of 3 classes as shown in the main body. These classes generally have less similarity than the classes shown in the main body of the paper causing the loss bound to increase more slowly as epsilon increases. However, we find that patterns observed for the 3 classes present in the main body of the paper are also present here: the gap between $L^*(2)$ and $L^*(3)$ is only visible at large values of 0-1 loss (ie. loss of 0.4).

E.8 Impact of architecture size

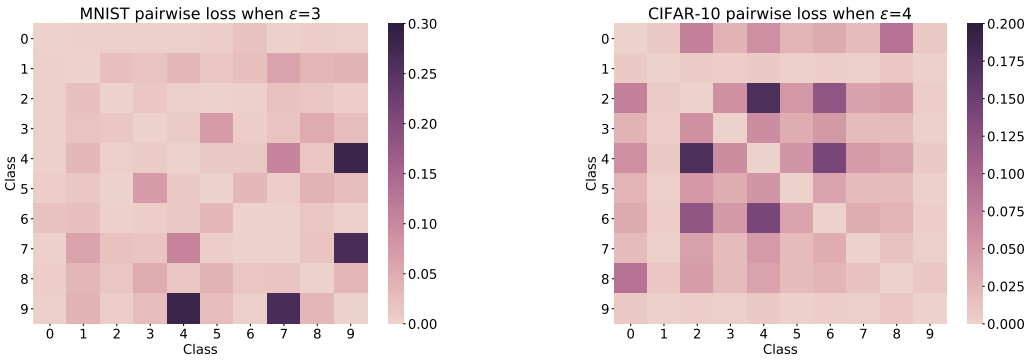
Previously, we saw that adversarial training with larger values of ϵ generally fails to converge (leading to losses matching random guessing across 3 classes). We now investigate whether increasing model capacity can resolve this convergence issue. In Figure 13, we plot the training losses of 3 WRN architectures commonly used in adversarial ML research across attack strength ϵ for the 3-class CIFAR-10 (classes 0, 2, 8) problem. All models are trained with TRADES adversarial training. Interestingly, we find that the benefit of larger architecture size only appears for the smallest value of epsilon plotted ($\epsilon = 1$) at which the optimal classifier can theoretically obtain 0 loss. At larger values of epsilon, the improvement in using larger architecture generally disappears.

E.9 Impact of dropping "hard" examples

From our experiments involving adversarial training, we found that training with large values of ϵ generally fails to converge. A potential way of trying to improve convergence using the results from our LP is to drop examples with optimal classification probability less than 1 and train with the remaining examples. Since even the optimal classifier cannot classify these examples correctly, these examples can be considered "hard" to learn. We find that in practice this does not lead to lower training loss; specifically, training without "hard" examples leads to a loss of 0.64 for CIFAR-10 with $\epsilon = 3$ while training with "hard" examples leads to a loss 0.57. We note that this loss is computed over the entire training dataset (including "hard" examples) for both settings.

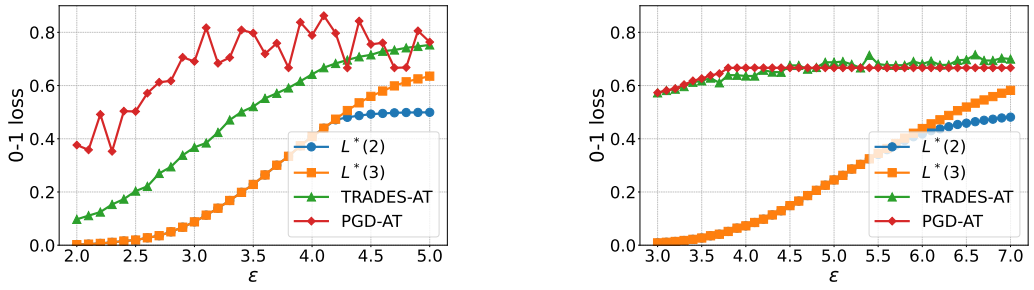
E.10 Lower bounds on test set

In the main text, we computer lower bounds on the train set as this would measure how well existing training algorithms are able to fit to the training data. In Table 3, we compute lower bounds on the test set (which contains 1000 samples per class) for MNIST 3-class classification between classes 1, 4, and 7. We find that the computed loss is similar to what is computed on a subset of the train set which contains the same number of samples per class.



(a) MNIST ($\epsilon = 3$) (b) CIFAR-10 ($\epsilon = 4$)

Figure 9: Heat maps for optimal loss for each pair of 1v1 classification problems.



(a) MNIST (1, 7, 9) (b) CIFAR-10 (0, 2, 8)

Figure 10: Lower bounds on error for MNIST and CIFAR-10 3-class problems (1000 samples per class) computed using only constraints from edges ($L^*(2)$) and up to degree 3 hyperedges ($L^*(3)$) in comparison to TRADES adversarial training (TRADES-AT) and PGD adversarial training (PGD-AT) loss.

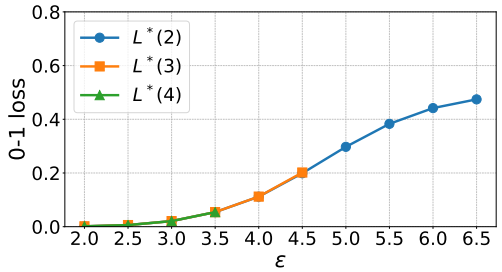
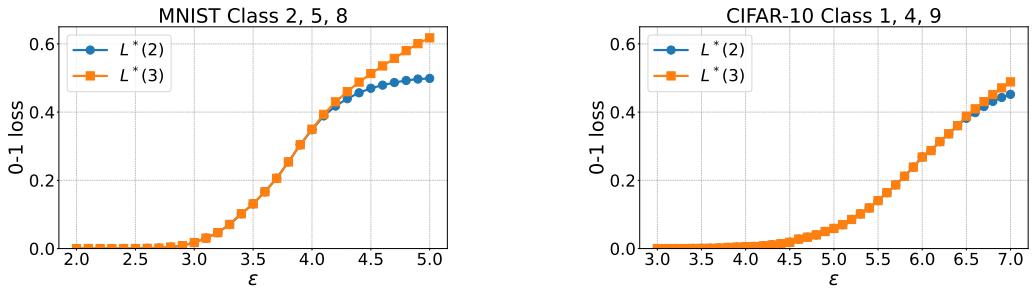


Figure 11: Lower bounds for optimal 0-1 loss the for CIFAR-100 train set



(a) MNIST (2, 5, 8) (b) CIFAR-10 (1, 4, 9)

Figure 12: Lower bounds on error for MNIST and CIFAR-10 3-class problems (1000 samples per class) computed using only constraints from edges ($L^*(2)$) and up to degree 3 hyperedges ($L^*(3)$)

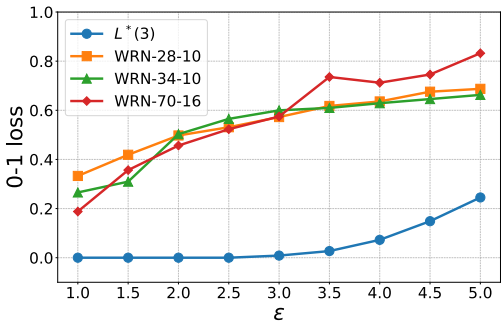


Figure 13: Impact of architecture size on training loss for 3-class CIFAR (0, 2, 8) at different strengths ϵ for models trained with TRADES adversarial training.

ϵ	train set	train set (1000 samples per class)	test set
2	0.0045	0.0020	0.0025
2.5	0.0260	0.0193	0.0149
3	0.1098	0.0877	0.0773
3.5	0.2587	0.2283	0.2181
4	-	0.4083	0.3987

Table 3: Optimal losses $L^*(3)$ for MNIST 3 class problem (classes 1, 4, 7) computed across the train set, the first 1000 samples of each class in the train set, and computed across the test set.

References

[1] M. S. Andersen, J. Dahl, and L. Vandenberghe. Cvxopt: Python software for convex optimization, 2013.

[2] M. ApS. *MOSEK Optimizer API for Python 10.0.22*, 2019.

[3] A. N. Bhagoji, D. Cullina, and P. Mittal. Lower bounds on adversarial robustness from optimal transport. In *Advances in Neural Information Processing Systems*, pages 7496–7508, 2019.

[4] S. Gowal, C. Qin, J. Uesato, T. Mann, and P. Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

[5] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[6] Y. LeCun and C. Cortes. The MNIST database of handwritten digits. 1998.

[7] M. S. Pydi and V. Jog. Adversarial risk via optimal transport and optimal couplings. In *Proceedings of the 37th International Conference on Machine Learning*, pages 7814–7823, 2020.

[8] E. R. Scheinerman and D. H. Ullman. *Fractional Graph Theory: A Rational Approach to the Theory of Graphs*. Wiley, Sept. 1997. Google-Books-ID: KujuAAAAMAAJ.

[9] N. G. Trillos, M. Jacobs, and J. Kim. The multimarginal optimal transport formulation of adversarial multiclass classification. *Journal of Machine Learning Research*, 24(45):1–56, 2023.