# Approximation speed of quantized vs. unquantized ReLU neural networks and beyond

Antoine Gonon

ENS Lyon

Joint work with: Nicolas Brisebarre, Rémi Gribonval, Elisa Riccietti

Curves & Surfaces, Arcachon, June 22, 2022

# ReLU neural networks

parameters

$$(\underbrace{A_1, \ldots, A_L}_{\text{matrices}}, \underbrace{b_1, \ldots, b_L}_{\text{vectors}})$$

function represented

# ReLU neural networks

parameters
$$\underbrace{(A_1, \ldots, A_L, b_1, \ldots, b_L)}_{\theta}$$

function represented

# ReLU neural networks

| parameters | | function represented |
| --- | --- | --- |
| $\underbrace{(A_1, \ldots, A_L, b_1, \ldots, b_L)}_{\theta}$ | $x \mapsto$ | $A_1 x + b_1$ |

# ReLU neural networks

$$\underbrace{(A_1, \ldots, A_L, b_1, \ldots, b_L)}_{\theta}^{\text{parameters}} \quad x \mapsto \quad \overset{\text{function represented}}{\rho(A_1 x + b_1)}$$

with $\rho(x) = \max(0, x) = x_+$ the ReLU function

# ReLU neural networks

$$\overbrace{\underbrace{(A_1, \ldots, A_L, b_1, \ldots, b_L)}_{\theta}}^{\text{parameters}} \quad x \mapsto \qquad \overbrace{A_2\rho(A_1 x + b_1) + b_2}^{\text{function represented}}$$

with $\rho(x) = \max(0, x) = x_+$ the ReLU function

# ReLU neural networks

$$\underbrace{(A_1, \ldots, A_L, b_1, \ldots, b_L)}_{\theta}^{\text{parameters}} \quad x \mapsto \quad \overbrace{\rho(A_2\rho(A_1 x + b_1) + b_2)}^{\text{function represented}}$$

with $\rho(x) = \max(0, x) = x_+$ the ReLU function

# ReLU neural networks

$$\underbrace{\overset{\text{parameters}}{(A_1, \ldots, A_L, b_1, \ldots, b_L)}}_{\theta} \quad \overset{\text{function represented}}{x \mapsto A_L \cdots \rho(A_2 \rho(A_1 x + b_1) + b_2) \cdots + b_L}$$

with $\rho(x) = \max(0, x) = x_+$ the ReLU function

# ReLU neural networks

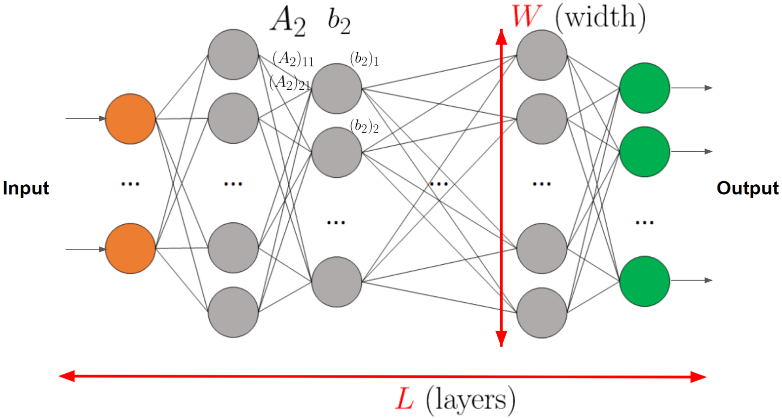
$$\underbrace{(A_1, \ldots, A_L, b_1, \ldots, b_L)}_{\theta}^{\text{parameters}} \quad \underbrace{x \mapsto A_L \cdots \rho(A_2\rho(A_1 x + b_1) + b_2) \cdots + b_L}_{R_\theta(x)}^{\text{function represented}}$$

with $\rho(x) = \max(0, x) = x_+$ the ReLU function

# ReLU neural networks

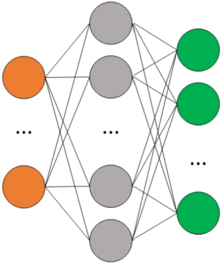
$$\underbrace{(A_1, \ldots, A_L, b_1, \ldots, b_L)}_{\theta}^{\text{parameters}} \quad \underbrace{x \mapsto A_L \cdots \rho(A_2\rho(A_1 x + b_1) + b_2) \cdots + b_L}_{R_\theta(x)}^{\text{function represented}}$$

with $\rho(x) = \max(0, x) = x_+$ the ReLU function



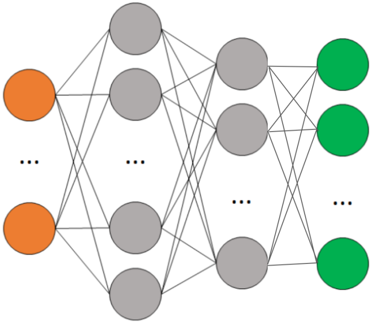
# Context: approximation with increasing complexity

**Consider:** $\Sigma_M$ set of networks with complexity increasing with $M$



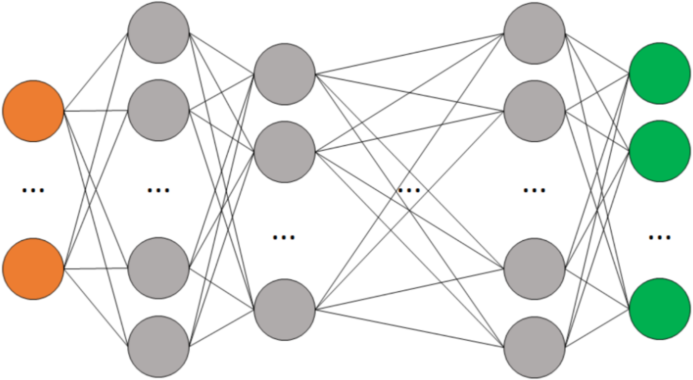
# Context: approximation with increasing complexity

**Consider:** $\Sigma_M$ set of networks with complexity increasing with $M$



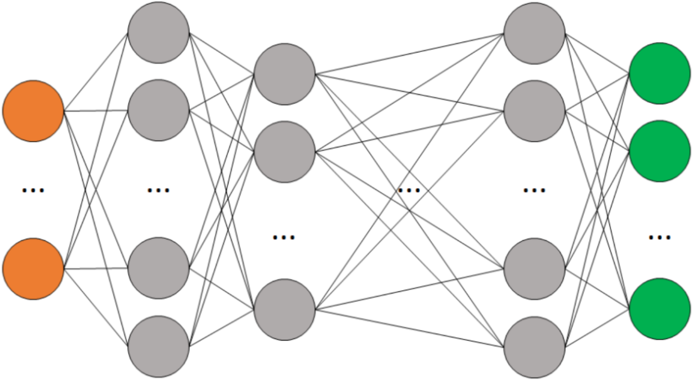
# Context: approximation with increasing complexity

**Consider:** $\Sigma_M$ set of networks with complexity increasing with $M$



# Context: approximation with increasing complexity

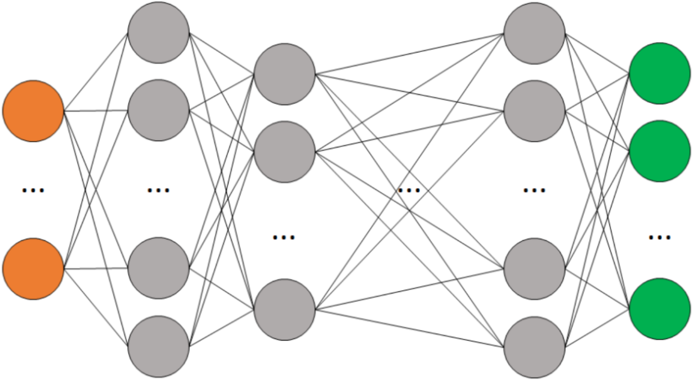
**Consider:** $\Sigma_M$ set of networks with complexity increasing with $M$



**complexity** $\nearrow \Longrightarrow$ **approximation error** $\searrow$

# Context: approximation with increasing complexity

**Consider:** $\Sigma_M$ set of networks with complexity increasing with $M$




$$\textbf{complexity} \nearrow \Longrightarrow \textbf{approximation error} \searrow$$

**Typical known result:**

$$f \text{ "}\gamma\text{-smooth"} \Longrightarrow d(f, \Sigma_M) \lesssim M^{-\gamma}$$

# Challenges

**Definition:**

$$\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) := \text{largest } \gamma > 0 \text{ s.t. } \sup_{f\in\mathcal{C}} d(f, \Sigma_M) \underset{M\to\infty}{=} \mathcal{O}(M^{-\gamma})$$

# Challenges

**Definition:**
$\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) :=$ largest $\gamma > 0$ s.t. $\sup_{f\in\mathcal{C}} d(f, \Sigma_M) \underset{M\to\infty}{=} \mathcal{O}(M^{-\gamma})$

1. **Quantization vs. approximation**

# Challenges

**Definition:**

$$\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) := \text{largest } \gamma > 0 \text{ s.t. } \sup_{f\in\mathcal{C}} d(f, \Sigma_M) \underset{M\to\infty}{=} \mathcal{O}(M^{-\gamma})$$

1. **Quantization vs. approximation**

**Context:** $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma)$ known for ReLU neural networks with **weights in** $\mathbb{R}$ and certain $\mathcal{C}$'s

# Challenges

**Definition:**

$$\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) := \text{largest } \gamma > 0 \text{ s.t. } \sup_{f\in\mathcal{C}} d(f, \Sigma_M) \underset{M\to\infty}{=} \mathcal{O}(M^{-\gamma})$$

1. **Quantization vs. approximation**

**Context:** $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma)$ known for ReLU neural networks with **weights in $\mathbb{R}$** and certain $\mathcal{C}$'s

**Question: Quantized weights?**

# Challenges

**Definition:**

$$\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) := \text{largest } \gamma > 0 \text{ s.t. } \sup_{f\in\mathcal{C}} d(f, \Sigma_M) \underset{M\to\infty}{=} \mathcal{O}(M^{-\gamma})$$

1. **Quantization vs. approximation**

2. **Relation between approximation- and information-theoretic quantities**

**Context:** $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma)$ known for ReLU neural networks with **weights in $\mathbb{R}$** and certain $\mathcal{C}$'s

**Question: Quantized weights?**

# Challenges

**Definition:**

$$\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) := \text{largest } \gamma > 0 \text{ s.t. } \sup_{f\in\mathcal{C}} d(f, \Sigma_M) \underset{M\to\infty}{=} \mathcal{O}(M^{-\gamma})$$

1. **Quantization vs. approximation**

**Context:** $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma)$ known for ReLU neural networks with **weights in $\mathbb{R}$** and certain $\mathcal{C}$'s

**Question: Quantized weights?**

2. **Relation between approximation- and information-theoretic quantities**

**Context:** known inequality in **certain situations**

$$\underbrace{\gamma^{*\text{approx}}(\mathcal{C}|\Sigma)}_{\text{approximation theory}} \leqslant \underbrace{\text{complexity}(\mathcal{C})}_{\text{information theory}}$$

# Challenges

**Definition:**

$$\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) := \text{largest } \gamma > 0 \text{ s.t. } \sup_{f\in\mathcal{C}} d(f, \Sigma_M) \underset{M\to\infty}{=} \mathcal{O}(M^{-\gamma})$$

1. **Quantization vs. approximation**

**Context:** $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma)$ known for ReLU neural networks with **weights in $\mathbb{R}$** and certain $\mathcal{C}$'s

**Question: Quantized weights?**

2. **Relation between approximation- and information-theoretic quantities**

**Context:** known inequality in **certain situations**

$$\underbrace{\gamma^{*\text{approx}}(\mathcal{C}|\Sigma)}_{\text{approximation theory}} \leqslant \underbrace{\text{complexity}(\mathcal{C})}_{\text{information theory}}$$

**Question: Unified framework?**

# Part 1: quantization vs. approximation

# A key tool: bound on the Lipschitz constant of the parameterization

$$\|f - R_{Q(\theta)}\| \leqslant \|f - R_\theta\| + \underbrace{\|R_\theta - R_{Q(\theta)}\|}_{\textit{quantization error}}$$

**Question:** Tradeoff number of bits/quantization error

# A key tool: bound on the Lipschitz constant of the parameterization

$$\|f - R_{Q(\theta)}\| \leqslant \|f - R_\theta\| + \underbrace{\|R_\theta - R_{Q(\theta)}\|}_{\textit{quantization error}}$$

**Question:** Tradeoff number of bits/quantization error

**Known result**[1]**:** On every bounded set of parameters $\Theta$, there exists $K_\Theta > 0$ s.t. for every $\theta, \theta' \in \Theta$:

$$\|R_\theta - R_{\theta'}\|_{L^p} \leqslant K_\Theta \|\theta - \theta'\|_\infty$$

*Explicit bounds on $K_\Theta$?*

[1] Neural network approximation. R. DeVore et al. 2021.

# A key tool: bound on the Lipschitz constant of the parameterization

$$\|f - R_{Q(\theta)}\| \leqslant \|f - R_\theta\| + \underbrace{\|R_\theta - R_{Q(\theta)}\|}_{\textit{quantization error}}$$

**Question:** Tradeoff number of bits/quantization error

**Known result**[1]**:** On every bounded set of parameters $\Theta$, there exists $K_\Theta > 0$ s.t. for every $\theta, \theta' \in \Theta$:

$$\|R_\theta - R_{\theta'}\|_{L^p} \leqslant K_\Theta \|\theta - \theta'\|_\infty$$

*Explicit bounds on* $K_\Theta$*?*

**To understand the tradeoff: new explicit bounds in terms of the depth, the width and a bound on the weights of the network**

[1] Neural network approximation. R. DeVore et al. 2021.

# Contribution 1: explicit bounds on the Lipschitz parameterization of ReLU networks and its consequences

Under mild assumptions, there exists $c > 0$ s.t.

$$\boxed{\frac{1}{c} L B^{L-1} \leqslant K_{\Theta_{L,W}(B)} \leqslant cWL \times LB^{L-1}}$$

**Definition of $\Theta_{L,W}(B)$:**

- depth $= L \in \mathbb{N}^*$
- width $= W \in \mathbb{N}^*$
- bound $B \geqslant 1$ on $\theta = (A_1, \ldots, A_L, b_1, \ldots, b_L) : \|A_\ell\|_2, \|b_\ell\|_2 \leqslant B$

# Contribution 1: explicit bounds on the Lipschitz parameterization of ReLU networks and its consequences

Under mild assumptions, there exists $c > 0$ s.t.

$$\boxed{\frac{1}{c} L B^{L-1} \leqslant K_{\Theta_{L,W}(B)} \leqslant c \underbrace{WL}_{\textbf{can be improved?}} L B^{L-1}}$$

**Definition of $\Theta_{L,W}(B)$:**

- depth $= L \in \mathbb{N}^*$
- width $= W \in \mathbb{N}^*$
- bound $B \geqslant 1$ on $\theta = (A_1, \dots, A_L, b_1, \dots, b_L)$ : $\|A_\ell\|_2, \|b_\ell\|_2 \leqslant B$

# Setup to get guarantees on quantization with our results

**Recipe for quantization guarantees of $\theta$**

# Setup to get guarantees on quantization with our results

**Recipe for quantization guarantees of $\theta$**

**Fixed:**

- depth $= L$
- width $= W$
- bound $B$ on the parameters
- desired quantization error $\varepsilon > 0$

# Setup to get guarantees on quantization with our results

**Recipe for quantization guarantees of $\theta$**

**Fixed:**

- depth $= L$
- width $= W$
- bound $B$ on the parameters
- desired quantization error $\varepsilon > 0$

---

---

**Result:** necessary and sufficient number of bits/weight to get $\varepsilon$-quantization error on $\Theta_{L,W}(B)$

# Setup to get guarantees on quantization with our results

**Recipe for quantization guarantees of $\theta$**

**Fixed:**

- depth $= L$
- width $= W$
- bound $B$ on the parameters
- desired quantization error $\varepsilon > 0$

---

**Ingredient provided by our work:** bounds on $K_{\Theta_{L,W}(B)}$

---

**Result:** necessary and sufficient number of bits/weight to get $\varepsilon$-quantization error on $\Theta_{L,W}(B)$

# Consequence 1: $\varepsilon$-accuracy with naive uniform quantization

**Fixed:** depth $L$, width $W$, bound $B$ on the parameters, desired quantization error $\varepsilon > 0$

**Look for:** smallest number of bits/weight to provide $\varepsilon$-error with $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise:

$$\|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon.$$

# Consequence 1: $\varepsilon$-accuracy with naive uniform quantization

**Fixed:** depth $L$, width $W$, bound $B$ on the parameters, desired quantization error $\varepsilon > 0$

**Look for:** smallest number of bits/weight to provide $\varepsilon$-error with $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise:

$$\max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon.$$

# Consequence 1: $\varepsilon$-accuracy with naive uniform quantization

**Fixed:** depth $L$, width $W$, bound $B$ on the parameters, desired quantization error $\varepsilon > 0$

**Look for:** smallest number of bits/weight to provide $\varepsilon$-error with $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise:

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon.$$

# Consequence 1: $\varepsilon$-accuracy with naive uniform quantization

**Fixed:** depth $L$, width $W$, bound $B$ on the parameters, desired quantization error $\varepsilon > 0$

**Look for:** smallest number of bits/weight to provide $\varepsilon$-error with $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise:

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \| R_\theta(x) - R_{Q_\eta(\theta)}(x) \|_p \leqslant \varepsilon.$$

**Necessary condition?**

# Consequence 1: $\varepsilon$-accuracy with naive uniform quantization

**Fixed:** depth $L$, width $W$, bound $B$ on the parameters, desired quantization error $\varepsilon > 0$

**Look for:** smallest number of bits/weight to provide $\varepsilon$-error with $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise:

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon.$$

**Necessary condition?**

$$\boxed{\textbf{Number of bits/weight} \propto \log(1/\varepsilon) + L \log(B)}$$

# Consequence 1: $\varepsilon$-accuracy with naive uniform quantization

**Fixed:** depth $L$, width $W$, bound $B$ on the parameters, desired quantization error $\varepsilon > 0$

**Look for:** smallest number of bits/weight to provide $\varepsilon$-error with $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise:

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon.$$

**Necessary condition?**

$$\boxed{\textbf{Number of bits/weight} \propto \log(1/\varepsilon) + L \log(B)}$$

**Sufficient condition?**

# Consequence 1: $\varepsilon$-accuracy with naive uniform quantization

**Fixed:** depth $L$, width $W$, bound $B$ on the parameters, desired quantization error $\varepsilon > 0$

**Look for:** smallest number of bits/weight to provide $\varepsilon$-error with $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise:

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon.$$

**Necessary condition?**

$$\boxed{\textbf{Number of bits/weight} \propto \log(1/\varepsilon) + L\log(B)}$$

**Sufficient condition?**

$$\boxed{\textbf{Number of bits/weight} \propto \log(1/\varepsilon) + L\log(B) + \underbrace{\log(WL)}_{\textbf{can be improved?}}}$$

# Consequence 1: $\varepsilon$-accuracy with naive uniform quantization

**Fixed:** depth $L$, width $W$, bound $B$ on the parameters, desired quantization error $\varepsilon > 0$

**Look for:** smallest number of bits/weight to provide $\varepsilon$-error with $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise:

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon.$$

**Necessary condition?**

$$\boxed{\textbf{Number of bits/weight} \propto \log(1/\varepsilon) + L\log(B)}$$

**Sufficient condition?**

$$\boxed{\textbf{Number of bits/weight} \propto \log(1/\varepsilon) + L\log(B) + \underbrace{\log(WL)}_{\textbf{can be improved?}}}$$

Improvement of the multiplicative constant compared to a known result[2]

[2] Deep Neural Network Approximation Theory. D. Elbrächter et al. 2021.

# Consequence 2: approximation with quantized networks

**Recipe for quantized approximation of $f$**

# Consequence 2: approximation with quantized networks

**Recipe for quantized approximation of $f$**

**Fixed:** quantization error $\varepsilon > 0$, function $f$

---

---

# Consequence 2: approximation with quantized networks

**Recipe for quantized approximation of $f$**

**Fixed:** quantization error $\varepsilon > 0$, function $f$

---

---

**Result:** depth $L$, width $W$, bound $B$ on the parameters, and number of bits/weight to get error $\varepsilon$

# Consequence 2: approximation with quantized networks

**Recipe for quantized approximation of $f$**

**Fixed:** quantization error $\varepsilon > 0$, function $f$

---

**Ingredient independent of our work:** depth $L$, width $W$, bound $B$ and $\theta \in \Theta_{L,W}(B)$ s.t. $\|f - R_\theta\|_p \leqslant \varepsilon/2$

---

**Result:** depth $L$, width $W$, bound $B$ on the parameters, and number of bits/weight to get error $\varepsilon$

# Consequence 2: approximation with quantized networks

**Recipe for quantized approximation of $f$**

**Fixed:** quantization error $\varepsilon > 0$, function $f$

---

**Ingredient independent of our work:** depth $L$, width $W$, bound $B$ and $\theta \in \Theta_{L,W}(B)$ s.t. $\|f - R_\theta\|_p \leqslant \varepsilon/2$
**Ingredient provided by our work:** Lipschitz bound to quantize $\theta$ within error $\varepsilon/2$

---

**Result:** depth $L$, width $W$, bound $B$ on the parameters, and number of bits/weight to get error $\varepsilon$

# Consequence 2: approximation with quantized networks

**Recipe for quantized approximation of $f$**

**Fixed:** quantization error $\varepsilon > 0$, function $f$

---

**Ingredient independent of our work:** depth $L$, width $W$, bound $B$ and $\theta \in \Theta_{L,W}(B)$ s.t. $\|f - R_\theta\|_p \leqslant \varepsilon/2$
**Ingredient provided by our work:** Lipschitz bound to quantize $\theta$ within error $\varepsilon/2$

---

**Result:** depth $L$, width $W$, bound $B$ on the parameters, and number of bits/weight to get error $\varepsilon$

**Applying this recipe:** **Recovery of a known result**[3] in Sobolev spaces using an external ingredient [4]

[3] On the Universal Approximability and Complexity Bounds of Quantized ReLU Neural Networks. Y. Ding et al. 2019.

[4] Error bounds for approximations with deep ReLU networks. D. Yarotsky. 2017

# Consequence 3: approximation speed of quantized ReLU networks

**Consider:** Increasingly complex $\Sigma = (\Sigma_M)_{M \in \mathbb{N}}$ with weights in $\mathbb{R}$

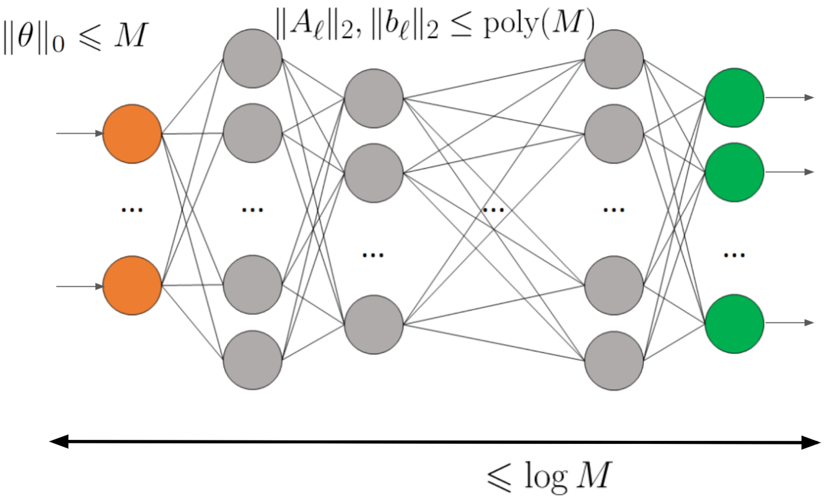
**Can we guarantee:**

$$\gamma^{*\text{approx}}(\mathcal{C}|\boldsymbol{Q(\Sigma)}) = \gamma^{*\text{approx}}(\mathcal{C}|\Sigma)\textbf{?}$$

Recall the definition:
$\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) :=$ largest $\gamma > 0$ s.t. $\sup_{f \in \mathcal{C}} d(f, \Sigma_M) \underset{M \to \infty}{=} \mathcal{O}(M^{-\gamma})$

# Consequence 3: same approximation speed with quantized networks

$(\log M)^2$ **bits per weight is enough** for $\Sigma_M :=$ functions represented by a ReLU network:



## Part 2: on a relation between approximation- and information-theoretic quantities

# Challenge: unified framework for a known inequality

**Known:** in **certain situations** (arbitrary $C$, sequence $\Sigma$: dictionaries[5], ReLU neural networks[6])

$$\underbrace{\gamma^{*\mathsf{approx}}(\mathcal{C}|\Sigma)}_{\text{approximation theory}} \leqslant \underbrace{\gamma^{*\mathsf{encod}}(\mathcal{C})}_{\text{information theory}}$$

$\gamma^{*\mathsf{encod}}(\mathcal{C})$ measures how well $\mathcal{C}$ can be encoded into bit sequences

**Challenge:** **Unified framework?**

[5] Optimally sparse data representations. P. Grohs. 2015

[6] Deep Neural Network Approximation Theory. D. Elbrächter et al. 2021

# We need to control the increasing complexity

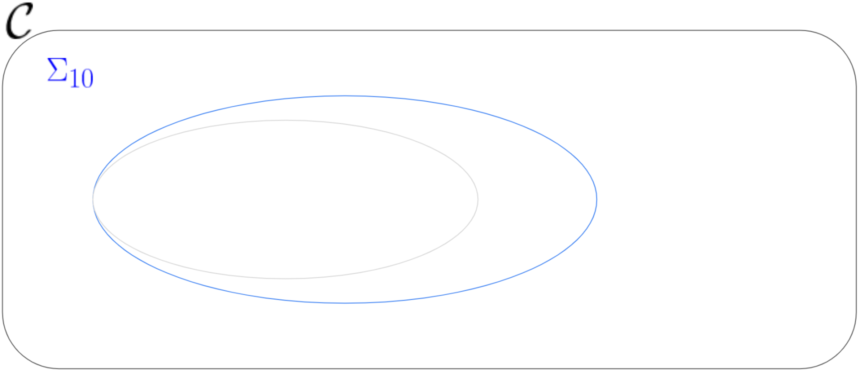
**Can we:** identify a property of $\Sigma$ to guarantee $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\text{encod}}(\mathcal{C})$

# We need to control the increasing complexity

**Can we:** identify a property of $\Sigma$ to guarantee $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\text{encod}}(\mathcal{C})$

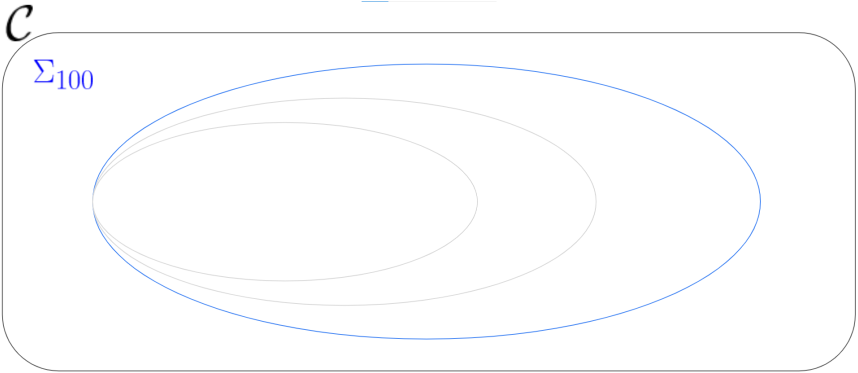
# We need to control the increasing complexity

**Can we:** identify a property of $\Sigma$ to guarantee $\gamma^{*\mathsf{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\mathsf{encod}}(\mathcal{C})$



# We need to control the increasing complexity

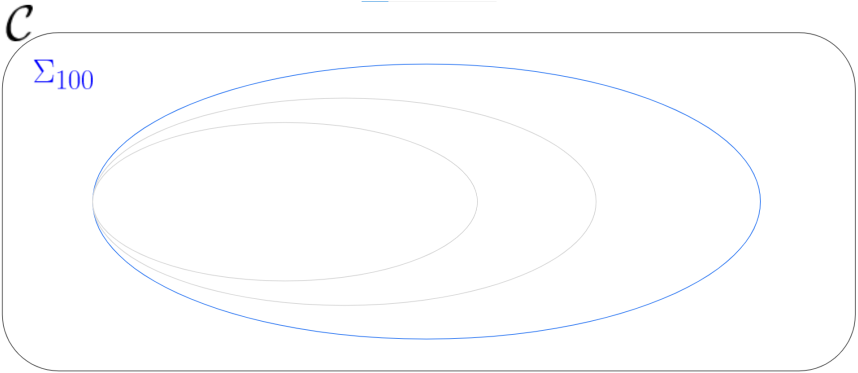
**Can we:** identify a property of $\Sigma$ to guarantee $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\text{encod}}(\mathcal{C})$




**Observation:** intuitively, if $\Sigma_M$ "grows" too fast then $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma)$ may be unreasonably large

# We need to control the increasing complexity

**Can we:** identify a property of $\Sigma$ to guarantee $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\text{encod}}(\mathcal{C})$

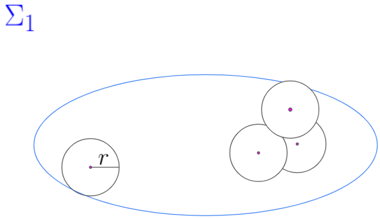



**Observation:** intuitively, if $\Sigma_M$ "grows" too fast then $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma)$ may be unreasonably large

Extreme case : $\Sigma_1 = \cdots = \Sigma_M = \cdots = \mathcal{C}$ where $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) = \infty$

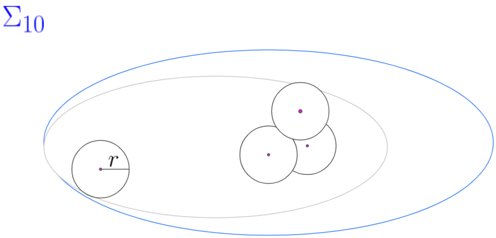
# Contribution 2: encodability property that allows unification

**A way to measure the "size" of $\Sigma_M$:** covering numbers



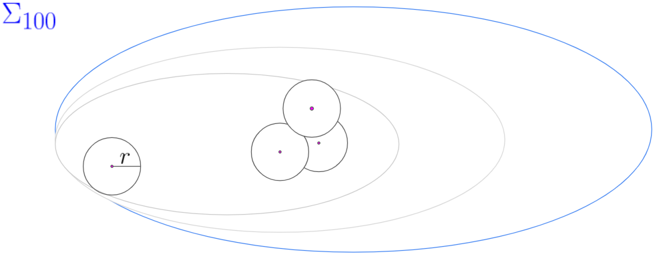
# Contribution 2: encodability property that allows unification

**A way to measure the "size" of $\Sigma_M$:** covering numbers



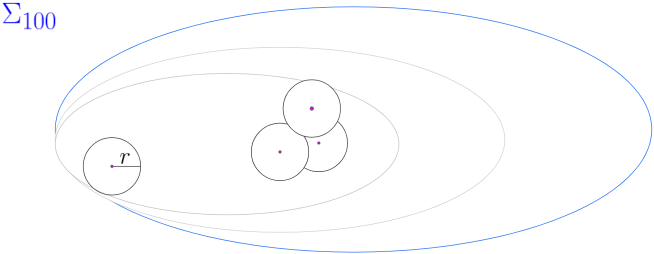
# Contribution 2: encodability property that allows unification

**A way to measure the "size" of $\Sigma_M$:** covering numbers



# Contribution 2: encodability property that allows unification

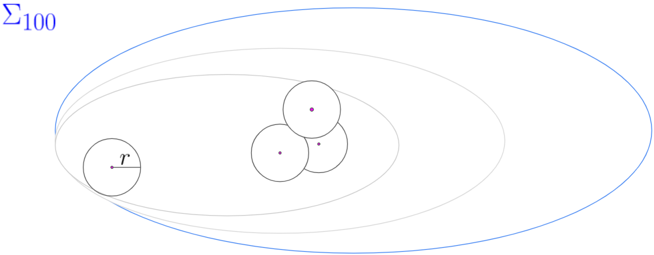
**A way to measure the "size" of $\Sigma_M$:** covering numbers




**Introduced property:** **controlled growth (with respect to $M$) of the covering numbers** of $\Sigma_M$

# Contribution 2: encodability property that allows unification

**A way to measure the "size" of $\Sigma_M$:** covering numbers




**Introduced property: controlled growth (with respect to $M$) of the covering numbers** of $\Sigma_M$

**Unification with this encodability property**

# Contribution 2: recovery and generalization of $\gamma^{*\mathrm{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\mathrm{encod}}(\mathcal{C})$

1. encodability property $\Longrightarrow \gamma^{*\mathrm{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\mathrm{encod}}(\mathcal{C})$ for every $\mathcal{C}$

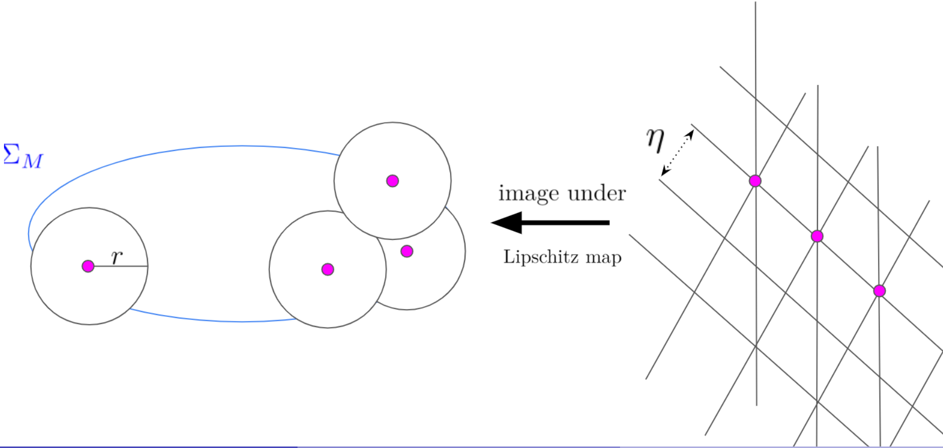
# Contribution 2: recovery and generalization of $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\text{encod}}(\mathcal{C})$

1. encodability property $\Longrightarrow \gamma^{*\text{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\text{encod}}(\mathcal{C})$ for every $\mathcal{C}$
2. satisfied for $\Sigma$: dictionaries, ReLU neural networks

# Contribution 2: recovery and generalization of $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\text{encod}}(\mathcal{C})$

1. encodability property $\Longrightarrow \gamma^{*\text{approx}}(\mathcal{C}|\Sigma) \leqslant \gamma^{*\text{encod}}(\mathcal{C})$ for every $\mathcal{C}$
2. satisfied for $\Sigma$: dictionaries, ReLU neural networks

**Proof of the encodability property for ReLU networks:**



# Summary and perspectives

# Conclusion

1. **Quantization vs. approximation**

# Conclusion

1. **Quantization vs. approximation**

**New key tool: Numbers of bits vs. quantization error controlled** via new explicit Lipschitz bounds

# Conclusion

1. **Quantization vs. approximation**

**New key tool: Numbers of bits vs. quantization error controlled** via new explicit Lipschitz bounds

**New consequences:**

- **characterize $\varepsilon$-quantization error** for $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise

# Conclusion

1. **Quantization vs. approximation**

**New key tool: Numbers of bits vs. quantization error controlled** via new explicit Lipschitz bounds

**New consequences:**

- **characterize $\varepsilon$-quantization error** for $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise
- sufficient number of bits/weight to keep **same approximation speed with quantized networks** for every function class $\mathcal{C}$

# Conclusion

1. **Quantization vs. approximation**

**New key tool: Numbers of bits vs. quantization error controlled** via new explicit Lipschitz bounds

**New consequences:**

- **characterize $\varepsilon$-quantization error** for $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise
- sufficient number of bits/weight to keep **same approximation speed with quantized networks** for every function class $\mathcal{C}$

2. **Relation between approximation- and information-theory**

# Conclusion

1. **Quantization vs. approximation**

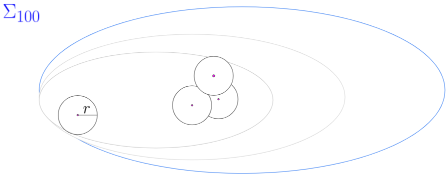
**New key tool: Numbers of bits vs. quantization error controlled** via new explicit Lipschitz bounds

**New consequences:**

- **characterize $\varepsilon$-quantization error** for $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise
- sufficient number of bits/weight to keep **same approximation speed with quantized networks** for every function class $\mathcal{C}$

2. **Relation between approximation- and information-theory**

**New key tool: Encodability property**



# Conclusion

1. **Quantization vs. approximation**

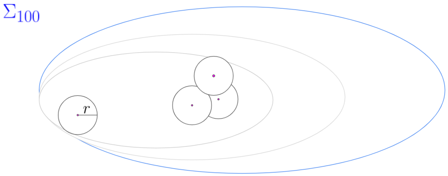
**New key tool: Numbers of bits vs. quantization error controlled** via new explicit Lipschitz bounds

**New consequences:**

- **characterize $\varepsilon$-quantization error** for $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ applied coordinatewise
- sufficient number of bits/weight to keep **same approximation speed with quantized networks** for every function class $\mathcal{C}$

2. **Relation between approximation- and information-theory**

**New key tool: Encodability property**




**New consequence: Unifies** situations where $\gamma^{*\text{approx}}(\mathcal{C}|\Sigma) \leqslant \text{complexity}(\mathcal{C})$

# Possible future directions on quantization

**Current theory = number of bits/weight must be linear in the depth:** if $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ gives $\varepsilon$-quantization error

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon$$

**Practice = 1 bit/weight is enough[7]:** quantization-aware training, same performance on MNIST for a 3 hidden-layers with 1024 neurons per layer

[7] BinaryConnect: Training Deep Neural Networks with binary weights during propagations. Courbariaux et al. 2015.

# Possible future directions on quantization

**Current theory = number of bits/weight must be linear in the depth:** if $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ gives $\varepsilon$-quantization error

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon$$

**Practice = 1 bit/weight is enough[7]:** quantization-aware training, same performance on MNIST for a 3 hidden-layers with 1024 neurons per layer

**Fill the gap theory/practice?**

[7] BinaryConnect: Training Deep Neural Networks with binary weights during propagations. Courbariaux et al. 2015.

# Possible future directions on quantization

**Current theory = number of bits/weight must be linear in the depth:** if $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ gives $\varepsilon$-quantization error

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon$$

**Practice = 1 bit/weight is enough[7]:** quantization-aware training, same performance on MNIST for a 3 hidden-layers with 1024 neurons per layer

**Fill the gap theory/practice?**

- Less naive quantization?
- $\varepsilon$-quantization error on a smaller set $\Theta \subset \Theta_{L,W}(B)$? e.g., *parameters that can be learned in practice*
- not interested in every $\varepsilon > 0$?

[7] BinaryConnect: Training Deep Neural Networks with binary weights during propagations. Courbariaux et al. 2015.

# Possible future directions on quantization

**Current theory = number of bits/weight must be linear in the depth:** if $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ gives $\varepsilon$-quantization error

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon$$

**Practice = 1 bit/weight is enough[7]:** quantization-aware training, same performance on MNIST for a 3 hidden-layers with 1024 neurons per layer

**Fill the gap theory/practice?**

- Less naive quantization?
- $\varepsilon$-quantization error on a smaller set $\Theta \subset \Theta_{L,W}(B)$? e.g., *parameters that can be learned in practice*
- not interested in every $\varepsilon > 0$?

More on agonon.github.io

[7]BinaryConnect: Training Deep Neural Networks with binary weights during propagations. Courbariaux et al. 2015.

# Possible future directions on quantization

**Current theory = number of bits/weight must be linear in the depth:** if $Q_\eta(x) = \lfloor x/\eta \rfloor \eta$ gives $\varepsilon$-quantization error

$$\max_{\theta \in \Theta_{L,W}(B)} \max_{x \in [0,1]^d} \|R_\theta(x) - R_{Q_\eta(\theta)}(x)\|_p \leqslant \varepsilon$$

**Practice = 1 bit/weight is enough**[7]**:** quantization-aware training, same performance on MNIST for a 3 hidden-layers with 1024 neurons per layer

**Fill the gap theory/practice?**

- Less naive quantization?
- $\varepsilon$-quantization error on a smaller set $\Theta \subset \Theta_{L,W}(B)$? e.g., *parameters that can be learned in practice*
- not interested in every $\varepsilon > 0$?

More on agonon.github.io

**Thank you!**

[7] BinaryConnect: Training Deep Neural Networks with binary weights during propagations. Courbariaux et al. 2015.

# Definition of the encoding speed

$$L(\varepsilon, \mathcal{C}) := \inf \Big\{ \ell \in \mathbb{N}, \exists E : \mathcal{C} \mapsto \{0,1\}^\ell, \\ \exists D : \{0,1\}^\ell \mapsto \mathcal{F}, \sup_{f \in \mathcal{C}} d\left(f, D(E(f))\right) \leqslant \varepsilon \Big\}$$

$$\gamma^{*\mathrm{encod}}(\mathcal{C}) := \sup \left\{ \gamma > 0, L(\varepsilon, \mathcal{C}) = \mathcal{O}_{\varepsilon \to 0}\left(\varepsilon^{-1/\gamma}\right) \right\}.$$

# Known approximation speeds[8]

| $\mathcal{C}$ := unit ball of | | $\Sigma$ | $\gamma^{*\text{approx}}(\mathcal{C}\|\Sigma) = \gamma^{*\text{encod}}(\mathcal{C})$ |
|---|---|---|---|
| $\alpha$-Hölder | $C^\alpha([0,1])$ | Wavelet basis | $\alpha$ |
| $L^p$-Sobolev[a] | $W_p^m([0,1]^d)$ | Wavelet frame | $\frac{m}{d}$ |
| Besov[b] | $B_{p,q}^m([0,1]^d)$ | Wavelet frame | $\frac{m}{d}$ |

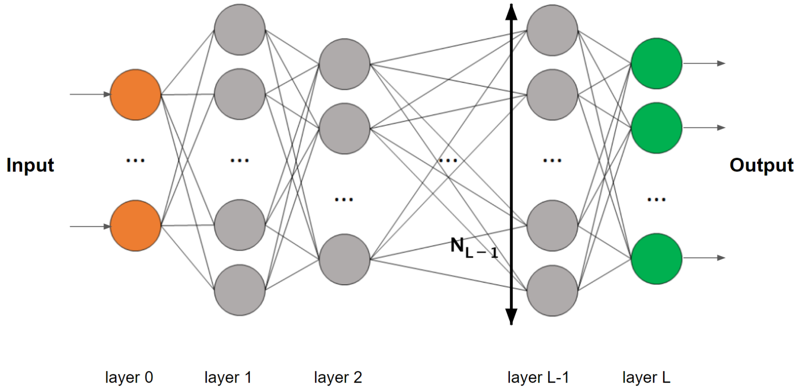
[a] where $p \in [1,\infty]$, $m > d\max(1/p - 1/2, 0)$
[b] where $p, q \in (0,\infty]$, $m > d\max(1/p - 1/2, 0)$

[8] Deep Neural Network Approximation Theory. D. Elbrächter et al. 2021.

# Notations : ReLU neural networks

**Architecture:** $(L, \mathbf{N})$ where

- $L \in \mathbb{N}$ is the number of layers : the depth
- $\mathbf{N} = (N_0, \ldots, N_L) \in \mathbb{N}^{L+1}$ with $N_\ell$ the width (number of neurons) of layer $\ell$



# Notations : ReLU neural networks

**Architecture:** $(L, \mathbf{N})$ where

- $L \in \mathbb{N}$ is the number of layers : the depth
- $\mathbf{N} = (N_0, \ldots, N_L) \in \mathbb{N}^{L+1}$ with $N_\ell$ the width (number of neurons) of layer $\ell$

**Parameters:** $\theta = (A_1, \ldots, A_L, b_1, \ldots, b_L)$ with $A_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $b_\ell \in \mathbb{R}^\ell$

# Notations : ReLU neural networks

**Architecture:** $(L, \mathbf{N})$ where

- $L \in \mathbb{N}$ is the number of layers : the depth
- $\mathbf{N} = (N_0, \ldots, N_L) \in \mathbb{N}^{L+1}$ with $N_\ell$ the width (number of neurons) of layer $\ell$

**Parameters:** $\theta = (A_1, \ldots, A_L, b_1, \ldots, b_L)$ with $A_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $b_\ell \in \mathbb{R}^\ell$

**Realization of the network:** $\boxed{R_\theta : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L} \text{ the realization of } \theta\text{:}}$

$$R_\theta(x) = A_L\rho(\cdots(A_2\rho(A_1x + b_1) + b_2)\cdots) + b_L \text{ avec } \rho(x) = \max(0, x).$$