The Power of Max Pooling Layer

Jiří Šíma

sima@cs.cas.cz

Institute of Computer Science
Czech Academy of Sciences, Prague, Czechia

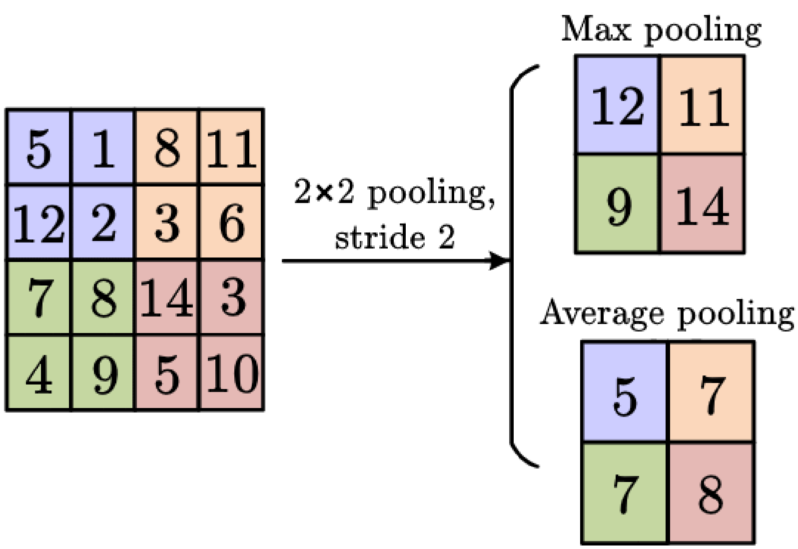
joint work with **Jérémie Cabessa**

jeremie.cabessa@uvsq.fr

DAVID Laboratory
Paris-Saclay University, Versailles, France

Max Pooling Layers

- convolutional neural networks (CNNs) are widely used across AI domains such as computer vision, natural language processing, speech recognition
- pooling layers are basic building blocks of CNNs
- two main types of pooling layers commonly used are maximum and average



  - downsample the spatial dimensions of feature maps
  - remove redundant information
  - improve robustness to variations, distortions, and noise in input data
  - enhance efficiency in computation and memory
  - help mitigate overfitting

- numerous studies show that max pooling is effective in practice

Motivations

CNNs are composed of heterogeneous layers:

Convolutional & **Fully Connected**: ReLU unit with the rectified linear activation:

$$R\left(w_0 + \sum_{i=1}^{n} w_i\, x_i\right) \quad \text{where } R(x) = \mathrm{ReLU}(x) = \max(0, x)$$

vs. **Max** or **Average Pooling:** $\max\left(x_1, \ldots, x_n\right)$ or $\dfrac{1}{n}\sum_{i=1}^{n} x_i$

for $x_1, \ldots, x_n \geq 0$ (as outputs from ReLU units)

unification: **Can max pooling layers be implemented by ReLU units?**

(e.g., trivial for average pooling: $w_0 = 0$ and $w_i = 1/n$ for $i \in \{1, \ldots, n\}$)

- unified computation by matrix processor (e.g. TPU, Tensor Core GPU, MMU)
- applying algorithms for deep neural networks (DNNs) with ReLU to CNNs
  e.g., AppMax for error estimation of approximated DNNs (our initial motivation)
- theoretical issue: computational power of max pooling in terms of ReLU units

Potential Depth Hierarchy for (Maximum) ReLU DNNs

Arora et al., 2018: DNNs (with ReLU units and linear output) compute exactly the class of **continuous piecewise linear (CPWL) functions**

Upper Bound: any CPWL function in n variables is computable by a ReLU DNN with $\lceil \log_2(n+1) \rceil$ hidden layers

$\rightarrow$ $\max(x_1, \ldots, x_n)$ (for all $x_1, \ldots, x_n \in \mathbb{R}$) can thus be implemented using $\lceil \log_2(n+1) \rceil$ hidden layers of (linearly many) ReLU units

Depth Hierarchy Conjecture: the classes of functions computable by ReLU DNNs form a strict hierarchy as the depth increases, up to the logarithmic upper bound

equivalent to the Lower Bound: any ReLU DNN computing $\max(x_1, \ldots, x_n, 0)$ requires strictly more than $k = \log_2 n$ hidden layers (Hertrich et al., 2023)

- holds for $k = 1$: any DNN computing $\max(x_1, x_2, 0)$ (for all $x_1, x_2 \in \mathbb{R}$) requires at least two hidden layers of ReLU units (Mukherjee, Basu, 2017)
- $k = \log_2 \log_2 n$ hidden layers insufficient for H-conforming ReLU DNNs: each ReLU unit acts linearly (as ReLU is identity or 0) under any fixed ordering of input values (Grillo et al., 2025)
- matching lower bound: $\lceil \log_2(n+1) \rceil$ hidden layers required for integer weights (Haase et al., 2023)

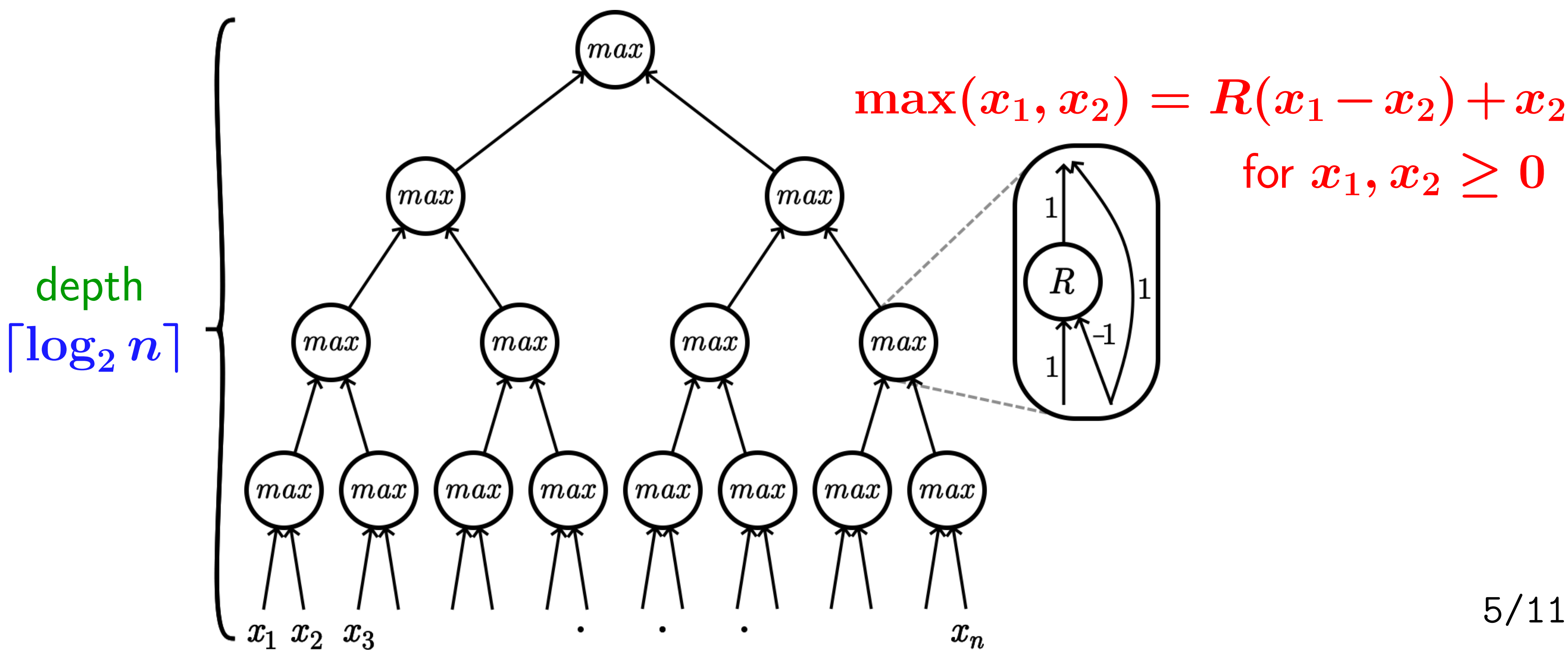
Log-Depth ReLU DNN for Computing the Maximum

constructions of a DNN computing $\max(x_1, \ldots, x_n)$ (for all $x_1, \ldots, x_n \in \mathbb{R}$), e.g., using $\lceil \log_2 n \rceil$ hidden layers, $3(n-1)$ ReLU units, and weights $-1, 1$ (folklore: Arora et al., 2018; Hertrich et al., 2023; Matoba et al., 2023)

our slight improvement in size: restricted to nonnegative inputs (i.e. ReLU unit outputs) and allows layer-skipping connections:

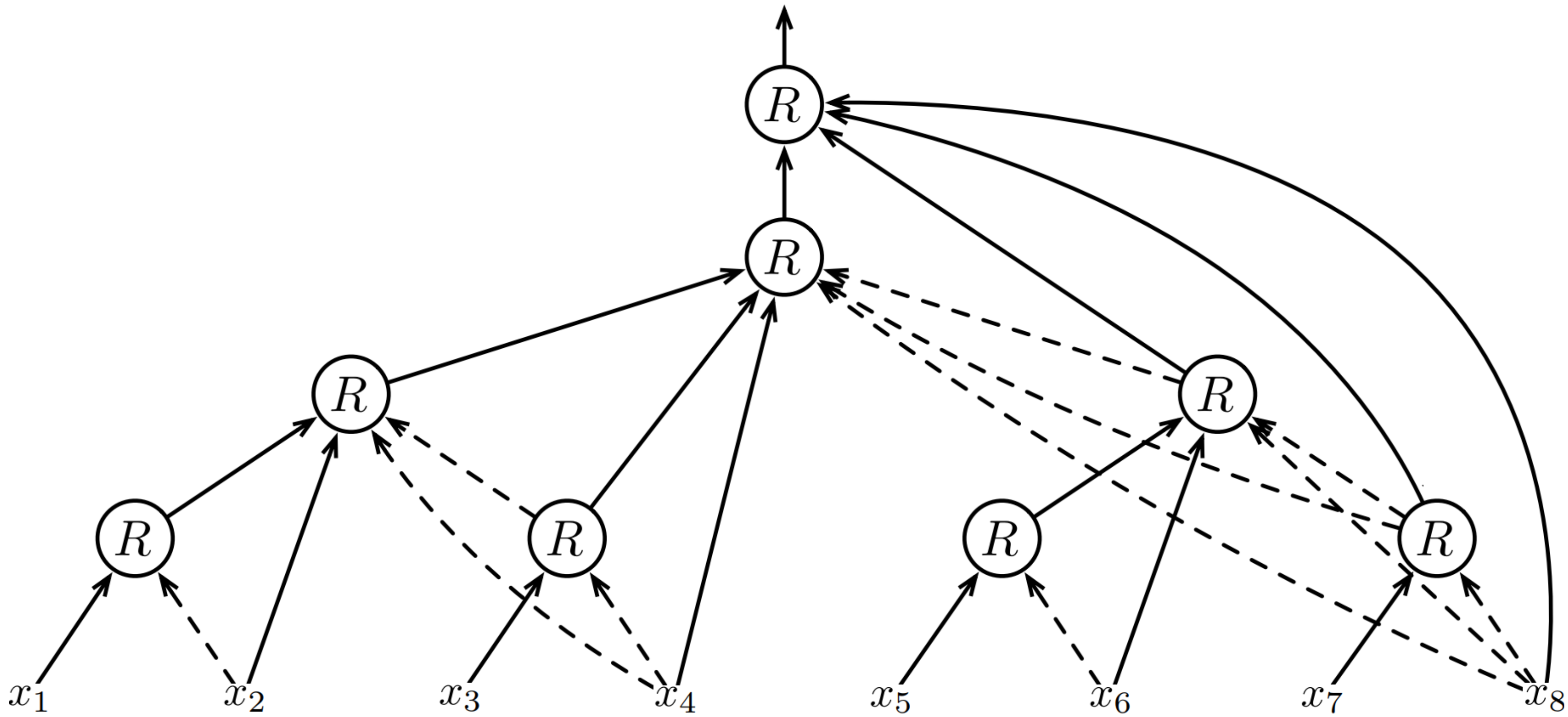
Theorem 1. *The maximum of $n \geq 2$ nonnegative real numbers $x_1, \ldots, x_n$ can be computed by a DNN of size n ReLU units, depth $\lceil \log_2 n \rceil + 1$ non-input layers, and bipolar weights $-1, 1$.*

Idea of Proof: binary tree composed of max units



Example of a ReLU NN Computing $max(x_1, \ldots, x_8)$

(based on Theorem 1)



real inputs: $\boldsymbol{x_1, \ldots, x_8} \geq \mathbf{0}$, depth: $\mathbf{4}$ (non-input) layers , size: $\mathbf{8}$ ReLU units ,

weights: $\mathbf{-1}$ (dashed arrows) or $\mathbf{1}$ (solid arrows)

Drawback: logarithmic depth and sparse connections $\longrightarrow$ inefficient for evaluation, e.g., via matrix operations

Constant-Depth ReLU DNN for Computing the Maximum

in general, contradicts the depth hierarchy hypothesis $\longrightarrow$ additional assumption:
maximum and the gap between the largest two numbers are bounded

our quadratic-size construction where the bounds trade off depth vs. weight:

Theorem 2. *Let* $x_1, \ldots, x_n$ *be* $n \geq 2$ *nonnegative real numbers. Denote by* $\mu_1 = \max\{x_1, \ldots, x_n\}$ *and* $\mu_2 = \max\{x_1, \ldots, x_n\} \setminus \{\mu_1\} \cup \{0\}$ *their largest two values (or zero). Then for any integer* $r \geq 0$, *the maximum* μ_1 *can be computed by a ReLU DNN* $\mathcal{N}_r$ *of size* $rn^2 + n + 1$, *depth* $2r + 2$, *and weights* $-1, 1, -\sqrt{w}, \sqrt{w}$ $(w \geq 1)$ *such that*

$$(w+1)^r \geq \frac{\mu_1}{\mu_1 - \mu_2} \quad \text{if } \mu_1 > 0\,, \quad \text{or } w = 1 \text{ otherwise.}$$

Idea of Proof: let $W \geq \mu_1/(\mu_1 - \mu_2) > 0$ be sufficiently large, then for each $j \in \{1, \ldots, n\}$, decide if $x_j = \mu_1 = \max(x_1, \ldots, x_n)$ using:

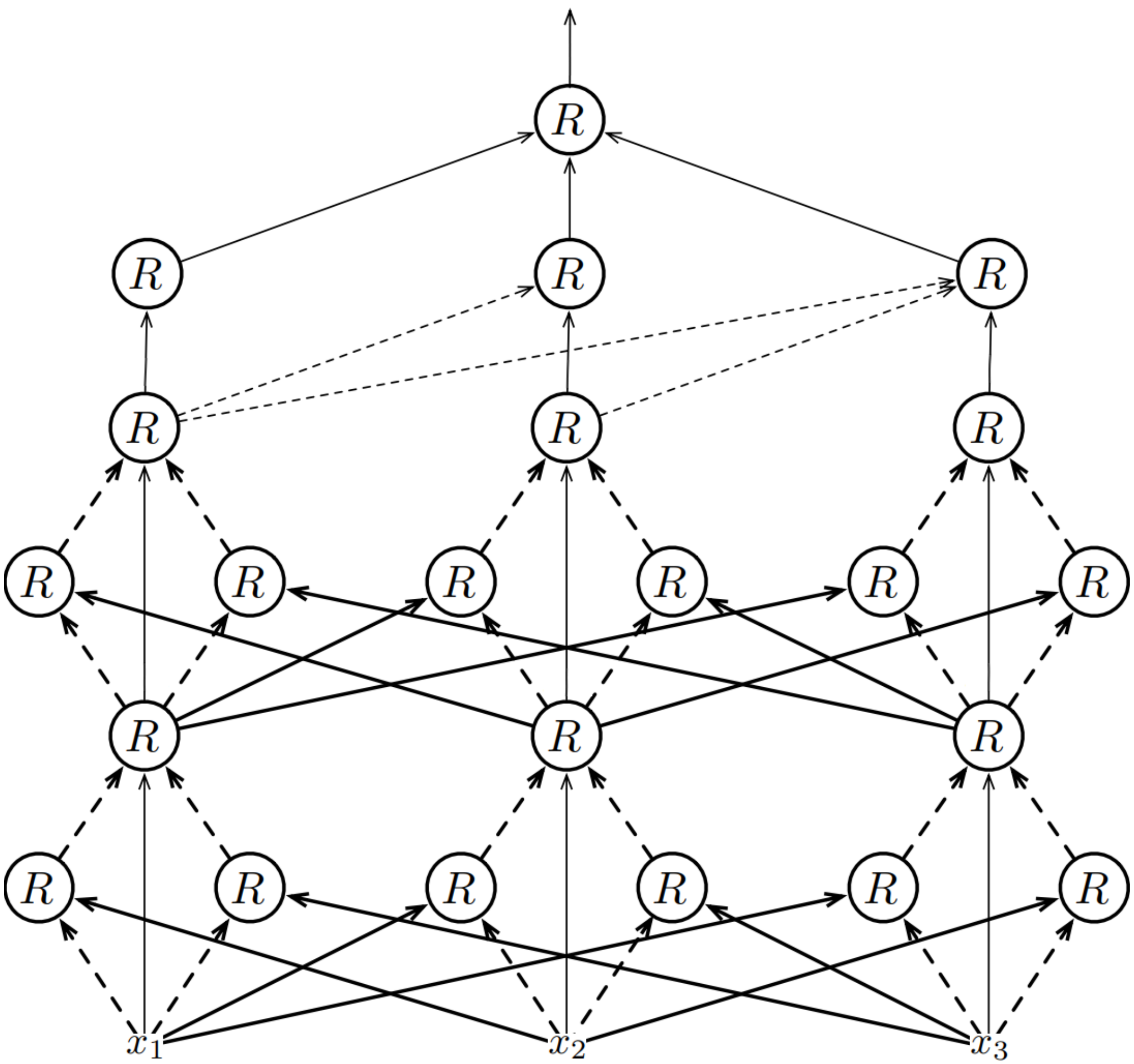
$$R\left(x_j - \sum_{i=1}^{n} R(W(x_i - x_j))\right) = \begin{cases} x_j & \text{if } x_j = \mu_1 \\ 0 & \text{if } x_j < \mu_1 \end{cases} :$$

since $\sum_{i=1}^{n} R(W(x_i - x_j)) \geq W(\mu_1 - \mu_2) \geq \mu_1$ whenever $x_j < \mu_1$

the large weight $W = (w+1)^r$ is split into $2r$ layers with weights $\sqrt{w}$

Example of a ReLU NN $\mathcal{N}_2$ Computing $max(x_1, x_2, x_3)$

(based on Theorem 2)



$\mathcal{N}_2$: $r = 2$

depth: **6** (non-input) layers

size: **22** ReLU units

weights:

- dashed thin arrows: -1
- solid thin arrows: 1
- dashed thick arrows: $-\sqrt{w}$
- solid thick arrows: $\sqrt{w}$

real inputs: $x_1, x_2, x_3 \geq 0$

Example of Calculating the Maximum for Integers:

integer $x_1, \ldots, x_n$, i.e. $\mu_1 - \mu_2 \geq 1$ (for $\mu_1 \neq \mu_2$)

$\rightarrow$ $\mathcal{N}_1$: depth: 4 (non-input) layers, size: $n^2 + n + 1$ ReLU units,

weights: $-1, 1, \sqrt{w}, -\sqrt{w}$ for $w \geq \max(\mu_1 - 1, 1)$

the fixed weights depend on an **a priori unknown maximum** to compute $\mathcal{N}_1$ correctly **vs.** the maximum is bounded in realistic computer numerics:

Depth–Weight Trade-off For Computer Data Types

- **unsigned integer** with standard b-bit precision for $b = 16, 32, 64$

$$\rightarrow \quad \mu_1 \leq 2^b - 1 \text{ and } \mu_1 - \mu_2 \geq 1$$

the weight $\max(\sqrt{w}, 1)$ of $\mathcal{N}_r$ in Theorem 2, valid for $w = \sqrt[r]{2^b - 1} - 1$:

depth of $\mathcal{N}_r$	4 ($\mathcal{N}_1$)	6 ($\mathcal{N}_2$)	8 ($\mathcal{N}_3$)	10 ($\mathcal{N}_4$)	12 ($\mathcal{N}_5$)	22 ($\mathcal{N}_{10}$)	32 ($\mathcal{N}_{15}$)
16-bit ushort	256.00	15.97	6.28	3.88	2.87	1.43	1.05
32-bit uint	65536.00	256.00	40.31	15.97	9.14	2.87	1.85
64-bit ulong	4294967296.00	65536.00	1625.50	256.00	84.45	9.14	4.28

- **floating-point** (IEEE 754) with standard b-bit precision for $b = 16, 32, 64$ including $e = 4, 7, 10$ bits for exponent

$$\rightarrow \quad \mu_1 \leq 2^{2^e}(1 - 2^{-b+e+1}) \text{ and } \mu_1 - \mu_2 \geq 2^{-2^b-b+e+4}$$

the weight $\max(\sqrt{w}, 1)$ of $\mathcal{N}_r$ in Theorem 2, valid for

$$w = \sqrt[r]{2^{2^{e+1}-3}(2^{b-e-1} - 1)} - 1 :$$

depth of $\mathcal{N}_r$	8 ($\mathcal{N}_3$)	22 ($\mathcal{N}_{10}$)	44 ($\mathcal{N}_{20}$)	102 ($\mathcal{N}_{50}$)	302 ($\mathcal{N}_{150}$)	1002 ($\mathcal{N}_{500}$)
16-bit half ($e = 4$)	101.59	3.88	1.74	1	1	1
32-bit single ($e = 7$)	7.90E13	14766.09	121.52	6.75	1.62	1
64-bit double ($e = 10$)	1.83E105	3.79E31	6.16E15	2068279.89	127.41	4.17

Lower Bound

Theorem 3. *There is no depth-**2** ReLU neural network that computes the maximum of more than **two** nonnegative real numbers.*

Summary

- investigated the computational power of max pooling in terms of ReLU units
- log-depth, linear-size ReLU DNN for $\max$ of n nonnegative numbers (Thm. 1)
- constant-depth, quadratic-size ReLU NN for $\max$ of bounded, limited-precision nonnegative inputs (Thm. 2), e.g., integer or floating-point data types
- unified evaluation of CNNs via matrix operations
- application example: AppMax method for estimating the maximum error of low-energy approximated CNNs, based on linear programming
  (Šíma, Vidnerová, ECML PKDD 2025 & ICONIP 2025)
- lower bound: no depth-$\mathbf{2}$ ReLU NN can compute $\max$ of $\mathbf{3}$ nonnegative reals (Thm. 3) $\longrightarrow$ max pooling cannot be replaced by a single convolutional layer
- open problem: extend the lower bound to nonconstant (logarithmic) depth
  $\longrightarrow$ establish strict depth hierarchy for ReLU DNNs