

A In what sense is evaluating $\max(x_1, x_2, x_3, x_4, x_5)$ hard?

To motivate the fundamental differences of the max function with higher dimensionality, in this short section, we argue that there is unlikely to be a simple generalization of the correspondence between maximum and ReLU in $d = 2$ to general d . The idea is entirely due to Blatter (2011), though any mistakes in the interpretation are ours alone.

Theorem 6. *There is no algebraic expression for*

$$(x_1, x_2, x_3, x_4, x_5) \mapsto \max(x_1, x_2, x_3, x_4, x_5).$$

Proof. $\max(x_1, x_2, x_3, x_4, x_5)$ is a root of the polynomial $(x - x_1)(x - x_2)(x - x_3)(x - x_4)(x - x_5)$. Suppose that there was an algebraic expression for the largest of $(x_1, x_2, x_3, x_4, x_5)$, say $f(x_1, x_2, x_3, x_4, x_5)$ then, then the roots of $(x - x_1)(x - x_2)(x - x_3)(x - x_4)(x - x_5)/(x - f(x_1, x_2, x_3, x_4, x_5))$ could be found via the quartic equation, and we would have an algebraic expression for all five roots. However, Abel’s Theorem states that there is no algebraic expression for general quintic polynomials. □

The argument is subtle, so consider the same argument applied to the maximum of two values. The two roots of $(x - x_1)(x - x_2)$ are well known to be $(x_1 + x_2)/2 \pm \sqrt{(x_1 + x_2)^2 - 4x_1x_2}/2$, and the larger corresponds to adding the discriminant:

$$\frac{(x_1 + x_2) + \sqrt{(x_1 + x_2)^2 - 4x_1x_2}}{2} = \frac{(x_1 + x_2) + \sqrt{(x_1 - x_2)^2}}{2} = \frac{(x_1 + x_2) + |x_1 - x_2|}{2}.$$

Which is a well-known trick for reasoning mathematically about the maximum of two variables. A corresponding expression comes from solving the cubic equation

$$\begin{aligned} \max(x_1, x_2, x_3) = & \frac{1}{2} \frac{x_1(|x_1 - x_2| + |x_1 - x_3|) + x_2(|x_1 - x_2| + |x_2 - x_3|) + x_3(|x_2 - x_3| + |x_1 - x_3|)}{|x_1 - x_2| + |x_2 - x_3| + |x_1 - x_3|} \\ & + \frac{|x_1 - x_2| + |x_2 - x_3| + |x_1 - x_3|}{4}. \end{aligned}$$

This equation is tractable because we can assess the min and max similar to above, and impute the third value from the average.³ It is not clear what the analogous interpretation is for the quartic equation, though one presumably exists.

However, for fifth and higher-order polynomials we cannot generally even write down an algebraic expression for the roots, so determining by inspection which will be the greatest seems unlikely.

B Proofs

B.1 Proof of Theorem 4

Proof. Let $\text{perm}(d)$ denote the set of all permutations of $\{1, 2, \dots, d\}$, and let

$$\mathcal{M}_d^s(R) = \{m \in \mathcal{M}_d(R) : m(x_1, x_2, \dots, x_d) = m(x_{\sigma_1}, x_{\sigma_2}, \dots, x_{\sigma_d}) \text{ for all } \sigma \in \text{perm}(d)\}$$

denote the restriction of $\mathcal{M}_d(R)$ to those elements that are invariant to a reordering of its arguments. Because max is symmetric in this sense, any optimal approximation to it must lie in $\mathcal{M}_d^s(R)$:

³See also the excellent exposition given at <https://math.stackexchange.com/a/89702/92999>.

Theorem 7. For all R ,

$$\min_{m \in \mathcal{M}_d(R)} \|m - \max\|_\infty = \min_{m \in \mathcal{M}_d^s(R)} \|m - \max\|_\infty.$$

Proof. Assume otherwise, that is:

$$\min_{m \in \mathcal{M}_d(R)} \|m - \max\|_\infty < \min_{m \in \mathcal{M}_d^s(R)} \|m - \max\|_\infty.$$

Meaning that there is an m that is not symmetric and an x such that $m(x') < m^s(x)$ for all symmetric m^s and all x' . In particular $m(x_\sigma) < m^s(x)$ for all permutations x_σ of x . Thus

$$\frac{1}{d!} \sum_{\sigma \in \text{perm}(x)} m(x'_\sigma) < m^s(x).$$

However, $x \mapsto \frac{1}{d!} \sum_{\sigma \in \text{perm}(x)} m(x'_\sigma) < m^s(x)$ is evidently symmetric, a contradiction. \square

Thus, it is without loss of generality to optimize over $\mathcal{M}^s(R)$ rather than $\mathcal{M}(R)$, and we turn to operationalizing the symmetry assumption in terms of the coefficients. For $r > 1$, $\omega_{jr}(1_d - e_{dk}) = 1^4$ for all $k = 1, 2, \dots, d$ and all $j = 1, \dots, \binom{d}{r}$ since there is only a single non-one value. Thus all terms of order greater than 1 are equal, and the sums are equal by symmetry, so we necessarily have that for all $f \in \mathcal{M}_d^s(R)$, $\beta_1^1 = \beta_1^2 = \dots = \beta_1^d$. Call this single value β_1 , and (for $0, 1 \in R$)

$$\mathcal{M}_d^s(R) = \left\{ x \mapsto \beta_0 + \beta_1 S(x; 1, d) + \sum_{r \in R \setminus \{0, 1\}} \sum_{j=1}^{\binom{d}{r}} \beta_r^j s(x; C(j, r, d)) : \beta_r^j, \beta_0, \beta_1 \in \mathbb{R} \right\}.$$

Repeating this process for pools consisting of entirely of 2, except for $3, 4, \dots, d-1$ zeros in turn implies that the estimator must be a function of $S(x; r, d)$ alone, and not the individual terms of the sum separately:

$$\mathcal{M}_d^s(R) = \left\{ x \mapsto \sum_{r \in R} \beta_r S(x; r, d) : \beta_r \in \mathbb{R} \right\}.$$

So, the maximization over all $m \in \mathcal{M}_d(R)$ can be reduced to the maximization of $|R|$ scalar coefficients.

We want to show that

$$\min_{\beta_0, \beta_1} \|x_{(1)} - \beta_0 - \beta_1 S(x; 1, d)\|_\infty = \frac{1}{2} \frac{d-1}{d} \quad (13)$$

$$\min_{\beta_{d-1}} \|x_{(1)} - \beta_{d-1} S(x; d-1, d)\|_\infty = 1/(2d-1) \quad (14)$$

$$\min_{\beta_0, \beta_{d-1}} \|x_{(1)} - \beta_0 - \beta_{d-1} S(x; d-1, d)\|_\infty = 1/(2d) \quad (15)$$

$$\min_{\beta_0, \beta_1, \beta_{d-1}} \|x_{(1)} - \beta_0 - \beta_1 S(x; 1, d) - \beta_{d-1} S(x; d-1, d)\|_\infty = 1/(2(d+1)) \quad (16)$$

$$\min_{\beta_0, \beta_1, \beta_{d-2}, \beta_{d-1}} \|x_{(1)} - \beta_0 - \beta_1 S(x; 1, d) - \beta_{d-2} S(x; d-2, d) - \beta_{d-1} S(x; d-1, d)\|_\infty = 1/d^2 \quad (17)$$

$$\min_{\beta_0, \beta_1, \dots, \beta_{d-1}} \|x_{(1)} - \beta_0 - \beta_1 S(x; 1, d) - \dots - \beta_{d-1} S(x; d-1, d)\|_\infty = 1/2^d \quad (18)$$

⁴ ω is defined in subsection 4.1, recall.

where the L_∞ norm is taken over values of x .

The proof of each is similar, to facilitate their analysis, we distill the essence of each into a “meta”-proof. There are three essential steps:

1. Conjecture coefficients, β^* with the help of Appendix D.
2. Lower bound: for any f and for any set of points $P \subseteq [0, 1]^d$,

$$\min_{\beta} \max_x |f(\beta, x)| \geq \min_{\beta} \max_{x \in P} |f(\beta, x)| = \max_{x \in P} |f(\beta^*, x)|$$

where we prove the equality by contradiction: suppose there is a better β' , and derive a contradiction.

3. Upper bound: for any f ,

$$\min_{\beta} \max_x |f(\beta, x)| \leq \max_x |f(\beta^*, x)|.$$

Then we show that $\max_x |f(\beta^*, x)| \leq \beta_0^*$ by writing it as a linear combination of order statistics.

For Equation 13, let $(\beta_0^* \ \beta_1^*) = ((d-1)/(2d) \ 1)$. Suppose that there were some $(\beta'_0 \ \beta'_1)$ achieving a criterion $< \beta_0^*$. Evaluating the error at $x = (1 \ 1 \ \dots \ 1)$ and $x = (1 \ 0 \ \dots \ 0)$, this implies

$$\begin{aligned} 1 - \beta_0 - \beta_1/d < +\beta_0^* \text{ and } 1 - \beta_0 - \beta_1 > -\beta_0^* &\implies (d-1)(1 - \beta_0) < (d+1)\beta_0^* \\ &\iff \beta_0 > 1 - \frac{d+1}{d-1}\beta_0^* = \beta_0^*. \end{aligned}$$

A contradiction to the condition at zero. Thus, $\min_{\beta_0, \beta_1} \|x_{(1)} - \beta_0 - \beta_1 S(x; 1, d)\|_\infty \geq \beta_0^*$.

On the other hand, from Theorem 3

$$x_{(1)} - \beta_0^* - S(x; 1, d) = \left(1 - \frac{1}{d}\right) x_{(1)} - \frac{1}{d} (x_{(2)} + \dots + x_{(d)}) - \beta_0^* \in [-\beta_0^*, +\beta_0^*].$$

■

For Equation 14, let $\beta_{d-1}^* = \frac{2d}{2d-1}$. Suppose that there were some β'_{d-1} achieving a criterion strictly less than $1/(2d-1)$. Evaluating the criterion at $x = (1 \ 1 \ \dots \ 1)$ and $x = (1 \ 1 \ \dots \ 1 \ 0)$, this is only possible if

$$1 - \beta'_{d-1} > -\frac{1}{2d-1} \text{ and } 1 - \beta'_{d-1} \frac{d-1}{d} < \frac{1}{2d-1} \iff \frac{2d}{2d-1} > \beta'_{d-1} \text{ and } \beta'_{d-1} > \frac{2d}{2d-1},$$

which is to say it is impossible. On the other hand, from Theorem 3

$$\begin{aligned} x_{(1)} - \frac{2d}{2d-1} S(x; d-1, d) &= x_{(1)} - \frac{2d}{2d-1} \left(\frac{d-1}{d} x_{(1)} + \frac{1}{d} x_{(2)} \right) \\ &= \frac{x_{(1)} - 2x_{(2)}}{2d-1} \in \left[-\frac{1}{2d-1}, +\frac{1}{2d-1} \right]. \end{aligned}$$

■

For Equation 15, let $(\beta_0^* \ \beta_{d-1}^*) = (1/(2d) \ 1)$. Suppose that there are some $(\beta'_0 \ \beta'_{d-1})$ achieving a criterion $< \beta_0^*$. Evaluating the error at $x = (1 \ 1 \ 1 \ \dots \ 1)$ implies

$$-1/(2d) < 1 - \beta'_0 - \beta'_{d-1} \iff \beta'_{d-1} < 1 - \beta'_0 + 1/(2d). \quad (19)$$

Evaluating the error at $x = (1 \ 0 \ 0 \ \dots \ 0)$ implies

$$\begin{aligned} 1 - \beta'_0 - \beta'_{d-1}(d-1)/d < +1/(2d) &\iff 1 - \beta'_0 - 1/(2d) < \beta'_{d-1}(d-1)/d \\ &\iff \frac{d}{d-1} (1 - \beta'_0 - 1/(2d)) < \beta'_{d-1}. \end{aligned} \quad (20)$$

Combining Equation 19 and Equation 20

$$\begin{aligned} \frac{d}{d-1} (1 - \beta'_0 - 1/(2d)) < 1 - \beta'_0 + 1/(2d) &\iff \left(\frac{d}{d-1} - 1 \right) (1 - \beta'_0) < \frac{1}{2d} \left(1 + \frac{d}{d-1} \right) \\ &\iff (1 - \beta'_0) < \frac{2d-1}{2d} \\ &\iff \beta'_0 > \frac{1}{2d}. \end{aligned} \quad (21)$$

An obvious contradiction to the condition at 0 that $|\beta'_0| < 1/(2d)$.

On the other hand, from Theorem 3

$$x_{(1)} - \frac{1}{2d} - S(x; d-1, d) = \frac{1}{d} (x_{(1)} - x_{(2)}) - \frac{1}{2d} \in \left[-\frac{1}{2d}, +\frac{1}{2d} \right].$$

■

For Equation 16, let $(\beta_0^* \ \beta_1^* \ \beta_{d-1}^*) = (1/2 \ -d/(d-2) \ d(d-1)/(d-2)) / (d+1)$. Suppose that there were a $(\beta'_0 \ \beta'_1 \ \beta'_{d-1})$ achieving a criterion less than β_0^* , then the condition at $x = (1, 1, \dots, 1), (1, 1, 0, \dots, 0), (1, 0, \dots, 0)$ imply, respectively:

$$\begin{aligned} 1 - \beta'_0 - \beta'_1 - \beta'_{d-1} < +\beta'_0 &\iff 1 - \beta'_1 - \beta'_{d-1} < +2\beta'_0 \\ 1 - \beta'_0 - \beta'_1 \frac{2}{d} - \beta'_{d-1} > -\beta'_0 &\iff 1 - \beta'_1 \frac{2}{d} - \beta'_{d-1} > 0 \\ 1 - \beta'_0 - \beta'_1 \frac{1}{d} - \beta'_{d-1} \frac{d-1}{d} < +\beta'_0 &\iff 1 - \beta'_1 \frac{1}{d} - \beta'_{d-1} \frac{d-1}{d} < +2\beta'_0 \end{aligned}$$

combining the first and the second implies $-\frac{2d}{d-2}\beta'_0 \leq \beta'_1$, while combining the second and third implies $\beta'_1 \leq \frac{2d^2}{d-2}\beta'_0 - \frac{d}{d-2}$. Combining these

$$-\frac{2d}{d-2}\beta'_0 < \frac{2d^2}{d-2}\beta'_0 - \frac{d}{d-2} \iff \beta'_0 > \frac{1}{2(d+1)},$$

a contradiction to the condition at $x = 0$.

On the other hand, from Theorem 3

$$\begin{aligned} x_{(1)} - \frac{1}{2(d+1)} + \frac{1}{(d+1)(d-2)}(x_{(1)} + \dots + x_{(d)}) - \frac{(d-1)}{(d+1)(d-2)}((d-1)x_{(1)} + x_{(2)}) \\ = \frac{1}{d+1}(x_{(1)} - x_{(2)}) + \frac{1}{(d+1)(d-2)}(x_{(3)} + \dots + x_{(d)}) - \frac{1}{2(d+1)} \in \left[-\frac{1}{2(d+1)}, +\frac{1}{2(d+1)} \right]. \end{aligned}$$

For Equation 17: $(\beta_0^*, \beta_1^*, \beta_{d-2}^*, \beta_{d-1}^*) = \left(\frac{1}{d^2}, \frac{2}{d(d-3)}, -1 - \frac{2}{d(d-3)}, 2\right)$. Suppose that there were a $(\beta'_0, \beta'_1, \beta'_{d-2}, \beta'_{d-1})$ achieving a criterion less than β_0^* . In order to scale up the proof by contraction, we use Theorem 8.

Theorem 8 (Carver (1922), Theorem 3). $Ax < b$ is consistent $\iff y = 0$ is the only solution for $y \geq 0$, $y^\top A = 0, y^\top b \leq 0$.

Applying the assumed condition at $x = (0, 0, \dots, 0)$, $(1, 1, \dots, 1)$, $(1, 1, 1, 0, \dots, 0)$, $(1, 1, 0, \dots, 0)$, and $(1, 0, \dots, 0)$ imply, respectively that:⁵

$$\begin{aligned} \beta'_0 &< +\beta_0^* \\ 1 - \beta'_0 - \beta'_1 - \beta'_{d-2} - \beta'_{d-1} &> -\beta_0^* \\ 1 - \beta'_0 - \beta'_1 \frac{3}{d} - \beta'_{d-2} - \beta'_{d-1} &< +\beta_0^* \\ 1 - \beta'_0 - \beta'_1 \frac{2}{d} - \beta'_{d-2} \frac{(d+1)(d-2)}{d(d-1)} - \beta'_{d-1} &> -\beta_0^* \\ 1 - \beta'_0 - \beta'_1 \frac{1}{d} - \beta'_{d-2} \frac{d-2}{d} - \beta'_{d-1} \frac{d-1}{d} &< +\beta_0^* \end{aligned}$$

or $Ax < b$ for

$$A = \begin{pmatrix} +1 & 0 & 0 & 0 \\ +1 & +1 & +1 & +1 \\ -1 & -\frac{3}{d} & -1 & -1 \\ +1 & +\frac{2}{d} & \frac{(d+1)(d+2)}{d(d-1)} & +1 \\ -1 & -\frac{1}{d} & -\frac{d-2}{d} & -\frac{d-1}{d} \end{pmatrix}, x = \begin{pmatrix} \beta'_0 \\ \beta'_1 \\ \beta'_{d-2} \\ \beta'_{d-1} \end{pmatrix}, \text{ and } b = \begin{pmatrix} \beta_0^* \\ \beta_0^* + 1 \\ \beta_0^* - 1 \\ \beta_0^* + 1 \\ \beta_0^* - 1 \end{pmatrix}.$$

It is straightforward to verify that

$$y = \begin{pmatrix} 1/d \\ (d-2)/(2d) \\ (d/2-1) \\ (d-1)/2 \\ 1 \end{pmatrix}$$

satisfies $y^\top A = 0$ and $y^\top b = (\beta_0^* d - 1/d) = 0$. The last equality is because $\beta_0^* = 1/d^2$. We have demonstrated a $y \geq 0, y \neq 0$ with $y^\top b \leq 0$, thus the system is not consistent, which forms a contradiction to the supposition that there exists a $(\beta'_0, \beta'_1, \beta'_{d-2}, \beta'_{d-1})$ achieving a criterion less than β_0^* .

On the other hand, from Theorem 3

$$\begin{aligned} x_{(1)} - \frac{1}{d^2} - \frac{2}{d(d-3)} \frac{1}{d} (x_{(1)} + \dots + x_{(d)}) + \frac{(d-2)(d-1)}{d(d-3)} 2S(x; d-2, d) - \frac{2}{d} ((d-1)x_{(1)} + x_{(2)}) \\ = \frac{2}{d^2} x_{(1)} - \frac{2}{d^2} x_{(2)} + \frac{2}{d^2} x_{(3)} - \frac{2}{d^2} \frac{1}{d-3} (x_{(4)} + \dots + x_{(d)}) - \frac{1}{d^2} \in \left[-\frac{1}{d^2}, +\frac{1}{d^2}\right]. \end{aligned}$$

5

$$\begin{aligned} S(x; d-2, d) &= \frac{1}{\binom{d}{d-2}} \left(\binom{d-1}{d-3} x_{(1)} + \binom{d-2}{d-3} x_{(2)} + \binom{d-3}{d-3} x_{(3)} \right) \\ &= \frac{1}{d} \left((d-2)x_{(1)} + \frac{2(d-2)}{(d-1)} x_{(2)} + \frac{2}{(d-1)} x_{(3)} \right) \end{aligned}$$

■

For Equation 18. The idea is the same, but handling $d+1$ equations simultaneously requires more powerful notation. Let $B(d)$ be the $d-1 \times d$ matrix with (r, c) th element $\binom{d-c}{r-1} / \binom{d}{r}$ if $r+c \leq d+1$, and zero otherwise. We can write the condition Equation 4 simultaneously for all r as

$$S(x; d)^\top \triangleq \begin{pmatrix} S(x; 0, d) \\ S(x; 1, d) \\ S(x; 2, d) \\ S(x; 3, d) \\ \vdots \\ S(x; d-1, d) \end{pmatrix}^\top = \begin{pmatrix} 1 \\ x_{(1)} \\ x_{(2)} \\ \vdots \\ x_{(d)} \end{pmatrix}^\top \begin{pmatrix} 1 & 0_{1 \times d-1} \\ 0_{d \times 1} & B(d)^\top \end{pmatrix} \in \mathbb{R}^{1 \times d}. \quad (22)$$

We have indicated the dimensionality of the zero vectors with subscripts.

Let $V(d)$ be the $d \times d+1$ matrix

$$\begin{pmatrix} 0 & 1 & 1 & \dots & 1 & 1 \\ 0 & 0 & 1 & \dots & 1 & 1 \\ \vdots & \dots & & \vdots & & \\ 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix} \quad (23)$$

of points at which we evaluate the estimator. Let $s(d)$ be the $d+1$ -dimensional vector starting and ending with j th element $(-1)^{j-1}$; $\text{diag}(s(d))$ encodes the signs of the binding inequalities. Let

$$A = \text{diag}(s(d)) \begin{pmatrix} 1_{1 \times d+1} \\ B(d)V(d) \end{pmatrix}^\top, b = 1/2^d + \text{diag}(s(d)) \begin{pmatrix} 0 \\ 1 \\ 1 \\ \dots \\ 1 \end{pmatrix}.$$

where we indicate the dimensionality of the $d+1$ -dimensional vector of ones. Here

$$y = \begin{pmatrix} \binom{d}{0} \\ \binom{d}{1} \\ \binom{d}{2} \\ \vdots \\ \binom{d}{d-1} \\ \binom{d}{d} \end{pmatrix}$$

satisfies $y^\top A = 0$ and $y^\top b = 0$. Thus, there are no parameters achieving a criterion $< 1/2^d$, by Theorem 8.

For the other direction, let

$$\beta^\star = \begin{pmatrix} \beta_0^\star \\ \beta_1^\star \\ \vdots \\ \beta_{d-1}^\star \end{pmatrix} = -1 \times \begin{pmatrix} -1/2^d \\ (-1/2)^{d-1} \binom{d}{1} \\ (-1/2)^{d-2} \binom{d}{2} \\ \vdots \\ (-1/2)^1 \binom{d}{d} \end{pmatrix}.$$

Then, by the binomial theorem⁶

$$-\begin{pmatrix} 1 & 0_{1 \times d-1} \\ 0_{d \times 1} & B(d)^\top \end{pmatrix} \beta^\star = \begin{pmatrix} 1/2^d \\ 1 - 2/2^d \\ +2/2^d \\ -2/2^d \\ \vdots \end{pmatrix}.$$

Thus

$$x_{(1)} - S(x; d)^\top \beta^\star = 1/2^d + \sum_{j=1}^d \frac{2}{2^d} x_{(j)} (-1)^{j+1} \in \left[-\frac{1}{2^d}, +\frac{1}{2^d} \right].$$

□

Theorem 9 (Fundamental perturbation analysis of L_∞ optimization over Δ_d). *Given a vector $\gamma \in \mathbb{R}^d$:*

$$(\gamma_{(1)} + \gamma_{(d)})/2 = \arg \min_{a \in \mathbb{R}} \max_{\lambda \in \Delta_d} |\gamma^\top \lambda - a| \quad (24)$$

$$(\gamma_{(1)} - \gamma_{(d)})/2 = \min_{a \in \mathbb{R}} \max_{\lambda \in \Delta_d} |\gamma^\top \lambda - a|. \quad (25)$$

Proof. The maximum absolute value element is given by $\|x\|_\infty = \max_{\lambda \in \Delta_d} |x^\top \lambda| = \max\{|x_{(1)}|, |x_{(d)}|\}$.

For Equation 24, for any $\lambda \in \Delta_d$, $|\gamma^\top \lambda - a| = |(\gamma - 1_d \times a)^\top \lambda|$. For any fixed a , the largest element (in magnitude) of $\gamma - 1_d \times a$ will be achieved at the coordinate of either the highest or lowest element of γ , and the inner optimization evaluates to

$$\max\{|\gamma_{(1)} - a|, |\gamma_{(d)} - a|\}.$$

Let $a^\star = (\gamma_{(1)} + \gamma_{(d)})/2$. Suppose that $a = a^\star + \epsilon$ for $\epsilon > 0$. Then $|\gamma_{(1)} - a| < |\gamma_{(d)} - a| = |(-\gamma_{(1)} + \gamma_{(d)})/2 - \epsilon| = (\gamma_{(1)} - \gamma_{(d)})/2 + \epsilon$, this quantity can be reduced by setting ϵ to zero. Thus the optimal a is $\leq a^\star$.

Now, suppose that $a = a^\star - \epsilon$ for $\epsilon > 0$. Then $|\gamma_{(d)} - a| < |\gamma_{(1)} - a| = |(\gamma_{(1)} - \gamma_{(d)})/2 - \epsilon| = (\gamma_{(1)} - \gamma_{(d)})/2 + \epsilon$, this quantity can be reduced by setting ϵ to zero. Thus the optimal a is $\geq a^\star$.

Equation 25 follows straightforwardly from plugging Equation 24 into the criterion $(\gamma_{(1)} + \gamma_{(d)})/2$ implies that the largest value of $\gamma - 1_d \times a$ is $(\gamma_{(1)} - \gamma_{(d)})/2$, and the smallest value is $(\gamma_{(d)} - \gamma_{(1)})/2$, with all other values in-between. □

Theorem 10. *Let $L(\beta) = (0 \ 1 \ \dots 1) - \beta^\top B(d)V(d)$, then*

$$\beta_0^\star, \beta_1^\star, \dots, \beta_{d-1}^\star \triangleq \arg \min_{\beta_0, \beta_1, \dots, \beta_{d-1}} \max_{\lambda \in \Delta_d} |L(\beta_1, \dots, \beta_{d-1})\lambda - \beta_0|$$

satisfies $L(\beta_1^\star, \dots, \beta_{d-1}^\star) \geq 0$, $\beta^\star = \arg \min_\beta \|L(\beta)\|_\infty$, and $\beta_0^\star = \|L(\beta)\|_\infty$.

Proof. Note that the first column of $B(d)V(d)$ is entirely zero, so $L(\beta_1, \dots, \beta_{d-1})_1 = 0$ for all $\beta_1, \dots, \beta_{d-1}$. Thus, if $L(\beta_1, \dots, \beta_{d-1}) \geq 0$, then the smallest element will be zero, and the last two assertions follow directly from Equation 24 and Equation 25.

⁶ $\sum_{k=0}^n \binom{n}{k} r^k = (1+r)^n$.

Thus, we need only to show that at any candidate optimum $L(\beta_1, \dots, \beta_{d-1}) \geq 0$ for all $i \iff (\beta_1 \ \beta_2 \ \dots \ \beta_{d-1}) B(d)V(d) \leq 1$ for all i . Note that no candidate solution would have $((\beta_1 \ \dots \ \beta_{d-1}) B(d)V(d))_i < 0$ for any i since this would result in a criterion $> 1/2$, which is trivially attainable with $\beta_1 = \dots = \beta_{d-1} = 0$.

Suppose that at a candidate β , $m \triangleq \max_i (\beta^\top B(d)V(d))_i > 1$, then the minimum element of L will be $1 - m < 0$, and by Equation 25, the attained criterion will then be

$$(\max_i L(\beta)_i - (1 - m))/2.$$

And $\max_i L(\beta)_i = \max(0, 1 - \min_i (\beta^\top B(d)V(d))_i)$. We have shown that

$$(\beta_1 \ \dots \ \beta_{d-1}) B(d)V(d) \geq 0,$$

thus $\max_i L(\beta/m)_i \geq \max_i L(\beta)_i$, and we have that

$$\max_i L(\beta/m)_i \leq (\max_i L(\beta)_i - (1 - m)) \iff (1 - m) \leq \max_i L(\beta)_i - \max_i L(\beta/m)_i.$$

We need to consider three separate cases:

1. $0 = \max_i L(\beta/m)_i \implies 0 = \max_i L(\beta)_i$ in which case the inequality holds strictly.
2. $L(\beta)_i = 0$ but $\max_i L(\beta/m)_i > 0$ then $\max_i L(\beta/m)_i = 1 - \frac{1}{m} \min_i (\beta^\top B(d)V(d))_i$. This holds only if $\min_i (\beta^\top B(d)V(d))_i > 1$, so

$$1 - m \leq -1 \times \left(1 - \frac{1}{m} \min_i (\beta^\top B(d)V(d))_i\right) \iff 2 < m + \min_i (\beta^\top B(d)V(d))_i/m$$

which follows from the AM-GM inequality: $a > 1, b > 1 \implies (a + b/a)/2 \geq \sqrt{b} > 1$.

3. If both terms are nonzero, then

$$1 - m \leq \frac{1 - m}{m} \times \min_i (\beta^\top B(d)V(d))_i \iff m \geq \min_i (\beta^\top B(d)V(d))_i.$$

□

Theorem 11. *An optimal estimator is increasing.*

Proof. Two points x_1, x_2 will have $x_1 \geq x_2$ if and only if their V representations are similarly ordered, thus f will be increasing if and only if $\beta^\top B(d)V(d) \geq 0$.

Suppose otherwise, that for some i $(\beta^\top B(d)V(d))_i < 0$. Then the criterion will be $\geq (1 - \beta^\top B(d)V(d))_i/2 > 1/2$ by Equation 24. However, by setting $\beta = 0$, a criterion of $1/2$ can always be achieved, thus β cannot be optimal. □

B.2 Proof of Theorem 5

Proof. Theorem 11 shows that an optimal estimator is weakly increasing. Thus, for $\delta < \beta_0^*$, $p \triangleq (\delta, 0, \dots, 0) \implies f(p) \geq \beta_0 = f(0) = \beta_0^* > \delta = \max(p)$ and means an error of at least $\beta_0^* - \delta$. So at p , the error will be at least ϵ iff $\beta_0^* - \epsilon \geq \delta$. This is true for all $p \in [0, \delta]^d$, thus $[0, \beta_0^* - \epsilon]^d \subseteq W(\epsilon; R) \implies \text{vol}(W(\epsilon; R)) \geq (\beta_0^* - \epsilon)^d$.

Theorem 10 shows that $\beta_0^* = \text{dist}(R)/2$, and the assertion is proven. □

This volume bound could be improved by including more than just the intercept in the computation of the bound. And, as one might intuit, a similar analysis holds at all vertices.

C L_2 problem

For brevity in what follows, let $\kappa(d) = B(d)V(d) \in \mathbb{R}^{d-1 \times d+1}$, then the difference between the fitted and actual values, as a function of λ is:

$$\lambda \mapsto (1_{d+1} - e_{d+1,1})^\top \lambda - \beta_0 - \beta^\top \kappa(d) \lambda = -\beta_0 + ((1_{d+1} - e_{d+1,1}) - \kappa(d)^\top \beta)^\top \lambda. \quad (26)$$

From Equation 26 the squared L_2 error is⁷

$$\int_{\Delta_d} (\beta_0 - ((1_d - e_{d1}) - \kappa(d)^\top \beta)^\top \lambda)^2 d\lambda. \quad (27)$$

To lighten the notation, we wrap this optimization problem into Theorem 12.

Theorem 12. *Let $\alpha_0 \in \mathbb{R}$, $\alpha \in \mathbb{R}^d$, $A \in \mathbb{R}^{d+1}$, and $\Xi \in \mathbb{R}^{d+1 \times d}$. Let $v(d) = \int_{\Delta_d} d\lambda$, then*

$$\min_{\alpha_0, \alpha} \int_{\Delta_d} (\alpha_0 - (A - \Xi\alpha)^\top \lambda)^2 d\lambda = v(d)A^\top \left(I - \Sigma(d)\Xi (\Xi^\top \Sigma(d)\Xi)^\dagger \Xi^\top \Sigma(d) \right) A. \quad (28)$$

Proof. Expanding the criterion above:

$$\alpha_0^2 v(d) - 2\alpha_0 (A - \Xi\alpha)^\top \left(\int_{\Delta_d} \lambda d\lambda \right) + (A - \Xi\alpha)^\top \left(\int_{\Delta_d} \lambda \lambda^\top d\lambda \right) (A - \Xi\alpha).$$

The first order criterion for optimality of α_0 evidently requires that

$$\alpha_0 = (A - \Xi\alpha)^\top \left(\int_{\Delta_d} \lambda d\lambda \right) / v(d),$$

thus, the criterion equals

$$v(d) \times (A - \Xi\alpha)^\top \left(\int_{\Delta_d} \lambda \lambda^\top d\lambda / v(d) - \int_{\Delta_d} \lambda d\lambda / v(d) \int_{\Delta_d} \lambda^\top d\lambda / v(d) \right) (A - \Xi\alpha).$$

Write the inner term – the covariance matrix of a Dirichlet(1, 1, ..., 1) distribution – as $\Sigma(d)$, then this weighted least squares problem is solved by

$$\alpha^* = (\Xi^\top \Sigma(d)\Xi)^\dagger \Xi^\top \Sigma(d)A.$$

Plugging this equation into the criterion gives Equation 28. □

Phrasing Equation 27 in terms of Equation 28, we have that the squared L_2 error of the optimal coefficients is:

$$v(d)(1_d - e_{d1})^\top \left(I - \Sigma(d)\kappa(d)^\top (\kappa(d)\Sigma(d)\kappa(d)^\top)^\dagger \kappa(d)\Sigma(d) \right) (1_d - e_{d1}). \quad (29)$$

⁷ Δ_d denotes the d -dimensional unit simplex.

The hat matrix $\Sigma(d)\kappa(d)^\top (\kappa(d)\Sigma(d)\kappa(d)^\top)^\dagger \kappa(d)\Sigma(d)$ has rank $d - 1$, thus $(1_d - e_{d1})$ cannot possibly lie in the nullspace of the projection operator, and we have a strictly positive error. We skip deriving the exact expressions as a function of d and note that the same analysis could be straightforwardly conducted constraining different coefficients to equal zero.

D Bounding the error and complexity of a general R -estimator

In this section, we present a method for computing nontrivially tight lower bounds on the error from a general $R \subseteq \{0, 1, 2, \dots, d - 1, d\}$ estimator. For any set of points $P \subseteq [0, 1]^d$, we have that

$$\begin{aligned} \|m - \max\|_\infty &\geq \max_{x \in P} |m(x) - \max(x)| \implies \\ \min_{m \in \mathcal{M}_d^s(R)} \|m - \max\|_\infty &\geq \min_{m \in \mathcal{M}_d^s(R)} \max_{x \in P} |m(x) - \max(x)|. \end{aligned}$$

Apply this with P equal to $d + 1$ corners of the unit cube containing zero, one, two, etc. ones:

$$\begin{aligned} \|m - \max\|_\infty &\geq \min_{m \in \mathcal{M}_d^s(R)} \max \{ |m((0 \ 0 \ \dots \ 0 \ 0))|, \\ &\quad |m((1 \ 0 \ \dots \ 0 \ 0)) - 1|, \\ &\quad |m((1 \ 1 \ \dots \ 0 \ 0)) - 1|, \\ &\quad \dots \\ &\quad |m((1 \ 1 \ \dots \ 1 \ 0)) - 1|, \\ &\quad |m((1 \ 1 \ \dots \ 1 \ 1)) - 1| \}. \end{aligned}$$

Finally, we write this as the convex optimization problem, in $2(d + 1)$ constraints, and $1 + |R|$ variables, using a standard trick for rewriting L_∞ optimization (see Boyd & Vandenberghe (2004)).

$$\begin{aligned} \min_{g, \beta_0, (\beta_r, r \in R)} g \text{ subject to } & \left| \beta_0 + \sum_{r \in R \setminus \{0\}} \beta_r S((0 \ 0 \ \dots \ 0 \ 0); r, d) \right| \leq g, \\ & \left| \beta_0 - \sum_{r \in R \setminus \{0\}} \beta_r S((1 \ 0 \ \dots \ 0 \ 0); r, d) - 1 \right| \leq g, \\ & \left| \beta_0 - \sum_{r \in R \setminus \{0\}} \beta_r S((1 \ 1 \ \dots \ 0 \ 0); r, d) - 1 \right| \leq g, \\ & \dots \\ & \left| \beta_0 - \sum_{r \in R \setminus \{0\}} \beta_r S((1 \ 1 \ \dots \ 1 \ 0); r, d) - 1 \right| \leq g, \\ & \left| \beta_0 - \sum_{r \in R \setminus \{0\}} \beta_r S((1 \ 1 \ \dots \ 1 \ 1); r, d) - 1 \right| \leq g. \end{aligned}$$

This computation scales well, essentially linear programs such as this can be simply solved on a desktop computer using standard software for thousands of variables and constraints.

E Implementing an R estimator as a purely feedforward network

In this section, we show how to cast a general R estimator as the forward pass of a purely feedforward network. This analysis is necessary to give a benchmark against which to compare stochastic gradient descent fitting. The reasoning is complex, so will skip some low-level details. The code, in idiomatic PyTorch and accompanied by extensive test cases, is available at xxx.com.

First, we describe a concept called the R -mapping, then we describe an algorithm for computing R -mappings, and then we show how to use an R -mapping to construct a purely feedforward network that is an R -estimator.

E.1 R -mapping definition and motivation

An R -mapping describes an R -estimator as a sequence of pairwise maxes. For $d \in \mathbb{N}$ and $r \in \{1, 2, \dots, d\}$ let $C(r, d) = \{\{1, 2, \dots, r\}, \{1, 3, \dots, r+1\}, \dots, \{d-r, \dots, d\}\}$ denote the set of size $\binom{d}{r}$ of all subsets of $\{1, 2, \dots, d\}$ of size r .

Definition 1. An R -mapping for $R \subseteq \{0, 1, 2, \dots, d\}$ is a sequence of sets t_1, t_2, \dots, t_s with $s \leq \lceil \log_2 d \rceil$ satisfying:

- for all $r \in R, r > 0$ there is some j such that $C(r, d) \subseteq t_j$, and
- $t_{j+1} \subseteq \{\tau_1 \cup \tau_2 : \tau_1, \tau_2 \in t_j\}$.

At a high level: Each element of an R -mapping corresponds to a set of indices into the input, and at the j th layer computes the max of the input over all indices in t_j . The first defining characteristic of an R -mapping ensures that the indices permit the evaluation of all required subpool max averages. And the second condition insures that a purely feedforward network can compute the maxes over the implied indices.

To simplify the subsequent discussion, we hereafter assume that $\{0, 1\} \subseteq R$ for all R . Since the first two terms are trivially uncomplicated – a constant bias, and the grand mean of the inputs are linear features – this assumption is without any loss of generality and could be easily relaxed.

E.2 Computing R -mappings

In this section, we show how to compute an R -estimator. We say that R is *adequate* if $\max(R) \leq 2 \times \max(R \cap [0, 2^{\lceil \log_2(\max(R)-1) \rceil}])$. If R is adequate then it is possible to form the greatest remaining term from pairwise maxes of terms that are a lower power of two – a condition necessary to enforce the second condition in 1. If R is not adequate, then it can be made adequate by appending an additional term.

Let $\tilde{R} = a(R)$, where $a : \{0, \dots, d\} \mapsto \{0, \dots, d\}$ is defined recursively as:

$$a(R) = \begin{cases} R & \text{if } R = \{0, 1\} \\ \{\max(R)\} \cup a(R \setminus \{\max(R)\}) & \text{if } R \text{ is adequate} \\ \{\max(R)\} \cup a(R \setminus \{\max(R)\}) \cup \{\lceil \max(R)/2 \rceil\} & \text{otherwise.} \end{cases} \quad (30)$$

The third case covers the situation where it would not be possible to compute an R -mapping out of terms in R , and so an additional term is appended. For example, $a(\{0, 1, 2, 5, 6\}) = \{0, 1, 2, 3, 5, 6\}$: 3 has been appended since it is impossible to compute the maxes of five and six terms using only pairwise maxes ($r = 2$) of pairs of variables. $a(R)$ essentially reduces its argument by one term with each recursive call, and thus it is fast and straightforward to evaluate.

By construction, every truncation of \tilde{R} is adequate, thus for every $\tilde{r} \in \tilde{R}$ with $\tilde{r} > 1$ there exists an $\tilde{r}' \in \tilde{R}$ with $\tilde{r}' \geq \tilde{r}/2$. This means that $C(1, d), C(2, d), \cup_{r \in R \cap [3, 4]} C(r, d), \dots, \cup_{r \in R \cap [d/2, d]} C(r, d)$, is a R -mapping, however if $R \subsetneq \tilde{R}$, then there will be smaller R -mappings since it is possible to skip the computation of some terms in $C(r, d)$. For example, continuing our example above, there are 56 subsets of size 3 of $d = 8$ values, but 36 terms of length 3 can be combined to form all subsets of size 5 and 6.

Algorithm 1 Computation of an R -mapping

Input: $R \subseteq \{0, 1, \dots, d\}$
Output: R -mapping suitable for evaluating f_R^*
 $\tilde{R} \leftarrow a(R)$ // Augment R with needed terms
 $s = \lceil \log_2(\max R) \rceil$
 $R_j \leftarrow R \cap \{i : i \in \mathbb{N}, \lceil \log_2 i \rceil = j\}$ for $j = 1, 2, \dots, s$ // group R by power of 2
 $\tilde{R}_j \leftarrow \tilde{R} \cap \{i : i \in \mathbb{N}, \lceil \log_2 i \rceil = j\}$ for $j = 1, 2, \dots, s$ // group \tilde{R} by power of 2
for $i = 0, 1, \dots, s - 1$ **do**
 $j \leftarrow s - i - 1$ // backward index
 if $i = 0$ **then**
 $N_j \leftarrow \emptyset$
 else
 $N_j \leftarrow t_{j+1}$ // terms needed for subsequent layer
 end if
 $X_j \leftarrow \{C(k, r, d) : k = 1, \dots, \binom{d}{r}, r \in R_j\}$ terms needed for this layer
 $Y_j \leftarrow X_j \cup N_j$
 $k_j \leftarrow \max(\tilde{R}_j)$ // tuple width in this mapping term
 $t_j \leftarrow \text{FLATSPLIT}_{k_j}(Y_j)$
end for
Return t_1, t_2, \dots, t_s

For a vector $A \in \mathbb{R}^\ell$ with $\ell > k$, let $\text{SPLIT}_k : \mathbb{R}^\ell \rightarrow \mathbb{R}^k \times \mathbb{R}^k$ be the function that splits its argument into the first and last k elements: $\text{SPLIT}_k(A) = (A_{(1)}, \dots, A_{(k)}), (A_{(\ell-k)}, \dots, A_{(\ell)})$. And let FLATSPLIT_k be the set-valued function that applies SPLIT to each of its inputs and collects the outputs into a single set

$$\begin{aligned}
& \text{FLATSPLIT}_k(\{A^1, A^2, \dots, A^n\}) \\
&= \{(A^1_{(1)}, \dots, A^1_{(k)}), (A^2_{(\ell-k)}, \dots, A^2_{(\ell)}), \dots, (A^n_{(1)}, \dots, A^n_{(k)}), (A^n_{(\ell-k)}, \dots, A^n_{(\ell)})\}.
\end{aligned}$$

Algorithm 1 gives one approach to compute an R -mapping that improves upon naively computing all subpool maxes of order $\tilde{r} \in \tilde{R}$: The idea is to include only those terms in \tilde{R} only minimally in order to support the computation of subsequent terms. Despite being awkward to state this algorithm runs quickly, requiring $O(\sum_{r \in R} \binom{d}{r})$ space and time.

E.3 R -mappings to neural network R -estimators

An R -mapping, as introduced in subsection E.2, is a sequence of sets that is defined so that each element of a constituent set is associated with a pair of elements in the previous set. Thus, it is well-suited to compute pairwise maxes via a simple linear-ReLU-linear block as shown in Equation 2. Computing the average of all subpooled values is of course a linear operation.

An important book-keeping challenge with this approach is to enable the network to convey the average of low-order subpool maxes through the network. To do this, we append to the network a “memory” – additional neurons which carry forward values computed earlier in the network via identity mappings (propagated through ReLUs via $x = \text{ReLU}(+x) - \text{ReLU}(-x)$) through until the end – a layer of width $|R|$, containing all needed subpool max averages. Finally, then, these values are aggregated according to β .

Our code is written in three stages: (1) compute a base network consisting of the linear layers implied by Equation 2, then (2) append the subpool averages and their attendant memory neurons, and finally (3) aggregate the penultimate layer value with the coefficients. One helpful trick to developing this logic is to leave each step above as consecutive linear layers, then once everything is complete, to fuse them all together.

This approach perhaps does not result in the smallest possible purely feedforward network that could implement an R -estimator but it is relatively simple to code, fast to run, and the architecture is very descriptive

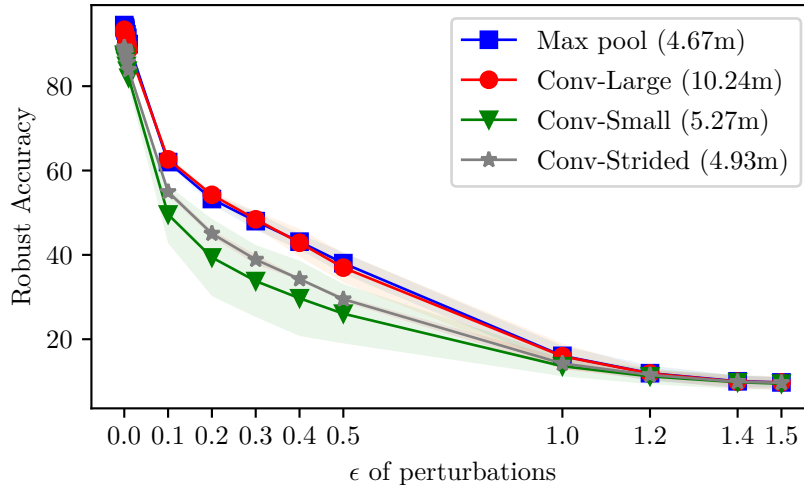


Figure 5: Effect of perturbations with $L_\infty \leq \epsilon$ on the accuracy of the ResNet model variants on the CIFAR10 dataset. Mean accuracy and ± 1.96 standard deviation over the course of three runs are depicted. The legend indicates the number of parameters of each model in parentheses. The models omitting max pool pay a price in either (robust) accuracy or model complexity (in terms of parameters).

of the logic the network implements. It seems unlikely that a large improvement on this general scheme is possible, though we do not attempt to prove this speculation.

F Analysis

We have strictly looked at the accuracy implications of approximating the max function in isolation. In this short and speculative section, we examine briefly the practical takeaways for adversarial robustness.

F.1 Adversarial robustness

Our hypothesis is that max pooling can be more robust than strided convolution and ReLU nonlinearity, since genuine max pooling admits only a single direction along which features can change – the max. ReLU, by contrast, can be moved with only low correlation changes, and a random perturbation will in general change the output.

Our experiments corroborate this intuition; omitting max pooling results in lower robust accuracy in several different model classes, ranging from simple Convolutional Neural Networks to ResNets. Our experimental approach is to adversarially attack models with and without max pooling. We use the Fast Gradient Sign Method by Goodfellow et al. (2015). Specifically, starting from a model incorporating max pool layers, we replace them with strided convolution + ReLU. We examine four different models and report the robust accuracy in Figure 5 on the CIFAR10 dataset (Krizhevsky (2009)). The Max pool model is from Page (2018) and includes four max pool layers. The baselines make the following modifications: Conv-Small and Conv-Large replace the max pool layers with a convolutional one, with kernel size one and three, respectively. The Conv-Strided model uses a strided convolution in lieu of the max pool layer and the convolution that precedes it. More details on the models as well as experiments with LeNet architecture can be found in Appendix G of the appendix.

We performed the computation on an internal computation cluster of Tesla V100-SXM2-32GB GPUs. All experiments presented here can be done in less than seven Tesla V100 days. All software and data are standard academic tools and present no licensing issues.

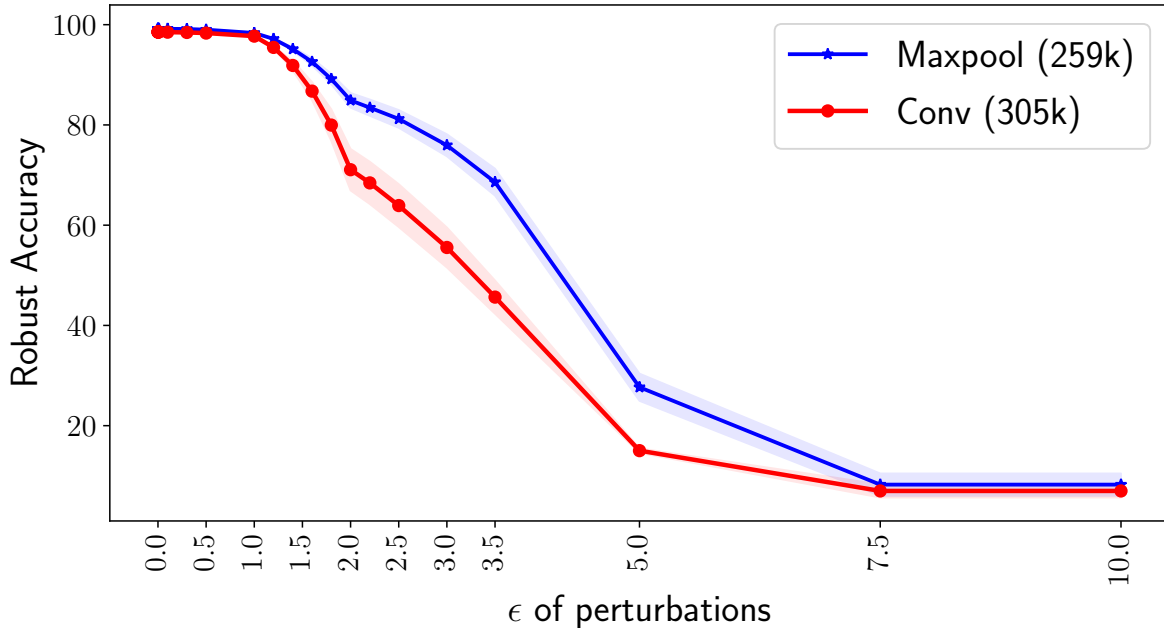


Figure 6: Effect of perturbations with $\ell_\infty \leq \epsilon$ on the accuracy of the LeNet model variants on the MNIST dataset.

The experimental results showcase a tradeoff between model complexity (in terms of number of parameters) and robust accuracy. The Max pool model is more adversarially robust than the baselines Conv-Small and Conv-Strided. This effect is further highlighted for larger perturbations ϵ . An exception to this trend lies in the Conv-Large model which is able to match the robust accuracy of the Max pool model, but requires more than twice as many parameters.

G Experimental Details

In this section, we present in greater detail our experimental framework. The goal is to show the greater adversarial robustness of models *with* max pool layers than those *without*. Specifically, Convolutional Neural Nets (CNNs) and Residual Networks (ResNets) are trained on the MNIST and CIFAR10 data sets. Then, an adversarial attack is performed for various levels of perturbations, denoted by ϵ , and the robust accuracy (mean and standard deviation) of the various models is reported over three different random seeds. We present results for the Fast Gradient Sign Method Goodfellow et al. (2015) attack.

In each experiment, we create a model incorporating max pool layer(s). Then, the network is modified by replacing each max pool layer with a trainable variant, ensuring that the output of the original layer and the modified one have the same shape.

The legend of each figure presents the name of the model variant as well as the number of trainable parameters in parentheses. The width of the lines is proportional to the number of parameters in the model. The max pool model variant is always depicted in blue.

G.1 Technology stack

The experiments are developed in PyTorch Paszke et al. (2017) and PyTorch Lightning Falcon (2019) with the help of the foolbox Rauber et al. (2017) library for the adversarial attack. We use a ResNet variant in our experiments Page (2018). We use the publicly available datasets MNIST (LeCun et al., 2010) and CIFAR10 Krizhevsky (2009).

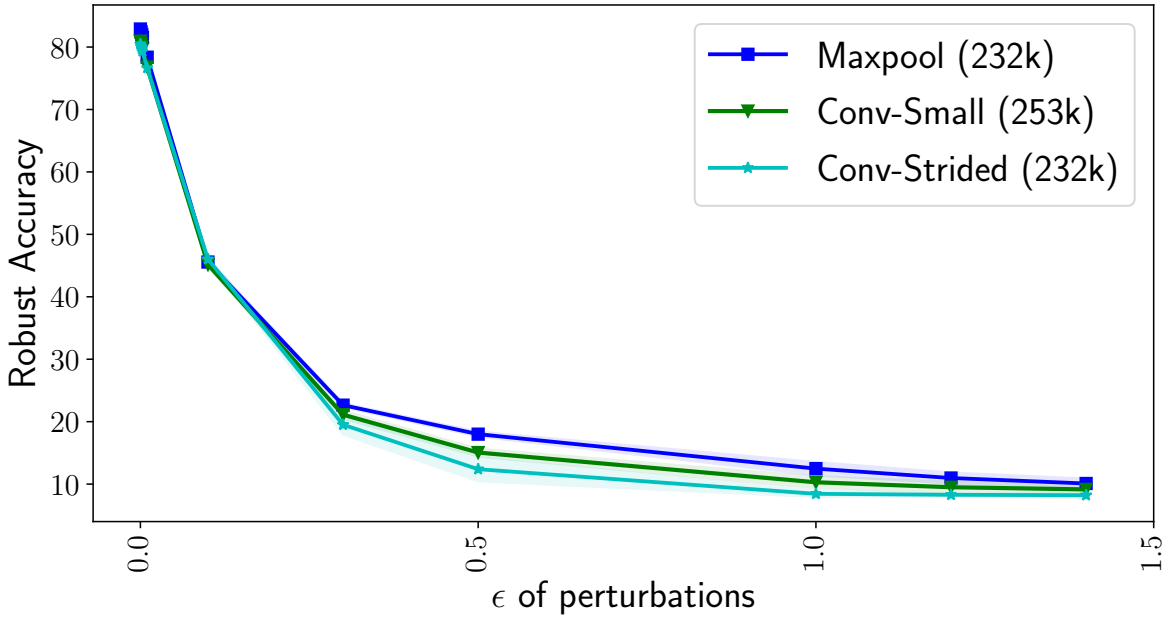


Figure 7: Effect of perturbations with $\ell_\infty \leq \epsilon$ on the accuracy of the LeNet model variants on the CIFAR10 dataset.

G.2 Experimental configurations

LeNet experiment on MNIST We train two convolutional neural networks (CNNs) on the digit classification dataset MNIST. The results are shown in Figure 6.

The first model, in blue, has 259 106 trainable parameters and consists of two convolutional layers, with 32 and 64 channels. Their kernel size is equal to five. Both layers are succeeded by a two-dimensional max pool with kernel size, stride and padding equal to three, two and one, respectively. The network is completed with two fully-connected layers of 1024 and 200 neurons, leading to the output of ten logits. The second model, in red, has 305 282 trainable parameters and has the same structure as the previous model. However, the max pool layers are replaced by a convolutional layer with the same number of input channels as the output of the preceding convolutional layer, hence the increase in parameters. Both models are trained for ten epochs of Stochastic Gradient Descent with learning rate and momentum equal to 0.01 and 0.9, respectively.

LeNet experiment on CIFAR10 A similar experiment is performed on the more challenging CIFAR10 dataset. The results are shown in Figure 7.

The max pool model, in blue, has 232 162 parameters and consists of three convolutional layers of 32, 64, 128 channels, respectively. Again, each of these layers is succeeded by max pool module identical to the MNIST experiment. The convolutional variant, depicted in green, has 253 890 parameters has a similar modification as before, i.e. the max pool layer is replaced by a convolutional one with kernel size, stride and padding identical to the corresponding max pool layer. Finally, the strided variant, in gray, has the same number of parameters as the original max pool model. In this case, we replace the block of convolution and max pool with a strided convolution of stride equal to two. Hence, this modification does not incur an increase in number of parameters, while maintaining the same output shape at all intermediate steps. The models are trained for 100 epochs of SGD with learning rate and momentum equal to 0.01 and 0.9 respectively. The learning rate is decayed by a parameter $\gamma = 10$ in epochs 50, 70 and 90. The batch size is 128.

ResNet experiment on CIFAR10 We use the ResNet variant proposed in Page (2018). This model consists of a preparatory whitening layer, three residual blocks and a classifier layer. First, the preparatory layer has two convolutions and Ghost Batch Normalization Hoffer et al. (2017). The three residual blocks

have identical structure, with the exception of the number of channels in the convolutional layer; the channels are doubled with each layer, from 64 to 128 to 256. Each of these layers consists of two blocks: the first one has a convolutional layer, a max pool layer with kernel size and stride equal to two and Ghost Batch Norm, while the second block employs a residual connection with similar structure as the previous block modulo the max pool. Finally, the classifier layer is comprised of a max pooling layer of kernel size and stride equal to four and a fully connected layer resulting in ten outputs.

Overall, the Maxpool model, in blue, has four max pooling layers and 4 666 265 parameters. The large convolutional model, depicted in red, has 10 238 233 parameters and replaces the Maxpool layers by convolutional ones with the same kernel size, stride and padding. The small convolution variant, depicted in green, has 5 273 881 parameters and the kernel size is set to one for all max pooling replacements. Finally, the strided model, depicted in gray, has 4 928 921 parameters and, for each residual layer, the pair of convolution and max pooling is replaced by a strided convolution, while the max pooling layer preceding the fully connected one is replaced with a convolutional layer of kernel size equal to one. The stride remains equal to four, as in the corresponding max pooling layer.

The models are trained for 50 epochs using float 16 precision. The learning rate follows a piecewise linear schedule; starting at zero the learning rate linearly increases to 0.4 until the fifth epoch and then linearly decays to zero until the final epoch.

G.3 Analysis

Our objective lies in showing the superior adversarial robustness of models incorporating max pooling. In each experiment, we use a max pooling model, drawn from widely used neural networks such as LeNet and ResNet. We modify the original model to produce comparisons. Specifically, the modifications simply replace the max pooling layer with a convolutional layer, or the pair of convolutional layer and max pooling (which traditionally come succession) with a strided convolution. In both cases, the new layer produces outputs of the same shape as the original layer, lending itself to an one-to-one comparison in terms of performance on (robust) accuracy. It is important to note that the first modification results in an increase in the number of trainable parameters. Subsequently, we perform an adversarial attack, FGSM in our case, to illuminate the adversarial robustness properties of each model. A common theme of all experiments is that the exclusion of the max pooling layer results in a tradeoff between (robust) accuracy and model complexity.

First, in the MNIST experiment both variants reach similar levels of performance on the clean accuracy ($\epsilon = 0$); the max pool variant achieves 99.19 ± 0.06 while the convolutional model 98.54 ± 0.14 . This is not surprising given the low difficulty of the dataset. Nevertheless, the model *with* max pooling is characterized by strictly higher adversarial robustness, since the difference in performance heightens for larger ϵ . In the CIFAR10 experiment, the observations are similar in nature; replacing max pooling with a trainable layer renders the model more susceptible to adversarial perturbations. However, the modified models do not exhibit the same level of clean accuracy, despite the increase in model complexity. Specifically, the mean clean accuracies (over 3 runs with different random seeds) of the max pool, small convolutional and strided convolution models are $82.89 \pm 0.08\%$, $80.94 \pm 0.26\%$ and $80.45 \pm 0.66\%$, respectively. It is important to note that the LeNet architecture does not achieve state-of-the-art results on any of the model variants presented. However, it serves as a direct comparison with the previous experiment. Finally, the ResNet experiment perhaps illuminates the tradeoff more clearly. Figure 5 (see main text) presents a dichotomy due to the exclusion of the max pool layer; the practitioner should choose between model complexity (measured in number of trainable parameters and, by extension, training and inference times) and (robust) accuracy. The large convolution model achieves a clean accuracy of $93.37 \pm 0.14\%$ compared to $94.49 \pm 0.20\%$ of the original model and is able to match its robust accuracy for different ϵ , while using more than double the parameters. The other two variants, however, have lowest clean accuracies ($89.22 \pm 0.09\%$ for the strided model and $87.46 \pm 0.99\%$ for the small convolutional) and present a faster deterioration in adversarial robustness.

G.4 Detailed Results

For completeness, we present the experimental results in tabular form. The experiments were repeated three times (with different random seeds) and the mean \pm standard deviation is reported.

Table 2: Detailed results on MNIST.

ϵ	Maxpool	Conv
0.000	99.19 ± 0.06	98.54 ± 0.14
0.001	99.19 ± 0.06	98.54 ± 0.14
0.002	99.19 ± 0.06	98.54 ± 0.14
0.003	99.19 ± 0.06	98.54 ± 0.14
0.010	99.19 ± 0.06	98.54 ± 0.14
0.100	99.18 ± 0.05	98.50 ± 0.16
0.300	99.14 ± 0.04	98.45 ± 0.16
0.500	99.00 ± 0.07	98.33 ± 0.11
1.000	98.34 ± 0.09	97.72 ± 0.07
1.200	97.16 ± 0.08	95.47 ± 0.15
1.400	95.14 ± 0.38	91.86 ± 0.97
1.600	92.59 ± 0.72	86.74 ± 1.81
1.800	89.16 ± 1.05	79.97 ± 3.15
2.000	84.90 ± 1.46	71.07 ± 4.14
2.200	83.44 ± 1.62	68.43 ± 4.22
2.500	81.20 ± 1.77	63.91 ± 4.27
3.000	75.95 ± 2.20	55.55 ± 3.99
3.500	68.58 ± 2.69	45.65 ± 3.32
5.000	27.67 ± 2.71	15.02 ± 0.46
7.500	8.24 ± 2.22	6.98 ± 1.31
10.000	8.24 ± 2.22	6.98 ± 1.31

Table 3: Detailed results on CIFAR10 with LeNet.

ϵ	Maxpool	Conv-small	Strided
0.000	82.89 ± 0.09	80.94 ± 0.27	80.45 ± 0.67
0.001	82.42 ± 0.11	80.51 ± 0.19	80.08 ± 0.60
0.002	82.00 ± 0.08	80.09 ± 0.20	79.70 ± 0.58
0.003	81.56 ± 0.13	79.68 ± 0.21	79.31 ± 0.66
0.010	78.37 ± 0.03	76.80 ± 0.17	76.64 ± 0.30
0.100	45.56 ± 0.54	45.16 ± 0.41	46.02 ± 0.30
0.300	22.64 ± 0.32	21.12 ± 0.68	19.51 ± 1.61
0.500	18.00 ± 0.42	15.05 ± 0.65	12.39 ± 1.96
1.000	12.48 ± 1.08	10.29 ± 0.96	8.44 ± 0.52
1.200	10.97 ± 0.90	9.50 ± 0.90	8.28 ± 0.21
1.400	10.10 ± 0.81	9.15 ± 0.75	8.22 ± 0.46