Intriguing Properties of Robust Classification

Supplementary Material

8. Proof of Theorem 2

In this section we prove Theorem 2. Recall

Theorem 2. *Assume that there exists a L_∞ robust classifier (margin δ) on data distribution $\mathcal{D}$, where the data points are in $[0,1]^d$. Then as long as we have $n \geqslant 37 \left\lceil \frac{1}{\delta} \right\rceil^d$ training points independently sampled from $\mathcal{D}$, the 1-nearest neighbor classifier achieves average L_∞ robust test accuracy of $\geqslant 99\%$ (average over sampling training sets).*

Proof. In order to prove this theorem, we first show that for any test point, the nearest point of a different class is at least an L_∞ distance of 2δ away. Assume that f is a robust classifier on $\mathcal{D}_\phi$. Consider a test point $\boldsymbol{x}$ and the nearest training point $\boldsymbol{x_j}$ that is of a different class. We know that f robustly classifies both $\boldsymbol{x}$ and $\boldsymbol{x_j}$ with radius δ. This implies that any point of distance $\leqslant \delta$ to either of the points must share a label with that point, and therefore $\boldsymbol{x}$ and $\boldsymbol{x_j}$ must be at least 2δ apart.

For a test point $\boldsymbol{x}$ with label y, suppose there exists a training point $\boldsymbol{x_i}$ that is less than δ away from $\boldsymbol{x}$. By our assumption, this training point also has label y. Consider any other point $\tilde{\boldsymbol{x}}$ with $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_\infty \leqslant \delta/2$. Furthermore, consider any training point $\boldsymbol{x_j}$ of a different class than $\boldsymbol{x}$. Using the triangle inequality we get that

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x_i}\|_\infty \leqslant \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_\infty + \|\boldsymbol{x} - \boldsymbol{x_i}\|_\infty < \frac{3\delta}{2} \tag{13}$$

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x_j}\|_\infty \geqslant \|\boldsymbol{x} - \boldsymbol{x_j}\|_\infty - \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_\infty \geqslant \frac{3\delta}{2}. \tag{14}$$

Therefore, we know that the nearest training point to $\tilde{\boldsymbol{x}}$ must have label y. This shows that the 1-nearest neighbor is robust to perturbation of size $\leqslant \delta/2$ of $\boldsymbol{x}$.

With this established it just remains to be shown that with enough training examples, for 99% of test points $\boldsymbol{x}$, there will be a training point close to $\boldsymbol{x}$. In order to prove that this is the case, we will split the hypercube into a set of disjoint boxes. The probability of a test point being close to a training point is at least as big as the probability of the test point being in a box that has at least one training point inside. We define the boxes by defining a set of *box centers*: For $D = \lceil 1/\delta \rceil$, set $\mathcal{C} = [\frac{1}{2}\delta, \frac{3}{2}\delta, \ldots, \frac{2D-1}{2}\delta]^d$. We define $B_r(C)$ as the L_∞ ball with radius r around C. Further, we set $\mathcal{B}$ to be the set of all boxes, $\mathcal{B} = \{B_{\delta/2}(C) : C \in \mathcal{C}\}$. We have that $|\mathcal{B}| = D^d = \lceil 1/\delta \rceil^d$. We will write p_B for the probability of a data point lying in box B under distibution $\mathcal{D}$. With this, we can write the probability of having at least 1 training point in box B as

$$\mathbb{P}(\exists i : \boldsymbol{x_i} \in B) = 1 - \mathbb{P}(\boldsymbol{x_i} \notin B \; \forall i) \tag{15}$$

$$= 1 - \prod_{i=1}^{n} (1 - \mathbb{P}(\boldsymbol{x_i} \in B)) \tag{16}$$

$$= 1 - (1 - p_B)^n \,. \tag{17}$$

We further have that $(1-p)^n = (1-p)^{\frac{1}{p}np} \leqslant \exp(-np)$, and therefore

$$\mathbb{P}(\exists i : \boldsymbol{x_i} \in B) \geqslant 1 - \exp(-np_B). \tag{18}$$

We will also use that $\exp(x) \geqslant 1 + x$ for all x, and therefore $\exp(x-1) \geqslant x$, and

$$x \exp(-x) \leqslant \exp(-1). \tag{19}$$

Then, putting everything together, for p the probability that a test point $\boldsymbol{x}$ is classified robustly we have that:

$$p \geqslant \mathbb{P}\left(\min_i \|\boldsymbol{x} - \boldsymbol{x_i}\|_\infty \leqslant \delta\right) \tag{20}$$

$$\geqslant \sum_{B \in \mathcal{B}} \mathbb{P}(\boldsymbol{x} \in B)\, \mathbb{P}(\exists i : \boldsymbol{x_i} \in B) \tag{21}$$

$$\geqslant \sum_{B \in \mathcal{B}} p_B (1 - \exp(-np_B)) \tag{22}$$

$$= 1 - \sum_{B \in \mathcal{B}} p_B \exp(-np_B) \tag{23}$$

$$= 1 - \frac{1}{n} \sum_{B \in \mathcal{B}} np_B \exp(-np_B) \tag{24}$$

$$\geqslant 1 - \frac{1}{n} \sum_{B \in \mathcal{B}} \frac{1}{e} \tag{25}$$

$$= 1 - \frac{|\mathcal{B}|}{ne} \tag{26}$$

Now since $|\mathcal{B}| = \lceil 1/\delta \rceil^d$ and we assumed that $n \geqslant 37 \lceil 1/\delta \rceil^d$ we get that $p \geqslant 99\%$. □

Note that we can adapt the proof to work for any margin $\delta' < \delta$, and not just for $\delta/2$. Furthermore, if we only assume L_2 robustness we might need many more data points, namely $\mathcal{O}(c^d d^{d/2})$ for some constant c.

9. Additional scaling law results

In this section we provide additional results for Section 5.1.

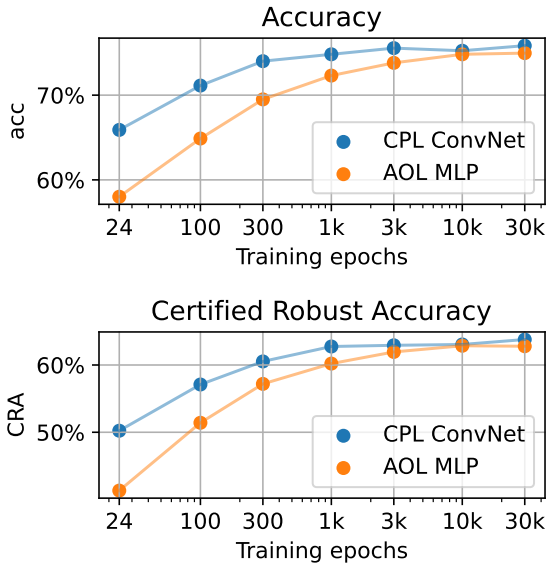

Figure 5. Scaling up the compute.

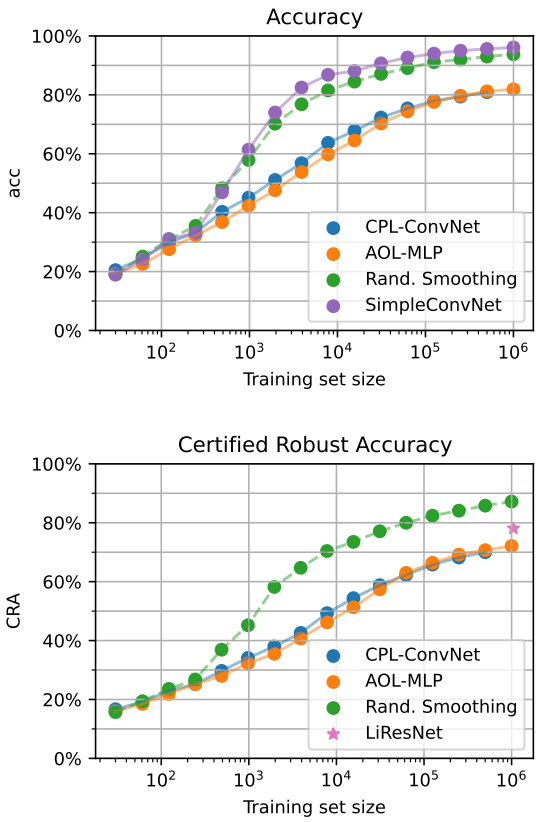

Figure 6. Scaling the size of the training data up by using additional data.

9.1. Scaling up compute

First we explore the question of whether increasing the amount of compute alone can have a positive effect similar to the one we observed when increasing the size of the dataset. The answer seems to be no. We analyze for 3 different models how they scale with compute when leaving the dataset size fixed. The results are visualized in Figure 5. Note that the x-axis is in log scale. Doubling the dataset size improves the performance less and less, and the curves for both accuracy and certified robust accuracy flatten with increasing training epochs.

9.2. Training on additional data

We are also interested whether the scaling law from Figure 1 also extends to larger training datasets, larger than the $50k$ images from the CIFAR-10 training dataset itself. Recently, there has been a lot of works that used data generated in the style of CIFAR-10 by a diffusion model [1, 16, 18, 19, 40].

We also followed this approach, and we used 1 million images from [40]. We subsampled this large dataset to obtain training sets of different sizes. For all experiments we set the number of epochs to 3000, so we did use more compute with larger datasets this time.

In addition to our own results, we also report the performance of the current best 1-Lipschitz model [19]. The authors generated 1 million additional CIFAR-10 style images with a diffusion model, using a model trained on 940 million images for data filtering. Training with this additional data allows them to achieve 78.1% certified robust accuracy.

We show the results in Figure 6. The scaling behavior from Figure 1 extends to larger datasets sizes and generated data for all models considered. However, the improvements from training 1-Lipschitz models naively on more data, without increasing the model size eventually diminish. The results from related work that carefully designs the training setup to maximize test performance show that it is possible to extend the improvement. This effect might also in part be due to the lower quality of generated data. For randomized smoothing our estimate of performance does scale very well with the amount of data.

9.3. Setup for MNIST experiments

Here we describe the setup we used for the additional results on MNIST in Figure 2. In order to be able to keep most of our setup the same, we padded the image (after subtracting the mean value) with value 0 to size 32×32. We reduced the size of our models slightly: we used 16 instead of 64 base channels for the ConvNet, and width 1024 instead of 3072 for the MLPs. We also simplified the augmentation a bit, and used only random cropping (size 4) and random flipping.

PCs	Var. Expl. %	Accuracy % Train	Test	CRA % Train	Test
1-16	72	99	43	64	31
1-512	98	100	91	89	61
513-3072	2	100	85	9	9
2049-3072	0.02	99	39	0	0
1-16 & 513-3072	74	100	86	65	35
1-3072	100	100	94	93	62

Table 1. Performance on different subsets of the principal components, as well as the proportion of variance explained by it.

9.4. Why linear-log?

It is very interesting that the performance in Figure 1 increases approximately linearly in the logarithm of the training datasets size. Whilst we do not know why this is the case we do want to provide two intuitions to the reader.

First, if we assume that the performance of a classifier on a test point depends only on a constant number of "useful" examples (*e.g.* the k-nearest neighbors), then the probability that an additional training example is "useful" for a test point is $O(1/n)$. We also have that $\sum_{i=1}^{n} \frac{1}{i} \sim \ln(n) + \gamma$, where $\gamma \sim 0.577$ is the Euler-Mascheroni constant, which might be the reason why we see this linear-log behavior.

Second, when we consider the minimum distance of a test point to a training point, this distance should behave approximately proportional to $n^{-\frac{1}{d^*}}$ for some $d^* \leqslant d$ (for details see Section 12). When $n \ll \exp(d^*)$, this term is approximately equal to $1 - \frac{\log(n)}{d^*}$, so the distance to the nearest neighbor is approximately linear in $\log(n)$. If our classifier improves (about linearly) as the nearest training examples get closer to the test points, the observation above would explain the scaling law we observe. We provide more details as well as some experimental evidence in Section 12.

10. Robust and non-robust features

In this section we provide additional visualizations for Section 5.2. For the variance explained by subsets of principal components see Figure 7, and some visualizations of images projected onto the first 16 components are in Figure 8.

In order to evaluate the capabilities of the models to overfit the training data projected onto different subsets of principal components, we repeated the experiments from Section 5.2 without data augmentation. The results are shown in Figure 9. Note that we can robustly overfit the training data, even when projected on just the first 16 principal components.

For the performance on further different subset of principal components see Figure 10.

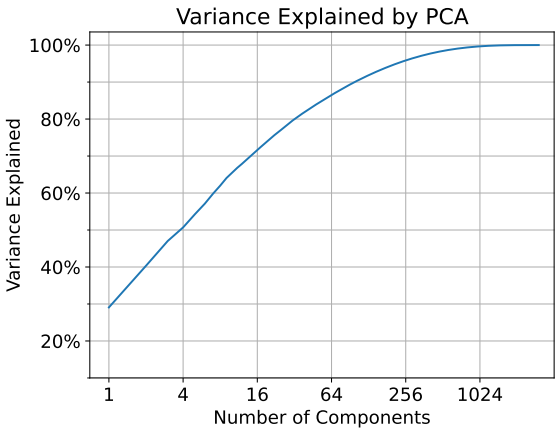

Figure 7. Variance explained by the first k principal components.

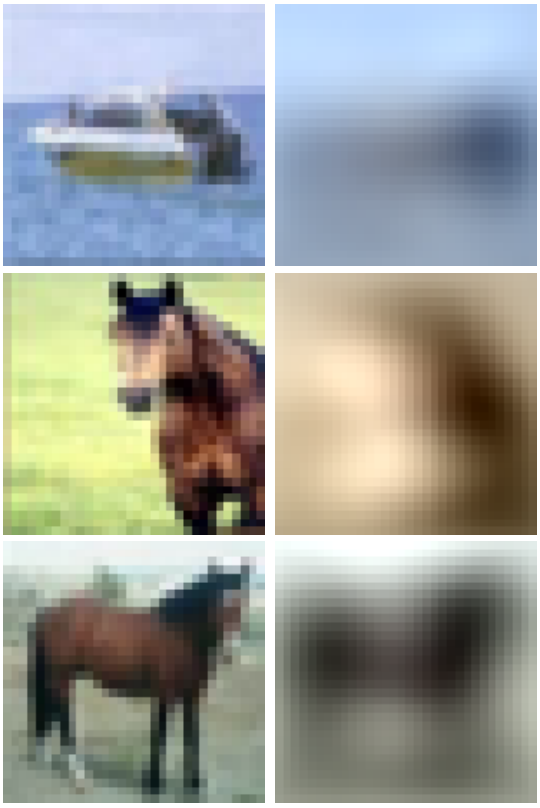

Figure 8. CIFAR-10 images (left) and their reconstruction using 16 principal components (right).

11. Robustness-accuracy trade-off

In this final experimental section we want to explore whether it is the model architecture that prevents robust models from generalizing. Often the architecture, layers, and the training pipeline in general is different depending on whether accuracy or robust accuracy is the goal metric.

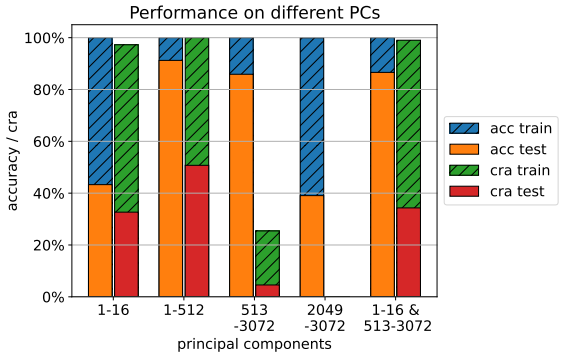

Figure 9. Performance of models when projected to a subset of principal components. Here, the robust models were trained without data augmentation.

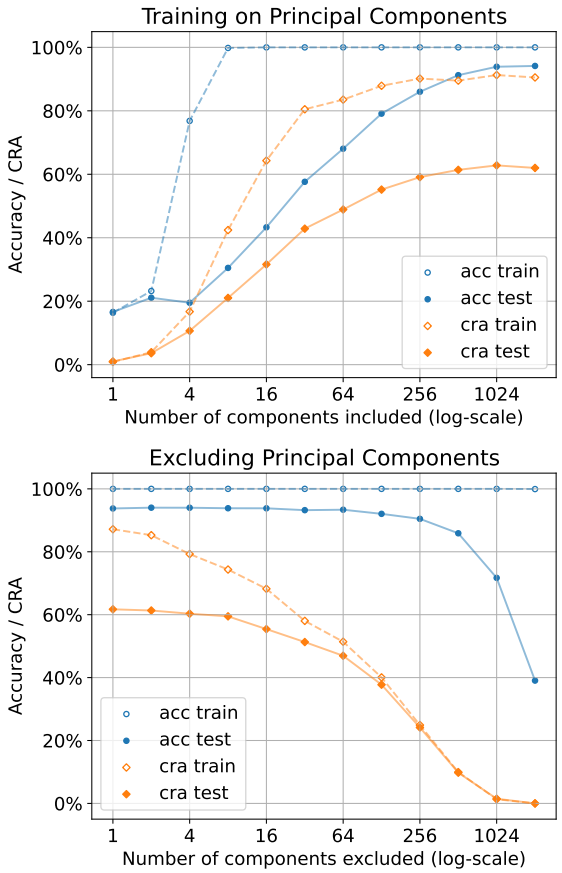

Figure 10. Training on the subset of principal components with the highest (top) or lowest (bottom) variance. Accuracy and certified robust accuracy reported come from two different models.

Therefore, we want to explore whether the change in architecture is responsible for the lack of generalization in robust networks. In order to prevent vanishing gradients and other problems when training 1-Lipschitz networks, there are a few important adaptation from standard convolutional networks. For example, in 1-Lipschitz model we use different activation layers, not MaxPooling, no BatchNorm and different initialization strategies, see also Table 2.

In order to evaluate whether the difference in architecture (*e.g.* the lack of global pooling in 1-Lipschitz networks) is the reason for the worse generalization, we carefully constructed a single architecture that can reach competitive accuracy and competitive certified robust accuracy.

Among all differences between the architectures, we have found that initialization and batch normalization cause most of the a trade-off between accuracy and certified robust accuracy, especially when training for a lower number of epochs. For initialization, it seems that identity or near-identity initialization is very useful for 1-Lipschitz networks [31, 42], whereas great accuracy requires some random initialization (*e.g. Kaiming uniform* [17] or orthogonal). For the experiments in this section we used orthogonal initialization. Getting rid of the batch normalization is slightly trickier when using 1-Lipschitz layers. However, it turns out we can use a single normalization layer applied to the output of the model to enable training to good accuracy. We furthermore can fold this normalization into our loss function, so that we can train the identical model to good accuracy and (with a different loss function) to good certified robust accuracy.

In order to smoothly interpolate between the two setting we introduce a loss function with a trade-off parameter t. It aims to be a version of the *OffsetCrossEntropy* [29], with additional normalization. We name the loss function *SelfNormalizingCrossEntropy* and define it as:

$$\text{CrossEntropy}\left(\text{Softmax}\left(\frac{s}{\text{std}(s)+t} - y\right), y\right), \quad (27)$$

where s is the vector of scores predicted by the model, y is a one-hot encoding of the label and $\text{std}(s)$ denotes the standard deviation of s.

We used this loss function to train a set of models that includes ones with good accuracy and some with good certified robust accuracy. For results see Figure 11 and Table 3. Setting $t = 0$ we can obtain 93.2% accuracy with this model, showing that the model itself allows to generalize comparably to traditional ConvNets, and we do not require layers such as MaxPooling for generalisation. With $t = \frac{1}{10}$ we can train the model to 61.7% certified robust accuracy. This shows that we can train the same architecture to competitive accuracy or competitive certified robust accuracy by only changing the loss function, and therefore that the architectural restrictions of 1-Lipschitz models are not the reason why those models fail to generalize well.

Interestingly, with our setup, there is no parameter value that is good for both tasks, but we do see a clear accuracy-robustness trade-off [37], as observed in the literature be-

ConvNets:	Standard	1-Lipschitz
Activation	ReLU	MaxMin
Blocks	3	5
Global Pooling	MaxPooling	None
Local Pooling	MaxPooling	PixelUnshuffle
Normalization	BatchNorm	None
Convolution	Standard	1-Lipschitz
Linear Layer	Standard	1-Lipschitz
Initialization	Random	Identity Map

Table 2. Difference in architecture of a SimpleConvNet and a standard 1-Lipschitz ConvNet.

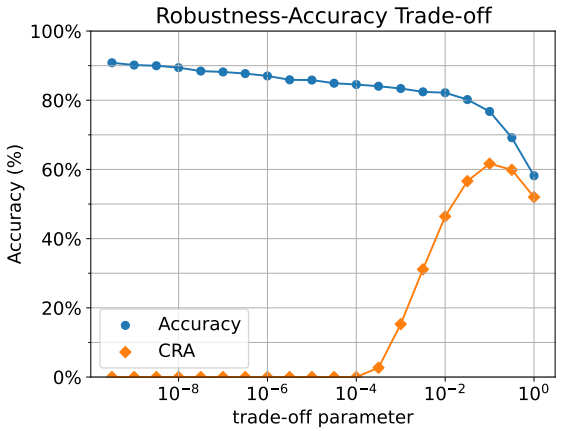

Figure 11. The same model can reach good accuracy as well as good certified robust accuracy.

t	Acc	CRA
0	93.2%	0.0%
$\frac{1}{10}$	76.8%	61.7%

Table 3. We can get high accuracy or good certified robust accuracy with the same setup, only by changing the value of trade-off parameter t in the loss function.

fore.

12. Details for linear-log behavior

In Figure 1 it seems that the certified robust accuracy depends on the logarithm of the size of the training set almost in a linear way. In this section we want to explore why this might be the case.

For a first intuition, we will consider the scenario where the performance of an architecture on every test example depends only on a few training examples. We think of those as *e.g.* the k-nearest neighbors or some support vectors. In this case, a new training data point can only have an influence on the performance (for a particular test point) if it is part of those few examples. The chance that this is the case for the n^{nt} training example added to the training set is $O(\frac{1}{n})$. Therefore, the amount of times *e.g.* a k-nearest neighbor classifier could be improved by adding an additional training example is $O(\frac{1}{n})$, and the total amount of times it might be improved when increasing the training set size from n_1 to n_2 is of order

$$\sum_{i=1+n_1}^{n_2} \frac{1}{i} \sim \ln(n_2) - \ln(n_1) = \ln\left(\frac{n2}{n1}\right). \tag{28}$$

If the amount of improvement does not increase with dataset size, this gives us an upper bound on the performance: For some value c, doubling the dataset size should at best increase the performance by c. This is in line with the behavior we observe in Figure 1.

We can also analyze this behavior in terms of distance to the nearest neighbor: We assume that for image datasets, for some dimension d^* (something like an "intrinsic dimension of the data"), it should hold that the probability p of a (test) data point being within radius r of another data point approximately follows $p \sim cr^{d^*}$ for come value c. We want to use this to make statements about the median of the distribution of the distance to the nearest neighbor, which we call r^*. In order to do this, consider the probability p_n that any of n datapoints is close to the test point. We have that

$$p_n = 1 - (1-p)^n \leqslant np \tag{29}$$

and

$$p_n = 1 - (1-p)^n \geqslant 1 - \exp(-np). \tag{30}$$

Now if we set $r = r_l$ for

$$r_l = \left(\frac{1}{2cn}\right)^{\frac{1}{d^*}}, \tag{31}$$

we get that $p_n \leqslant np = nc\frac{1}{2nc} = \frac{1}{2}$, and therefore $r^* \geqslant r_l$. Similarly, for

$$r_u = \left(\frac{1}{cn}\right)^{\frac{1}{d^*}}, \tag{32}$$

we get that $p_n \geqslant 1-\exp(-np) = 1-\frac{1}{e} > \frac{1}{2}$, and therefore $r^* \leqslant r_u$. Putting these together we have that

$$\left(\frac{1}{2cn}\right)^{\frac{1}{d^*}} \leqslant r^* \leqslant \left(\frac{1}{cn}\right)^{\frac{1}{d^*}}. \tag{33}$$

Furthermore note that as long as $\log n \ll d^*$, the following approximation should be close:

$$\left(\frac{1}{n}\right)^{\frac{1}{d^*}} = \exp\left(-\frac{\log(n)}{d^*}\right) \sim 1 - \frac{\log(n)}{d^*}. \tag{34}$$

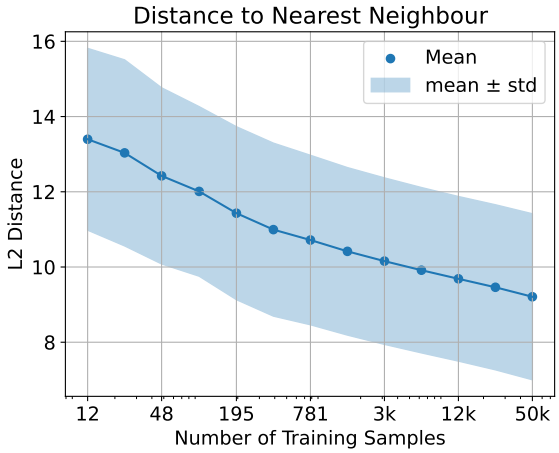

Figure 12. The distance to the nearest neighbor scales about linearly in $\log(n)$, for n the size of the training dataset.

Therefore, for some C it should approximately hold that

$$C\left(1-\frac{\log(n)}{d^*}\right) \leqslant r^* \leqslant C\left(1-\frac{\log(2n)}{d^*}\right), \qquad (35)$$

which does imply that r^* will behave approximately linearly in $\log(n)$ as long as $\log(n) \ll d^*$

We evaluated whether this relationship does hold on CIFAR-10. Our results are shown in Figure 12, where we show that indeed the distance to the nearest neighbor does behave similarly to what we expect from the theoretical analysis. If it is further the case that the (expected) certified robust accuracy of a classifier increases when a test point is closer to the training point, this would explain why the performance of this classifier scales about linearly with the logarithm of the dataset size. Note that at least when using the angular distance, it seems that on average 1-Lipschitz classifiers do better on test examples with a nearby training example.

When estimating d^* from the experimental data in Figure 12, we get an "intrinsic dimension" of about $d^* \sim 28$. This unfortunately implies that in order to get 1-nearest neighbors of distance close to 1 we will require $n \geqslant 10^{31}$. So while the 1-nearest neighbor algorithm will produce a great robust classifier with enough data, this amount of data does not seem to be reachable in practice.