A. Generic Randomized Smoothing Algorithm

Algorithm 2 Generic randomized smoothing procedure

Input: function $\phi : \mathcal{X} \to \{0, 1\}$, number of samples N, smoothing distribution μ, test point to predict x_0, failure probability $\delta > 0$.
for $i = 1, \ldots, N$ **do**
  Sample $x_i \sim \mu(x_0)$ and compute $y_i = \phi(x_i)$.
end for
Compute approximate smoothed output

$$\hat{g}(\mu, \phi) = \mathbf{1}\left\{\left(\frac{1}{N}\sum_{i=1}^{N} y_i\right) \geq \frac{1}{2}\right\}.$$

Compute bound $\hat{G}(\mu, \phi)$ such that with probability $\geq 1 - \delta$

$$\hat{G}(\mu, \phi)\begin{cases} \leq G(\mu, \phi) & \text{if } \hat{g}(\mu, \phi) = 1 \\ \geq G(\mu, \phi) & \text{if } \hat{g}(\mu, \phi) = 0. \end{cases}$$

Output: Prediction $\hat{g}(\mu, \phi)$ and probability bound $\hat{G}(\mu, \phi)$, or abstention if $\hat{g}(\mu, \phi) \neq \text{sign}(\hat{G}(\mu, \phi) - \frac{1}{2})$.

B. The Multi-Class Setting

Although the notation and algorithms are slightly more complex, all the methods we have discussed in the main paper can be extended to the multi-class setting. In this case, we consider a class label $y \in \{1, \ldots, K\}$, and we again seek some smoothed prediction such that the classifier's prediction on a new point will not change with some number r flips of the labels in the training set.

B.1. Randomized smoothing in the multi-class case

We here extend our notation to the case of more than two classes. Recall our original definition of G,

$$G(\mu, \phi) = \mathbf{E}_{x\sim\mu}[\phi(x)] = \int_{\mathcal{X}} \mu(x)\phi(x)dx,$$

where $\phi : \mathcal{X} \to \{0, 1\}$. More generally, consider a classifier $\phi : \mathcal{X} \to [K]$, outputting the index of one of K classes. Under this formulation, for a given class $c \in [K]$, we have

$$G(\mu, \phi, c) = \mathbf{E}_{x\sim\mu}[\phi_c(x)] = \int_{\mathcal{X}} \mu(x)\phi_c(x)dx,$$

where $\phi_c(x) = \mathbf{1}\{\phi(x) = c\}$ is the indicator function for if $\phi(x)$ outputs the class c. In this case, the hard threshold g is evaluated by returning the class with the highest probability. That is,

$$g(\mu, \phi) = \arg\max_c G(\mu, \phi, c).$$

B.2. Linearization and Chernoff bound approach for the multi-class case

Using the same linearization approach as in the binary case, we can formulate an analogous approach which forgoes the need to actually perform random sampling at all and instead directly bounds the randomized classifier using the Chernoff bound.

Adopting the same notation as in the main text, the equivalent least-squares classifier for the multi-class setting finds some set of weights

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$$

where $\mathbf{Y} \in \{0,1\}^{n \times K}$ is a binary matrix with each row equal to a one-hot encoding of the class label (note that the resulting $\hat{\boldsymbol{\beta}} \in \mathbb{R}^{k \times K}$ is now a matrix, and we let $\hat{\boldsymbol{\beta}}_i$ refer to the ith column). At prediction time, the predicted class of some new point x_{n+1} is simply given by the prediction with the highest value, i.e.,

$$\hat{y}_{n+1} = \arg\max_i \hat{\boldsymbol{\beta}}_i^T h(x_{n+1}).$$

Alternatively, following the same logic as in the binary case, this same prediction can be written in terms of the $\boldsymbol{\alpha}$ variable as

$$\hat{y}_{n+1} = \arg\max_i \boldsymbol{\alpha}^T \mathbf{Y}_i$$

where $\mathbf{Y}_i$ denotes the ith column of $\mathbf{Y}_i$.

In our randomized smoothing setting, we again propose to flip the class of any label with probability q, selecting an alternative label uniformly at random from the remaining $K-1$ labels. Assuming that the predicted class label is i, we wish to bound the probability that

$$P(\boldsymbol{\alpha}^T \mathbf{Y}_i < \boldsymbol{\alpha}^T \mathbf{Y}_{i'})$$

for all alternative classes i'. By the Chernoff bound, we have that

$$\begin{aligned}\log P(\boldsymbol{\alpha}^T \mathbf{Y}_i < \boldsymbol{\alpha}^T \mathbf{Y}_{i'}) &= \log P(\boldsymbol{\alpha}^T (\mathbf{Y}_i - \mathbf{Y}_{i'}) \leq 0) \\ &\leq \min_{t>0} \left\{ \sum_{j=1}^n \log \mathbf{E}\left[e^{-t\boldsymbol{\alpha}_j (\mathbf{Y}_{ji} - \mathbf{Y}_{ji'})} \right] \right\}.\end{aligned}$$

The random variable $\mathbf{Y}_{ji} - \mathbf{Y}_{ji'}$ takes on three different distributions depending on if $y_j = i$, if $y_j = i'$, or if $y_j \neq i$ and $y_j \neq i'$. Specifically, this variable can take on the terms $+1, 0, -1$ with the associated probabilities

$$\begin{aligned}P(\mathbf{Y}_{ji} - \mathbf{Y}_{ji'} = +1) &= \begin{cases} 1-q & \text{if } y_j = i, \\ q/(K-1) & \text{otherwise.} \end{cases} \\ P(\mathbf{Y}_{ji} - \mathbf{Y}_{ji'} = -1) &= \begin{cases} 1-q & \text{if } y_j = i', \\ q/(K-1) & \text{otherwise.} \end{cases} \\ P(\mathbf{Y}_{ji} - \mathbf{Y}_{ji'} = 0) &= \begin{cases} q(K-2)/(K-1) & \text{if } y_j = i \text{ or } y_j = i', \\ 1 - 2q/(K-1) & \text{otherwise.} \end{cases}\end{aligned}$$

Combining these cases directly into the Chernoff bound gives

$$\begin{aligned}\log P(\boldsymbol{\alpha}^T \mathbf{Y}_i < \boldsymbol{\alpha}^T \mathbf{Y}_{i'}) \leq \min_{t>0} \Bigg\{ &\sum_{j:y_j=i} \log\left((1-q)e^{-t\boldsymbol{\alpha}_j} + q\frac{K-2}{K-1} + \frac{q}{K-1} e^{t\boldsymbol{\alpha}_j} \right) + \\ &\sum_{j:y_j=i'} \log\left(\frac{q}{K-1} e^{-t\boldsymbol{\alpha}_j} + q\frac{K-2}{K-1} + (1-q)e^{t\boldsymbol{\alpha}_j} \right) + \\ &\sum_{j:y_j \neq i, y_j \neq i'} \log\left(\frac{q}{K-1} e^{-t\boldsymbol{\alpha}_j} + 1 - 2\frac{q}{K-1} + \frac{q}{K-1} e^{t\boldsymbol{\alpha}_j} \right) \Bigg\}.\end{aligned}$$

Again, this problem is convex in t, and so can be solved efficiently using Newton's method. And again since the reverse case can be computed via the same expression we can similarly optimize this in an unconstrained fashion. Specifically, we can do this for every pair of classes i and i', and return the i which gives the smallest lower bound for the worst-case choice of i'.

B.3. KL Divergence Bound

To compute actual certification radii, we will derive the KL divergence bound for the the case of K classes. Let μ, ρ be defined as in Section 4, except that as in the previous section when a label is flipped with probability q it is changed to one of the other $K-1$ classes uniformly at random. Let μ_i and ρ_i refer to the independent measures on each dimension which collectively make up the factorized distributions μ and ρ (i.e., $\mu(x) = \prod_{i=1}^d \mu_i(x)$). Further, let Y_1^i be the i^{th} element of Y_1,

meaning it is the "original" label which may or may not be flipped when sampling from μ. First noting that each dimension of the distributions μ and ρ are independent, we have

$$\begin{aligned}
\mathrm{KL}(\rho \parallel \mu) &= \sum_{i=1}^{n} \mathrm{KL}(\rho_i \parallel \mu_i) \\
&= \sum_{i:\rho_i \neq \mu_i} \mathrm{KL}(\rho_i \parallel \mu_i) \\
&= r\left(\sum_{j=1}^{K} \rho_i(j) \log\left(\frac{\rho_i(j)}{\mu_i(j)}\right)\right) \\
&= r\left(\rho_i(Y_1^i) \log\left(\frac{\rho_i(Y_1^i)}{\mu_i(Y_1^i)}\right) + \rho_i(Y_2^i) \log\left(\frac{\rho_i(Y_2^i)}{\mu_i(Y_2^i)}\right)\right) \\
&= r\left((1-q)\log\left(\frac{1-q}{\frac{q}{K-1}}\right) + \frac{q}{K-1}\log\left(\frac{\frac{q}{K-1}}{1-q}\right)\right) \\
&= r\left(1 - \frac{Kq}{K-1}\right)\log\left(\frac{(1-q)(K-1)}{q}\right).
\end{aligned}$$

Plugging in the robustness guarantee (3), we have that $g(\mu, \phi) = g(\rho, \phi)$ so long as

$$r \leq \frac{\log(4p(1-p))}{2(1-\frac{Kq}{K-1})\log\left(\frac{q}{(1-q)(K-1)}\right)}.$$

Setting $K = 2$ recovers the divergence term (4) and the bound (5).

C. Description of Label-Flipping Attacks

C.1. Attacks on Undefended Classifiers

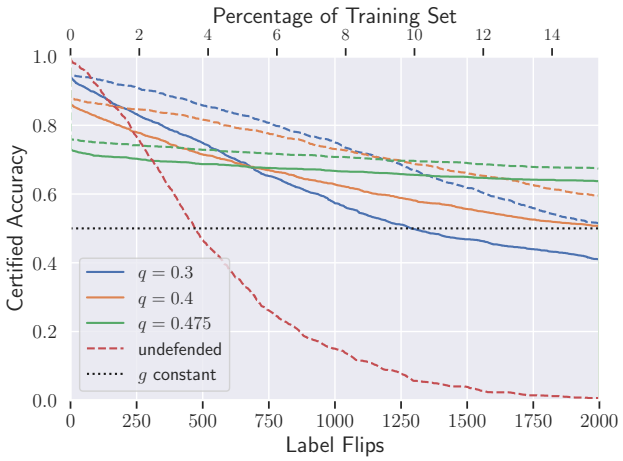
Due to the dearth of existing work on label-flipping attacks for deep networks, our attacks on MNIST and Dogfish were quite straightforward; we expect significant improvements could be made to tighten this upper bound.

For Dogfish, we used a pre-trained Inception network (Szegedy et al., 2016) to evaluate the influence of each training point with respect to the loss of each test point (Koh & Liang, 2017). As in prior work, we froze all but the top layer of the network for retraining. Once we obtained the most influential points, we flipped the first one and recomputed approximate influence using only the top layer for efficiency. After each flip, we recorded which points were classified differently and maintained for each test point the successful attack which required the fewest flips. When this was finished, we also tried the reverse of each attack to see if any of them could be achieved with even fewer flips.

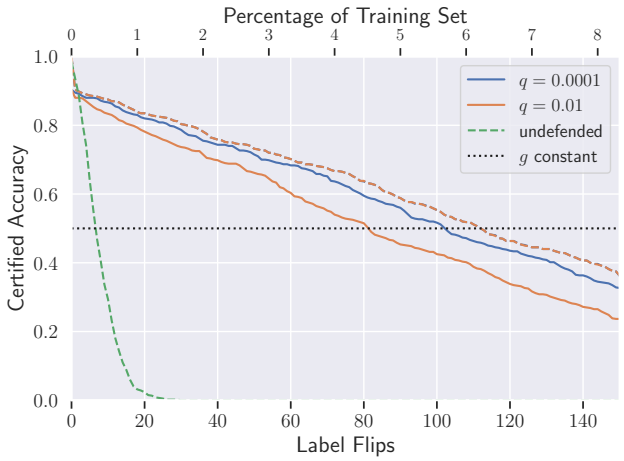
For MNIST we implemented two similar attacks and kept the best attack for each test point. The first attack simply ordered training labels by their ℓ_2 distance from the test point in feature space, as a proxy for influence. We then tried flipping these one at a time until the prediction changed, and we also tried the reverse. The second attack was essentially the same as the Dogfish attack, ordering the test points by influence. To calculate influence we again assumed a frozen feature map; specifically, using the same notation as Koh & Liang (2017), the influence of flipping the label of a training point $z = (x, y)$ to $z^- = (x, 1-y)$ on the loss at the test point z_{test} is:

$$\begin{aligned}
\frac{dL(z_{\text{test}}, \hat{\theta}_{\epsilon, z^-, -z})}{d\epsilon} &= \nabla_\theta L(z_{\text{test}}, \hat{\theta})^T \frac{d\hat{\theta}_{\epsilon, z^-, -z}}{d\epsilon} \\
&\approx -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \left(\nabla_\theta L(z^-, \hat{\theta}) - \nabla_\theta L(z, \hat{\theta})\right).
\end{aligned}$$

For logistic regression these values can easily be computed in closed form.



(a) Binary MNIST (classes 1 and 7)



(b) Dogfish

Figure C.1. MNIST 1/7 and Dogfish certified lower bounds (solid) compared to empirical upper bounds (dashed) of our classifier and the undefended classifier. Our classifier's upper and lower bounds are reasonably close, and they get closer as q decreases. The gap is due to the potential looseness of the Chernoff bound, though in practice we would expect the true robustness to be closer to the upper bound.

C.2. Attacks on Our Classifier

Recall that our theoretical classifier outputs a prediction based on $P(\boldsymbol{\alpha}^T\mathbf{y} \geq 1/2)$, where the randomness is over the label flips of $\mathbf{y}$. More specifically, the classifier's output is based on a weighted majority vote of "sub-classifiers", each of which is a simple linear classifier which outputs $\mathbf{1}\{\boldsymbol{\alpha}^T\hat{\mathbf{y}} \geq 1/2\}$ for its own labels $\hat{\mathbf{y}}$. The sub-classifier's vote is weighted by its probability under the smoothing distribution, which depends only on $\|\mathbf{y} - \hat{\mathbf{y}}\|_0$ (and is monotonically decreasing in this value). It is clear that the optimal attack to reduce $P(\boldsymbol{\alpha}^T\mathbf{y} \geq 1/2)$ is to flip the labels which will push the inner product $\boldsymbol{\alpha}^T\mathbf{y}$ as much as possible towards the incorrect label: flipping labels by their change to the inner product will add weight to the votes of the most overall number of incorrect sub-classifiers, pushing our smoothed classifier to be incorrect.

Here we make a subtle distinction: while this attack is optimal for the purpose of reducing $P(\boldsymbol{\alpha}^T\mathbf{y} \geq 1/2)$, it is *not necessarily optimal* against our classifier, even though this probability represents how our classifier (theoretically) makes a prediction. This is because in practice, we never actually compute $P(\boldsymbol{\alpha}^T\mathbf{y} \geq 1/2)$. Instead, recall from (6) that we use the Chernoff inequality to tightly bound this probability. Thus, while the attack described above is optimal for reducing the *true* probability (and therefore the theoretical robustness), it is technically possible that a different attack would cause a looser Chernoff bound, more effectively reducing our bound on the probability. In essence, our attack is optimal for modifying the LHS of (6), but not necessarily the RHS, which is ultimately how our classifier *actually* makes predictions.

With that said, the existence of an attack which causes the Chernoff bound to return a particularly sub-optimal bound seems debatable. So, while we present these results as an empirical upper bound, we believe it would not be inappropriate to also view them as an *approximate* lower bound. Of course, the actual lower bound returned by our classifier is still a genuinely guaranteed certificate. Figure C.1a displays the result of our attack on MNIST 1/7, with the undefended classifier for comparison. Observe that the empirical upper bounds (dashed lines) track the guaranteed lower bounds (solid lines) reasonably closely. The gap is under 10% accuracy and shrinks as the noise q decreases. Further, this empirical robust accuracy outperforms the undefended classifier's empirical robust accuracy by an even larger margin. Figure C.1b presents the same results on the Dogfish dataset. Our empirical attacks had very similar success rates for all values of q, so we only plot two values along with the undefended classifier. We again observe a tight correspondence between upper and lower bounds which gets tighter with smaller q.

D. Additional Tables of Results

To supplement the line plots, for each dataset and noise parameter we present here precise certified test set accuracy at specific numbers of label flips. When available, for comparison we also provide the undefended classifier's empirical accuracy when subjected to our label-flipping attack as detailed in Section C.1. For each number of label flips, the noise

hyperparameter setting which results in the highest certified accuracy is in bold.

MNIST 1/7 ($n = 13007$, 2 classes)							
	Number of Label Flips						
Noise $q \downarrow$	1	10	100	500	1000	1500	2000
(undefended)	(.9903)	(.9815)	(.9163)	(.4674)	(.1503)	(.0388)	(.0065)
0.3	**.9399**	**.9320**	**.8918**	**.7470**	.5726	.4681	.4089
0.4	.8659	.8571	.8248	.7152	.6283	.5566	.5072
0.45	.7855	.7767	.7540	.7004	.6556	.6218	.5950
0.475	.7294	.7262	.7118	.6873	**.6674**	**.6503**	**.6378**

Table 1. Certified test set accuracy on MNIST 1/7 (Figure 1a), with the undefended classifier's empirical robust accuracy for comparison. Random guessing or a constant classifier would attain 50% accuracy.

Full MNIST ($n = 60000$, 10 classes)							
	Number of Label Flips						
Noise $q \downarrow$	1	10	100	200	300	400	500
0.0125	.5693	.5689	**.5212**	**.4292**	.3333	.2446	.1706
0.025	**.5713**	**.5701**	.5053	.4040	.2999	.2096	.1407
0.05	.5495	.5486	.4954	.4160	**.3400**	**.2633**	**.2012**

Table 2. Certified test set accuracy on Full MNIST (Figure 1b). Random guessing or a constant classifier would attain 10% accuracy.

CIFAR10 ($n = 50000$, 10 classes)							
	Number of Label Flips						
Noise $q \downarrow$	1	10	50	100	200	300	400
0.012	**.7180**	**.7158**	**.6800**	**.6234**	**.4493**	**.2520**	**.1201**
0.025	.7068	.7017	.6597	.5949	.4051	.1870	.0548
0.1	.7040	.6876	.6230	.5384	.3019	.0981	.0213

Table 3. Certified test set accuracy on CIFAR10 (Figure 2). Random guessing or a constant classifier would attain 10% accuracy.

Dogfish ($n = 1800$, 2 classes)							
	Number of Label Flips						
Noise $q \downarrow$	1	10	25	50	75	100	150
(undefended)	(.9367)	(.2933)	(.0050)	(.0000)	(.0000)	(.0000)	(.0000)
0.0001	.8950	**.8667**	**.8083**	**.7150**	**.6233**	**.5167**	**.3267**
0.001	.8950	.8550	.7967	.6967	.5917	.4750	.3017
0.01	.8800	.8333	.7583	.6617	.5283	.4250	.2367
0.05	**.9367**	.7833	.7033	.5567	.4350	.3200	.1583

Table 4. Certified test set accuracy on Dogfish (Figure 3), with the undefended classifier's empirical robust accuracy for comparison. Random guessing or a constant classifier would attain 50% accuracy.

IMDB Sentiment Analysis ($n = 25000$, 2 classes)							
	Number of Label Flips						
Noise $q \downarrow$	1	10	25	50	100	200	300
0.01	.6275	.5980	.5882	.5686	**.5392**	**.4804**	**.4412**
0.025	.6364	.6154	.5944	.5594	.5105	.4406	.3287
0.05	.5878	.5344	.5038	.4656	.4160	.3206	.2519
0.1	**.7585**	**.7034**	**.6469**	**.5806**	.4788	.3263	.2135

Table 5. Certified test set accuracy on the IMDB Sentiment Analysis dataset (Figure 4). Random guessing or a constant classifier would attain 50% accuracy.

E. Additional Plots

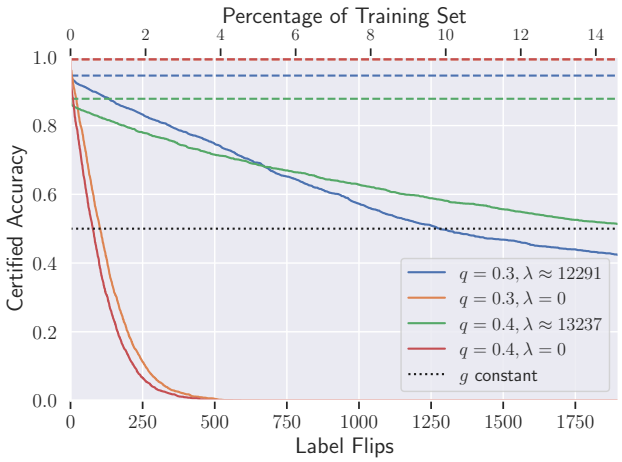


Figure E.1. MNIST 1/7 test set certified accuracy with and without ℓ_2 regularization in the computation of $\boldsymbol{\alpha}$. Note that the unregularized solution achieves almost 100% non-robust accuracy, but certifies significantly lower robustness. This implies that the "training" process is not robust enough to label noise, hence the lower margin by the ensemble. In comparison, the regularized solution achieves significantly higher margins, at a slight cost in overall accuracy.

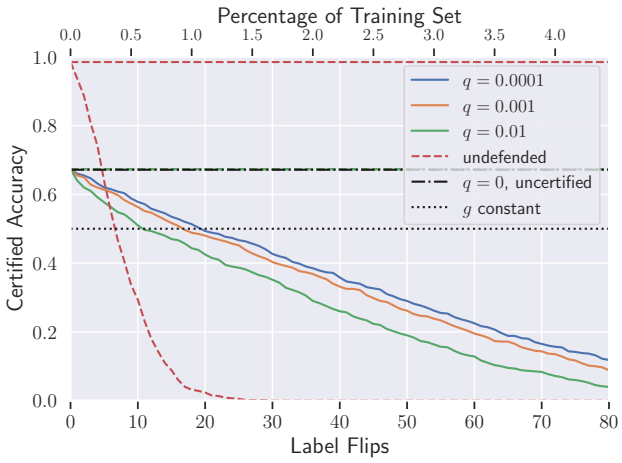


Figure E.2. Dogfish test set certified accuracy using features learned with RICA (Le, 2013). While not as performant as the pre-trained features, our classifier still achieves reasonable certified accuracy—note that the certified lines are lower bounds, while the undefended line is an upper bound. As demonstrated in the main body, deep unsupervised features significantly boost performance, but require a larger dataset.

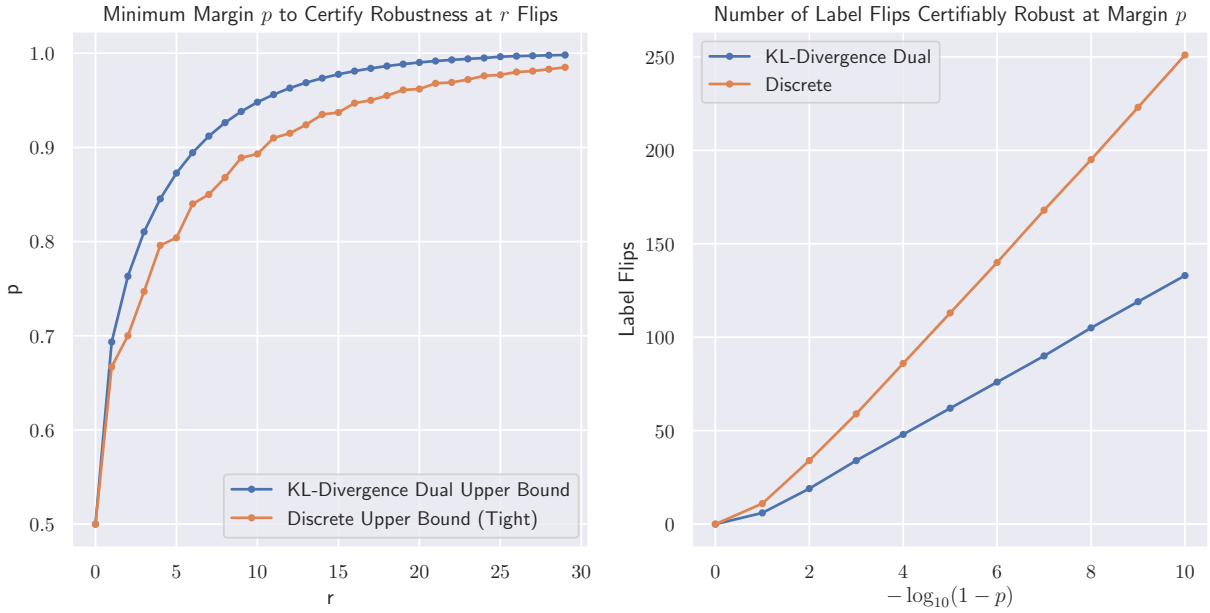


Figure E.3. **Left:** Required margin p to certify a given number of label flips using the generic KL bound (5) versus the tight discrete bound (2). **Right:** The same comparison, but inverted, showing the certifiable robustness for a given margin. The tight bound certifies robustness to approximately twice as many label flips. Both plots are with $q = 0.4$.