EE-559 – Deep learning

6.1. Benefits of depth

François Fleuret
https://fleuret.org/ee559/
Wed May 1 15:08:28 UTC 2019

For image classification for instance, there has been a trend toward deeper architectures to improve performance.

Network	Nb. layers
LeNet5 (leCun et al., 1998)	5
AlexNet (Krizhevsky et al., 2012)	8
VGG (Simonyan and Zisserman, 2014)	11–19
GoogleLeNet (Szegedy et al., 2015)	22
Inception v4 (Szegedy et al., 2016)	76
Resnet (He et al., 2015)	34–152
Resnet (He et al., 2016)	1001
Resnet (Huang et al., 2016)	1202

For image classification for instance, there has been a trend toward deeper architectures to improve performance.

Network	**Nb. layers**
LeNet5 (leCun et al., 1998)	5
AlexNet (Krizhevsky et al., 2012)	8
VGG (Simonyan and Zisserman, 2014)	11–19
GoogleLeNet (Szegedy et al., 2015)	22
Inception v4 (Szegedy et al., 2016)	76
Resnet (He et al., 2015)	34–152
Resnet (He et al., 2016)	1001
Resnet (Huang et al., 2016)	1202

"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)

For image classification for instance, there has been a trend toward deeper architectures to improve performance.

Network	**Nb. layers**
LeNet5 (leCun et al., 1998)	5
AlexNet (Krizhevsky et al., 2012)	8
VGG (Simonyan and Zisserman, 2014)	11–19
GoogleLeNet (Szegedy et al., 2015)	22
Inception v4 (Szegedy et al., 2016)	76
Resnet (He et al., 2015)	34–152
Resnet (He et al., 2016)	1001
Resnet (Huang et al., 2016)	1202

> "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.

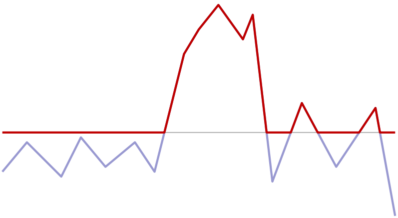
Let $\mathscr{F}$ be the set of piece-wise linear mappings on $[0, 1]$, and $\forall f \in \mathscr{F}$, let $\kappa(f)$ be the minimum number of linear pieces needed to represent f.

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 needed to represent 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 needed to represent f.



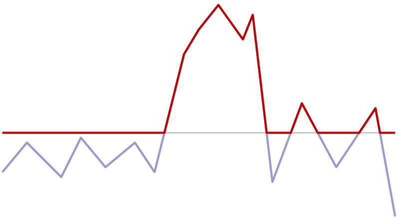
Let σ be the ReLU function

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

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

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

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 needed to represent 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

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

and we also have

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

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

$$x_1^0 = x,$$

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

$$y = x_1^D.$$

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

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 $\forall i, \kappa(s_i^1) = 1$, and

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

from which

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

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

$$x^0_1 = x,$$

$$\forall d = 1, \dots, D, \forall i, \quad \begin{cases} s^d_i & = \sum_{j=1}^{W^{d-1}} w^d_{i,j} x^{d-1}_j + b^d_i \\ x^d_i & = \sigma(s^d_i) \end{cases}$$

$$y = x^D_1.$$

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

$$\forall l, i, \kappa\left(x^l_i\right) = \kappa\left(\sigma(s^l_i)\right) \leq 2\kappa\left(s^l_i\right) \leq 2 \sum_{j=1}^{W_{l-1}} \kappa\left(x^{l-1}_j\right)$$

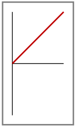
from which

$$\forall l, \max_i \kappa\left(x^l_i\right) \leq 2 W_{l-1} \max_j \kappa\left(x^{l-1}_j\right)$$

and we get the following bound for any ReLU MLP

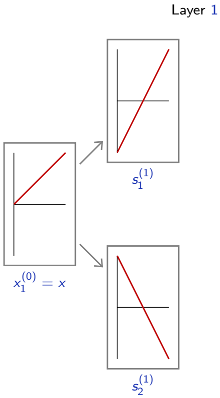
$$\kappa(y) \leq 2^D \prod_{d=1}^{D} W_d.$$

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

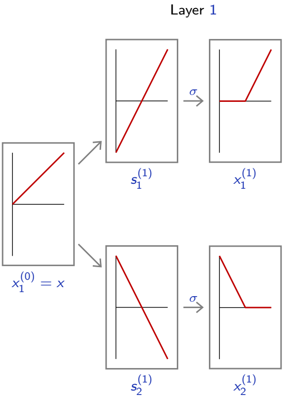


$x_1^{(0)} = x$

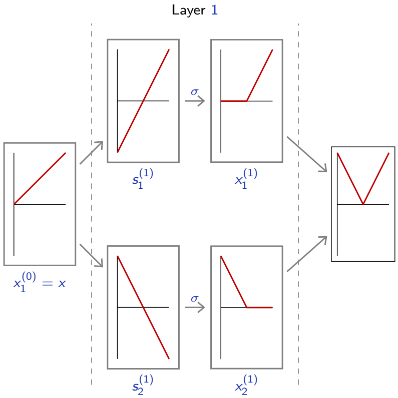
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



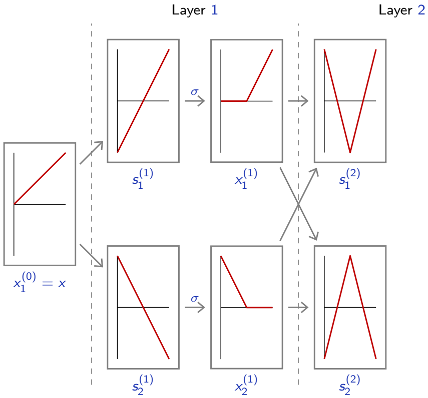
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



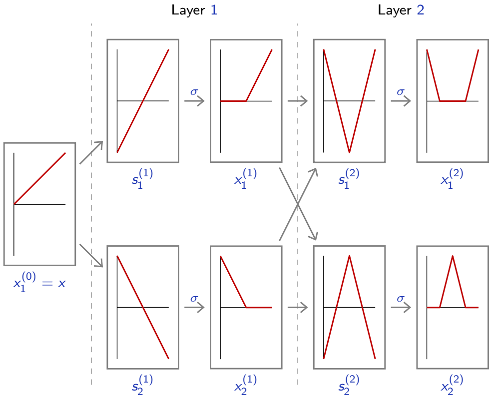
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



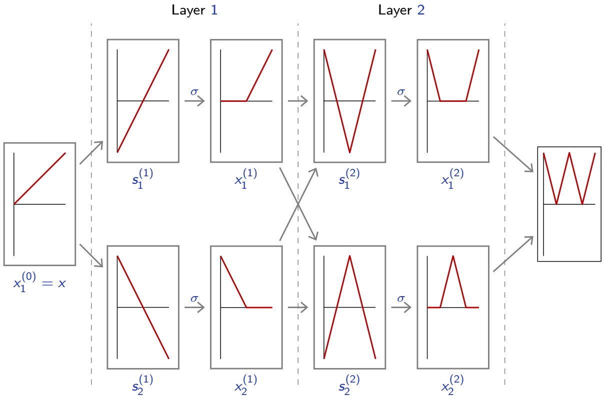
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



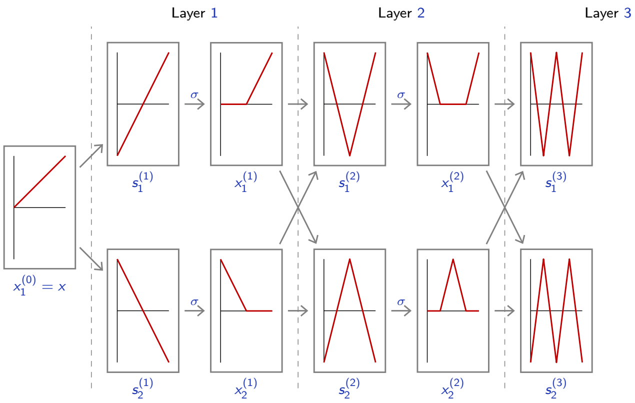
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



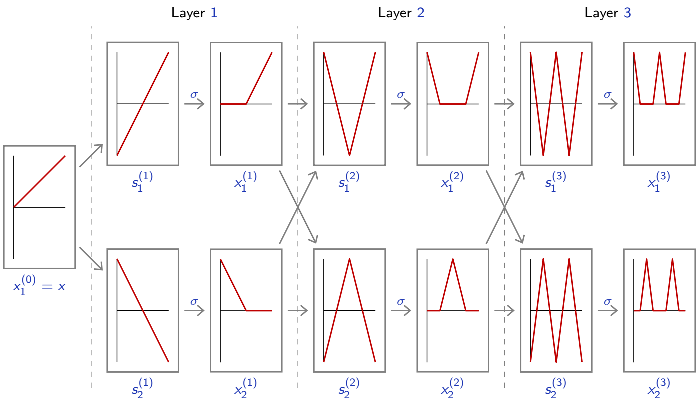
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



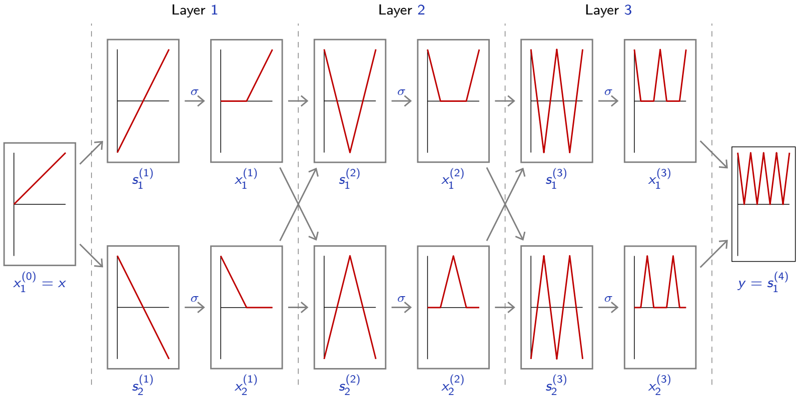
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:



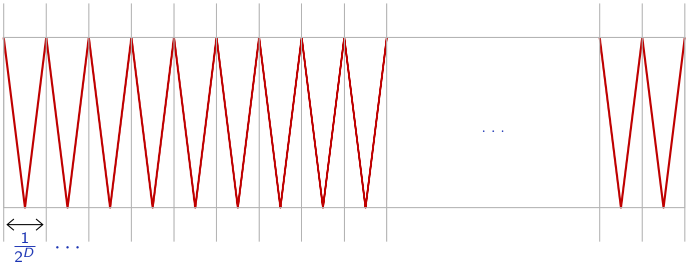
Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:

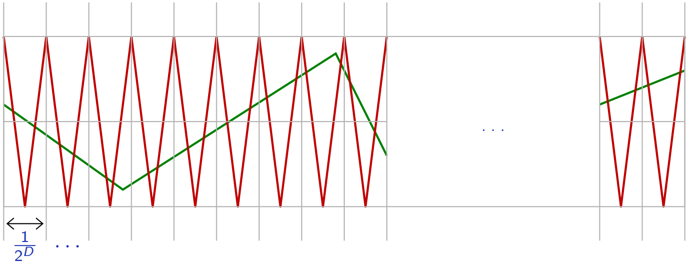


Although this seems quite a pessimist bound, we can hand-design a network that [almost] reaches it:

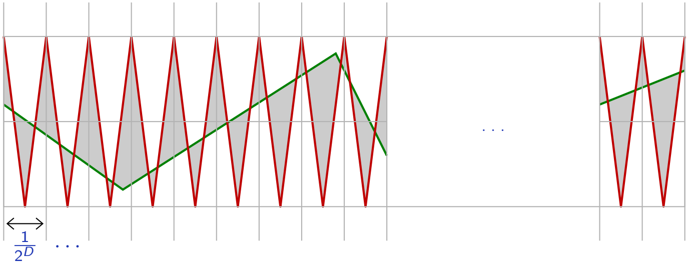


So for any D, there is a network with D hidden layers and $2D$ hidden units which computes an $f : [0, 1] \to [0, 1]$ of period $1/2^D$



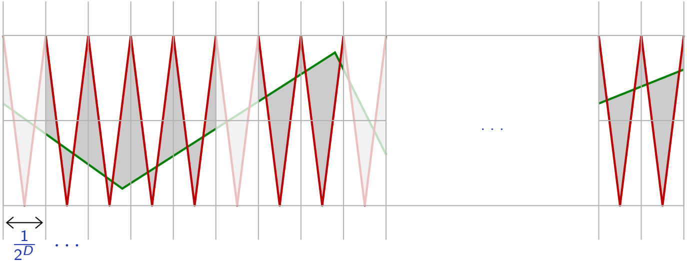


Given $g \in \mathscr{F}$



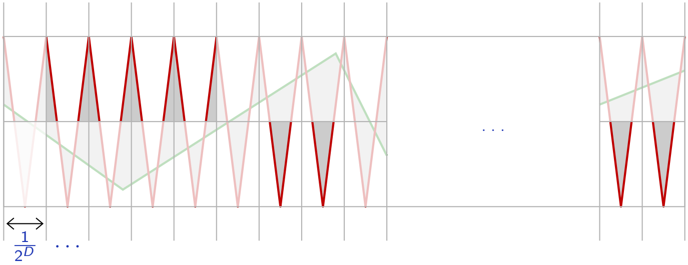
Given $g \in \mathscr{F}$

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



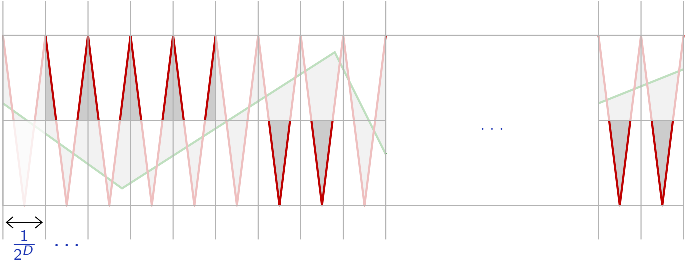
Given $g \in \mathscr{F}$, it crosses $\frac{1}{2}$ at most $\kappa(g)$ times

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



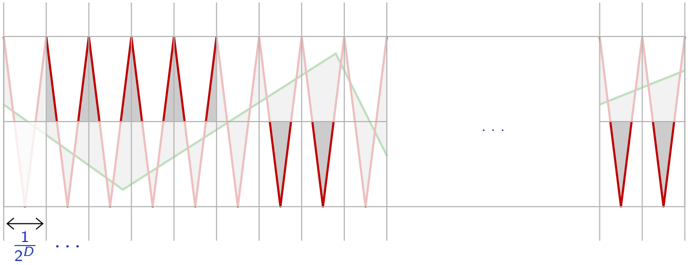
Given $g \in \mathscr{F}$, it crosses $\frac{1}{2}$ at most $\kappa(g)$ times, which means that on at least $2^D - \kappa(g)$ segments of length $1/2^D$, it is on one side of $\frac{1}{2}$, and

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



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

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



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

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

And we multiply f by 16 to get our final result.

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 $\{W_1, \dots, W_D\}$:

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

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 $\{W_1, \dots, W_D\}$:

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

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 $\{W_1, \dots, W_D\}$:

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

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).

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.

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.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. CoRR, abs/1603.05027, 2016.

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.

G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. CoRR, abs/1603.09382, 2016.

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In Neural Information Processing Systems (NIPS), 2012.

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.

Y. leCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

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

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR, abs/1602.07261, 2016.

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.