Deep learning

6.1. Benefits of depth

François Fleuret

https://fleuret.org/dlc/

Jan 1, 2021

idiap RESEARCH INSTITUTE

UNIVERSITÉ DE GENÈVE

EPFL

Using deeper architectures has been key in improving performance in many applications. For instance image classification:

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

(He et al., 2015)

Using deeper architectures has been key in improving performance in many applications. For instance image classification:

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

(He et al., 2015)

"Notably, we did not depart from the classical ConvNet architecture of LeCun et al. (1989), but improved it by substantially increasing the depth."

(Simonyan and Zisserman, 2014)

A theoretical analysis provides an intuition of how a network's output "irregularity" grows:

- linearly with its width and
- exponentially with its depth.

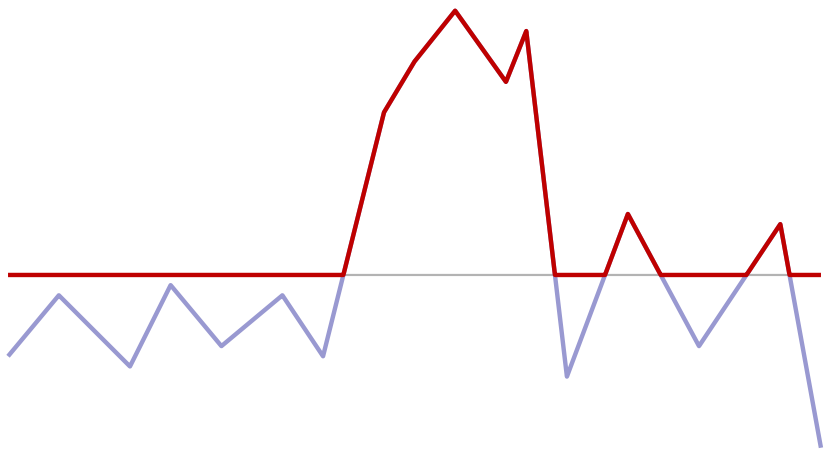
Let $\mathscr{F}$ be the set of piece-wise linear mappings on $[0,1]$, and $\forall f \in \mathscr{F}$, let $\kappa(f)$ be the minimum number of linear pieces in f.

Let $\mathscr{F}$ be the set of piece-wise linear mappings on $[0,1]$, and $\forall f \in \mathscr{F}$, let $\kappa(f)$ be the minimum number of linear pieces in f.

Let σ be the ReLU function

$$\begin{aligned}\sigma : \mathbb{R} &\to \mathbb{R}\\ x &\mapsto \max(0, x).\end{aligned}$$

Let $\mathscr{F}$ be the set of piece-wise linear mappings on $[0,1]$, and $\forall f \in \mathscr{F}$, let $\kappa(f)$ be the minimum number of linear pieces in f.



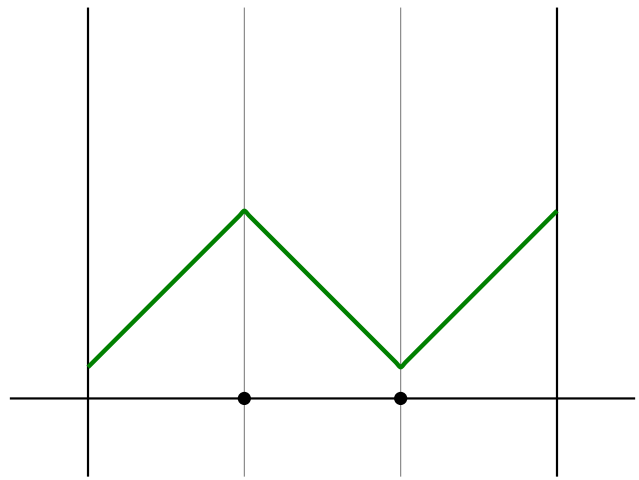
Let σ be the ReLU function

$$\begin{aligned}\sigma : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto \max(0, x).\end{aligned}$$

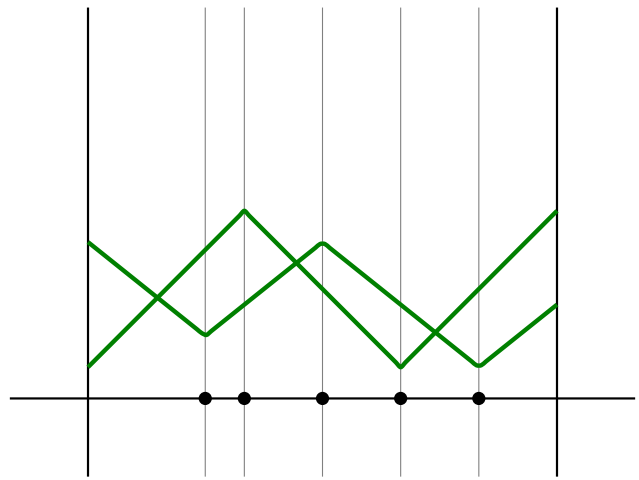
If we compose σ and $f \in \mathscr{F}$, any linear piece that does not cross 0 remains a single piece or disappears, and one that does cross 0 breaks into two, hence

$$\forall f \in \mathscr{F},\ \kappa(\sigma(f)) \leq 2\kappa(f)\,.$$

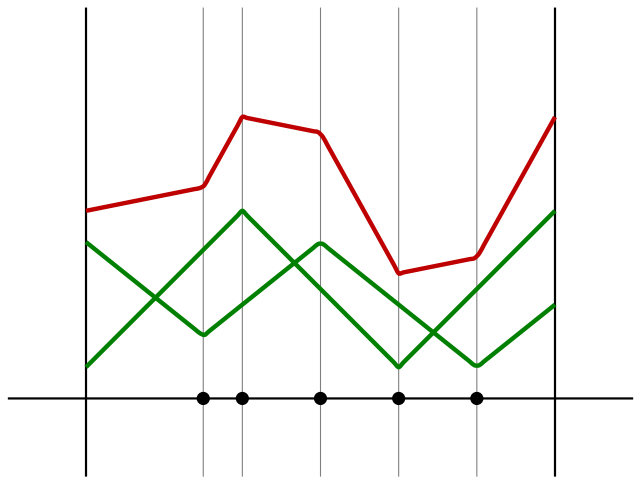
Also, when summing functions, a change of slop in the sum happens only if there was a change of slop in one of the operands.



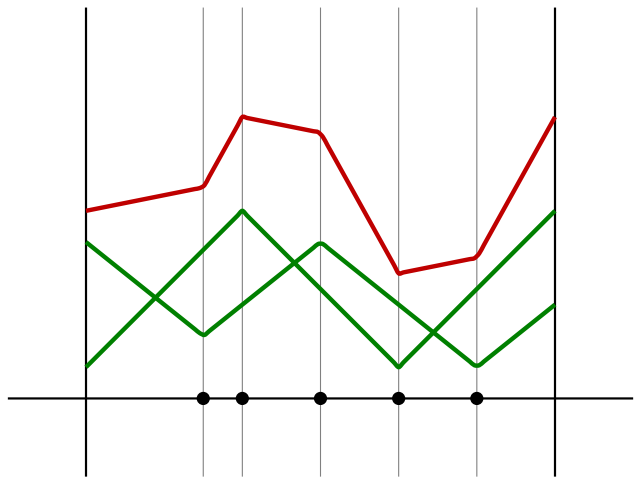
Also, when summing functions, a change of slop in the sum happens only if there was a change of slop in one of the operands.



Also, when summing functions, a change of slop in the sum happens only if there was a change of slop in one of the operands.



Also, when summing functions, a change of slop in the sum happens only if there was a change of slop in one of the operands.



Hence

$$\forall f_n \in \mathscr{F}, n = 1, \ldots, N, \ \kappa\left(\sum_n f_n\right) \leq \sum_n \kappa(f_n)\,.$$

Consider a MLP with ReLU, a single input unit, and a single output unit.

$$x_1^0 = x,$$

$$\forall d = 1, \ldots, D, \forall i, \quad \left\{ \begin{array}{ll} s_i^d & = \sum_{j=1}^{W^{(d-1)}} w_{i,j}^d x_j^{d-1} + b_i^d \\ x_i^d & = \sigma(s_i^d) \end{array} \right.$$

$$y = x_1^D.$$

Consider a MLP with ReLU, a single input unit, and a single output unit.

$$x_1^0 = x,$$

$$\forall d = 1, \ldots, D, \forall i, \quad \begin{cases} s_i^d & = \sum_{j=1}^{W^{(d-1)}} w_{i,j}^d x_j^{d-1} + b_i^d \\ x_i^d & = \sigma(s_i^d) \end{cases}$$

$$y = x_1^D.$$

All the s_i^ds and x_i^ds are piece-wise linear functions of x

Consider a MLP with ReLU, a single input unit, and a single output unit.

$$x_1^0 = x,$$

$$\forall d = 1, \dots, D, \forall i, \quad \begin{cases} s_i^d & = \sum_{j=1}^{W^{(d-1)}} w_{i,j}^d x_j^{d-1} + b_i^d \\ x_i^d & = \sigma(s_i^d) \end{cases}$$

$$y = x_1^D.$$

All the s_i^ds and x_i^ds are piece-wise linear functions of x with $\forall i, \kappa\left(s_i^1\right) = 1$, and

$$\forall d, i, \kappa\left(x_i^d\right) = \kappa\left(\sigma(s_i^d)\right) \leq 2\kappa\left(s_i^d\right) \leq 2 \sum_{j=1}^{W^{(d-1)}} \kappa\left(x_j^{d-1}\right)$$

from which

$$\forall d, \max_i \kappa\left(x_i^d\right) \leq 2W^{(d-1)} \max_j \kappa\left(x_j^{d-1}\right)$$

Consider a MLP with ReLU, a single input unit, and a single output unit.

$$x_1^0 = x,$$

$$\forall d = 1, \dots, D, \forall i, \quad \begin{cases} s_i^d & = \sum_{j=1}^{W^{(d-1)}} w_{i,j}^d x_j^{d-1} + b_i^d \\ x_i^d & = \sigma(s_i^d) \end{cases}$$

$$y = x_1^D.$$

All the s_i^ds and x_i^ds are piece-wise linear functions of x with $\forall i, \kappa\left(s_i^1\right) = 1$, and

$$\forall d, i, \kappa\left(x_i^d\right) = \kappa\left(\sigma(s_i^d)\right) \leq 2\kappa\left(s_i^d\right) \leq 2 \sum_{j=1}^{W^{(d-1)}} \kappa\left(x_j^{d-1}\right)$$

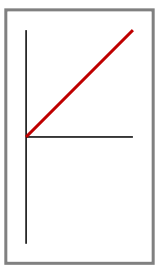
from which

$$\forall d, \max_i \kappa\left(x_i^d\right) \leq 2W^{(d-1)} \max_j \kappa\left(x_j^{d-1}\right)$$

and we get the following bound for any ReLU MLP

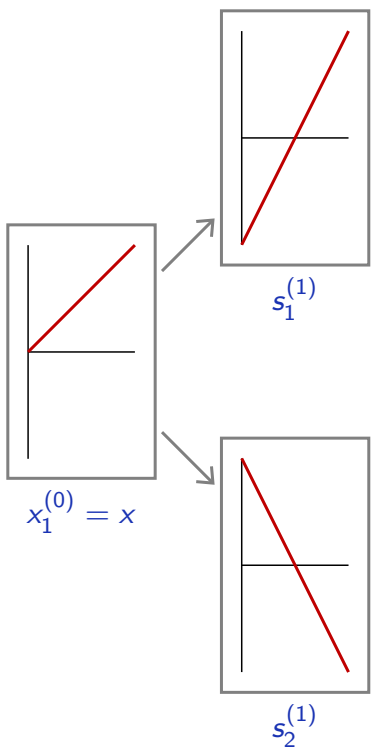
$$\kappa(y) \leq 2^D \prod_{d=1}^{D} W^{(d)}.$$

Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:

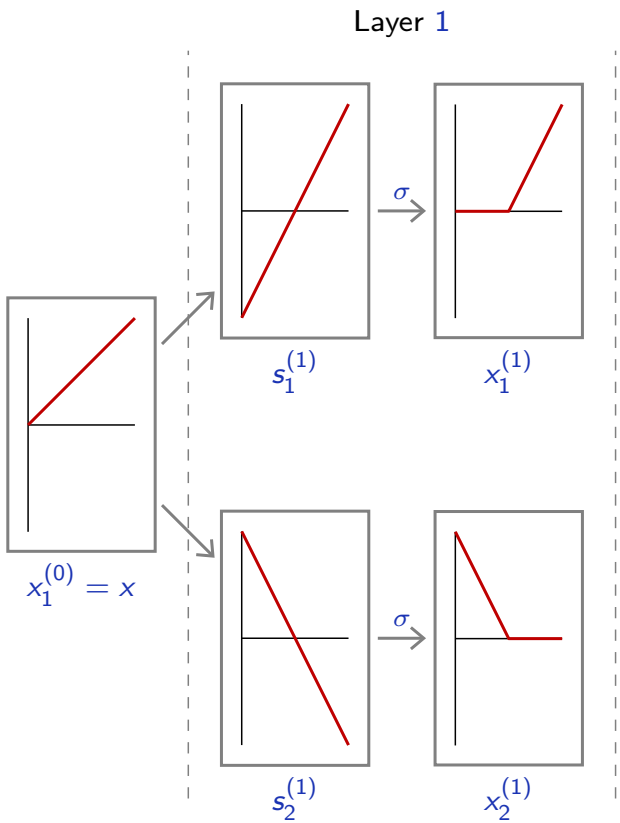


$x_1^{(0)} = x$

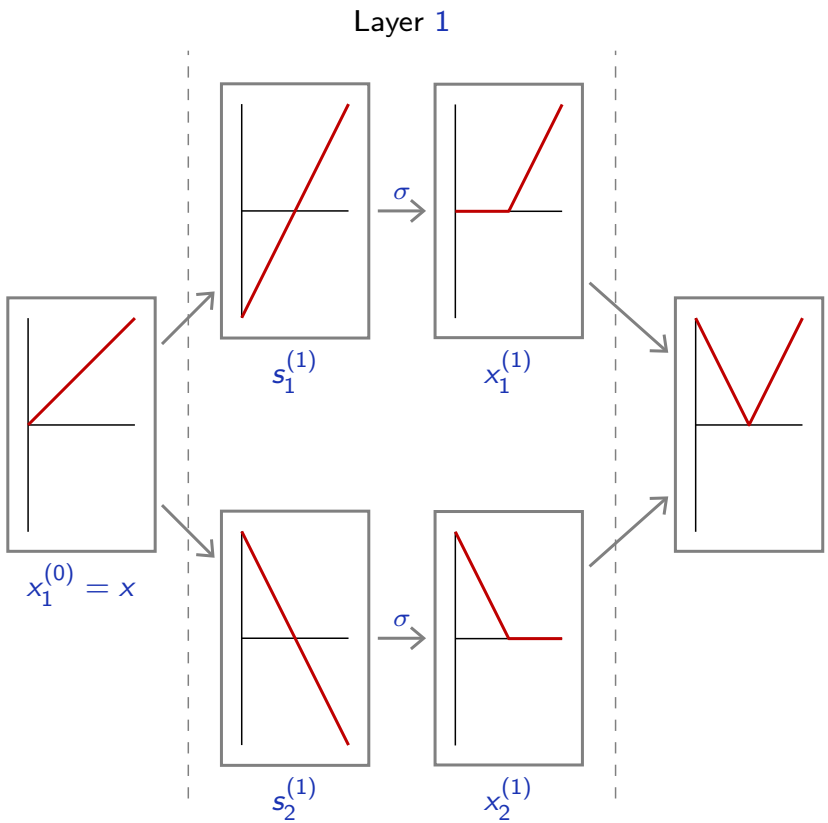
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



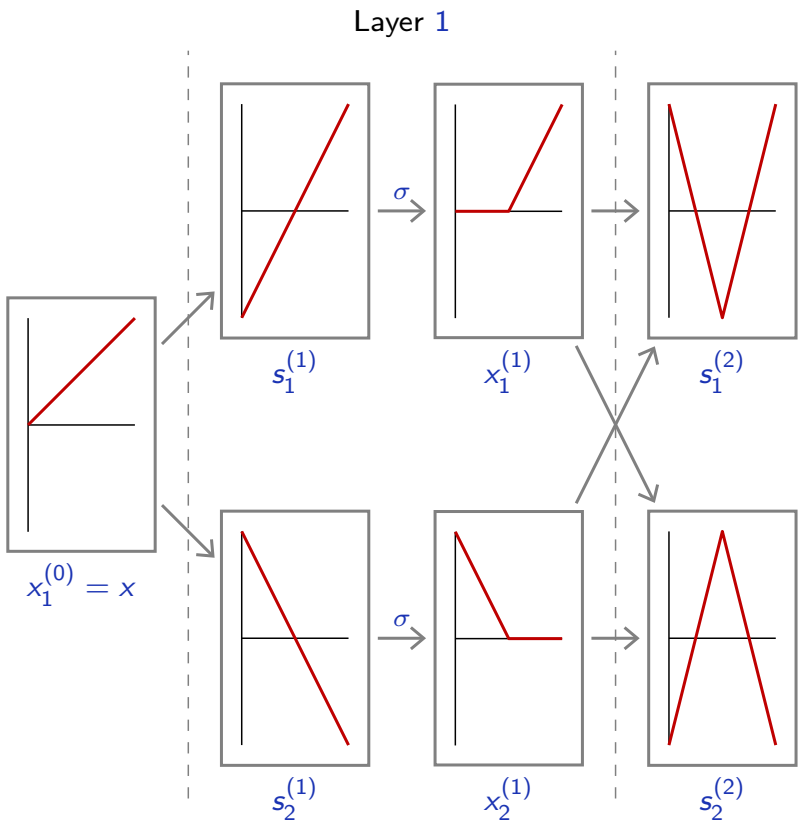
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



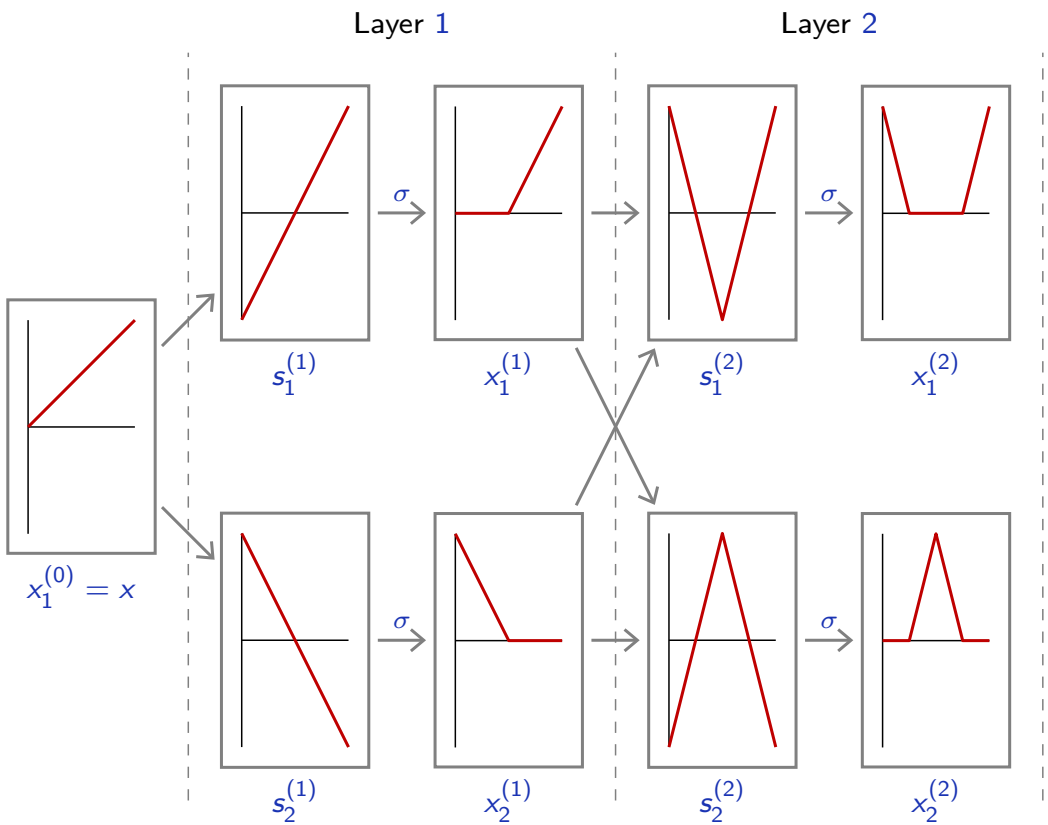
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



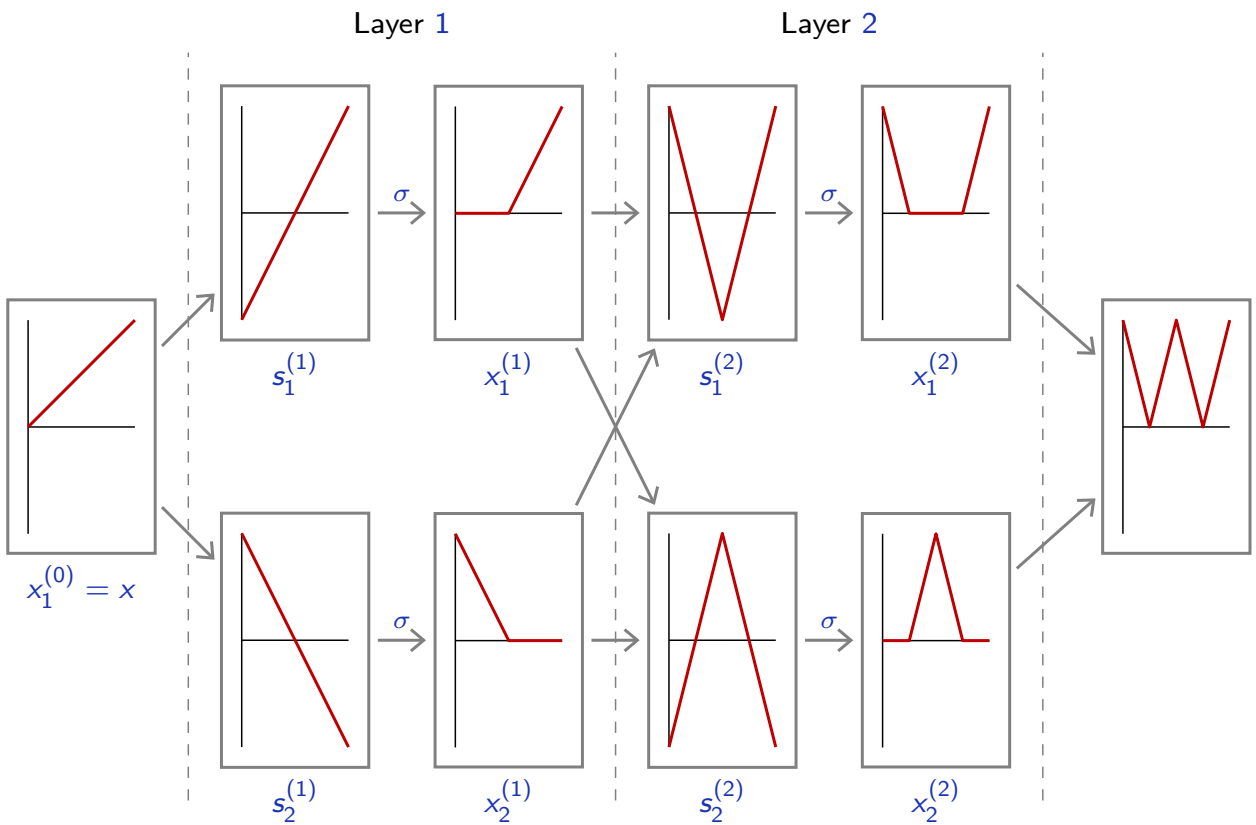
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



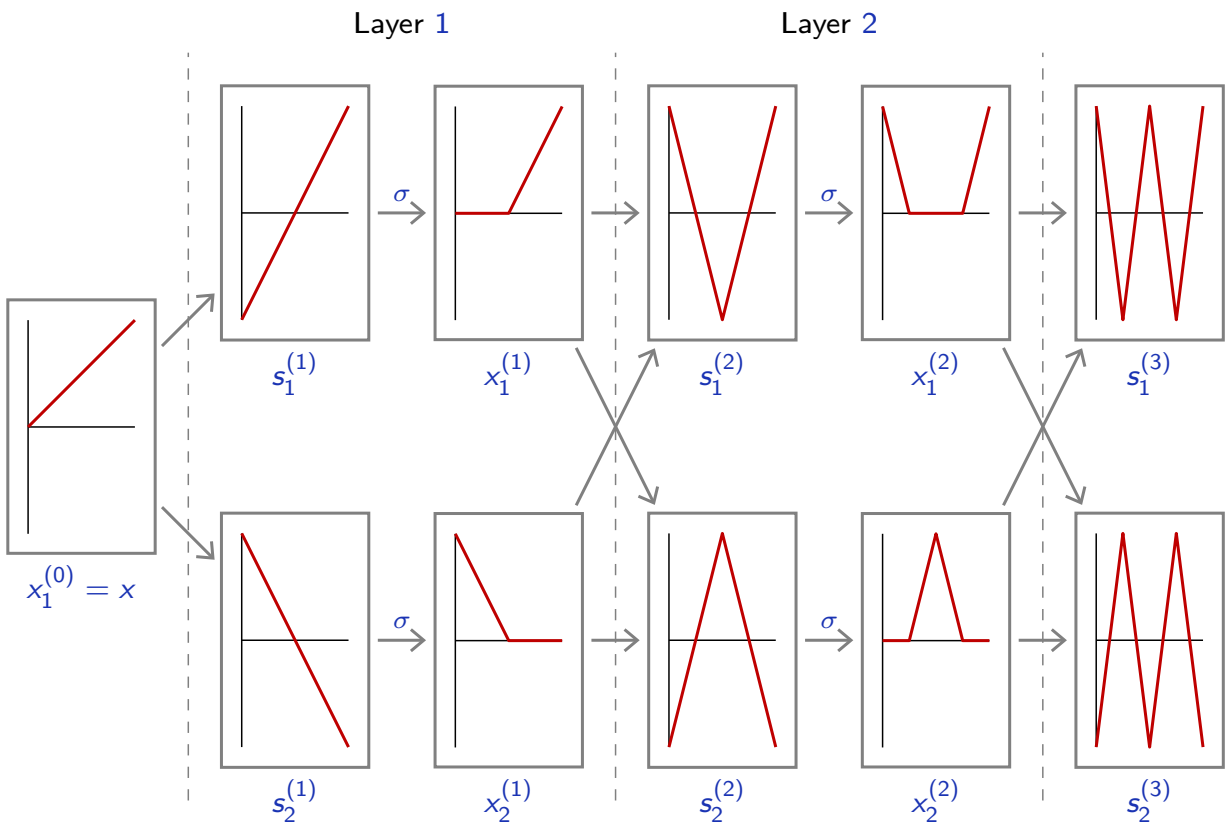
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



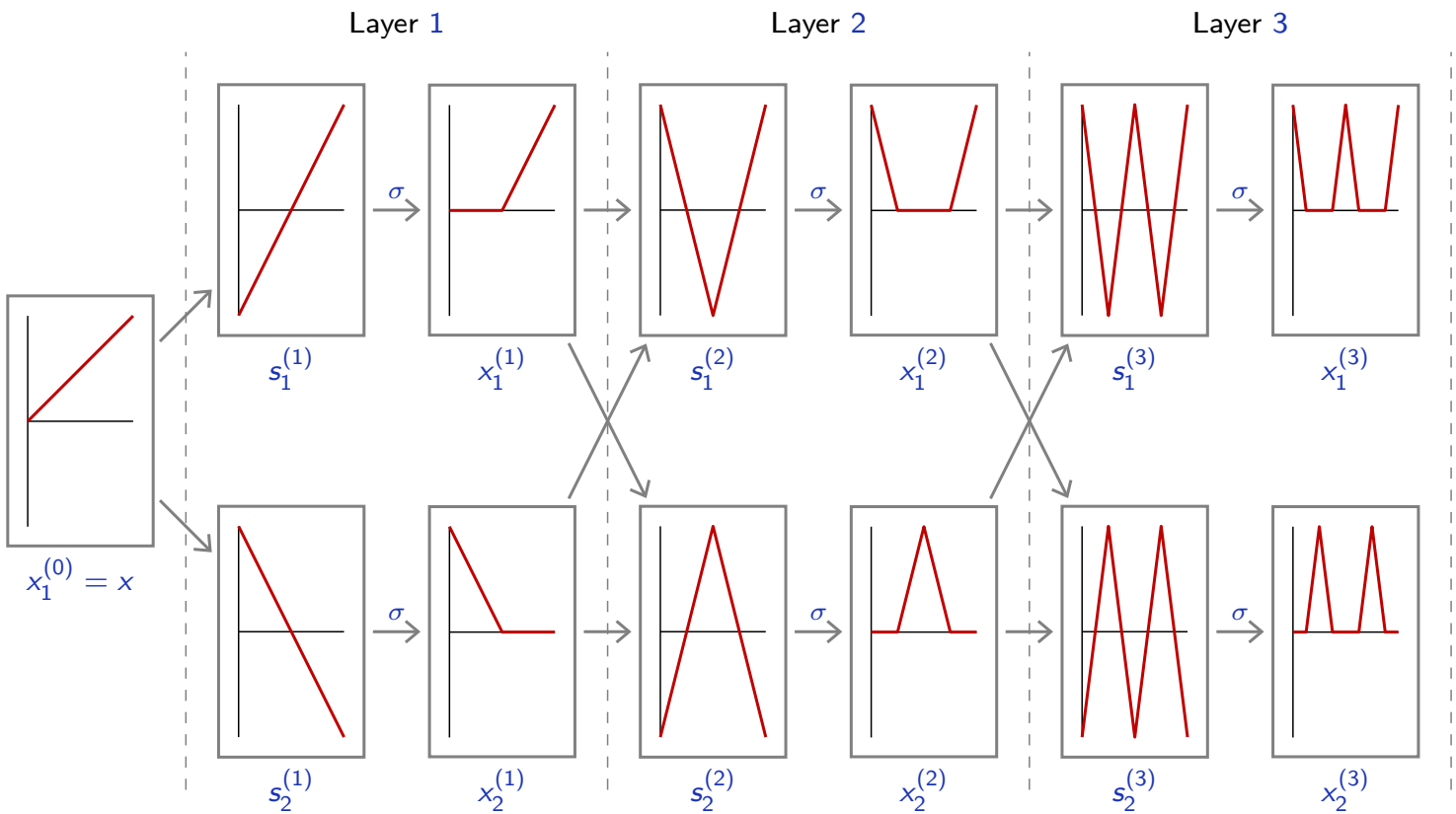
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



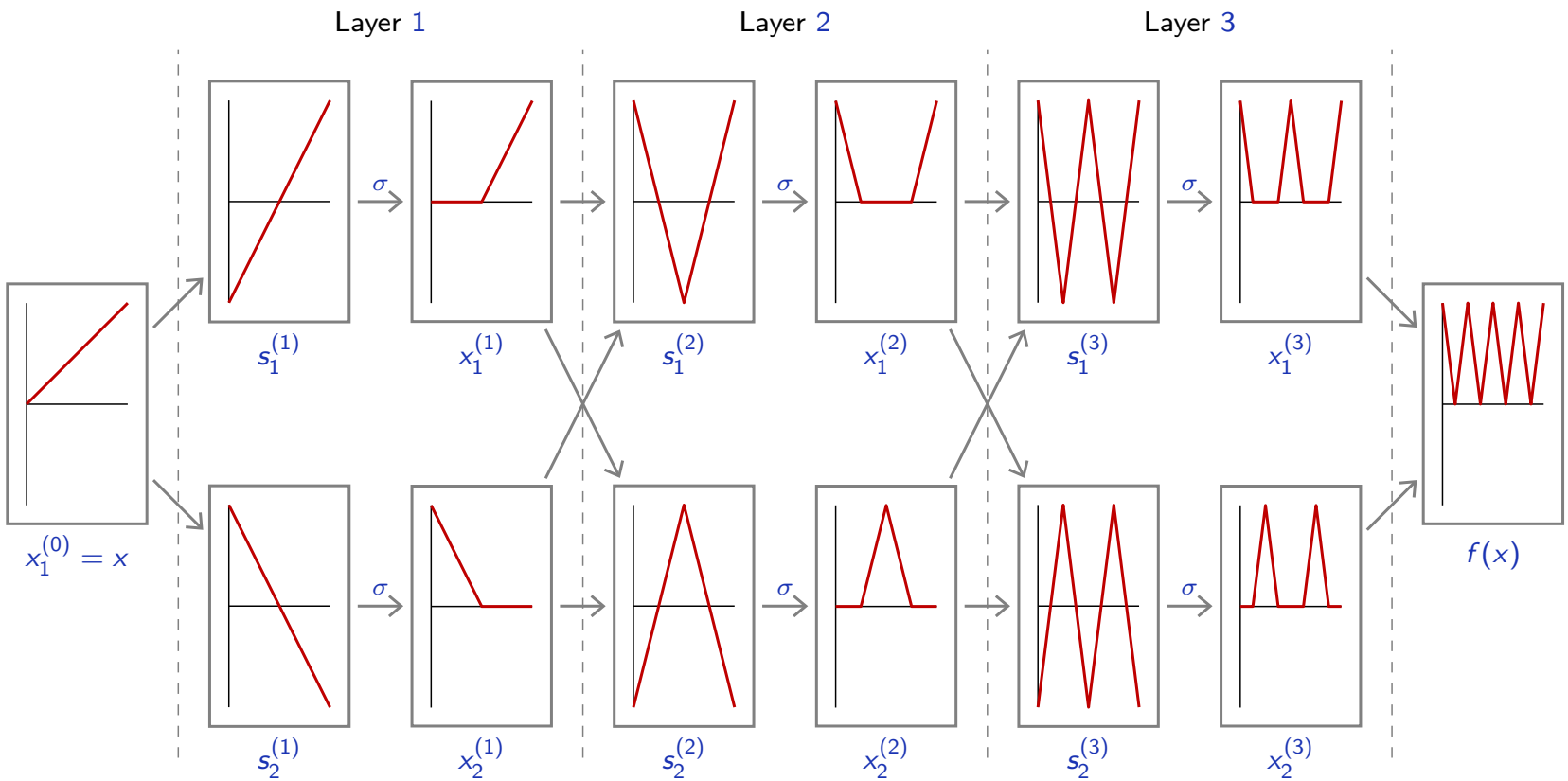
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



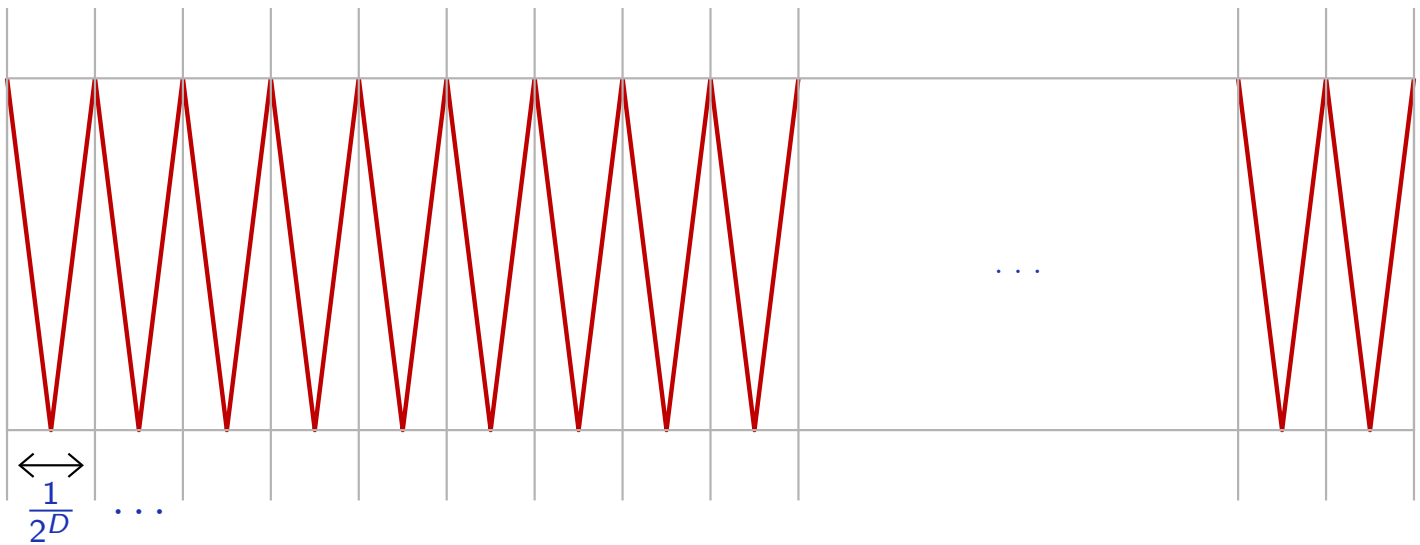
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:

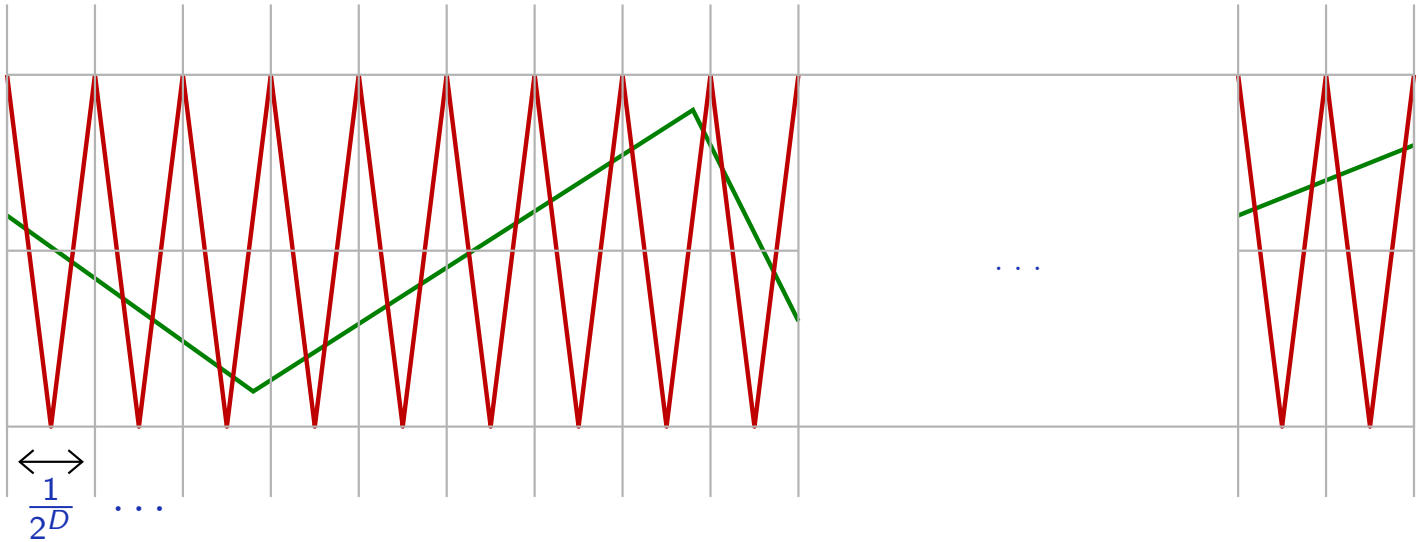


Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:

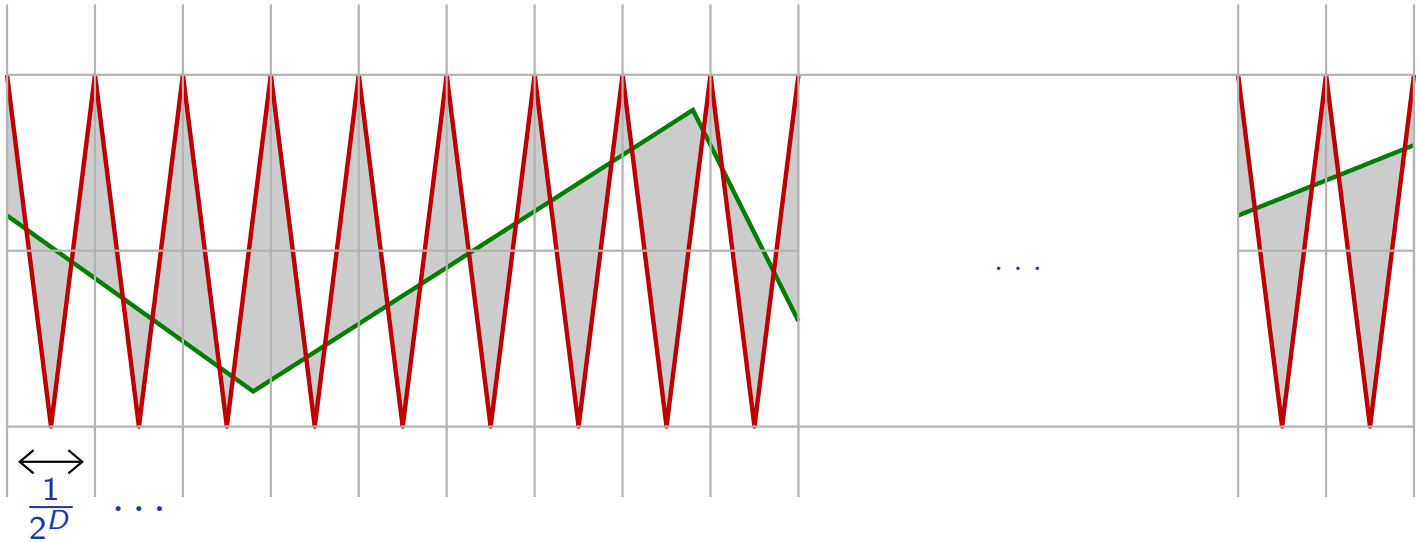


So for any D, there is a network with D hidden layers and $2D$ hidden units which computes an $f : [0, 1] \to [0, 1]$ of period $1/2^D$



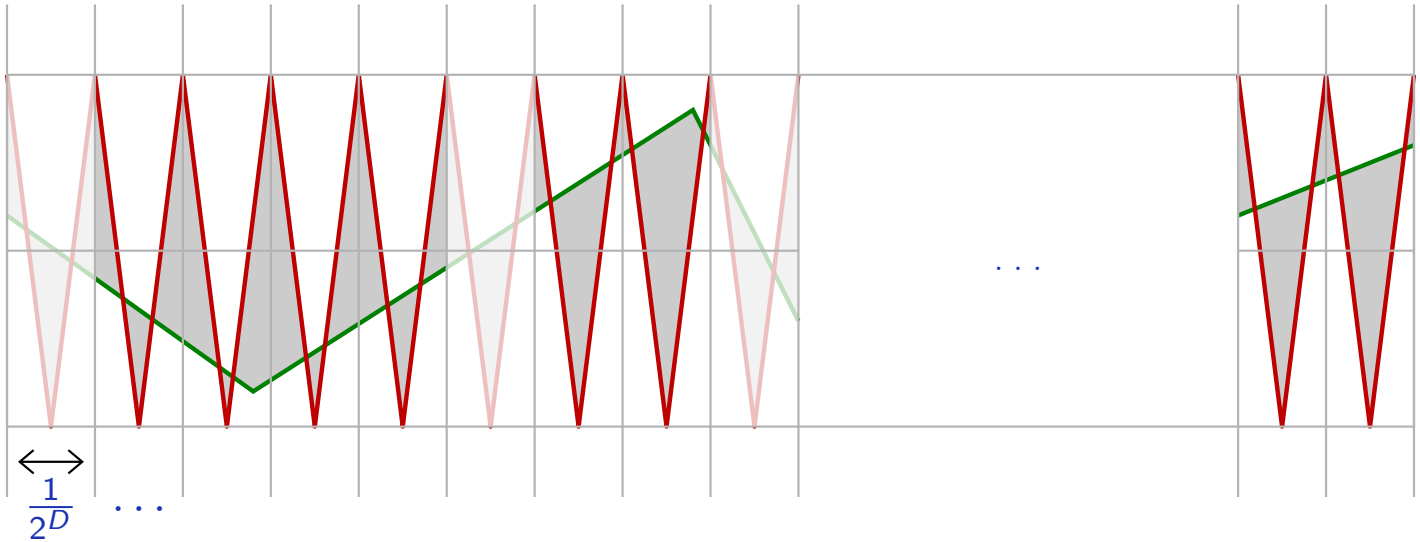


Given $g \in \mathscr{F}$



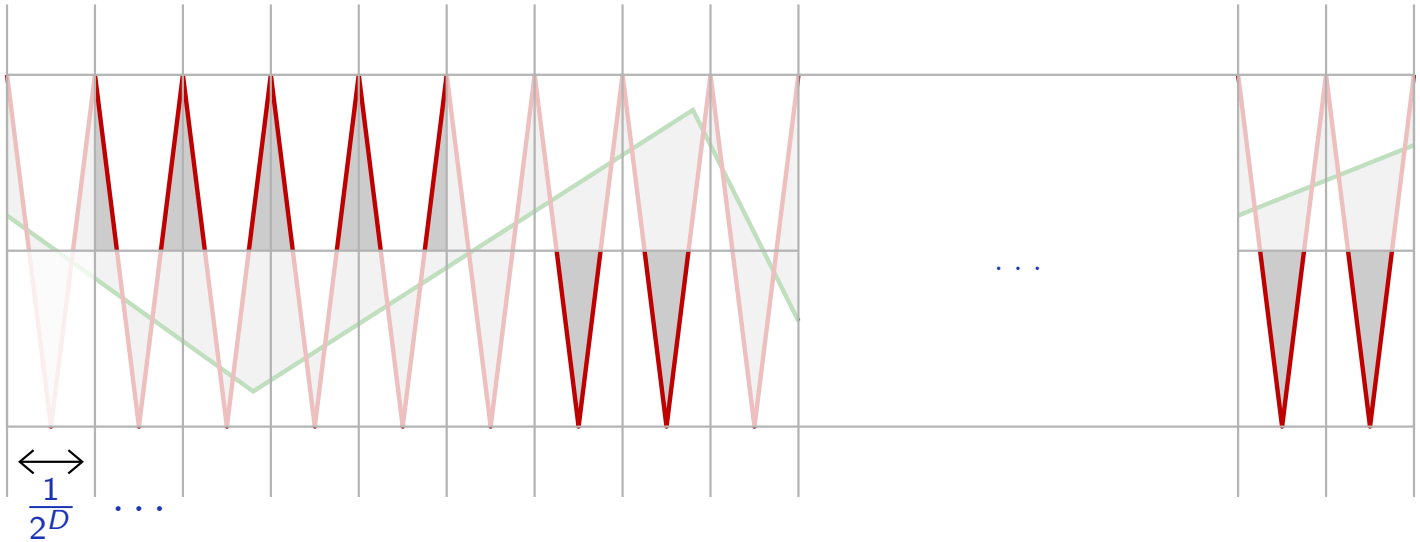
Given $g \in \mathcal{F}$

$$\int_0^1 |f(x) - g(x)|$$



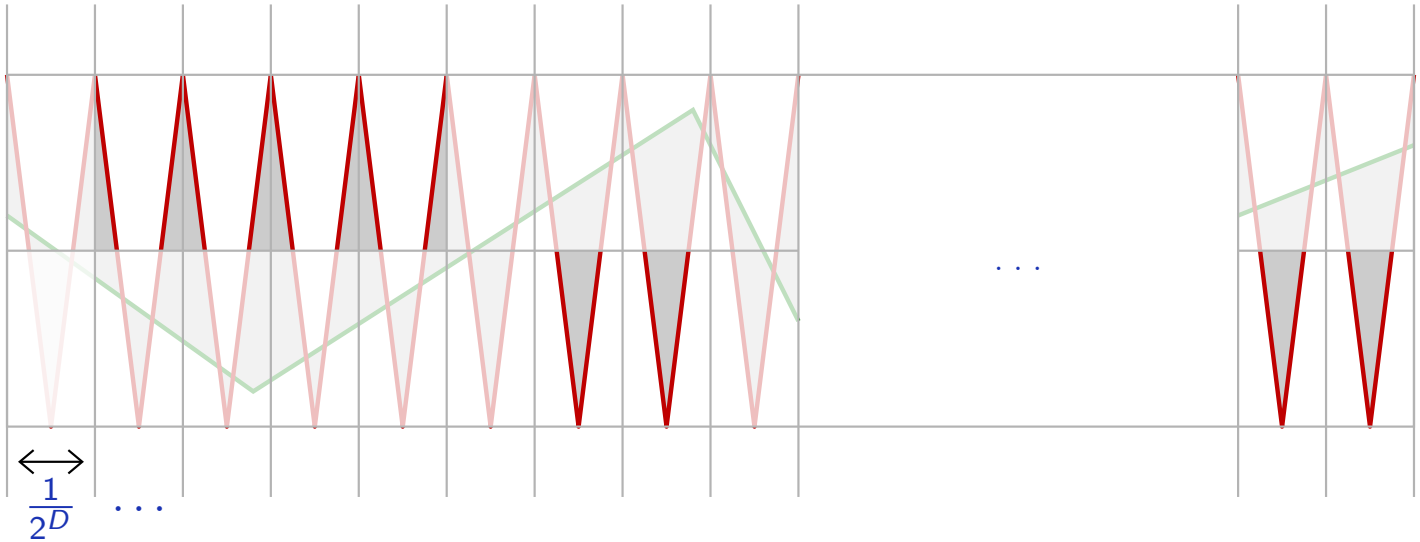
Given $g \in \mathscr{F}$, it crosses $\frac{1}{2}$ at most $\kappa(g)$ times

$$\int_0^1 |f(x) - g(x)|$$



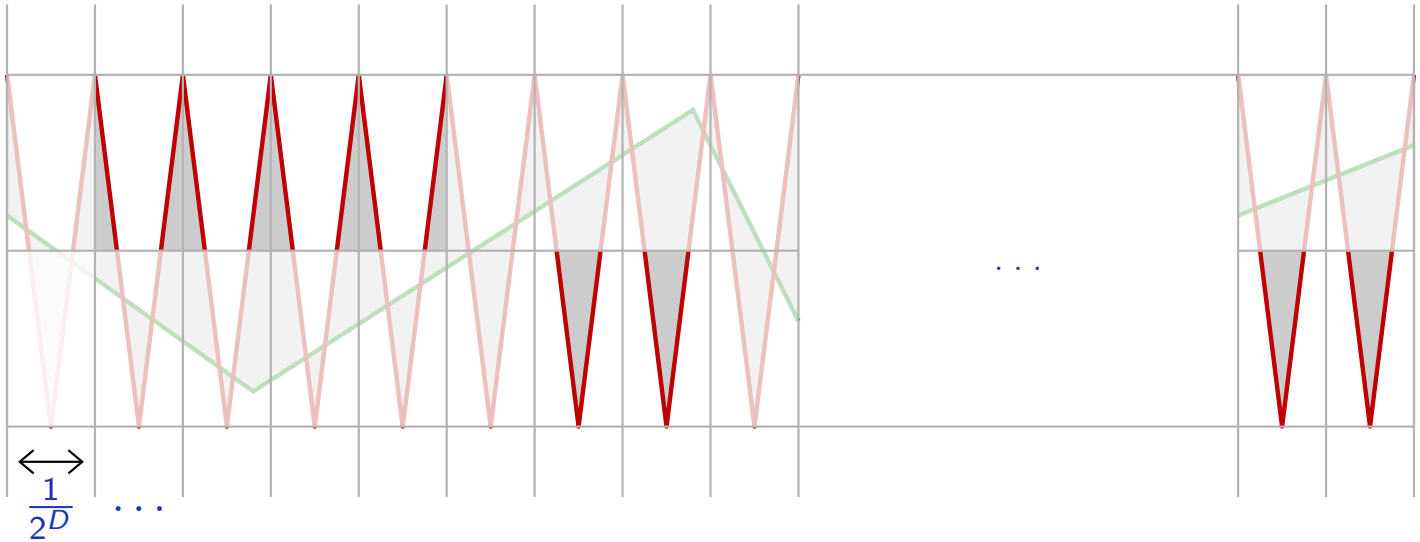
Given $g \in \mathscr{F}$, it crosses $\frac{1}{2}$ at most $\kappa(g)$ times, which means that on at least $2^D - \kappa(g)$ segments of length $1/2^D$, it is on one side of $\frac{1}{2}$, and

$$\int_0^1 |f(x) - g(x)|$$



Given $g \in \mathscr{F}$, it crosses $\frac{1}{2}$ at most $\kappa(g)$ times, which means that on at least $2^D - \kappa(g)$ segments of length $1/2^D$, it is on one side of $\frac{1}{2}$, and

$$\begin{aligned}\int_0^1 |f(x) - g(x)| &\geq \left(2^D - \kappa(g)\right) \frac{1}{2} \int_0^{1/2^D} \left| f(x) - \frac{1}{2} \right| \\ &= \left(2^D - \kappa(g)\right) \frac{1}{2} \frac{1}{2^D} \frac{1}{8} \\ &= \frac{1}{16} \left(1 - \frac{\kappa(g)}{2^D}\right).\end{aligned}$$



Given $g \in \mathcal{F}$, it crosses $\frac{1}{2}$ at most $\kappa(g)$ times, which means that on at least $2^D - \kappa(g)$ segments of length $1/2^D$, it is on one side of $\frac{1}{2}$, and

$$\begin{aligned}\int_0^1 |f(x) - g(x)| &\geq \left(2^D - \kappa(g)\right) \frac{1}{2} \int_0^{1/2^D} \left| f(x) - \frac{1}{2} \right| \\ &= \left(2^D - \kappa(g)\right) \frac{1}{2} \frac{1}{2^D} \frac{1}{8} \\ &= \frac{1}{16} \left(1 - \frac{\kappa(g)}{2^D}\right).\end{aligned}$$

And we multiply f by 16 to get rid of the $\frac{1}{16}$.

So, considering ReLU MLPs with a single input/output:

There exists a network f with D^* layers, and $2D^*$ internal units, such that, for any network g with D layers of sizes $\{W^{(1)}, \ldots, W^{(D)}\}$:

$$\|f - g\|_1 \geq 1 - \frac{2^D}{2^{D^*}} \prod_{d=1}^{D} W^{(d)}.$$

So, considering ReLU MLPs with a single input/output:

There exists a network f with D^* layers, and $2D^*$ internal units, such that, for any network g with D layers of sizes $\{W^{(1)}, \ldots, W^{(D)}\}$:

$$\|f - g\|_1 \geq 1 - \frac{2^D}{2^{D^*}} \prod_{d=1}^{D} W^{(d)}.$$

In particular, with g a single hidden layer network

$$\|f - g\|_1 \geq 1 - 2\frac{W^{(1)}}{2^{D^*}}.$$

To approximate f properly, the width $W^{(1)}$ of g's hidden layer has to increase exponentially with f's depth D^*.

So, considering ReLU MLPs with a single input/output:

There exists a network f with D^* layers, and $2D^*$ internal units, such that, for any network g with D layers of sizes $\{W^{(1)}, \ldots, W^{(D)}\}$:

$$\|f - g\|_1 \geq 1 - \frac{2^D}{2^{D^*}} \prod_{d=1}^{D} W^{(d)}.$$

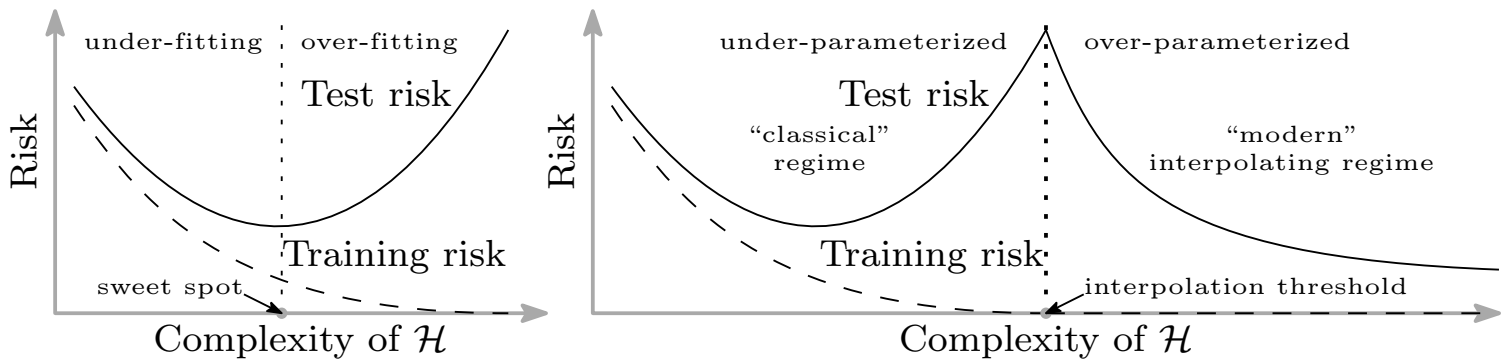
In particular, with g a single hidden layer network

$$\|f - g\|_1 \geq 1 - 2\frac{W^{(1)}}{2^{D^*}}.$$

To approximate f properly, the width $W^{(1)}$ of g's hidden layer has to increase exponentially with f's depth D^*.

This is a simplified variant of results by Telgarsky (2015, 2016).

Regarding over-fitting, over-parametrizing a deep model often improves test performance, contrary to what the bias-variance decomposition predicts (Belkin et al., 2018).



(a) U-shaped "bias-variance" risk curve

(b) "double descent" risk curve

Figure 1: Curves for training risk (dashed line) and test risk (solid line). (a) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (b) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high complexity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

(Belkin et al., 2018)

So we have good reasons to increase depth, but we saw that an important issue then is to control the amplitude of the gradient, which is tightly related to controlling activations.

So we have good reasons to increase depth, but we saw that an important issue then is to control the amplitude of the gradient, which is tightly related to controlling activations.

In particular we have to ensure that

- the gradient does not "vanish" (Bengio et al., 1994; Hochreiter et al., 2001),
- gradient amplitude is homogeneous so that all parts of the network train at the same rate (Glorot and Bengio, 2010),
- the gradient does not vary too unpredictably when the weights change (Balduzzi et al., 2017).

Modern techniques change the functional itself instead of trying to improve training "from the outside" through penalty terms or better optimizers.

Modern techniques change the functional itself instead of trying to improve training "from the outside" through penalty terms or better optimizers.

Our main concern is to make the gradient descent work, even at the cost of engineering substantially the class of functions.

Modern techniques change the functional itself instead of trying to improve training "from the outside" through penalty terms or better optimizers.

Our main concern is to make the gradient descent work, even at the cost of engineering substantially the class of functions.

An additional issue for training very large architectures is the computational cost, which often turns out to be the main practical problem.

The end

References

D. Balduzzi, M. Frean, L. Leary, J. Lewis, K. Wan-Duo Ma, and B. McWilliams. **The shattered gradients problem: If resnets are the answer, then what is the question?** CoRR, abs/1702.08591, 2017.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. **Reconciling modern machine learning and the bias-variance trade-off**. CoRR, abs/1812.11118, 2018.

Y. Bengio, P. Simard, and P. Frasconi. **Learning long-term dependencies with gradient descent is difficult**. IEEE Transactions on Neural Networks, 5(2):157–166, Mar. 1994.

X. Glorot and Y. Bengio. **Understanding the difficulty of training deep feedforward neural networks**. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2010.

K. He, X. Zhang, S. Ren, and J. Sun. **Deep residual learning for image recognition**. CoRR, abs/1512.03385, 2015.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies, pages 237–243. IEEE Press, 2001.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. **Backpropagation applied to handwritten zip code recognition**. Neural Computation, 1(4):541–551, 1989.

K. Simonyan and A. Zisserman. **Very deep convolutional networks for large-scale image recognition**. CoRR, abs/1409.1556, 2014.

M. Telgarsky. **Representation benefits of deep feedforward networks**. CoRR, abs/1509.08101, 2015.

M. Telgarsky. **Benefits of depth in neural networks**. CoRR, abs/1602.04485, 2016.