Neural Architecture Growth using Tangent Space

Manon Verbockhaven, Guillaume Charpiat
prenom.nom@inria.fr

TAU team, LISN, INRIA Saclay, Université Paris-Saclay

Avril 2023

Plan

- Introduction, common pratices
- Expressivity bottlenecks : definition and formula
- Best parameters move at each layer
- Best neurons to add
- Resultats
- Conclusion

I - Introduction

Introduction

- Medium architecture : approximate any function. (*Universal approximation theorem*)
- Huge architecture : good minimum following the gradient descent. (*Neural Tangent Kernel*)

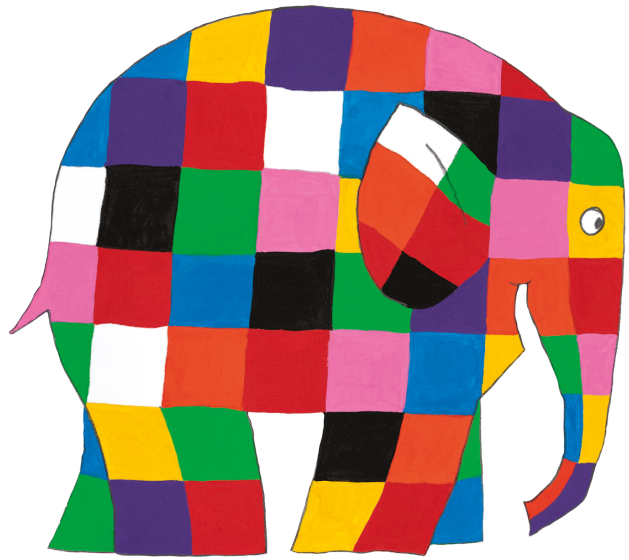
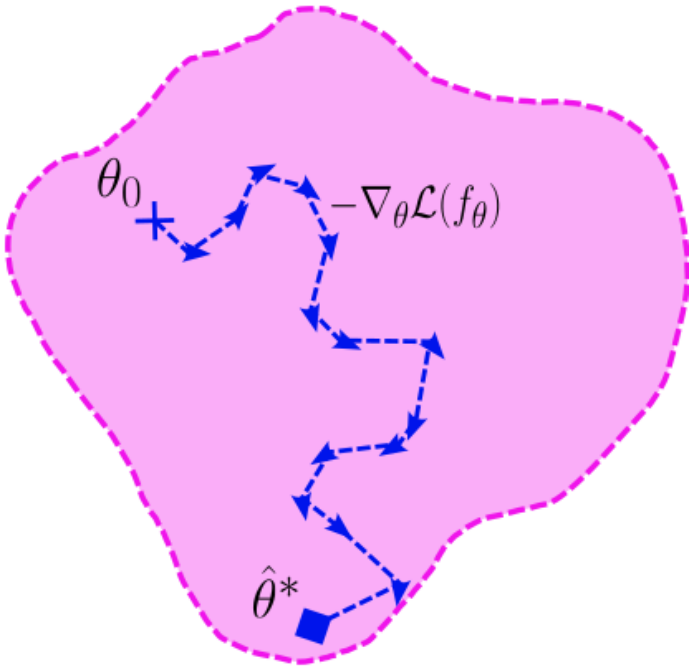


Figure: Neural Network with huge architecture

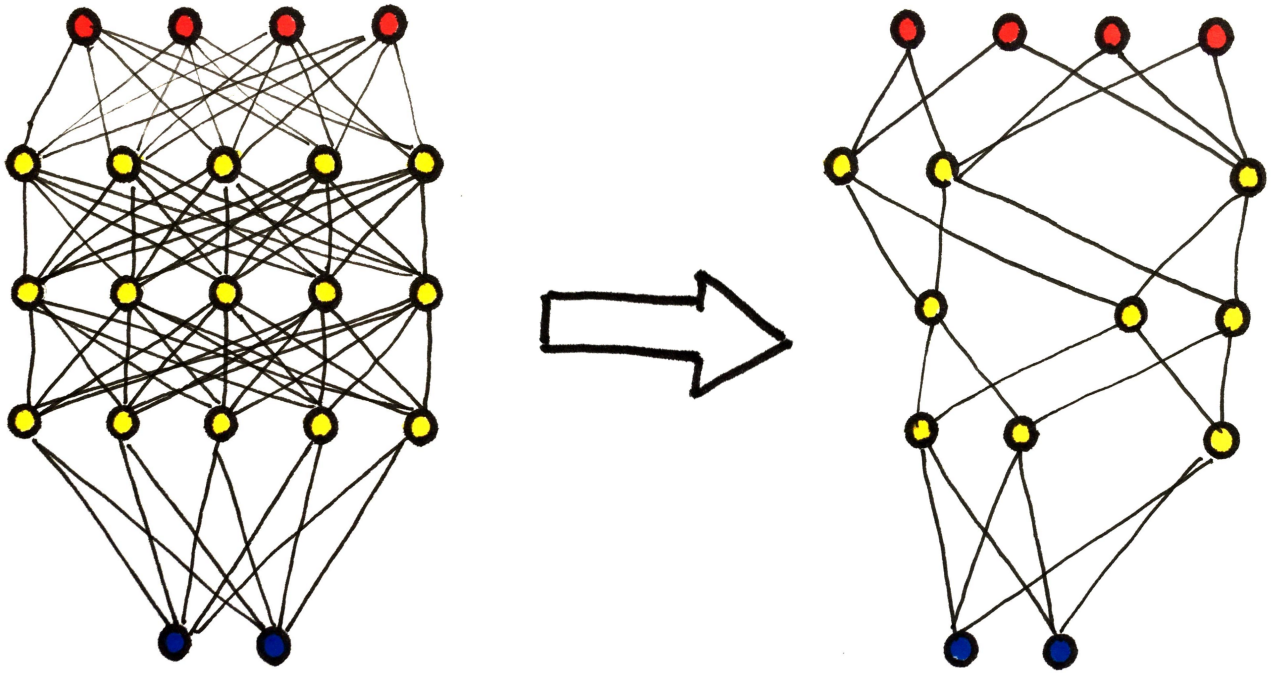
Optimizing NN



$$|\mathcal{L}(\hat{\theta}^*) - \mathcal{L}(\theta^*)| \to 0$$

Figure: Following the gradient descent on the set of parameters

pruning



large final	small final	large init	small init	large init large final	small init small final	magnitude increase	movement	random
$\|w_f\|$	$-\|w_f\|$	$\|w_i\|$	$-\|w_i\|$	$min(\alpha\|w_f\|, \|w_i\|)$	$-max(\alpha\|w_f\|, \|w_i\|)$	$\|w_f\| - \|w_i\|$	$\|w_f - w_i\|$	0

NAS : Neural Architecture Search

from scratch
Human design
random search
gradient bases method [1]

meta DL
reinforcement learning [2]
multi-task learning
genetic methods [3]

Sensitive to exploration hyper-parameters, need a lot of computational resources.

1 : GradMax: Growing Neural Networks using Gradient Information, Google Research, Brain Team
2 : Neural Architecture Search with Reinforcement Learning, Google Brain
3 : Compositional pattern producing networks: A novel abstraction of development, Stanley

Starting with a small neural network

Training a small network:

- faster learning, less memory
- the solution found by gradient descent is poor

Adapt the architecture:

- search for architecture usually done in a discrete way (trial/error)

We propose: to adapt the architecture during learning:

- estimate and localize the potential **expressivity Bottlenecks**, and fill them, without trial/error.

II - Expressivity Bottlenecks, definition and formula

Definitions, Goals, Objectives

For the NN $\mathcal{N}$ with the architecture $\mathcal{A}$:

- What are *expressivity bottlenecks* ?
  - Where are they ?
  - How to quantify them ?

Optimization Problem

Neural Network Settings

- Dataset $\mathcal{D} := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N \in \left(\mathbb{R}^p \times \mathbb{R}^d\right)^N iid \sim \mathcal{P}$
- Neural Network $f_\theta : \mathbb{R}^p \to \mathbb{R}^d$
- Loss function $\mathcal{L} : \left(\mathbb{R}^d\right)^2 \to \mathbb{R}^+$

Deplacement

Definition (desired update for the function f_θ)

For $\eta > 0$, we define :

$$v_{\text{goal}}(.) := -\eta \, \nabla_{f_\theta(.)} \mathcal{L}(f_\theta)$$
$$v_{\text{goal}}(x) := -\eta \, \nabla_{f_\theta(x)} \mathcal{L}(f_\theta(x), y) \qquad \text{for } (x, y) \sim \mathcal{P}$$

Definition (possible update for the function f_θ)

For $\delta\theta$ an incremental direction for the parameters θ, we define :

$$v(., \delta\theta) := \frac{\partial f_\theta(.)}{\partial \theta} \, \delta\theta$$
$$v(x, \delta\theta) := \frac{\partial f_\theta(x)}{\partial \theta} \, \delta\theta$$

Where $\frac{\partial f}{\partial \theta}$ is the Jacobian matrix.

Example

Example (Least Square regression)

Using $\mathcal{L}(f_\theta(\boldsymbol{x}), \boldsymbol{y}) := ||\theta^T \boldsymbol{x} - \boldsymbol{y}||^2$:

$$\begin{aligned} \boldsymbol{v}_{\text{goal}}(\boldsymbol{x}) &:= -\eta \nabla_{\theta^T \boldsymbol{x}} \mathcal{L}(\theta^T \boldsymbol{x}, \boldsymbol{y}) \\ &= -\eta\, 2(\theta^T \boldsymbol{x} - \boldsymbol{y}) \\ \boldsymbol{v}(\boldsymbol{x}, \delta\theta) &= (\delta\theta)^T \boldsymbol{x} \end{aligned}$$

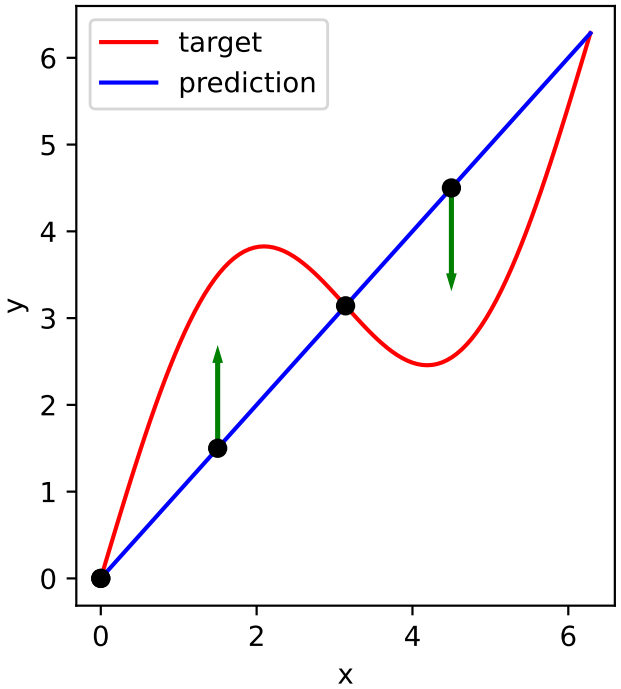
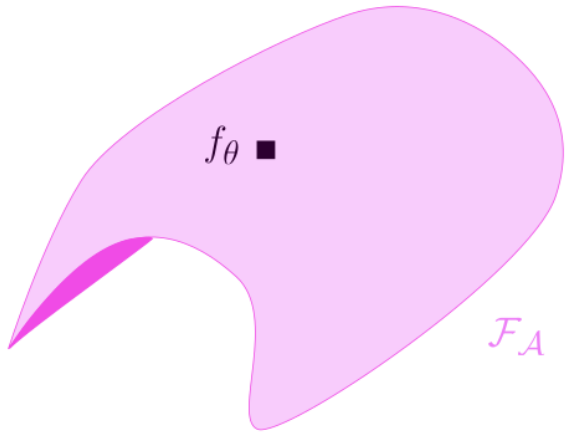


Figure: Linear prediction for a sinusoidal output

The Theory

mathematical objects

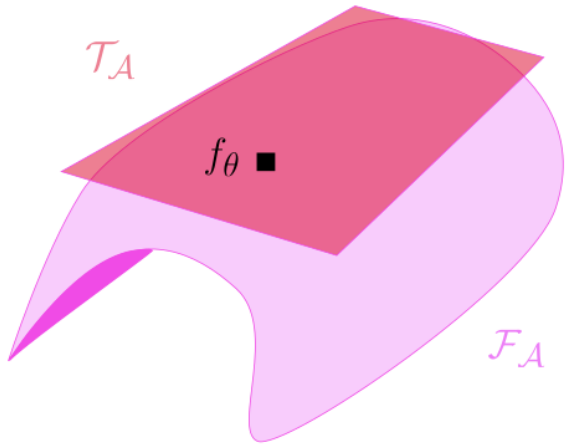
- Architecture space : $\Theta_{\mathcal{A}}$
- $\mathcal{F}_{\mathcal{A}} = \{f_\theta \mid \theta \in \Theta_{\mathcal{A}}\}$
-



The Theory

mathematical objects

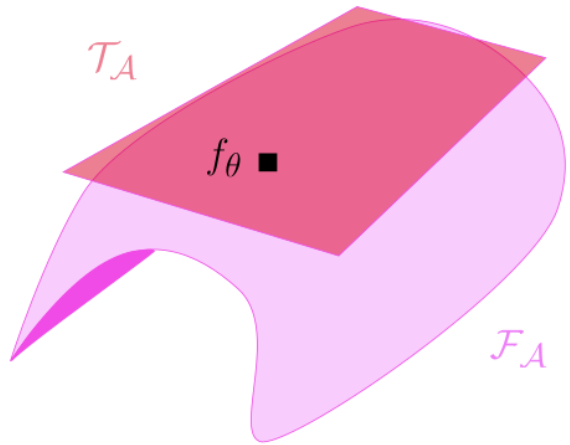
- Architecture space : $\Theta_{\mathcal{A}}$
- $\mathcal{F}_{\mathcal{A}} = \{f_\theta \mid \theta \in \Theta_{\mathcal{A}}\}$
- $\mathcal{T}_{\mathcal{A}}^{f_\theta} := \mathcal{T}_{\mathcal{A}} = \{\frac{\partial f_\theta}{\partial \theta}\ \delta\theta \mid \text{s.t. } \delta\theta \in \Theta\}$ ie Tangent space in f_θ



The Theory

mathematical objects

- Architecture space : $\Theta_{\mathcal{A}}$
- $\mathcal{F}_{\mathcal{A}} = \{f_\theta \mid \theta \in \Theta_{\mathcal{A}}\}$
- $\mathcal{T}_{\mathcal{A}}^{f_\theta} := \mathcal{T}_{\mathcal{A}} = \{\frac{\partial f_\theta}{\partial \theta}\ \delta\theta \mid \text{s.t. } \delta\theta \in \Theta\}$ ie Tangent space in f_θ

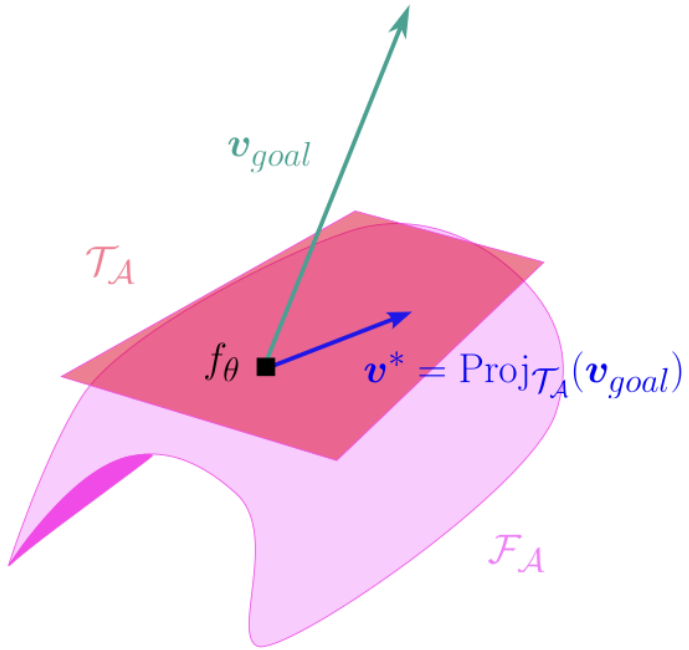


$$g \in \mathcal{T}_{\mathcal{A}} \iff \exists \delta\theta \ \ s.t. \ \ g(\boldsymbol{x}) = f_\theta(\boldsymbol{x}) + \frac{\partial f(\boldsymbol{x})}{\partial \theta}\ \delta\theta$$

The Theory

mathematical objects

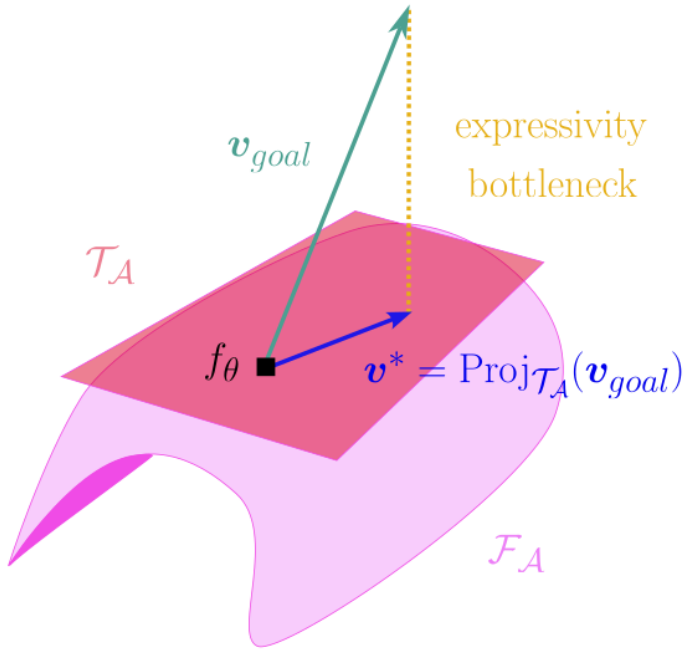
- Architecture space : $\Theta_{\mathcal{A}}$
- $\mathcal{F}_{\mathcal{A}} = \{f_\theta \mid \theta \in \Theta_{\mathcal{A}}\}$
- $\mathcal{T}_{\mathcal{A}}^{f_\theta} := \mathcal{T}_{\mathcal{A}} = \{\frac{\partial f_\theta}{\partial \theta}\ \delta\theta \mid \text{s.t. } \delta\theta \in \Theta\}$ ie Tangent space in f_θ



The Theory

mathematical objects

- Architecture space : $\Theta_{\mathcal{A}}$
- $\mathcal{F}_{\mathcal{A}} = \{f_\theta \mid \theta \in \Theta_{\mathcal{A}}\}$
- $\mathcal{T}_{\mathcal{A}}^{f_\theta} := \mathcal{T}_{\mathcal{A}} = \{\frac{\partial f_\theta}{\partial \theta}\ \delta\theta \mid \text{s.t. } \delta\theta \in \Theta\}$ ie Tangent space in f_θ



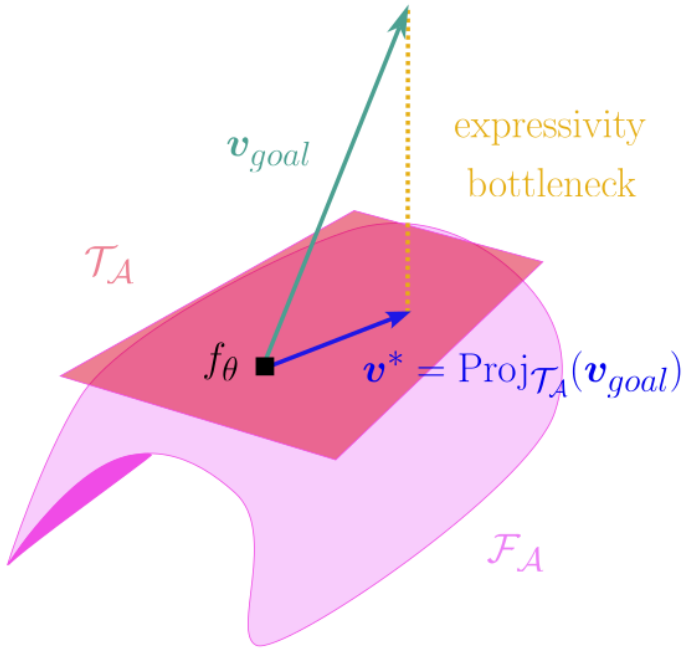
The Theory

Expressivity Bottleneck

$$\arg\min_{v \in \mathcal{T}_{\mathcal{A}}} \mathbb{E}_{(x,y)\sim\mathcal{P}} \left[||v_{\text{goal}}(x) - v(x)||^2 \right]$$

Expressivity Bottleneck approx.

$$\arg\min_{v \in \mathcal{T}_{\mathcal{A}}} ||V_{\text{goal}} - V||^2_{\text{Tr}}$$



$$V_{\text{goal}} = \begin{pmatrix} v_{\text{goal}}(x_1) & \dots & v_{\text{goal}}(x_n) \end{pmatrix} \text{ and } V = \begin{pmatrix} v(x_1) & \dots & v(x_n) \end{pmatrix}$$

Example

Example

$$f_\theta(x) := \sigma(\theta^T x) \quad \text{and} \quad \mathcal{L}(x, y) := ||y - f_\theta(x)||^2$$

$$v_{\text{goal}}(x) = -2(f_\theta(x) - y)$$

$$v(x, \delta\theta) = 2(\sigma(\theta^T x) - y)\sigma'(\theta^T x)x^T \delta\theta$$

$$\arg\min_{\delta\theta} \mathbb{E}_{\mathcal{P}}[||v_{\text{goal}}(x) - v(x, \delta\theta)||^2]$$

Notations

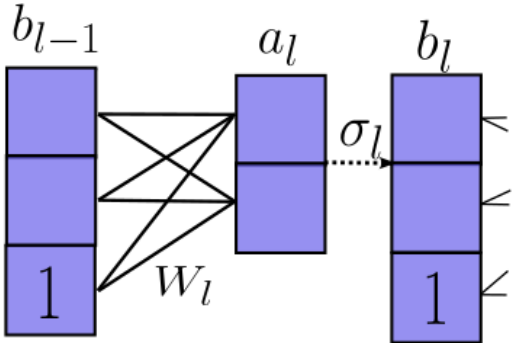


Figure: Feedforward NN

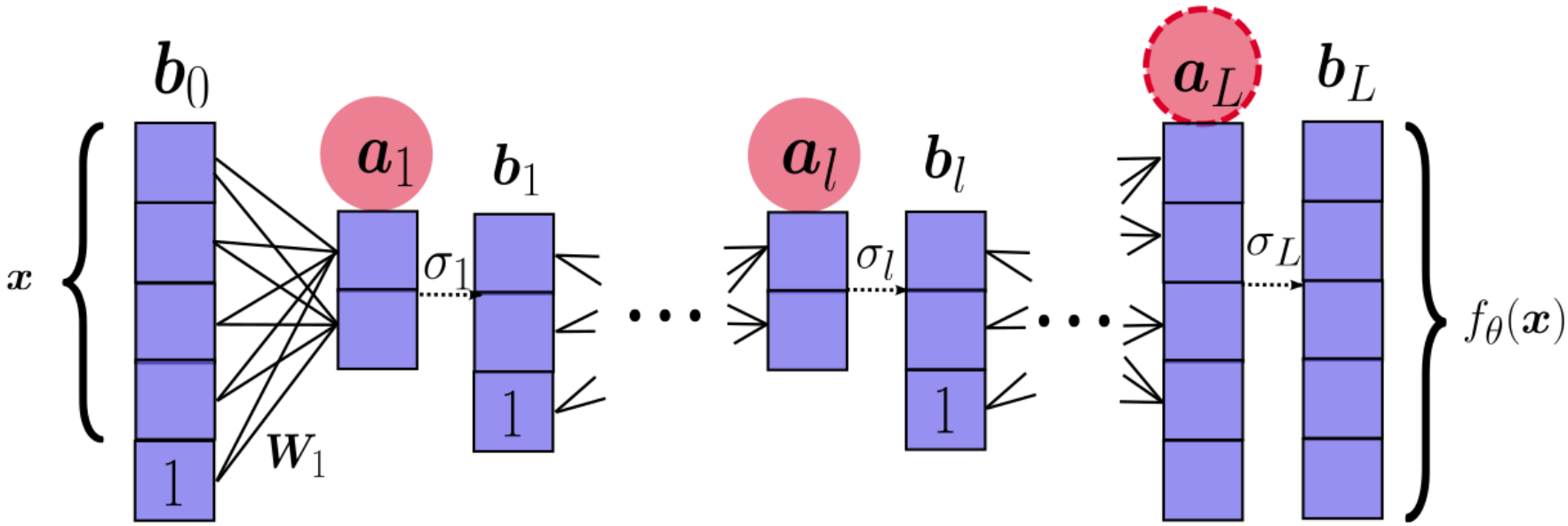
$$\boldsymbol{b}_0(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix}$$

$$\boldsymbol{a}_l(\boldsymbol{x}) = \boldsymbol{W}_l \boldsymbol{b}_{l-1}(\boldsymbol{x})$$

$$\boldsymbol{b}_l(\boldsymbol{x}) = \begin{pmatrix} \sigma_l(\boldsymbol{a}_l(\boldsymbol{x})) \\ 1 \end{pmatrix}$$

$$f_\theta(\boldsymbol{x}) = \sigma_L(\boldsymbol{a}_L(\boldsymbol{x}))$$

v and v_{goal} at layer l



Definition (v^l_{goal} desired update)

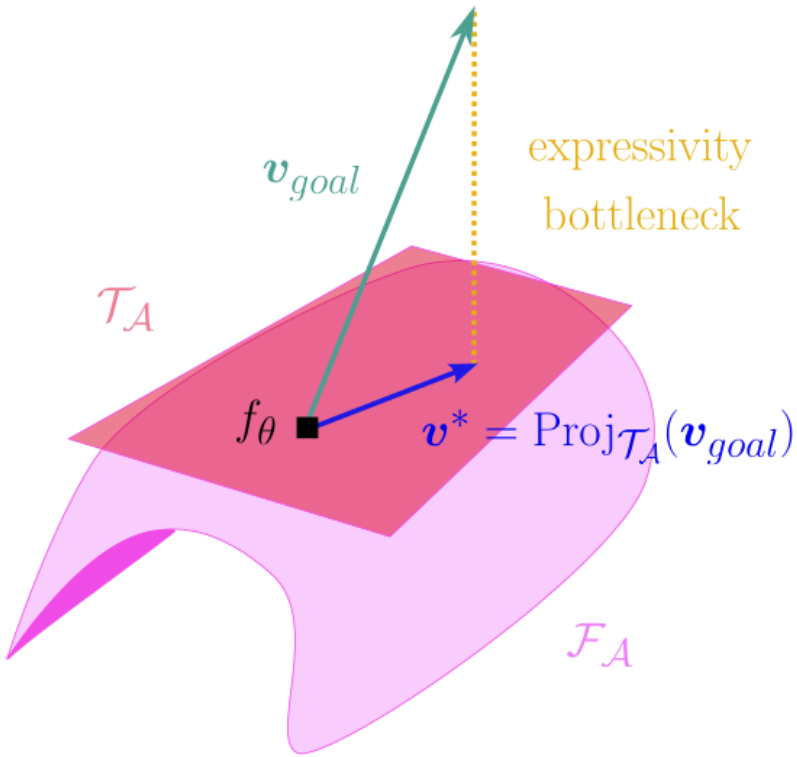
$$v_{goal}{}^l(x) := -\eta \nabla_{a_l(x)} \mathcal{L}(f_\theta(x), y)$$

Definition (v^l possible update)

$$v^l(x, \delta\theta) := \frac{\partial a_l(x)}{\partial \theta} \delta\theta$$

III - Best parameter move

v^*


$\boldsymbol{v}_{goal}$
expressivity
bottleneck
$\mathcal{T}_A$
f_θ
$\boldsymbol{v}^* = \mathrm{Proj}_{\mathcal{T}_A}(\boldsymbol{v}_{goal})$
$\mathcal{F}_A$

Before Fixing E.B

Non-convex optimization

Considering all the parameters :

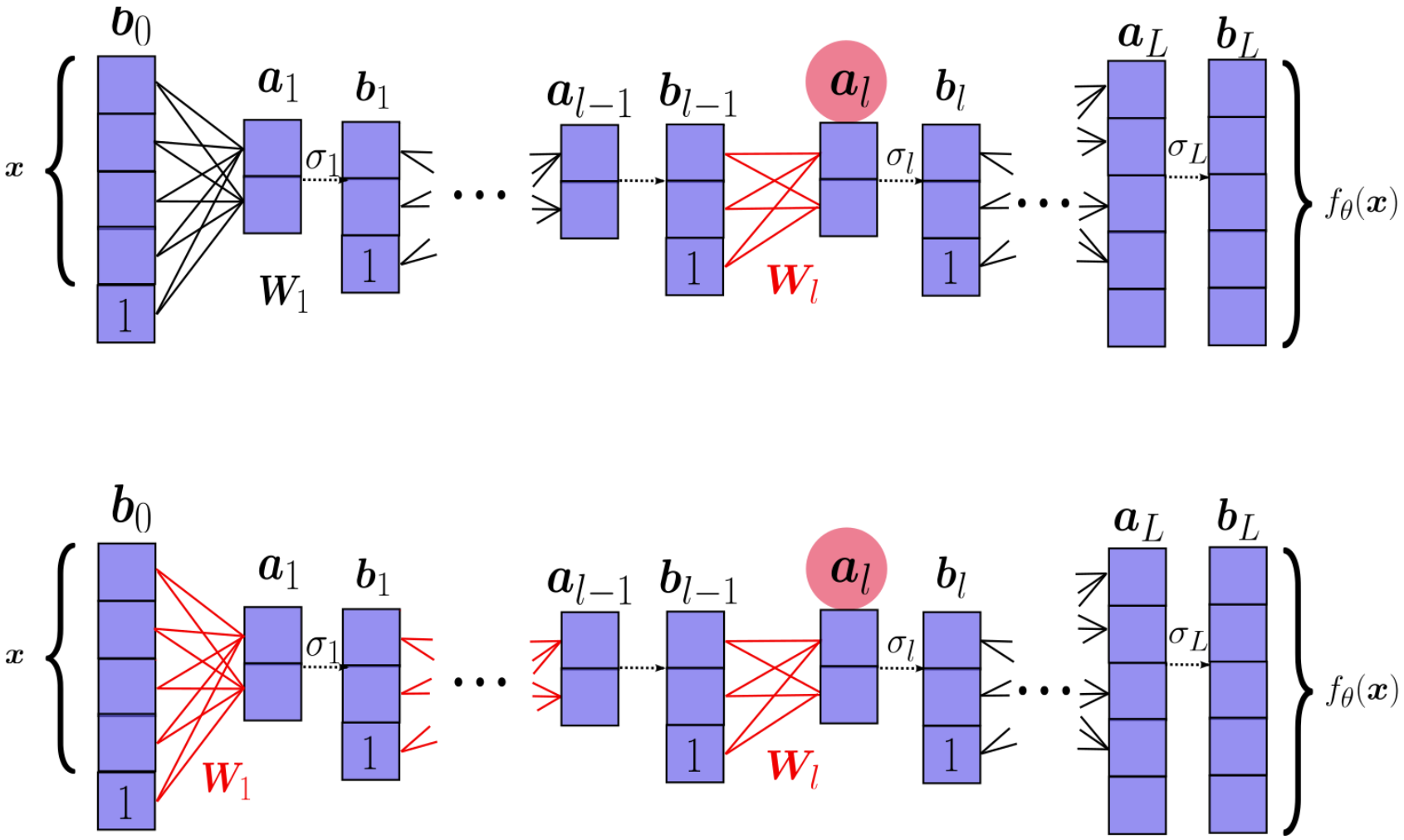
$$\delta\theta^* = \arg\min_{\delta\theta} ||\boldsymbol{V}_{\text{goal}}{}^{l} - \boldsymbol{V}(\delta\theta)||^2$$

Convex optimization

Considering only parameters at layer l :

$$\delta\boldsymbol{W}_l^* = \arg\min_{\delta\theta} ||\boldsymbol{V}_{\text{goal}}{}^{l} - \boldsymbol{V}(\delta\boldsymbol{W}_l)||^2$$

$\delta\theta^*$ vs δW_l^*



Convex optimization

Convex Optimization

Linear regression from $\boldsymbol{b}_{l-1}(\boldsymbol{x})$ to predict $\boldsymbol{v}^l_{\text{goal}}(\boldsymbol{x})$:

$$\delta \boldsymbol{W}_l^* := \arg\min_{\delta\theta} ||\boldsymbol{V}^l_{\text{goal}} - \boldsymbol{V}^l(\delta \boldsymbol{W}_l)||^2$$

$$\delta \boldsymbol{W}_l^* := \arg\min_{\delta\theta} ||\boldsymbol{V}^l_{\text{goal}} - \delta \boldsymbol{W}_l \boldsymbol{B}_{l-1}||^2$$

$$\delta \boldsymbol{W}_l^* = \frac{1}{n} \boldsymbol{V}^l_{\text{goal}} \boldsymbol{B}_{l-1}^T \left(\frac{1}{n} \boldsymbol{B}_{l-1} \boldsymbol{B}_{l-1}^T\right)^{-1}$$

$$\boldsymbol{B}_{l-1} := \begin{pmatrix} \boldsymbol{b}_{l-1}(\boldsymbol{x}_1) & \dots & \boldsymbol{b}_{l-1}(\boldsymbol{x}_n) \end{pmatrix}$$

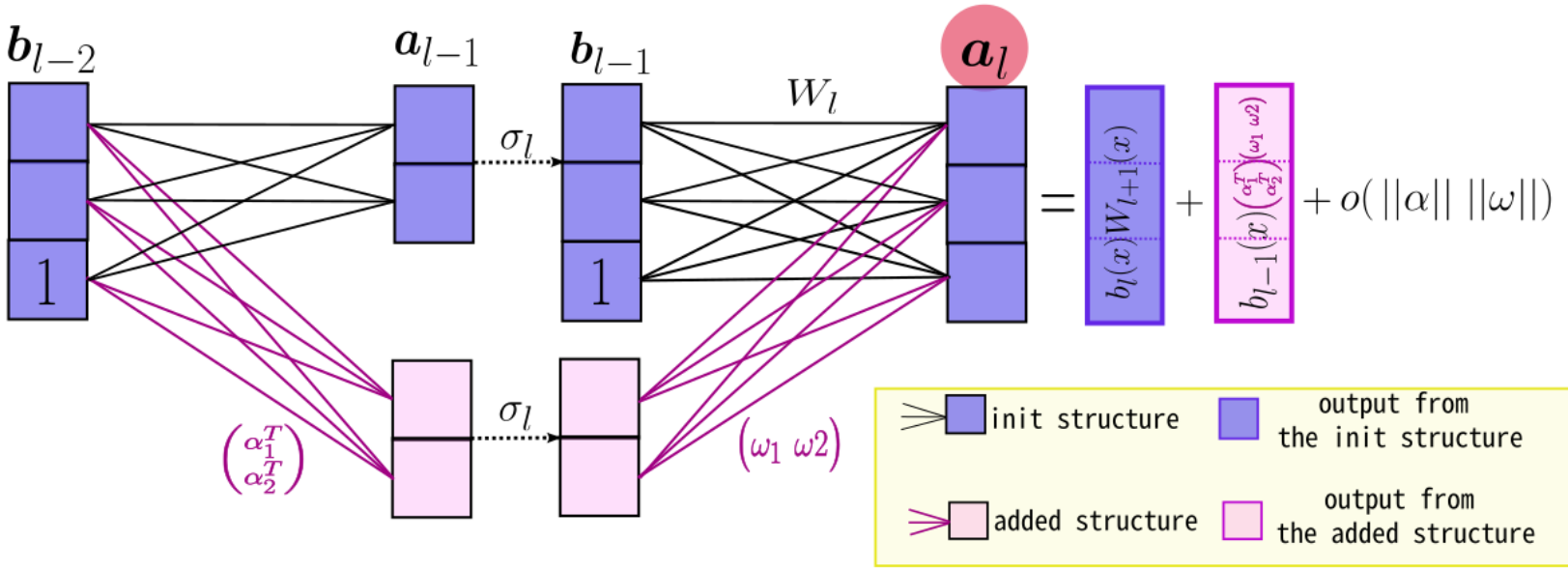
IV - Best neurons to add

Definitions, Goals, Objectives

For the NN $\mathcal{N}$ with the architecture $\mathcal{A}$:

- What is *expressivity bottlenecks* ?
- Where are they ?
- How to quantify them ?
- **Can we find a method to fix them ?**

Fix E.B and changing the structure



Definition ($V(\delta\theta^K_{l-1\leftrightarrow l})$)

$$\Delta a_l(x) := \sum_{k=1}^{K} \omega_k \sigma_l(\alpha_k^T b_{l-2}(x)) \quad \text{and} \quad v(x, \delta\theta^K_{l-1\leftrightarrow l}) := \sum_{k=1}^{K} \omega_k \left(\alpha_k^T b_{l-2}(x)\right)$$

Full Optimization Problem

The Optimal Neurons to reduce Expressivity Bottlenecks at layer l

The optimal neurons $n_1, ..., n_K$ are defined by $n_i := (\alpha_i, \omega_i)$ and are the solution of the optimization problem :

$$\underset{\delta W_l, K, \delta\theta^K_{l-1\leftrightarrow l}}{\arg\min} \overbrace{\left|\left| V^l_{\text{goal}} - \delta W_l B_{l-1} - \Omega A^T B_{l-2} \right|\right|^2_{\text{Tr}}}^{\text{Expressivity Bottleneck}}$$

$$\Omega := \begin{pmatrix} \omega_1 & ... & \omega_K \end{pmatrix} \text{ and } A := \begin{pmatrix} \alpha_1 & ... & \alpha_K \end{pmatrix}$$

Full Optimization Problem

The Optimal Neurons

The optimal neurons $n_1, ..., n_K$ are defined by $n_i := (\alpha_i, \omega_i)$ and are the solution of the optimization problem :

$$\underset{\delta W_l, \delta\theta^K_{l-1\leftrightarrow l}}{\arg\min} \left|\left| \overbrace{V^l_{\text{goal}} - \delta W_l B_{l-1}}^{V^l_{\text{goal}_{proj}}} - \Omega A^T B_{l-2} \right|\right|^2_{\text{Tr}}$$

$$\Omega := \begin{pmatrix} \omega_1 & ... & \omega_K \end{pmatrix} \text{ and } A := \begin{pmatrix} \alpha_1 & ... & \alpha_K \end{pmatrix}$$

Exact Solution

Theorem (Solution)

Define $\boldsymbol{S} := \frac{1}{n}\boldsymbol{B}_{l-2}\boldsymbol{B}_{l-2}^T$ *and* $\boldsymbol{N} := \frac{1}{n}\boldsymbol{B}_{l-2}\boldsymbol{V}^l_{goal_{proj}}$ *and* $\boldsymbol{S}^{1/2^{-1}}N = \sum_{c=1}^{R} \lambda_c \boldsymbol{u}_c \boldsymbol{v}_c^T$ *(SVD), then :*

$$\textit{maximum number of neuron to add} := R$$

$$\textit{initialization of new neuron } k,\ (\hat{\alpha}_k^*, \hat{\omega}_k^*) := (\boldsymbol{S}^{1/2^{T-1}} u_k, v_k)$$

Theorem

Theorem (Impact at first order on Loss)

In $\hat{\theta}^{K,*}_{l\leftrightarrow l+1} := (\hat{\alpha}^*_k, \hat{\omega}^*_k)_{k=1}^K$ *with* $K \leq K^* = R$, *then :*

$$\frac{1}{n}||\boldsymbol{V}_{goal}{}^{l+1}_{proj} - \boldsymbol{V}^{l+1}(\hat{\theta}^{K,*}_{l\leftrightarrow l+1})||^2_{\text{Tr}} = \frac{1}{n}||\boldsymbol{V}_{goal}{}^{l+1}_{proj}||^2_{\text{Tr}} - \boxed{\sum_{k=1}^{K} \lambda_k^2}$$

$$\mathcal{L}(f_{\theta \oplus \hat{\theta}^{K,*}_{l\leftrightarrow l+1}}) = \mathcal{L}(f_\theta) - \frac{\sigma_l'(0)}{\eta} \boxed{\sum_{k=1}^{K} \lambda_k^2} + o(||\hat{\theta}^{K,*}_{l\leftrightarrow l+1}||^2)$$

Intuition

Theorem (Loss variation)

Taylor at order 1 in $\boldsymbol{V}^{l+1}(\theta_{l\leftrightarrow l+1})$

$$\mathcal{L}(f_{\theta\oplus\theta^{K}_{l\leftrightarrow l+\mathbf{1}}}) \approx \mathcal{L}(f_{\theta}) - \sigma'_{l}(0)\frac{1}{\eta}\frac{1}{n}\sigma'_{l}(0)\Big\langle \boldsymbol{V}_{goal}{}^{l+1}_{proj}(\delta \mathrm{W}_{l}),\ \boldsymbol{V}^{l+1}(\theta_{l\leftrightarrow l+1})\Big\rangle_{\mathrm{Tr}}$$

Sum up

At layer l we have :

- we evaluate the expressivity bottleneck $||\boldsymbol{V}^l_{\text{goal}} - \boldsymbol{V}^l||_{\text{Tr}}$
- we compute the best parameter move without changing the architecture
- we compute the best neurons to add by changing the architecture (finite quantity of neurons)

Neurons selection

- $\lambda_k >$ seuil
- statistical hypothesis test
- Numerical stability
- batchsize for estimation

V - Results

Random neurons vs our approach

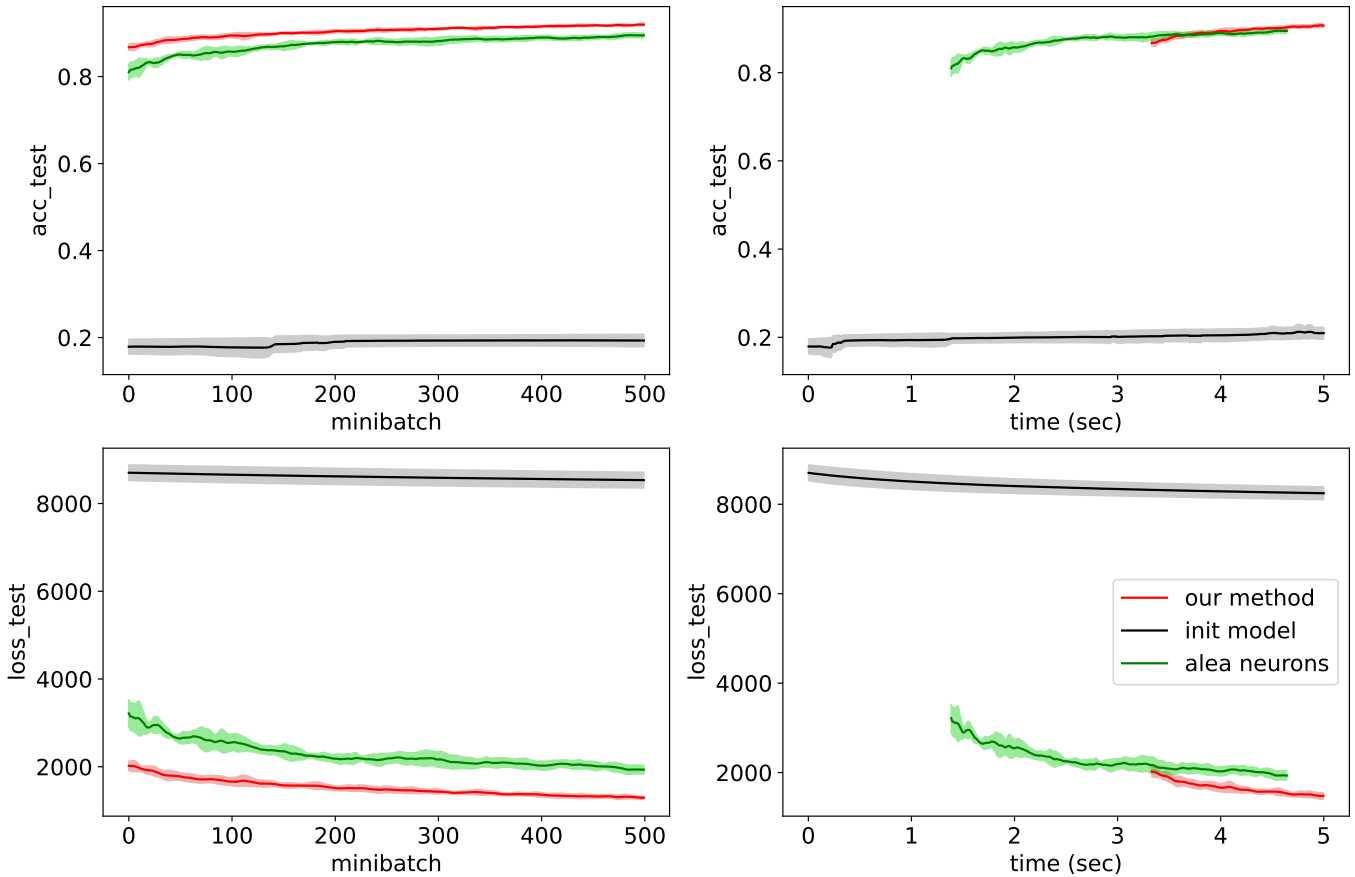


Figure: Acc and Loss on test set for the MNIST dataset. Comparaison random add / our method.

init architecture $[1, 1]$
alea/our architecture $[110, 61]$

MNIST

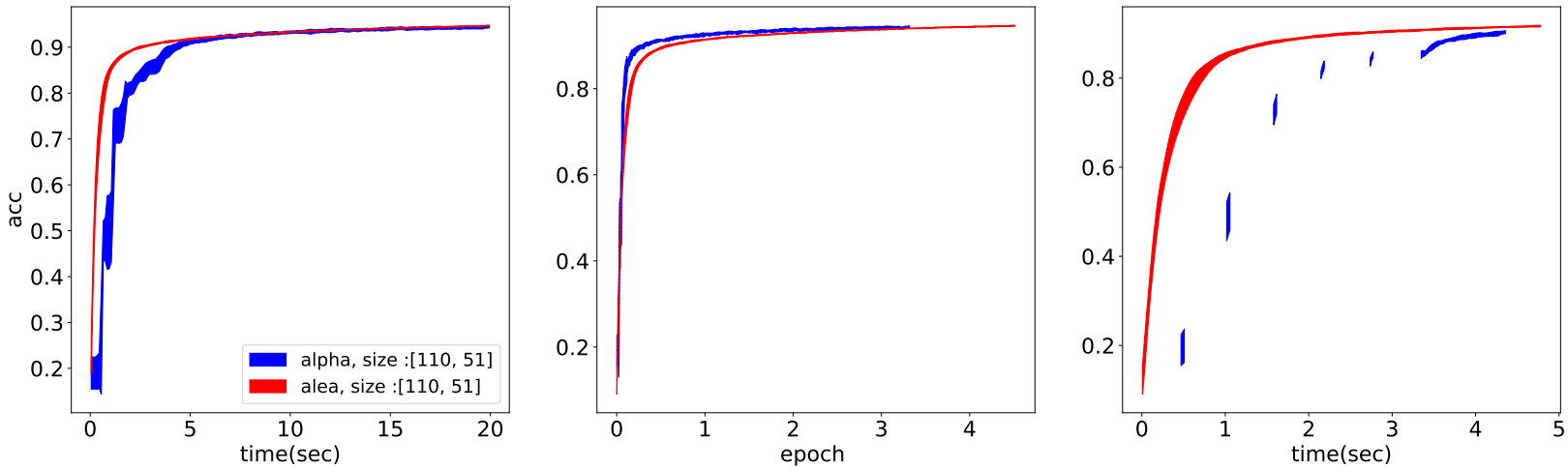


Figure: Accuracy on test set for the MNIST dataset, model : usual training with architecture $[110, 51]$ a time 0. model : our method with architecture $[110, 51]$.

MNIST

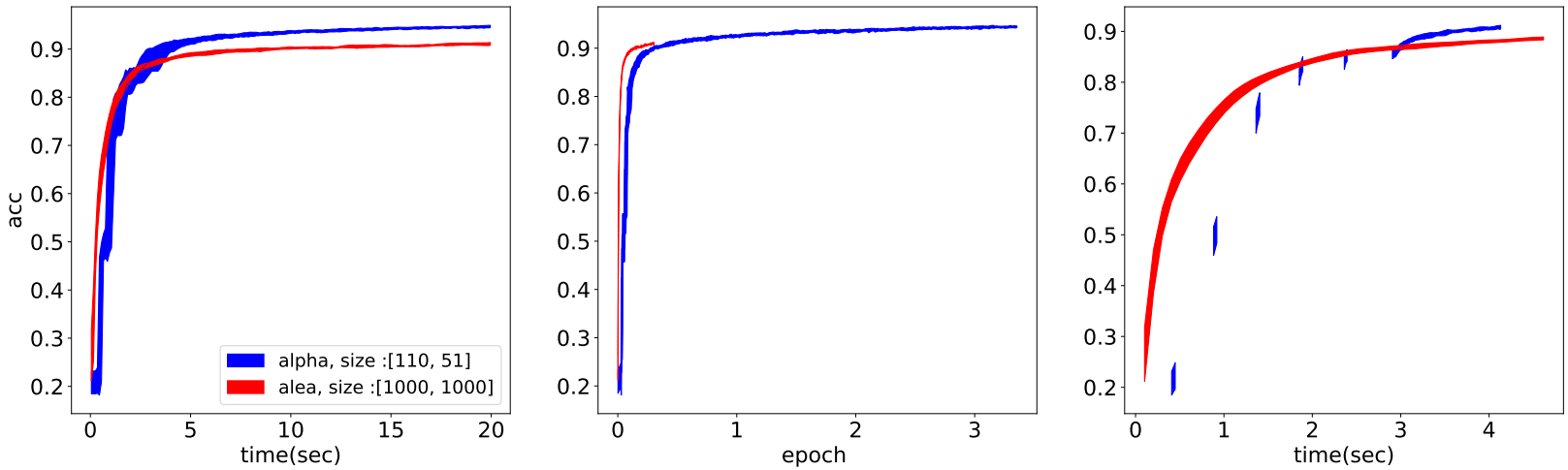
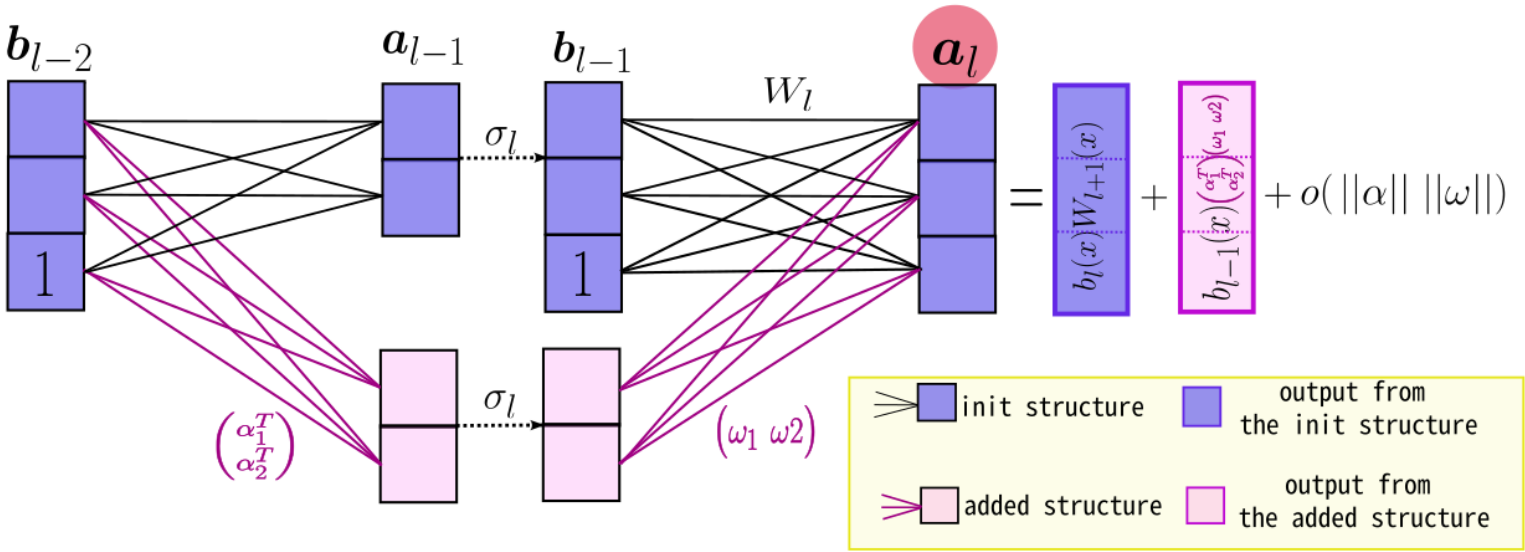


Figure: Accuracy on test set for the MNIST dataset, model : usual training with architecture $[1000, 1000]$ a time 0. model : our method with architecture $[110, 51]$.

comparaison 30 lancements rouge contre une bleue

Discussion

- looking for linear correlation between B_{l-2} and ${V_{goal}}^{l}$
  - connecting to others layers
  - can be extented to higher order moments (further Taylor expansion)
- greedy approach
- future work : convolution, extension to addition of layers (any DAG)

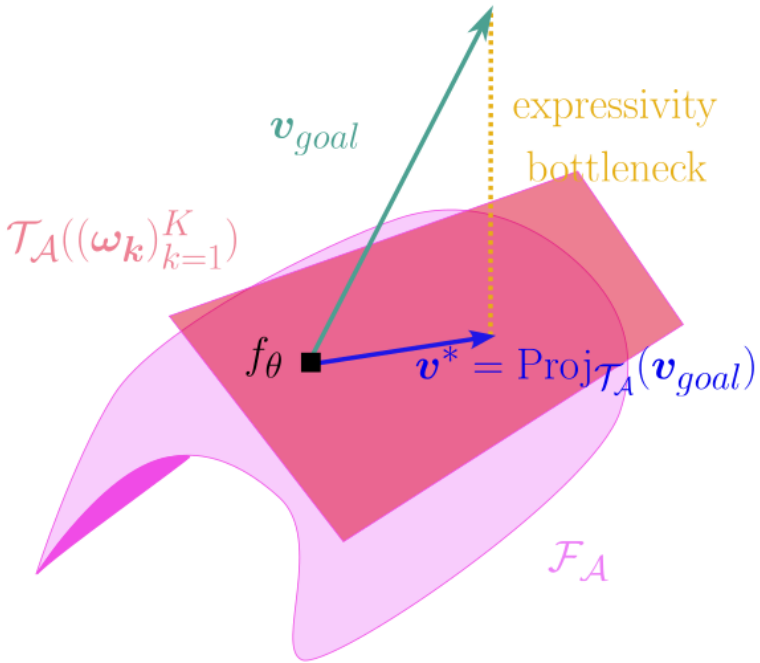


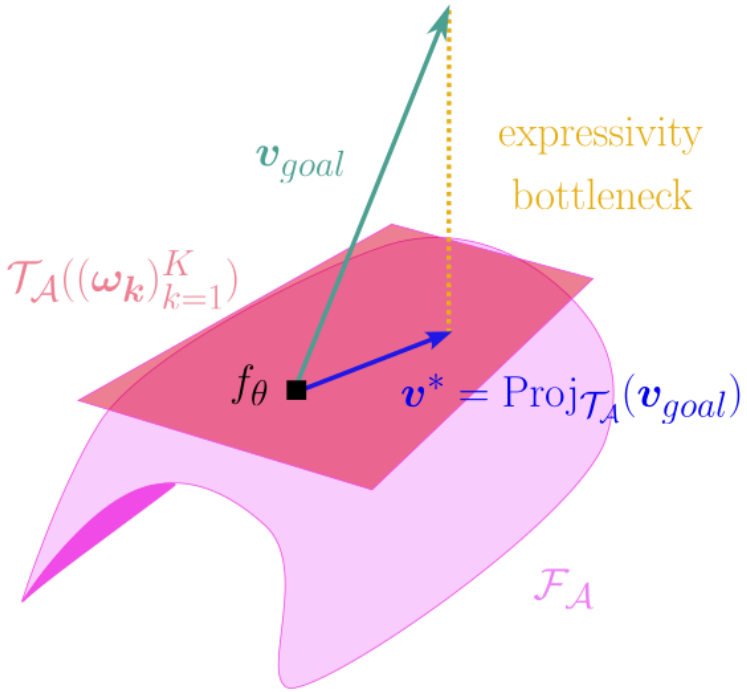
Article

https://www.lri.fr/~gcharpia/Expressivity_bottlenecks_preprint.pdf

Questions ?

?


$\boldsymbol{v}_{goal}$
expressivity
bottleneck
$\mathcal{T}_{\mathcal{A}}((\boldsymbol{\omega_k})_{k=1}^{K})$
f_θ
$\boldsymbol{v}^* = \text{Proj}_{\mathcal{T}_{\mathcal{A}}}(\boldsymbol{v}_{goal})$
$\mathcal{F}_{\mathcal{A}}$


$\boldsymbol{v}_{goal}$
expressivity
bottleneck
$\mathcal{T}_{\mathcal{A}}((\boldsymbol{\omega_k})_{k=1}^{K})$
f_θ
$\boldsymbol{v}^* = \text{Proj}_{\mathcal{T}_{\mathcal{A}}}(\boldsymbol{v}_{goal})$
$\mathcal{F}_{\mathcal{A}}$

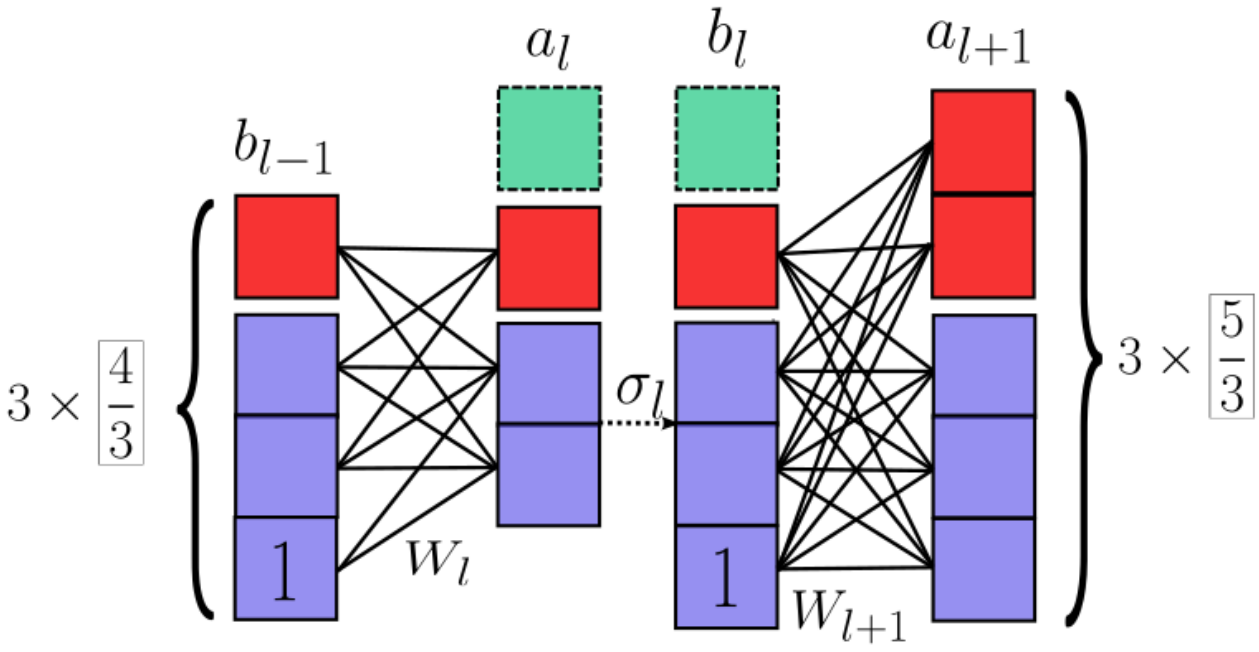
$$a_{l+1}(x) = a_{l+1}(x)_{|(\alpha_k,\omega_k)_{k=1}^K=0} + \sum_{k=1}^{K} \omega_k b_{l-1}(x)^T \alpha_k$$
$$+ o\left(\left(||(\alpha_k)_{k=1}^K|| + ||(\omega_k)_{k=1}^K||\right)^2\right)$$

Estimating the best neurons/update

$$\Big(\overbrace{(\hat{\boldsymbol{\alpha}}_k^*, \hat{\boldsymbol{\omega}}_k^*)_{k=1}^{K^*}}^{\hat{\theta}^{K,*}_{l\leftrightarrow l+1}}, K^*\Big) := \underset{(\boldsymbol{\alpha}_k, \boldsymbol{\omega}_k)_{k=1}^{K}, K}{\arg\min} \frac{1}{\boxed{n}} \left\{ || \boldsymbol{V}_{\text{goal}}{}^{l+1}_{proj} - \boldsymbol{V}^{l+1}(\theta^K_{l\leftrightarrow l+1}) ||^2_{\text{Tr}} \right\}$$

With $\boldsymbol{V} := \begin{pmatrix} v_{\text{goal}}(\boldsymbol{x}_1) & \dots & v_{\text{goal}}(\boldsymbol{x}_n) \end{pmatrix}$

Estimating the best neurons and the naturel gradient



$$n \leftarrow n \times max(\sqrt{\frac{4}{3}}, \sqrt{\frac{5}{3}})$$

amplitude factor

Defining the updates
$\theta^{\delta(l+1)}(\gamma_0) = (\boldsymbol{W}_1, ..., \boldsymbol{W}_{l+1} + \gamma_0\, \delta \boldsymbol{W}^*_{l+1}, ..., \boldsymbol{W}_L)$ and
$\hat{\theta}^{K,*}_{l \leftrightarrow l+1}(\gamma) = (\gamma \hat{\boldsymbol{\alpha}}^*_k, \hat{\boldsymbol{\omega}}^*_k)_{k=1}^K$, we apply a line search algorithm to find a local minimum of the loss function :

$$\gamma_0^* := \underset{\gamma_0 \in \mathcal{V}(0)}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f_{\theta^{\delta(l+1)}(\gamma_0)}(X_i), Y_i) \tag{1}$$

$$\gamma^* := \underset{\gamma \in \mathcal{V}(0)}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f_{\theta^{\delta(l+1)}(\gamma_0^*) \oplus \hat{\theta}^{K,*}_{l \leftrightarrow l+1}(\gamma)}(X_i), Y_i) \tag{2}$$

where $\mathcal{V}(0)$ is a positive neighbourhood of zero.