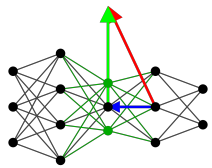
Spotting expressivity bottlenecks in neural networks and fixing them by optimal architecture growth

Manon Verbockhaven

Sylvain Chevallier, Guillaume Charpiat

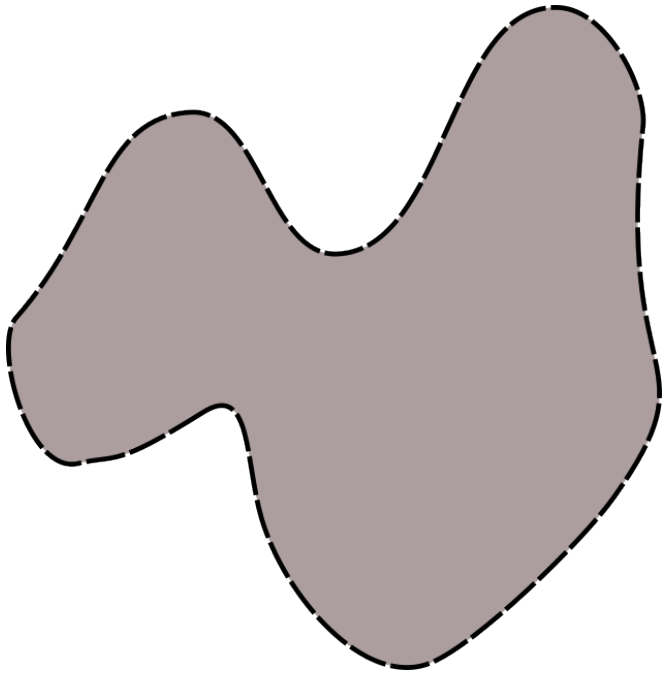
TAU team, LISN, INRIA Saclay, Université Paris-Saclay

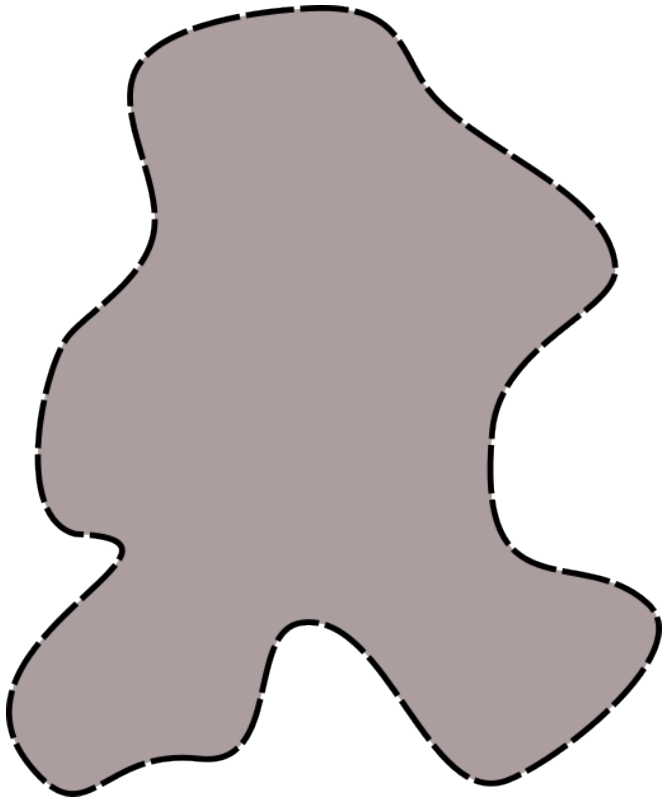
Rémi Gribonval Hervé Luga David Picard Guillaume Lecué Aurélie Névéol



Introduction

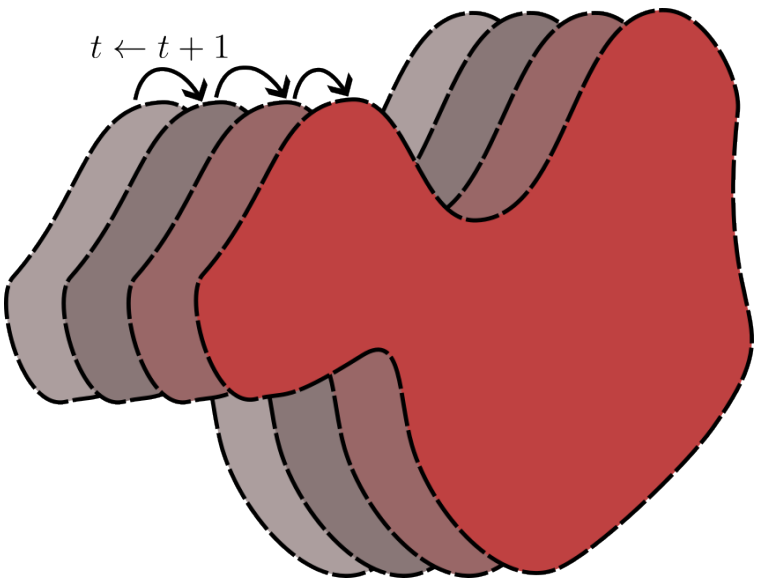
$\mathcal{T}$

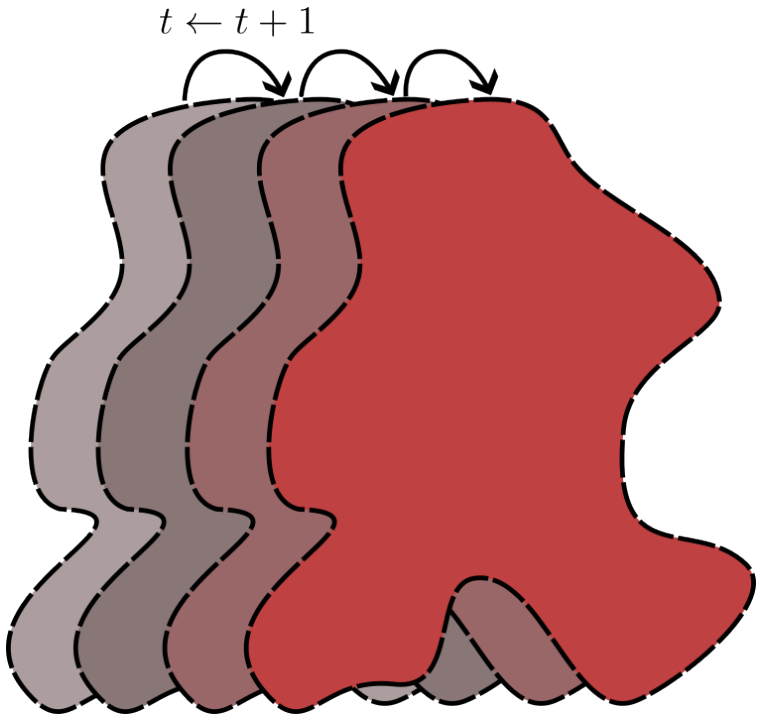




Introduction

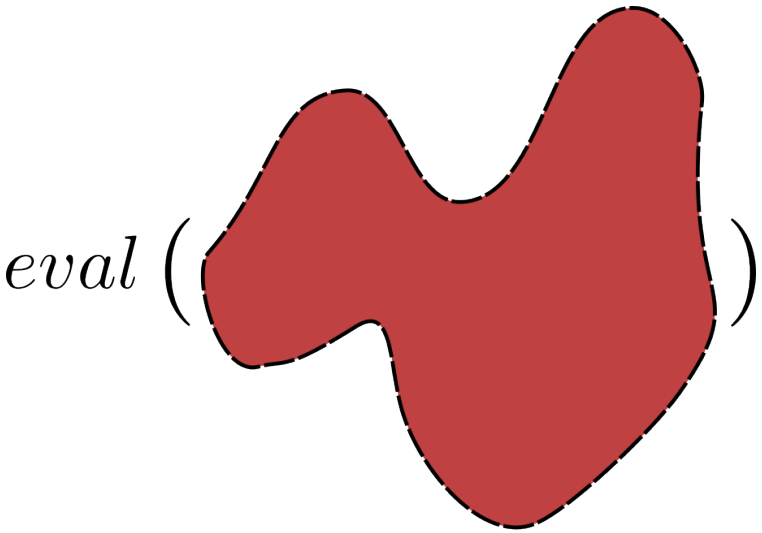
$$\mathcal{T}$$

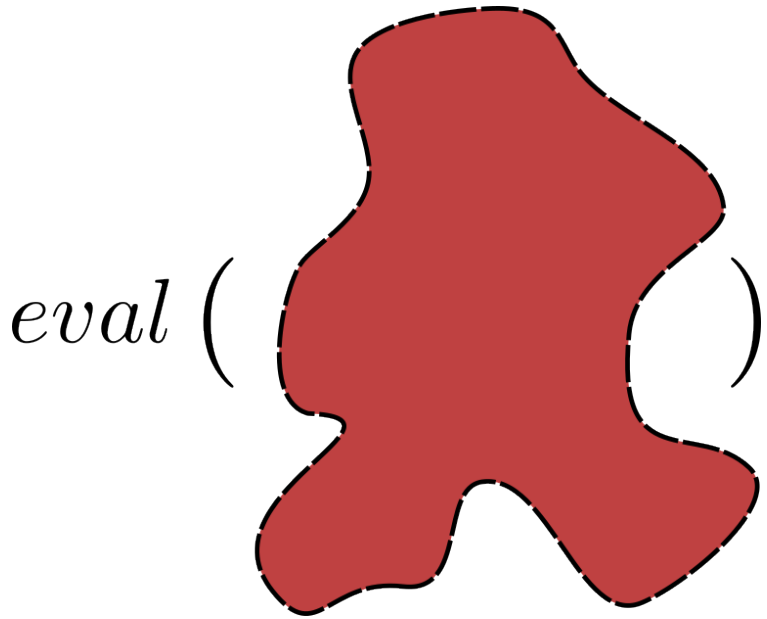




Introduction

τ





Plan

1. Literature review on Neural Architecture Search (NAS)
   - The definitions
   - The One-Shot techniques
   - Some expressivity metrics
2. The expressivity bottleneck
   - The intuition and the definitions
   - The best parameter move and the best neurons to add
3. Reformulation and extension
   - Revisiting GradMax and NORTH
   - Move away from first order approximation

Part I

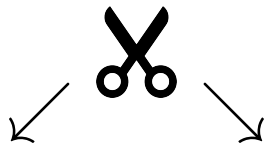
I Literature review on Neural Architecture Search (NAS)

- The definitions
- The One-Shot techniques
- Some expressivity metrics

NAS : the optimization problem

Consider a task $\mathcal{T}$ and a corresponding dataset $\mathcal{X}$. Let $\mathbb{A}$ a set of architecture and $\mathcal{L}$ a loss function, we would like to solve :

$$\underset{\mathcal{A}\in\mathbb{A},\ \theta\in\mathbb{R}^{d(\mathcal{A})}}{\arg\min} \quad \mathcal{L}_{train}(\mathcal{A}(\theta)) \tag{1}$$



$$\iff \underset{\mathcal{A}\in\mathbb{A}}{\arg\min}\ \mathcal{L}_{train}(\mathcal{A}(\theta^*)) \quad s.t. \quad \theta^* := \underset{\theta}{\arg\min}\ \mathcal{L}_{train}(\mathcal{A}(\theta))$$

NAS Optimization problem

$$\underset{\mathcal{A}\in\mathbb{A}}{\arg\min}\ \mathcal{L}_{val}(\mathcal{A}(\theta^*)) \quad s.t. \quad \theta^* := \underset{\theta}{\arg\min}\ \mathcal{L}_{train}(\mathcal{A}(\theta)) \tag{2}$$

NAS : a two-steps optimization procedure

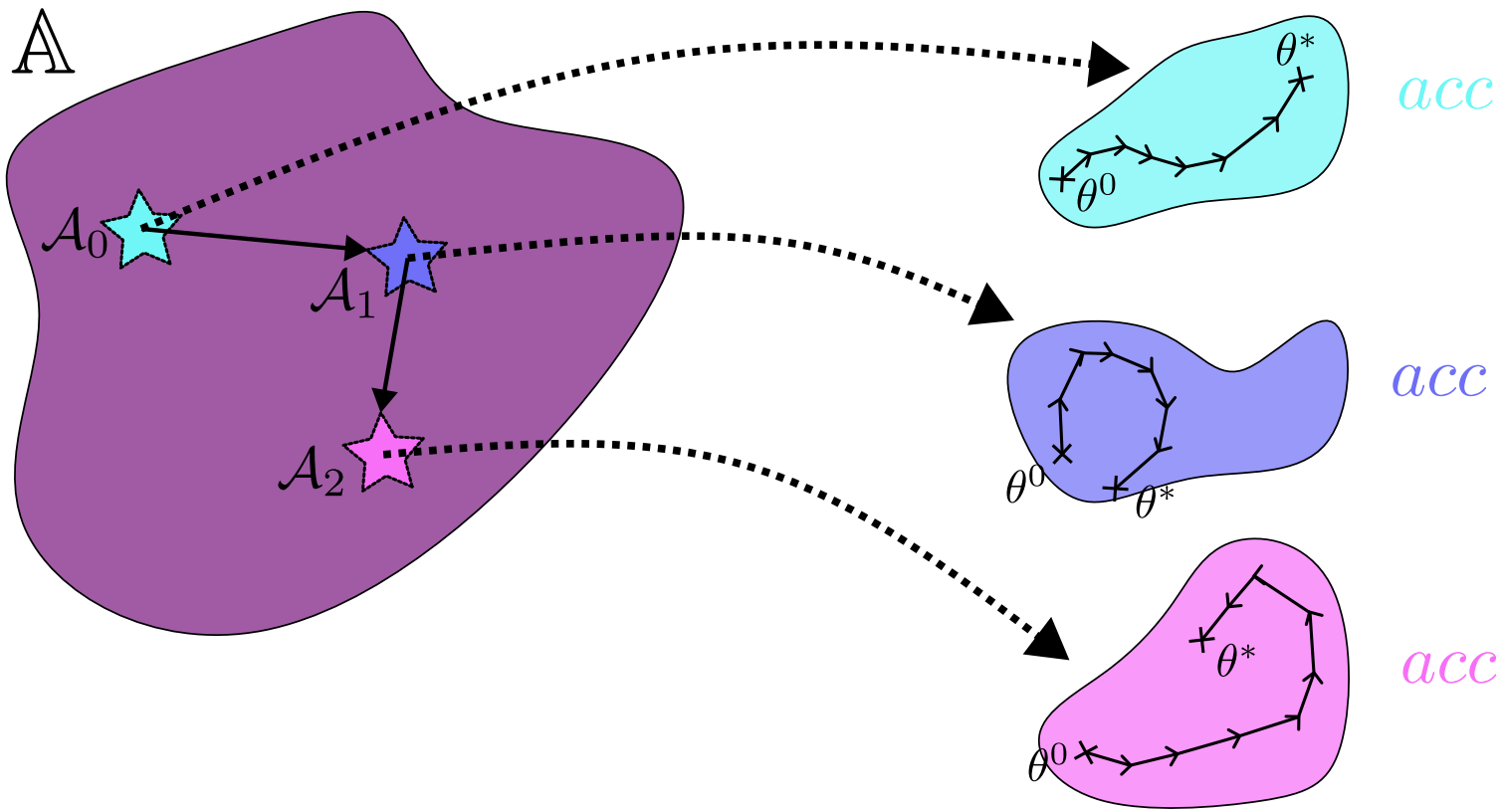
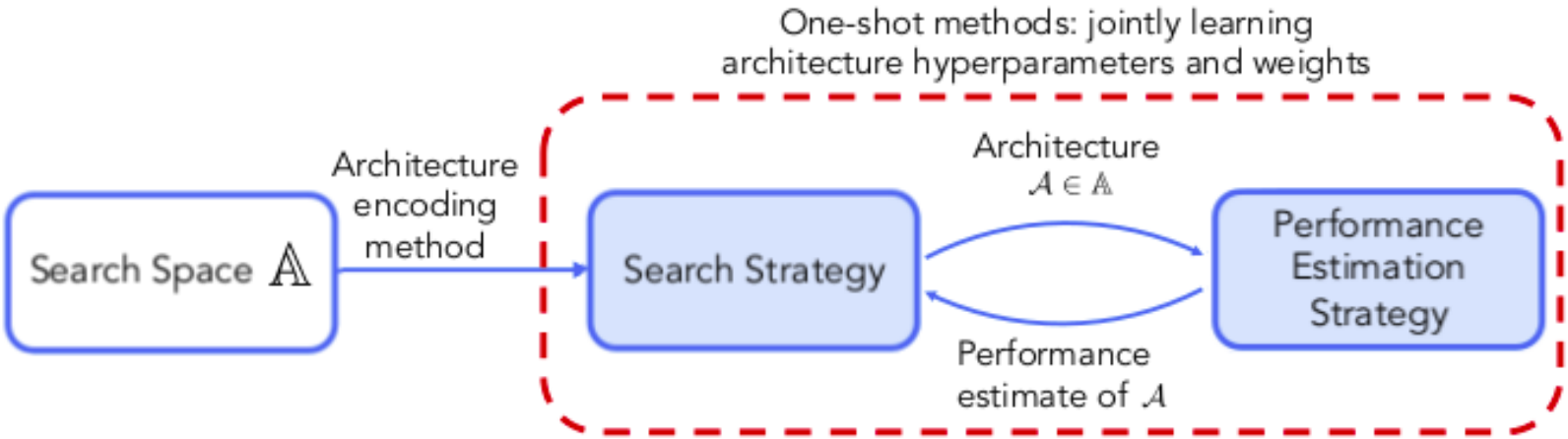


Figure: Optimization of a network architecture with the decoupling between the search into the space of architecture $\mathbb{A}$ and the space of parameters $\mathbb{R}^{d(\mathcal{A})}$.

NAS : the definitions

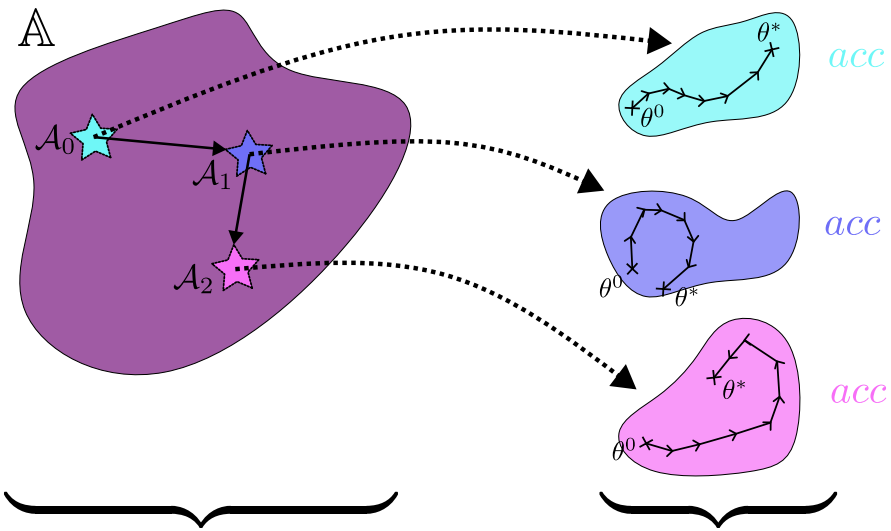
NAS : the definitions



NAS : the search strategy

Probabilistic techniques

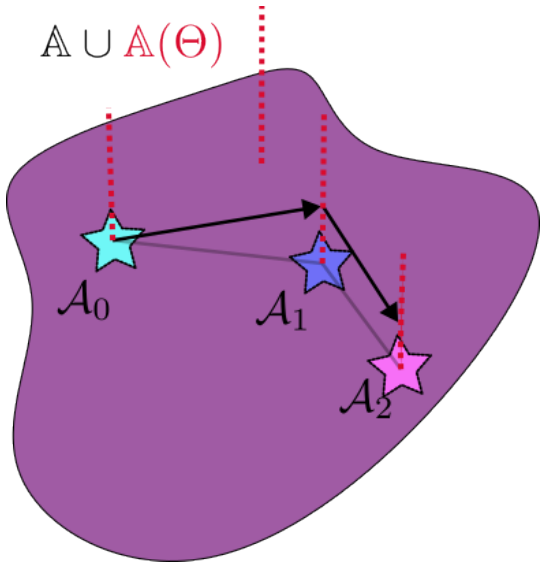
Search $\rightleftarrows$ Evaluation



Architecture Optimization

Parameters Optimization

One-shot techniques



Joint optimization of the architecture and its parameters

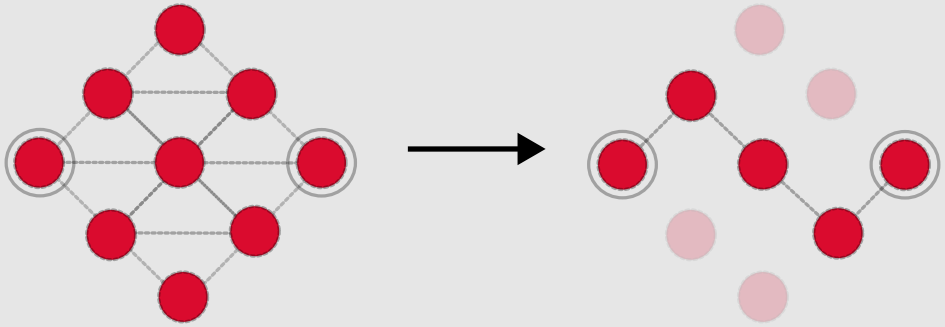
First formulation

$$\underset{\mathcal{A}\in\mathbb{A}}{\arg\min}\ \mathcal{L}_{train}(\mathcal{A}(\theta^*)) \quad s.t. \quad \theta^* := \underset{\theta}{\arg\min}\ \mathcal{L}_{train}(\mathcal{A}(\theta))$$

$$\underset{\mathcal{A}\in\mathbb{A},\ \theta\in\mathbb{R}^{d(\mathcal{A})}}{\arg\min}\ \mathcal{L}_{train}(\mathcal{A}(\theta))$$

NAS : categories of one-shot techniques

Supernetwork



Gradient-based

A gradient-based method searches in $\mathbb{A}$ using the back-products of the standard backpropagation (gradient estimation method to update network parameters) (Evci et al. [2022], Dai et al. [2019], Wu et al. [2020], Verbockhaven et al. [2024]).

Examples of gradient-based methods

GradMax

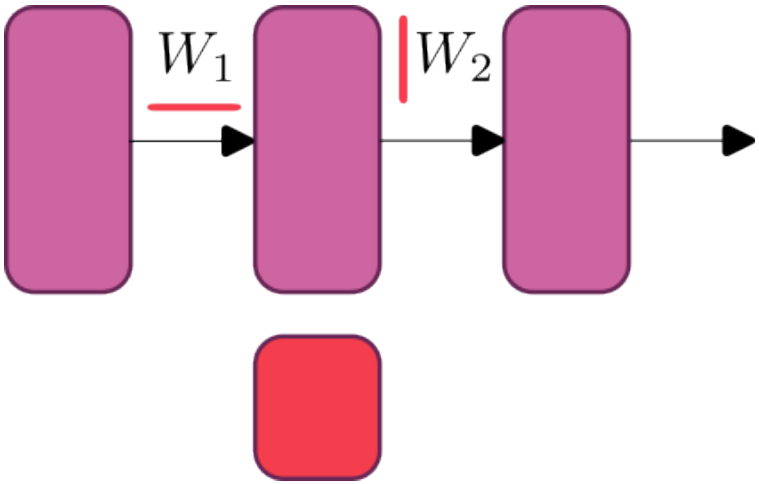


Figure: Adding one neuron (red) to a layer (pink) as in paper Evci et al. [2022]

Firefly

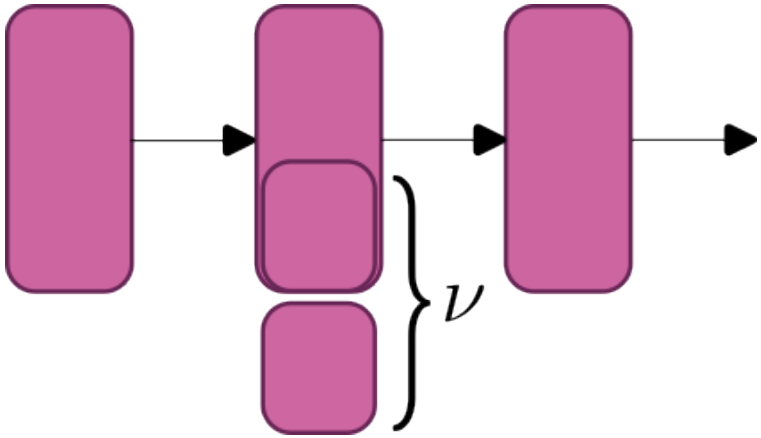
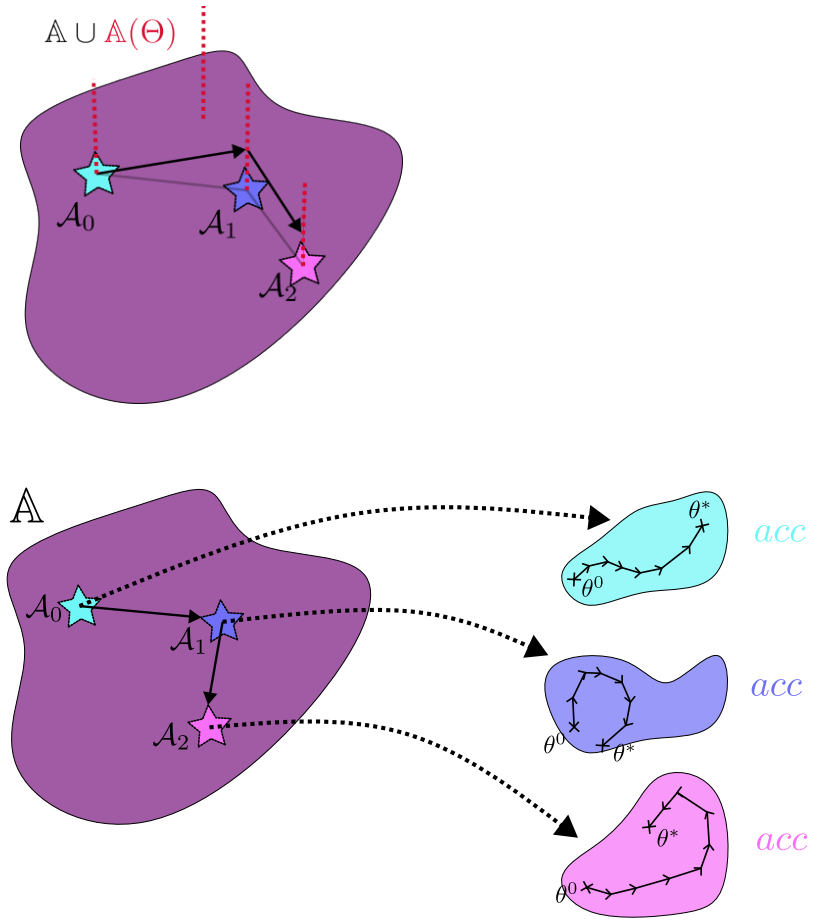


Figure: Dividing one neuron into two neurons (ν) as in paper Wu et al. [2020]

Orient the search

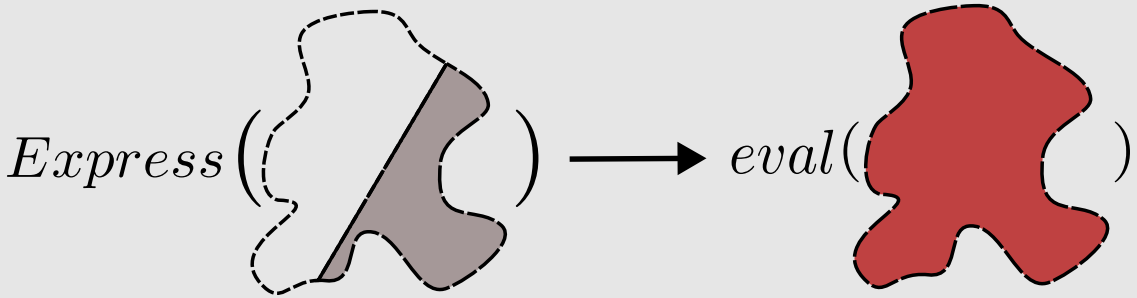
- Probabilistic techniques & One-Shot methods

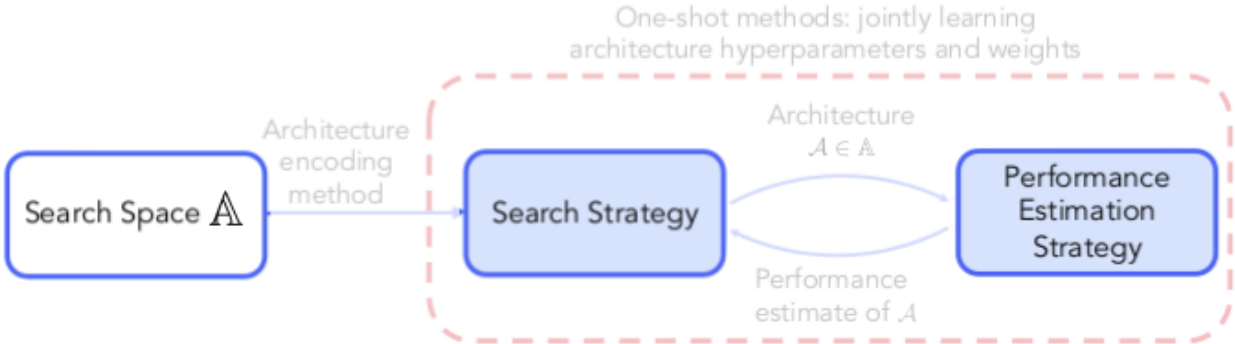
$\hookrightarrow$ optimization in $\mathbb{A}$ and θ



The expressivity metrics

Expressivity

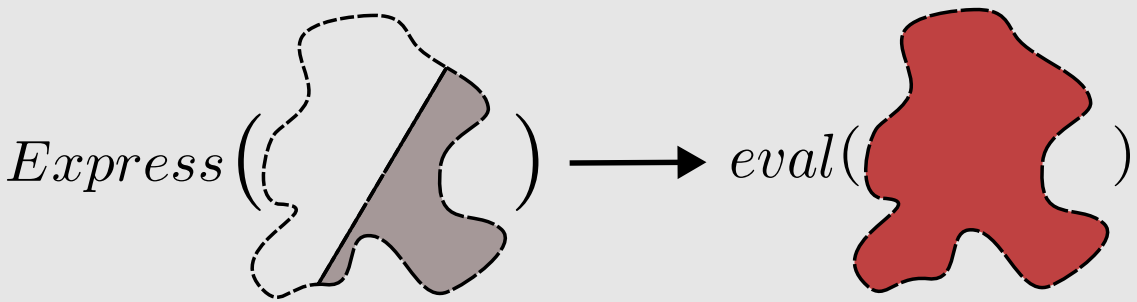




$$\arg\min_{\mathcal{A}\in\mathbb{A}} \mathcal{L}_{val}(\mathcal{A}(\theta^*)) \qquad s.t. \quad \theta^* := \arg\min_{\theta} \mathcal{L}_{train}(\mathcal{A}(\theta))$$

The expressivity metrics

Expressivity



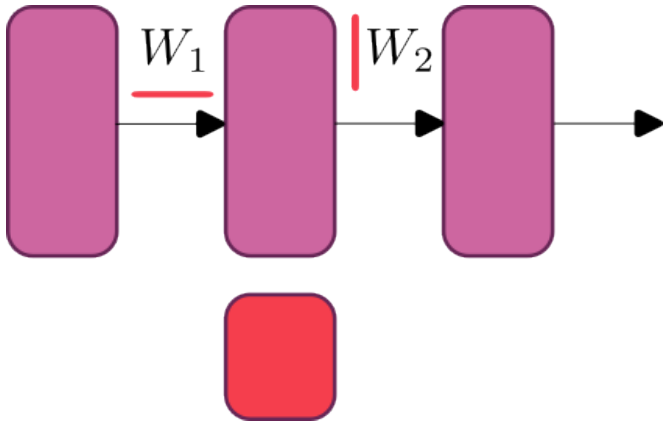
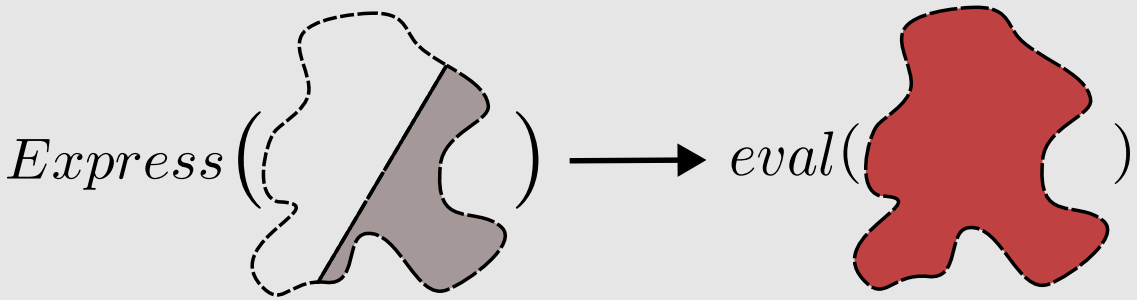


Figure: Adding one neuron (red) to a layer (pink) as in papers of Maile et al. [2022]

The expressivity metrics

Expressivity



	easy to compute	specific to the task $\mathcal{T}$
VC-dimension	✗	✗
Rademacher Complexity	✗	✓

Part II

I Literature review on Neural Architecture Search (NAS)
- The definitions
- The One-Shot techniques
- Some expressivity metrics

II The expressivity bottleneck
- The intuition and the definitions
- The best parameter move and the best neurons to add

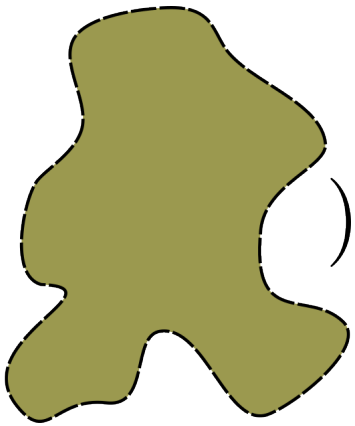
the problem statement

We would like to solve :

$$\underset{\mathcal{A}\in\mathbb{A},\ \theta\in\mathbb{R}^{d(\mathcal{A})}}{\arg\min} \quad \mathcal{L}_{train}(\mathcal{A}(\theta))$$

- Jointly optimize the architecture and its parametrization.

With the use of :

$Express($  $)$

- Easy to compute.
- Provides guidance in $\mathbb{A}$ without redundancy.

Intuition

Notations

- Dataset $\mathcal{D} := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ iid
- Neural Network $f_\theta := \mathcal{A}(\theta)$
- Loss function $\mathcal{L}$

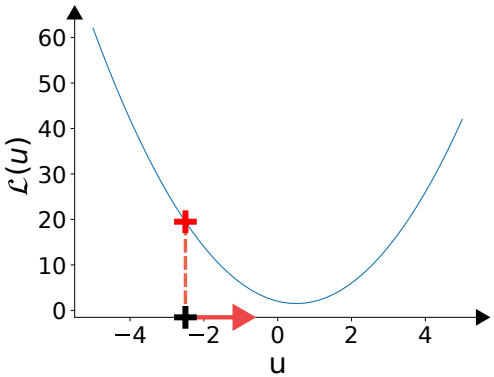
$$\underset{\mathcal{A}\in\mathbb{A},\ \theta\in\mathbb{R}^{d(\mathcal{A})}}{\arg\min} \quad \mathcal{L}_{train}(f_\theta)$$

Intuition

Notations

- Dataset $\mathcal{D} := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ iid
- Neural Network $f_\theta := \mathcal{A}(\theta)$
- Loss function $\mathcal{L}$

$$\underset{\mathcal{A}\in\mathbb{A},\ \theta\in\mathbb{R}^{d(\mathcal{A})}}{\arg\min} \quad \mathcal{L}_{train}(f_\theta)$$



Minimize $\mathcal{L} : U \to \mathbb{R}^+$ on $(U, \langle . \, , . \rangle_\dagger)$ with

$$u^{t+1} = u^t - \eta \nabla_u^\dagger \mathcal{L}(u)_{|u=u^t}$$

Where $\nabla_u^\dagger \mathcal{L}(u)$ verifies :

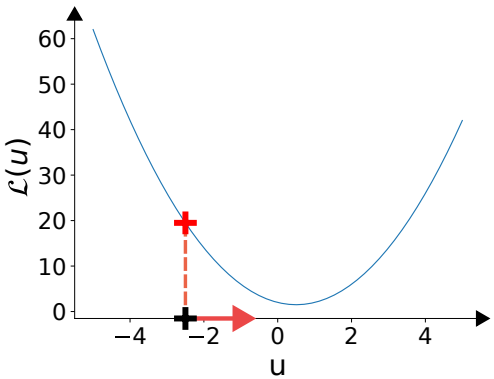
$$\mathcal{L}(u + \delta u) = \mathcal{L}(u) + \left\langle \nabla_u^\dagger \mathcal{L}(u), \delta u \right\rangle_\dagger + o(\|\delta u\|_\dagger)$$

Intuition

Notations

- Dataset $\mathcal{D} := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ iid
- Neural Network $f_\theta := \mathcal{A}(\theta)$
- Loss function $\mathcal{L}$

$$\underset{\mathcal{A}\in\mathbb{A},\ \theta\in\mathbb{R}^{d(\mathcal{A})}}{\arg\min} \quad \mathcal{L}_{train}(f_\theta)$$



Minimize $\mathcal{L} : \Theta \to \mathbb{R}^+$ on $(\Theta, \langle . , . \rangle_{\mathbb{R}^p})$ with

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta^{\mathbb{R}^p} \mathcal{L}(\theta)_{|\theta=\theta^t}$$

Where $\nabla_\theta^{\mathbb{R}^p} \mathcal{L}(\theta)$ verifies :

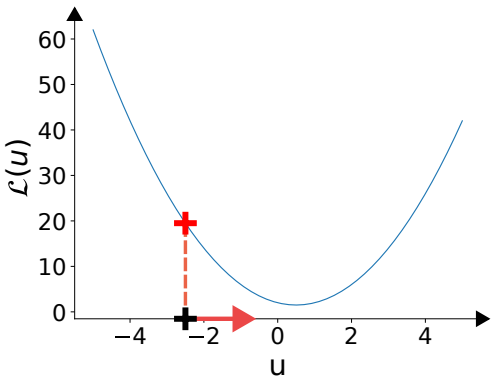
$$\mathcal{L}(\theta + \delta\theta) = \mathcal{L}(\theta) + \left\langle \nabla_\theta^{\mathbb{R}^p} \mathcal{L}(\theta), \delta\theta \right\rangle_{\mathbb{R}^p} + o(\|\delta\theta\|_{\mathbb{R}^p})$$

Intuition

Notations

- Dataset $\mathcal{D} := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ iid
- Neural Network $f_\theta := \mathcal{A}(\theta)$
- Loss function $\mathcal{L}$

$$\underset{\mathcal{A}\in\mathbb{A},\ \theta\in\mathbb{R}^{d(\mathcal{A})}}{\arg\min} \quad \mathcal{L}_{train}(f_\theta)$$



Minimize $\mathcal{L} : \mathcal{F} \to \mathbb{R}^+$ on $(\mathcal{F}, \langle . , . \rangle_{L_2})$ with

$$f^{t+1} = f^t - \eta \nabla_f^{L_2} \mathcal{L}(f)_{|f=f^t}$$

Where $\nabla_f^{L_2} \mathcal{L}(f)$ verifies :

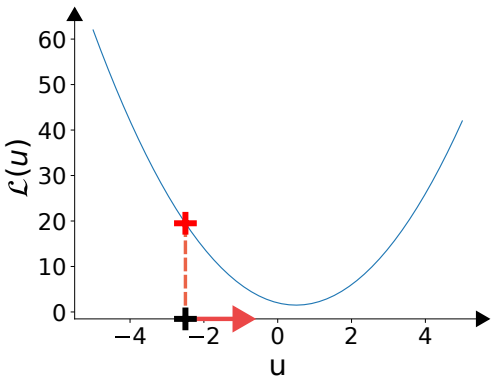
$$\mathcal{L}(f + \delta f) = \mathcal{L}(f) + \left\langle \nabla_f^{L_2} \mathcal{L}(f), \delta f \right\rangle_{L_2} + o(\|\delta f\|_{L_2})$$

Intuition

Notations

- Dataset $\mathcal{D} := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ iid
- Neural Network $f_\theta := \mathcal{A}(\theta)$
- Loss function $\mathcal{L}$

$$\underset{\mathcal{A}\in\mathbb{A},\ \theta\in\mathbb{R}^{d(\mathcal{A})}}{\arg\min} \quad \mathcal{L}_{train}(f_\theta)$$



Minimize $\mathcal{L} : \mathcal{F} \to \mathbb{R}^+$ on $(\mathcal{F}, \langle .\ ,. \rangle_{L_2})$ with

$$f^{t+1} = f^t \overbrace{-\eta \nabla_f^{L_2} \mathcal{L}(f)_{|f=f^t}}^{v_\nabla}$$

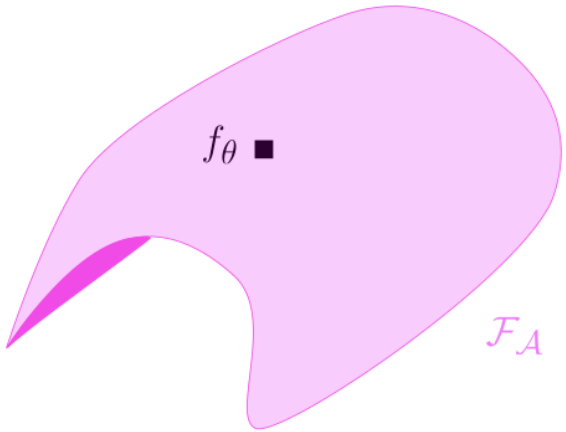
Where $\nabla_f^{L_2} \mathcal{L}(f)$ verifies :

$$\mathcal{L}(f + \delta f) = \mathcal{L}(f) + \left\langle \nabla_f^{L_2} \mathcal{L}(f), \delta f \right\rangle_{L_2} + o(\|\delta f\|_{L_2})$$

The expressivity bottleneck

Objects

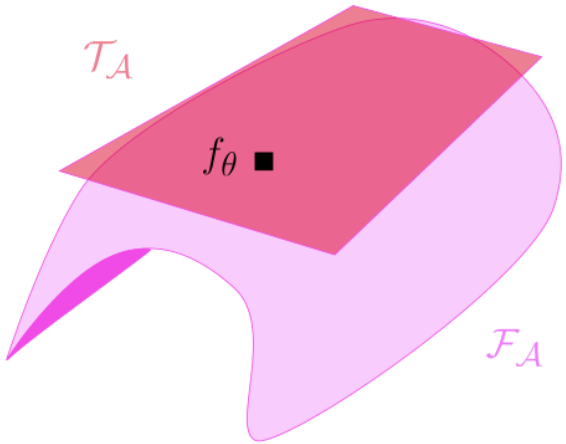
- Architecture space : $\Theta_{\mathcal{A}}$
- $\mathcal{F}_{\mathcal{A}} = \{f_\theta \mid \theta \in \Theta_{\mathcal{A}}\}$



The expressivity bottleneck

Objects

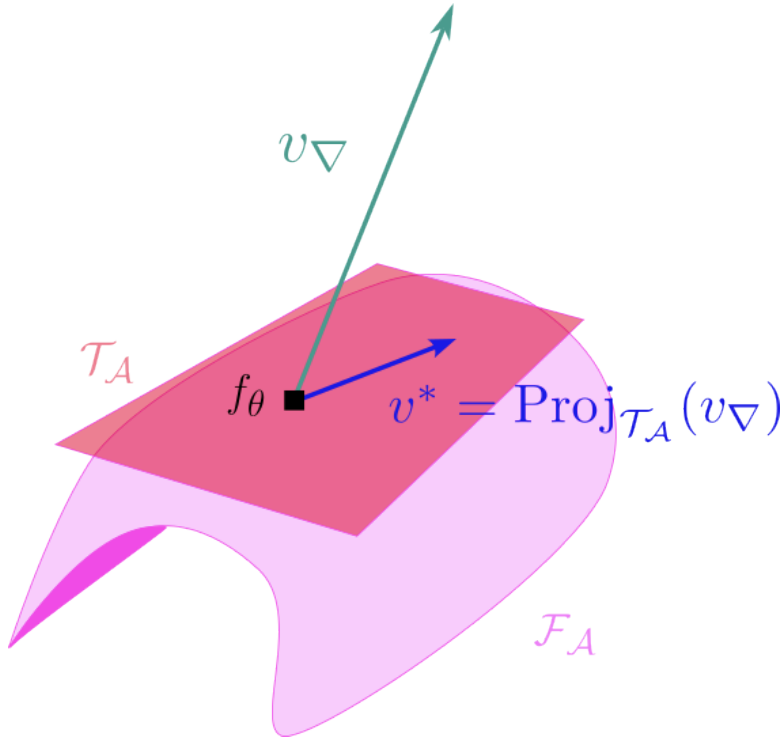
- Architecture space : $\Theta_{\mathcal{A}}$
- $\mathcal{F}_{\mathcal{A}} = \{f_\theta \mid \theta \in \Theta_{\mathcal{A}}\}$
- $\mathcal{T}_{\mathcal{A}}^{f_\theta}$ Tangent space in f_θ, ie $\{\frac{\partial f_\theta}{\partial \theta}\ \delta\theta \mid \text{s.t. } \delta\theta \in \Theta\}$



The expressivity bottleneck

Objects

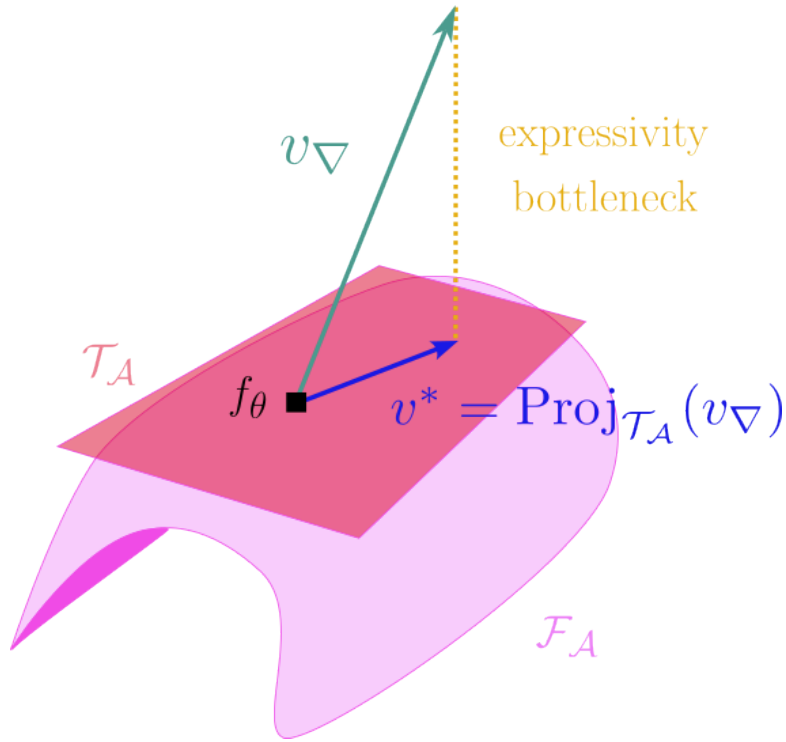
- Architecture space : $\Theta_{\mathcal{A}}$
- $\mathcal{F}_{\mathcal{A}} = \{f_\theta \mid \theta \in \Theta_{\mathcal{A}}\}$
- $\mathcal{T}_{\mathcal{A}}^{f_\theta}$ Tangent space in f_θ, ie $\{\frac{\partial f_\theta}{\partial \theta}\ \delta\theta \mid \text{s.t. } \delta\theta \in \Theta\}$



The expressivity bottleneck

Expressivity Bottleneck

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



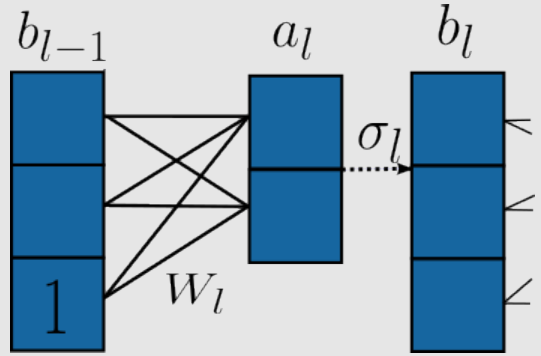
Definition (Feed-forward network)

We note $f_\theta : \mathbb{R}^p \to \mathbb{R}^d$ a feed-forward network with L hidden layers. Formally $\theta := (\boldsymbol{W}_l)_{l=1\dots L}$, and the function f_θ iteratively computes :

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

$$\forall l \in [1, L], \quad \begin{cases} \boldsymbol{a}_l(\boldsymbol{x}) &= \boldsymbol{W}_l \, \boldsymbol{b}_{l-1}(\boldsymbol{x}) \\ \boldsymbol{b}_l(\boldsymbol{x}) &= \begin{pmatrix} \sigma_l(\boldsymbol{a}_l(\boldsymbol{x})) \\ 1 \end{pmatrix} \end{cases}$$

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

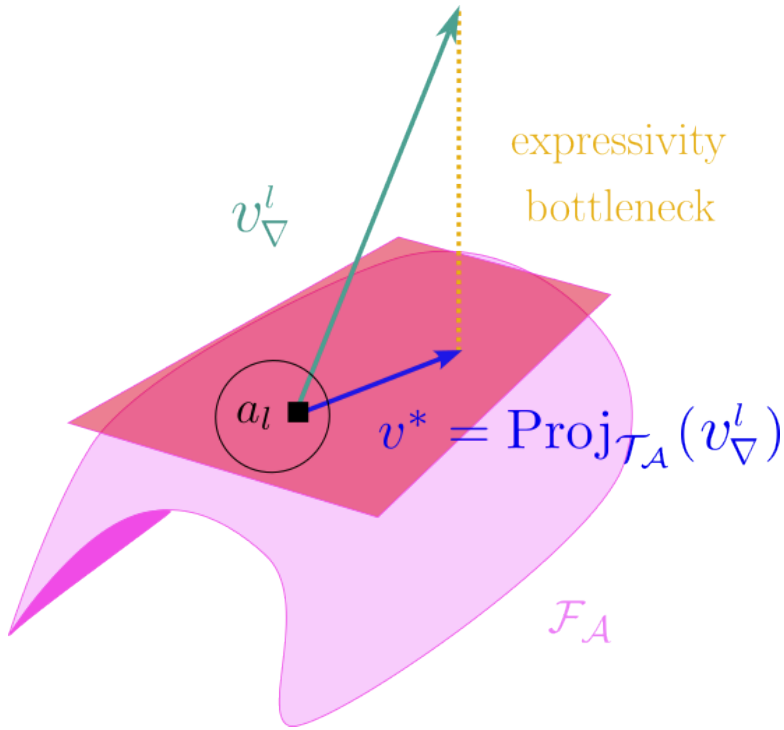


The expressivity bottleneck at layer l

Expressivity Bottleneck at l

$$\min_{\boldsymbol{v} \in \mathcal{T}_{\mathcal{A}}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})} \left[\left\| \boldsymbol{v}^l_{\nabla}(\boldsymbol{x}) - \boldsymbol{v}^l(\boldsymbol{x}) \right\|^2 \right]$$

$$\Psi^l := \min_{\boldsymbol{v}^l \in \mathcal{T}_{\mathcal{A}}} \frac{1}{n} \left\| \boldsymbol{V}^l_{\nabla} - \boldsymbol{V}^l \right\|^2_{\mathrm{Tr}}$$



$$\boldsymbol{V}_{\nabla} = \begin{pmatrix} \boldsymbol{v}_{\nabla}(\boldsymbol{x}_1) & \dots & \boldsymbol{v}_{\nabla}(x_n) \end{pmatrix} \text{ and } \boldsymbol{V} = \begin{pmatrix} \boldsymbol{v}(\boldsymbol{x}_1) & \dots & \boldsymbol{v}(x_n) \end{pmatrix}$$

Updating the current architecture

$$\delta \boldsymbol{W}^* := \underset{\delta \boldsymbol{W}^*}{\arg\min} \ \frac{1}{n} \left\| \boldsymbol{V}_\nabla^l - \boldsymbol{V}^l(\delta \boldsymbol{W}) \right\|^2$$

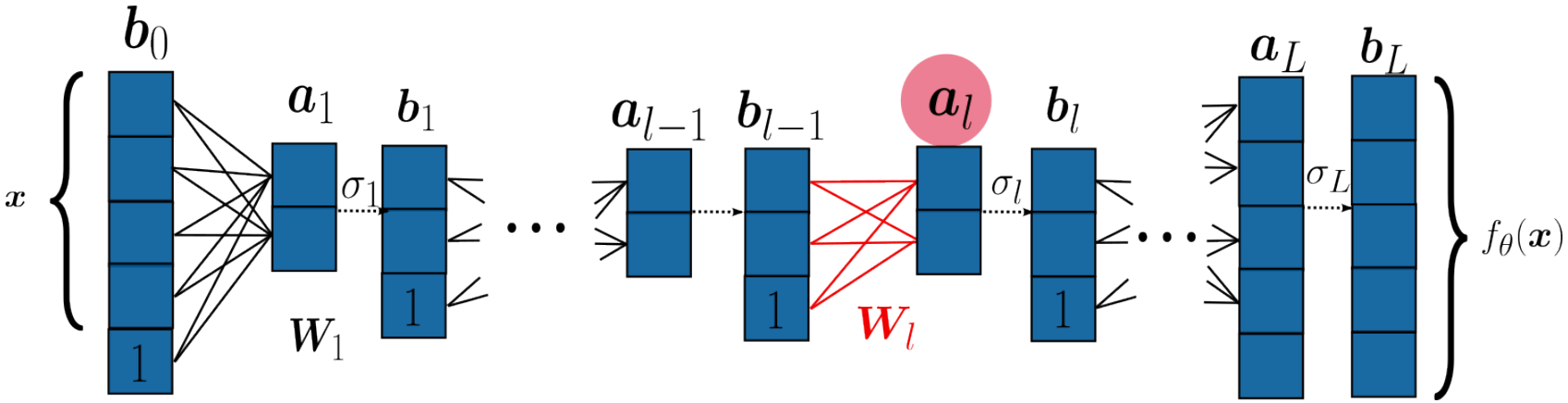


Figure: Approximating the expressivity bottleneck by modifying the parameters at layer l.

Adding neurons

$$\boldsymbol{A}^*, \boldsymbol{\Omega}^*, K^* := \underset{\boldsymbol{A},\boldsymbol{\Omega},K}{\arg\min} \frac{1}{n} \left\| \overbrace{\boldsymbol{V}_\nabla^l - \boldsymbol{V}^l(\delta \boldsymbol{W}^*)}^{\boldsymbol{V}_\perp^l} - \boldsymbol{V}^l(\boldsymbol{A}, \boldsymbol{\Omega}) \right\|^2$$

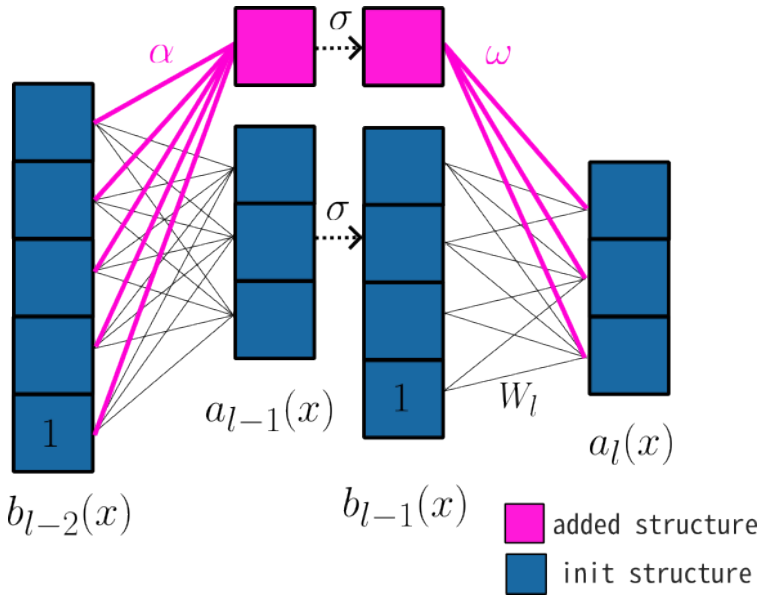
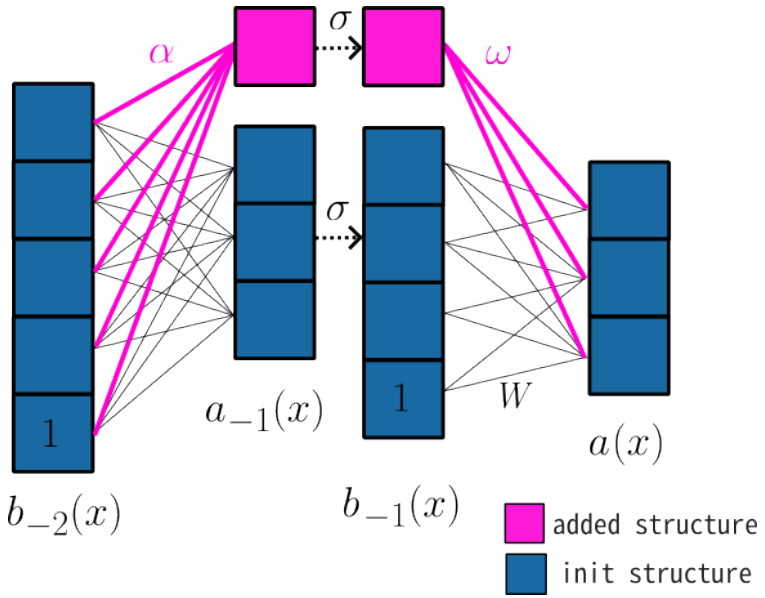


Figure: Adding one neuron to layer l in cyan, with connections in cyan. Here, $\boldsymbol{\alpha} \in \mathrm{R}^5$ and $\boldsymbol{\omega} \in \mathrm{R}^3$.

The optimization problems

$$\delta \boldsymbol{W}^* = \arg\min_{\delta \boldsymbol{W}} \frac{1}{n} \| \boldsymbol{V}_{\nabla} - \boldsymbol{V}(\delta \boldsymbol{W}) \|_{\mathrm{Tr}}^2$$

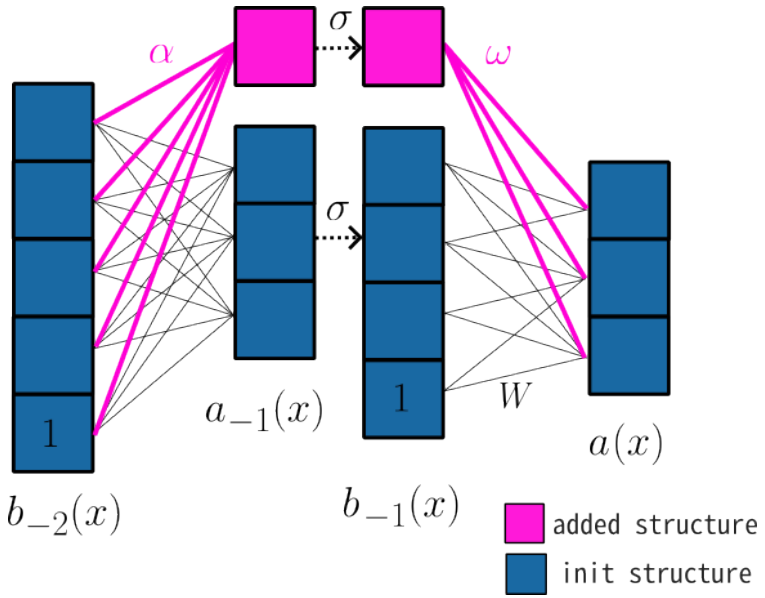
$$\boldsymbol{A}^*, \boldsymbol{\Omega}^*, K^* = \arg\min_{K,\, rk(\boldsymbol{\Omega}\boldsymbol{A}^T)=K} \frac{1}{n} \| \boldsymbol{V}_{\perp} - \boldsymbol{V}((\boldsymbol{A}, \boldsymbol{\Omega})) \|_{\mathrm{Tr}}^2$$



The optimization problems

$$\delta \boldsymbol{W}^* = \arg\min_{\delta \boldsymbol{W}} \frac{1}{n} \left\| \boldsymbol{V}_{\nabla} - \delta \boldsymbol{W} \boldsymbol{B}_{-1} \right\|_{\mathrm{Tr}}^2$$

$$\boldsymbol{A}^*, \boldsymbol{\Omega}^*, K^* = \arg\min_{K,\, rk(\boldsymbol{\Omega}\boldsymbol{A}^T)=K} \frac{1}{n} \left\| \boldsymbol{V}_{\perp} - \boldsymbol{\Omega} \boldsymbol{A}^T \boldsymbol{B}_{-2} \right\|_{\mathrm{Tr}}^2$$



The optimization problems

$$\delta \boldsymbol{W}^* = \arg\min_{\delta \boldsymbol{W}} \frac{1}{n} \left\| \boldsymbol{V}_{\nabla} - \delta \boldsymbol{W} \boldsymbol{B}_{-1} \right\|_{\mathrm{Tr}}^2$$

$$\boldsymbol{A}^*, \boldsymbol{\Omega}^*, K^* = \arg\min_{K,\, rk(\boldsymbol{\Omega}\boldsymbol{A}^T)=K} \frac{1}{n} \left\| \boldsymbol{V}_{\perp} - \boldsymbol{\Omega} \boldsymbol{A}^T \boldsymbol{B}_{-2} \right\|_{\mathrm{Tr}}^2$$

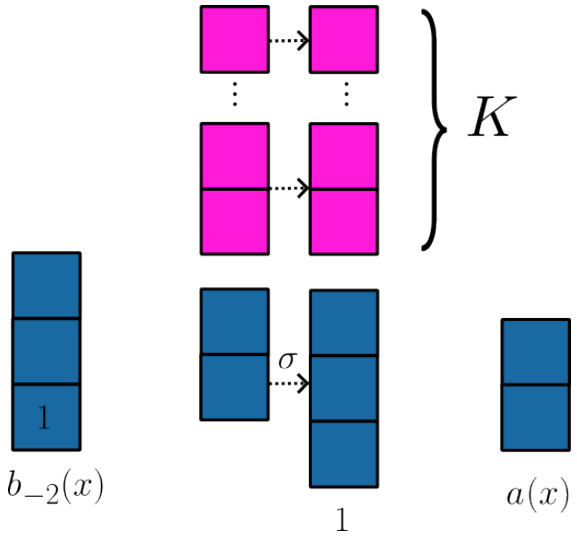
Best Update and new neurons

The optimal solution is $K^* = rank(\boldsymbol{V}_{\perp} \boldsymbol{B}_{-2}^T)$ and :

$$\delta \boldsymbol{W}^* = \boldsymbol{V}_{\nabla} \boldsymbol{B}_{-1}^T (\boldsymbol{B}_{-1} \boldsymbol{B}_{-1}^T)^+$$

$$\boldsymbol{\Omega}^* \boldsymbol{A}^{*T} = \boldsymbol{V}_{\perp} \boldsymbol{B}_{-2}^T (\boldsymbol{B}_{-2} \boldsymbol{B}_{-2}^T)^+$$

The new neurons

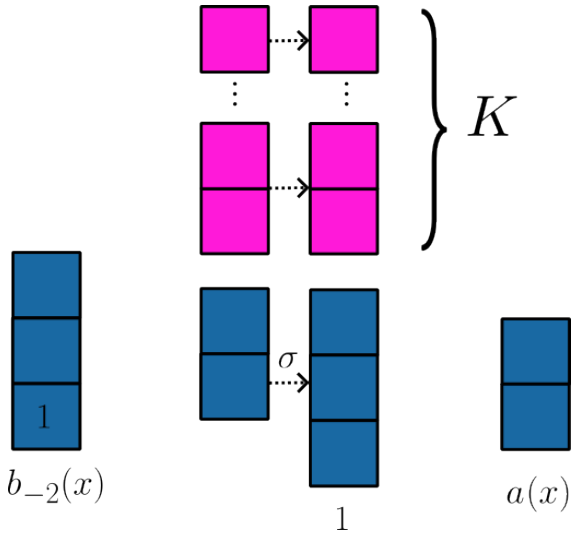


New neurons

The optimal solution is $K^* = rank(\boldsymbol{V}_\perp \boldsymbol{B}_{-2}^T)$ and :

$$\boldsymbol{\Omega}^* \boldsymbol{A}^{*T} = \boldsymbol{V}_\perp \boldsymbol{B}_{-2}^T (\boldsymbol{B}_{-2} \boldsymbol{B}_{-2}^T)^+$$

The new neurons

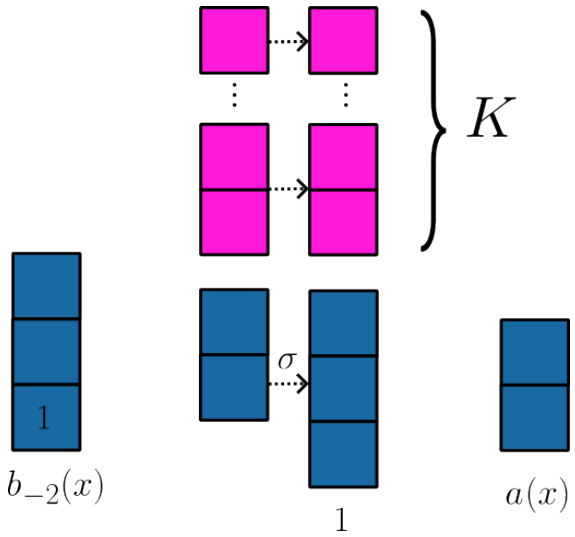


New neurons

Noting $\boldsymbol{S} := \frac{1}{n}\boldsymbol{B}_{-2}\boldsymbol{B}_{-2}^T$, $\boldsymbol{N}_\perp := \frac{1}{n}\boldsymbol{B}_{-2}{\boldsymbol{V}_\perp}^T$, the optimal solution is $K^* = rank(\boldsymbol{N}_\perp)$ and :

$$\boldsymbol{\Omega}^*{\boldsymbol{A}^*}^T = \boldsymbol{N}_\perp^T\boldsymbol{S}^+$$

The new neurons

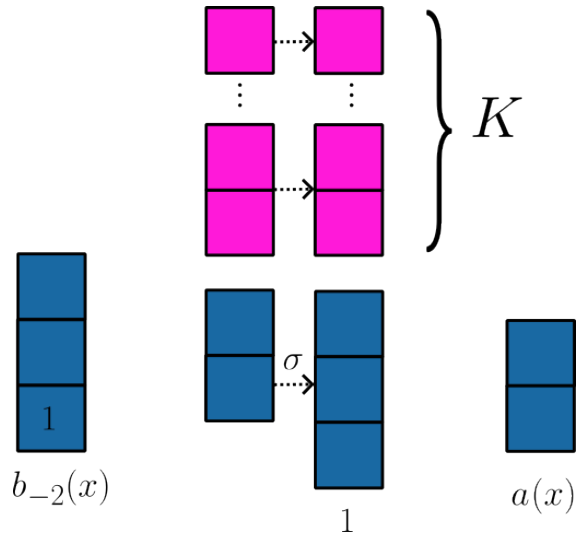


$$\boldsymbol{A}^*, \boldsymbol{\Omega}^* = \underset{rk(\boldsymbol{\Omega}\boldsymbol{A}^T)=K}{\arg\min} \frac{1}{n} \left\| \boldsymbol{V}_\perp - \boldsymbol{\Omega}\boldsymbol{A}^T\boldsymbol{B}_{-2} \right\|_{\text{Tr}}^2$$

Theorem

Imposing $K \leq rank(\boldsymbol{N}_\perp)$, then the optimal solution is :

$$\left(\boldsymbol{\Omega}_K^* {\boldsymbol{A}_K^*}^T\right) = \left(\boldsymbol{N}_\perp^T \boldsymbol{S}^{-\frac{1}{2}}\right)_{|\text{Approx rk}(K)} \boldsymbol{S}^{-\frac{1}{2}}$$



Theorem : the expressivity bottleneck decrease

If the new neurons are initialised as previously, then noting $\lambda_1 \geq ... \geq \lambda_K$ the K first eigenvalues of $\boldsymbol{N}_\perp^T \boldsymbol{S}^{-\frac{1}{2}}$, then :

$$\Psi^l_{\theta \oplus \theta^{K,*}_{\leftrightarrow}} = \Psi^l_\theta - \sum_{k=1}^{K} \lambda_k^2$$

Part III

- Reformulation and extension
  - Revisiting GradMax and NORTH
  - Move away from first order approximation

Similar new neurons initialization methods

GradMax (Evci et al. [2022]) & NORTH (Maile et al. [2022])

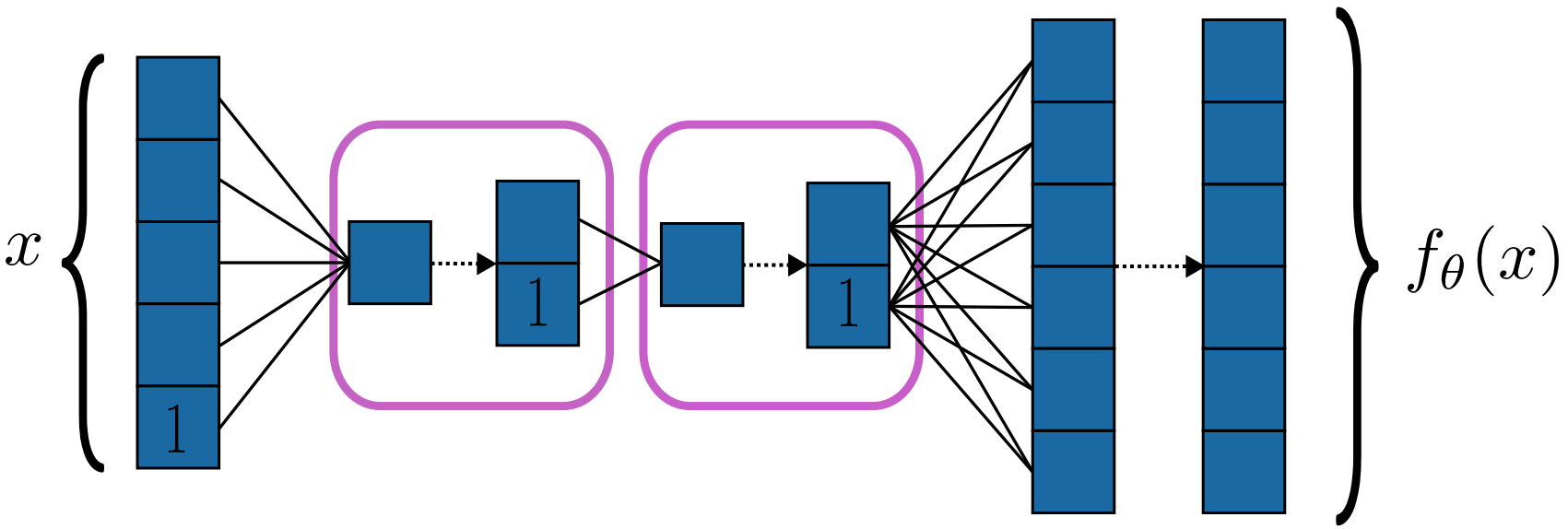
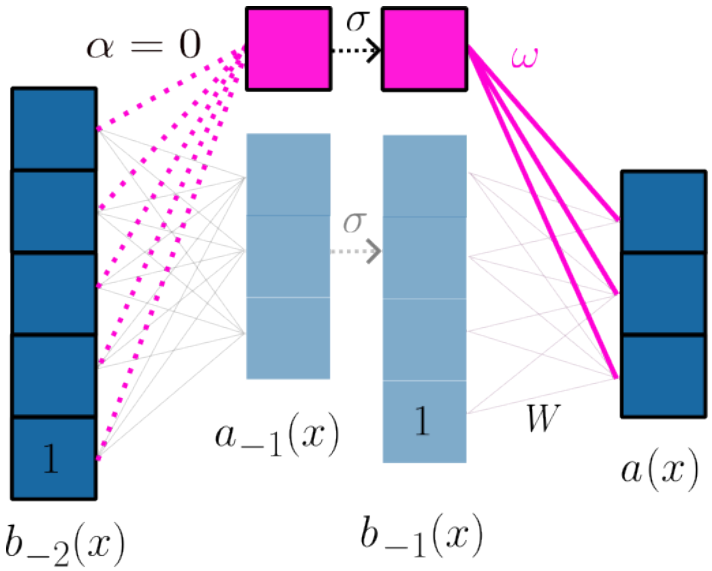
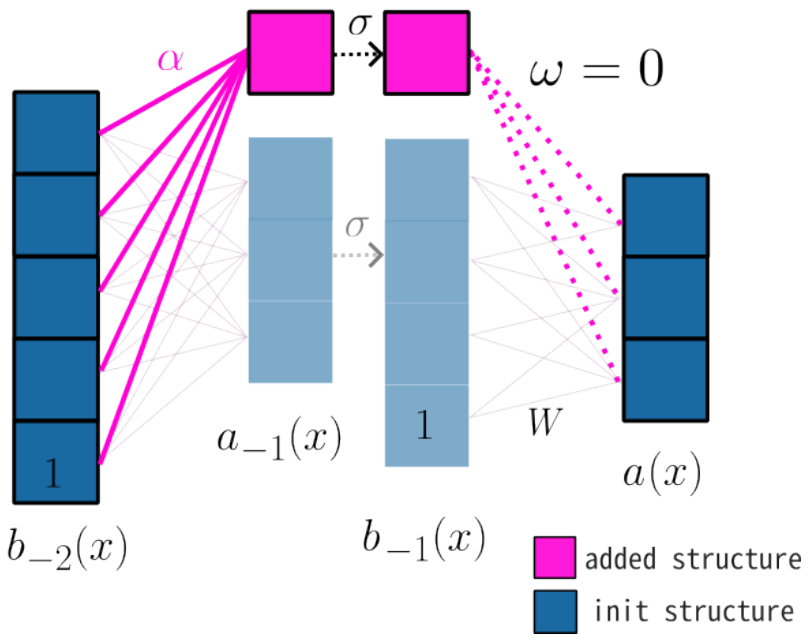


Figure: Thin feed forward network with two hidden layers.

GradMax & NORTH





GradMax initialization philosophy

objective : Accelerate the training of a network.

Updating the network parameters with $\delta\theta$:

$$\mathcal{L}^{t+1} = \mathcal{L}^t + \langle \nabla_\theta \mathcal{L}, \delta\theta \rangle + o(\|\delta\theta\|)$$

Choosing the gradient direction, i.e. $\delta\theta = -\eta \nabla_\theta \mathcal{L}$:

$$\mathcal{L}^{t+1} = \mathcal{L}^t - \eta \|\nabla_\theta \mathcal{L}\|^2 + o(\eta)$$

Idea : Maximizing the gradient according to the new parameters.

GradMax trick

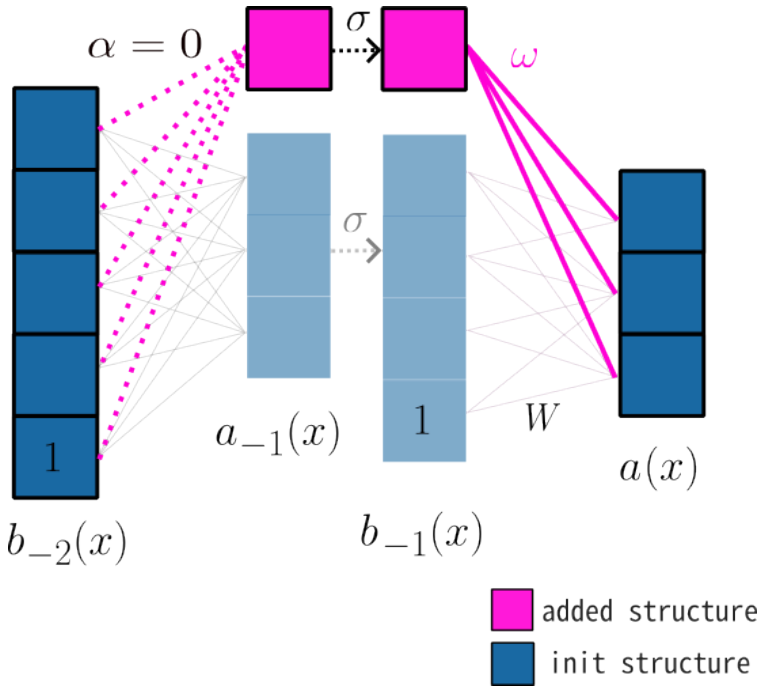
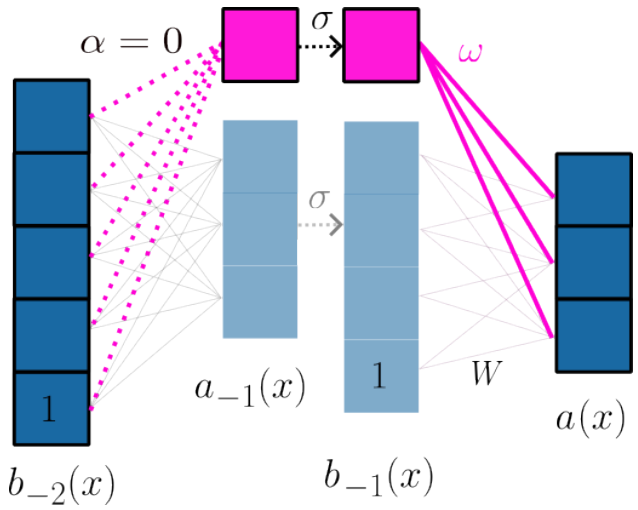


Figure: GradMax new neurons initialization.

$$\mathcal{L}^{t+1} \approx \mathcal{L}^{t} - \eta \left\| \nabla_{\theta} \mathcal{L} \right\|^2 - \eta \Big(\overbrace{\left\| \nabla_{\mathbf{\Omega}} \mathcal{L} \right\|^2}^{=0} + \overbrace{\left\| \nabla_{\mathbf{A}} \mathcal{L} \right\|^2}^{\text{quadratic in } \mathbf{\Omega}} \Big)$$

GradMax vs TINY



$$\boldsymbol{S} := \frac{1}{n}\boldsymbol{B}_{-2}\boldsymbol{B}_{-2}^T$$
$$\boldsymbol{N}_\perp := \frac{1}{n}\boldsymbol{B}_{-2}{\boldsymbol{V}_\perp}^T$$
$$\boldsymbol{N}_\nabla := \frac{1}{n}\boldsymbol{B}_{-2}{\boldsymbol{V}_\nabla}^T$$

GradMax	TINY
$\arg\max_{\boldsymbol{\Omega}} \left\|\nabla_{\boldsymbol{A}}\mathcal{L}(f_\theta)\right\|^2$ s.t. $\|\boldsymbol{\Omega}\| \leq c$	$\arg\max_{\boldsymbol{A},\boldsymbol{\Omega}} \left\langle \boldsymbol{A}\boldsymbol{\Omega}^T, \boldsymbol{N}_\perp \right\rangle$ s.t. $\left\|\boldsymbol{\Omega}\boldsymbol{A}^T\boldsymbol{B}\right\|^2 \leq c$
$\Downarrow$	$\Downarrow$
$\arg\max_{\boldsymbol{\Omega}} \left\|\boldsymbol{N}_\nabla\boldsymbol{\Omega}\right\|^2$ s.t. $\|\boldsymbol{\Omega}\|^2 \leq c$	$\arg\max_{\boldsymbol{\Omega}} \left\|\boldsymbol{N}_\perp\boldsymbol{\Omega}\right\|^2_{\boldsymbol{S}^{-1}}$ s.t. $\|\boldsymbol{\Omega}\| \leq \tilde{c}$

GradMax & NORTH

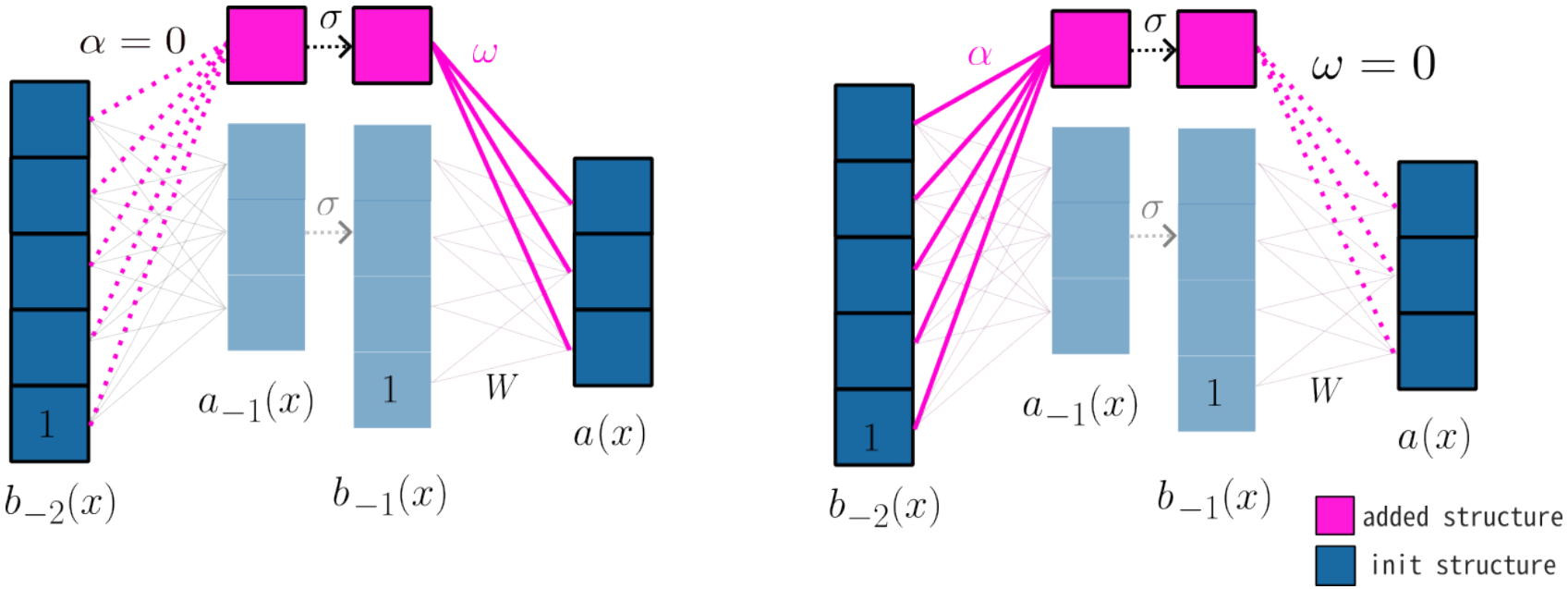
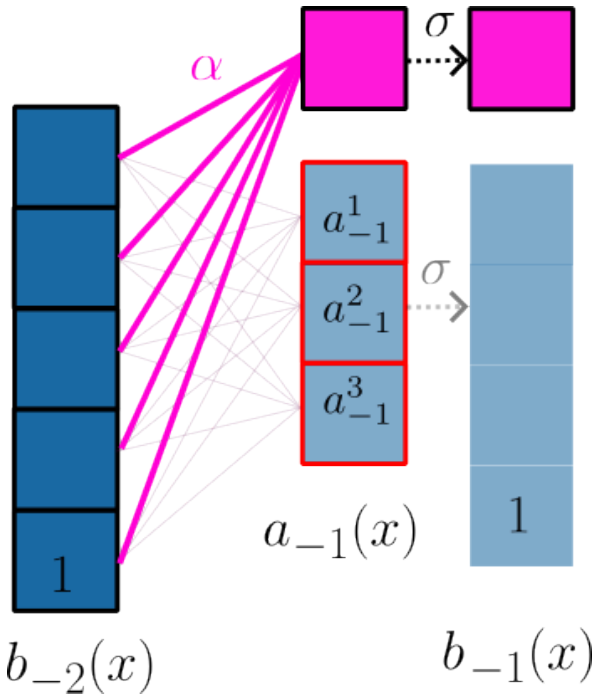


Figure: left : GradMax initialization; right : NORTH initialization

NORTH : activation-based



$$\boldsymbol{v}(., \boldsymbol{\alpha}) \in \{\boldsymbol{a}^1_{-1}(.), \boldsymbol{a}^2_{-1}(.), \boldsymbol{a}^3_{-1}(.)\}^{\perp}$$

NORTH vs TINY

$$\boldsymbol{S} := \frac{1}{n}\boldsymbol{B}_{-2}\boldsymbol{B}_{-2}^T$$

NORTH	TINY
$\boldsymbol{S}^+\boldsymbol{B}_{-2}\mathbf{V}_{\mathrm{Ker}(\boldsymbol{A}_{-1})}\boldsymbol{r}, \quad \boldsymbol{r} \sim \mathcal{N}(0, I)$	$\boldsymbol{S}^+\boldsymbol{B}_{-2}\overbrace{\mathcal{P}_{\mathrm{Ker}(\boldsymbol{B}_{-1})}\boldsymbol{V}_{\boldsymbol{\nabla}}{}^T}^{\boldsymbol{V}_{\perp}{}^T}\boldsymbol{v}_i$

Conclusion : Initialization differences

Input weights $\boldsymbol{A}$:

	NORTH	TINY
Init.	$\boldsymbol{S}^{+}\boldsymbol{B}_{-2}\mathbf{V}_{\mathrm{Ker}(\boldsymbol{A}_{-1})}\boldsymbol{r}, \quad \boldsymbol{r} \sim \mathcal{N}(0, I)$	$\boldsymbol{S}^{+}\boldsymbol{B}_{-2}\mathcal{P}_{\mathrm{Ker}(\boldsymbol{B}_{-1})}\,\boldsymbol{V}_{\nabla}{}^{T}\boldsymbol{v}_i$
Ortho.	✓	✓
Grad.	×	✓

Output weights $\boldsymbol{\Omega}$:

	GradMax	TINY
Init.	$\arg\max_{\boldsymbol{\Omega}, \|\boldsymbol{\Omega}\|^2 \leq c} \left\|\boldsymbol{N}_{\nabla}\boldsymbol{\Omega}\right\|^2$	$\arg\max_{\boldsymbol{\Omega}, \|\boldsymbol{\Omega}\| \leq \tilde{c}} \left\|\boldsymbol{N}_{\perp}\boldsymbol{\Omega}\right\|^2_{\boldsymbol{S}^{-1}}$
Ortho.	×	✓
Grad.	✓	✓

Comparaison on CIFAR-100

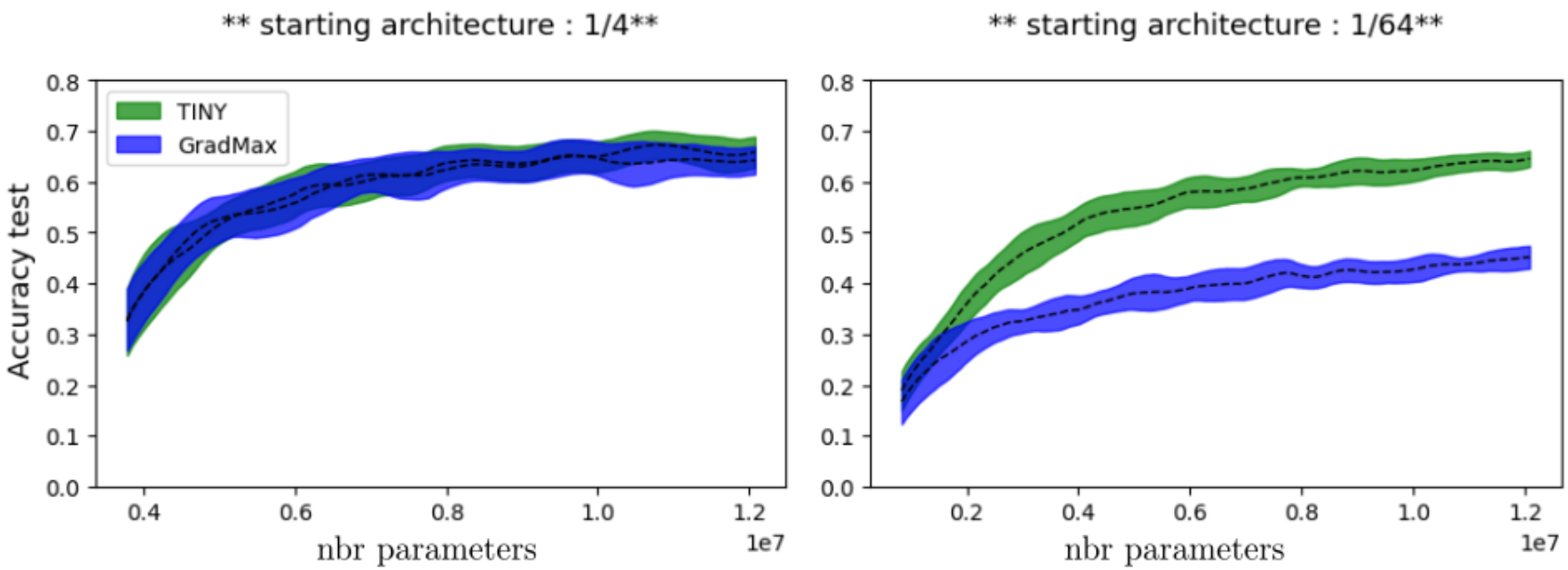


Figure: Test accuracy as a function of the number of parameters during architecture growth from ResNet$_s$ to ResNet18; 4 runs are performed for each setting.

The amplitude factor (go faster !)

The amplitude factor γ

A new neuron with input and output weights $\boldsymbol{\alpha}, \boldsymbol{\omega}$ is multiplied by the amplitude factor γ with the operation $\gamma(\boldsymbol{\alpha}, \boldsymbol{\omega})$ as :

$$(\boldsymbol{\alpha}, \boldsymbol{\omega}) \leftarrow (\sqrt{\gamma}\boldsymbol{\alpha}, \sqrt{\gamma}\boldsymbol{\omega}).$$

For an input $\boldsymbol{x}$, then :

$$\boldsymbol{v}(\boldsymbol{x}, (\boldsymbol{\alpha}, \boldsymbol{\omega})) = \sqrt{\gamma}\boldsymbol{\omega}\sigma(\sqrt{\gamma}\boldsymbol{\alpha}^T \boldsymbol{b}_{-2}(\boldsymbol{x})).$$

And for an input $\boldsymbol{x}$, if $\|\gamma\| \ll 1$, then :

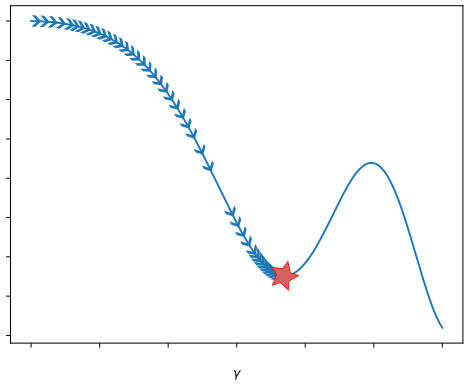
$$\boldsymbol{v}(\boldsymbol{x}, (\boldsymbol{\alpha}, \boldsymbol{\omega})) = O(\|\gamma\|).$$

The best amplitude factor (go faster !)

The best amplitude factor γ^*

For a batch $\boldsymbol{X}_\gamma$, we can compute the best amplitude factor γ^* as :

$$\gamma^* := \underset{\gamma \in [0,u]}{\arg\min} \sum_{\boldsymbol{x}_i \in \boldsymbol{X}_\gamma} \mathcal{L}(f_{\theta \oplus \gamma(\boldsymbol{A}, \boldsymbol{\Omega})}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$



For large γ^* :

$$\mathcal{L}(f_{\theta \oplus \gamma^*(\boldsymbol{A}, \boldsymbol{\Omega})}) = \mathcal{L}(f_\theta) - \gamma^* \frac{1}{n} \langle \boldsymbol{V}_\nabla, \boldsymbol{V}(\boldsymbol{A}, \boldsymbol{\Omega}) \rangle_{\mathrm{Tr}} + o(\gamma^*)$$

GradMax & NORTH : no impact

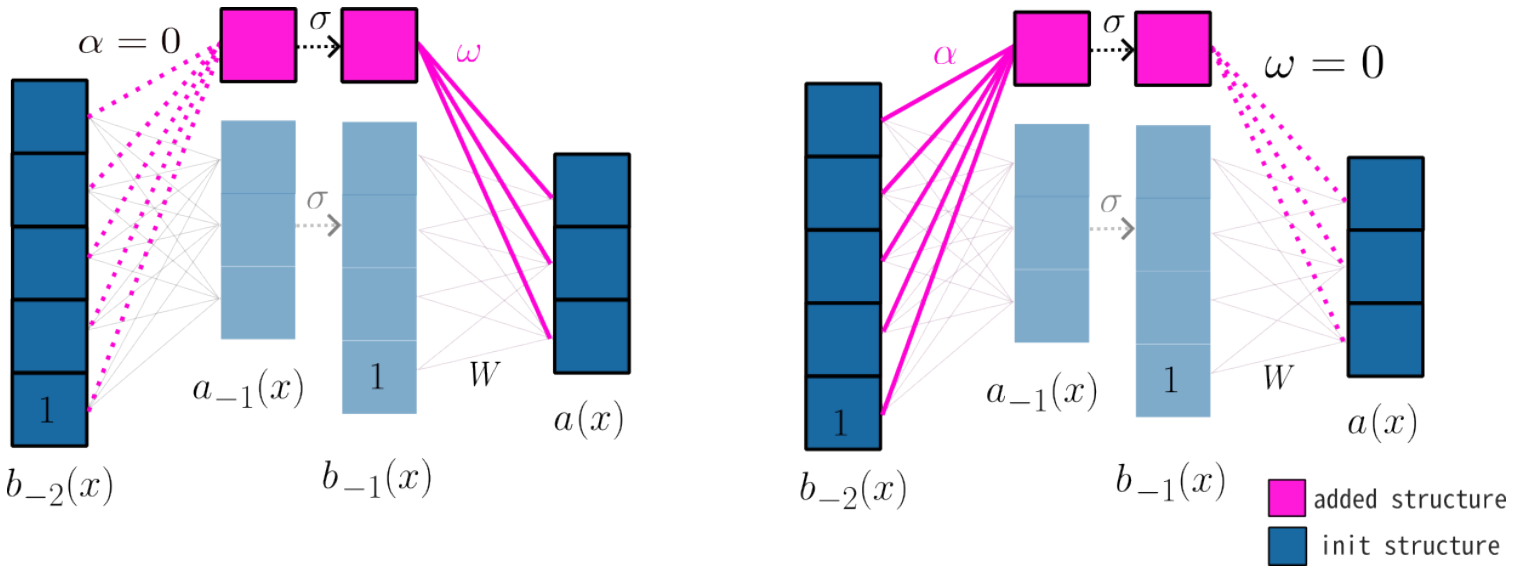


Figure: right : GradMax initialization; left : NORTH initialization

The amplitude factor for the strategy

Where to first add neurons ?

Using first order information, i.e. $\gamma \ll 1$:

$$l^* := \arg\max_l \lambda_1^l$$

Using non-linear information :

$$\Delta_l := \mathcal{L}(f_\theta) - \mathcal{L}(f_{\theta \oplus \gamma^*(\boldsymbol{A}, \boldsymbol{\Omega})})$$
$$l := \arg\max_l \Delta_l$$

Experiment : Algorithm (2)

Algorithm 1: Grow a network with the strategy s

Data: $AddStrat =$ depth selection strategy
Result: Neural Network N
Initialize a *thin* network with *few* neurons per layer;
for t *in* $[1, ..., T]$ **do**
    $depth = AddStrat(t)$;
    Compute the first neuron α_1, ω_1 of at $depth$;
    Compute the amplitude factor γ^* for this new neuron;
    Multiply the neuron with its amplitude factor;
    Add it to the architecture;
    Evaluate the performances of the network;
    for d *in [1, ..., deep]* **do**
        Compute the best update at layer d and its amplitude factor;
        Multiply the best update with its amplitude factor;
        Update the architecture with this best update;
        Evaluate the performances of the network;
    end
end

Grow

Train

Experiment : Algorithm (2)

Algorithm 2: Select depth to grow the network

Data: $s =$ Strategy, $t =$ iteration
Result: $depth$ where to add a neuron

if $s =$ *Naive* **then**
  $depth = np.mod(t, deep) + 1$;

else if $s =$ *MaxEigenValue* **then**
  for d *in [1, ..., deep]* **do**
    Compute the first neuron α_1, ω_1 at d;
    Compute the first order information $\lambda_1^d = \langle \boldsymbol{V}, \boldsymbol{V}_\perp \rangle$ at d;
  end
  $depth = \arg\max_{d=1,\dots,deep}\{\lambda_1^d\}$;

else if $s =$ *AmplitudeFactor* **then**
  for d *in [1, ..., depth]* **do**
    Compute the first neuron α_1, ω_1 at d;
    Compute the amplitude factor γ^* for this new neuron;
    Evaluate the decrease of loss Δ_d if the architecture is expanded with $(\sqrt{\gamma^*}\alpha_1, \sqrt{\gamma^*}\omega_1)$
  end
  $depth = \arg\max_{d=1,\dots,deep}\{\Delta_d\}$;

return $depth$;

Experiment (2)

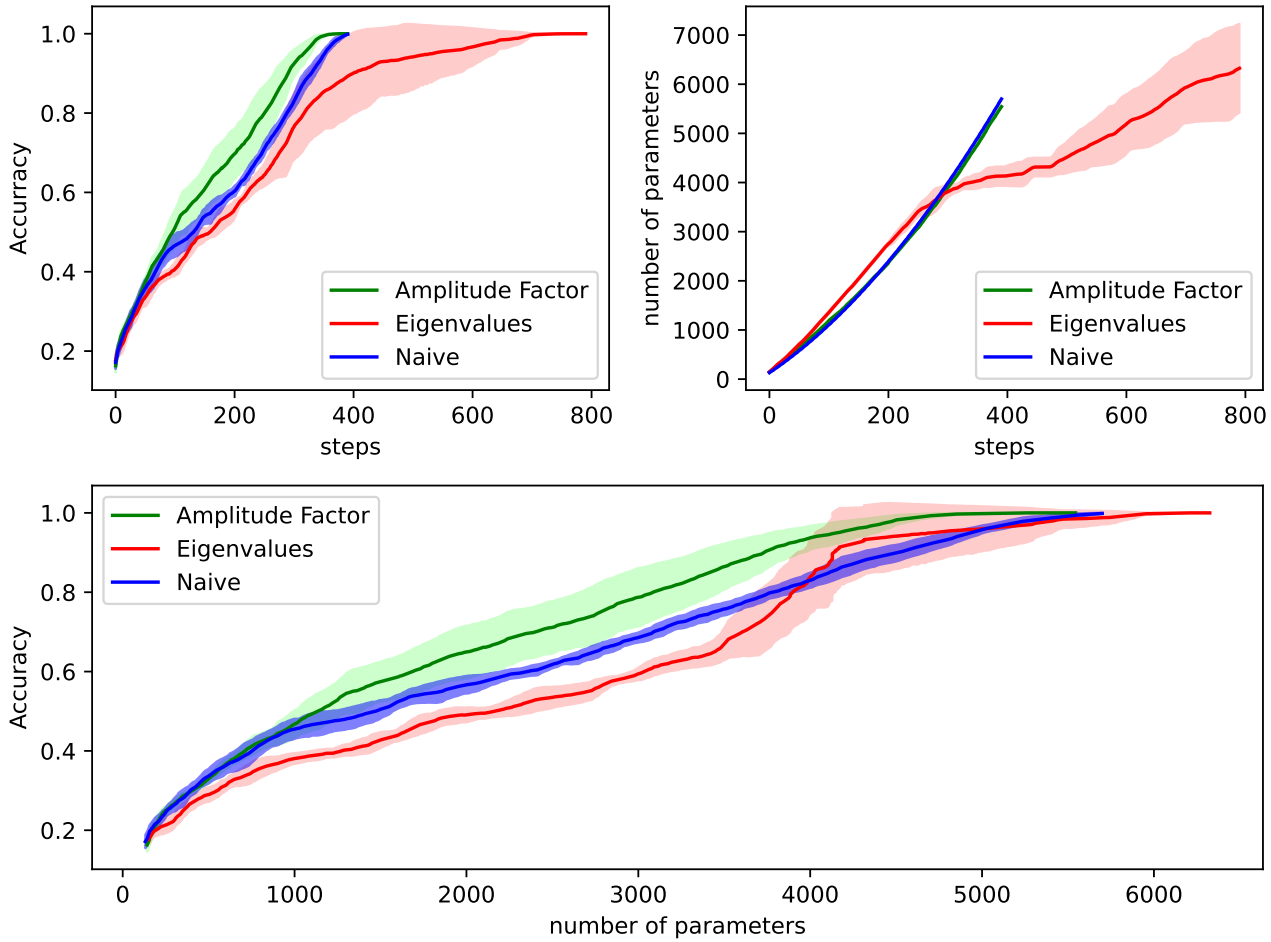


Figure: Accuracy and number of parameters of growing networks with different addition strategies for $\boldsymbol{x} \sim \mathcal{N}(0, I_{50})$ and $\boldsymbol{y} = \arg\max_{k=1,\dots,10} \left\{ \sin(k \sum_{j=1}^{50} \boldsymbol{x}[j] \right\}$.

Conclusion

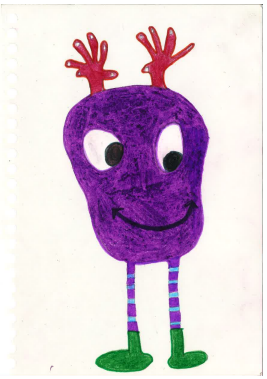
- The expressivity bottleneck to update the parameters and initialize the new neurons
- First order information to select the new neurons at a given layer
- Formal comparison between TINY, GradMax and NORTH
- The amplitude factor to select the depth at which neurons should be added first and build a strategy across layers.

Open-source TINYpub, statistical significance of the estimators, MNIST, CIFAR10

Growing Tiny Networks:
Spotting Expressivity Bottlenecks and Fixing Them Optimally Verbockhaven et al. [2024]
Growth strategies for arbitrary DAG neural architectures Douka et al. [2025]

Neural Architecture Search

Suppose that God's apprentices would like to have a personal chef and decide to create it. To do so, they need to *assemble parts of the human body* into a human puppet, which will be *trained* to cook. Considering all possible layouts of the body sections, they try a first idea and place the hands on top of the head. After several hours of training, they observe, quite embarrassed, that their puppet is incapable of cooking the simplest recipes. One apprentice suggests that this failure might be linked to the puppet's inability to see its actions while using its hands. Considering this remark, they decide to revise the joining of the puppet and restart the training...



Replace the action *assemble the parts of the human body* by *construct the architecture of a neural network*, and the apprentices become a computer program of some researchers in the NAS field.

$\mathbb{A}$ $\mathbf{A}$ A $\mathring{A}$ A A $\mathtt{A}$

$\mathbb{L}$ $\mathbf{L}$ L $\mathring{L}$ L L $\mathtt{L}$

$\prec$ $\mathbf{l}$ l $\mathring{l}$ l l l $\mathtt{l}$

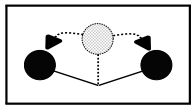
Method	Indicators				Dataset
	C_0	C_∞	T	P	
DARTS $\not\subset\!\!\!\!\!\!=\not\subset$	$\times$	$3.3e6$	$1.5+2.5$	97.0 ± 0.1	CIFAR-10
	$\times$	$3.3e6$	$4+2.5$	97.8 ± 0.1	
	$\times$	$4.7e6$	4	73.3	ImageNet
Firefly 	$1.4e6$	$1.4e7$	≈ 1 ?	~ 94	CIFAR-10
	$\times$	$5.8e5$	$\times$	~ 71	CIFAR-100
GradMax $0 \mid \|\nabla\mathcal{L}\|^2$	$1e6$	$9e6$	$\times$	84.4 ± 0.4	CIFAR-10
	$1e5$	$4e5$	$\times$	91.1 ± 0.1	
	$1e5$	$4e5$	$\times$	66.8 ± 0.2	CIFAR-100
	$1e6$	$4.2e6$	$\times$	68.6 ± 0.2	ImageNet
NORTH $\mathcal{X}_\perp \mid 0$	$1e6$	3e7	3.3	~ 76.2	CIFAR-10
	$3e4$	$3e5$	0.4	~ 82.5	
	$1e6$	$3e7$	1.6	~ 55.2	CIFAR-100

Figure: T : GPU days, P : accuracy on test (%) by-hand mean on all tested strategy, see figure 4 of paper Maile et al. [2022] for more precise results.

NAS : the search space

Search Space $\mathbb{A}$

$$\mathbb{A} := \{\text{macro}(\text{micro}) \mid \text{micro} \in S,\ \text{macro} \in S'\} \tag{3}$$

A layer, noted , is of the form $\boldsymbol{x} \to \sigma(\boldsymbol{W}\boldsymbol{x})$.

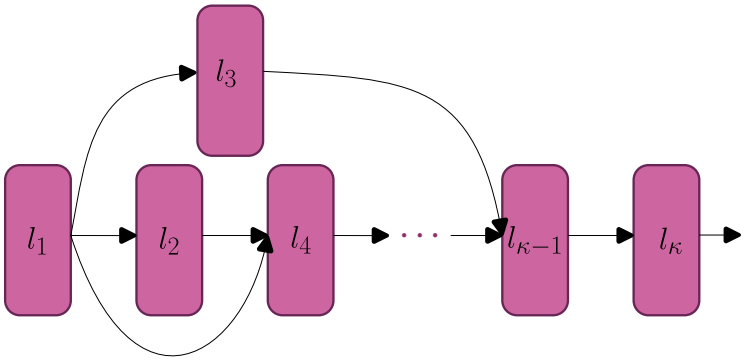


Figure: Example of a chain-structured architecture.

NAS : the search space

Search Space $\mathbb{A}$

$$\mathbb{A} := \{\text{macro}(\text{micro}) \mid \text{micro} \in S,\ \text{macro} \in S'\} \tag{3}$$

A layer, noted , is of the form $x \to \sigma(Wx)$.

A cell, noted , is a composition of layers.

Figure: Example of a chain-structured architecture.

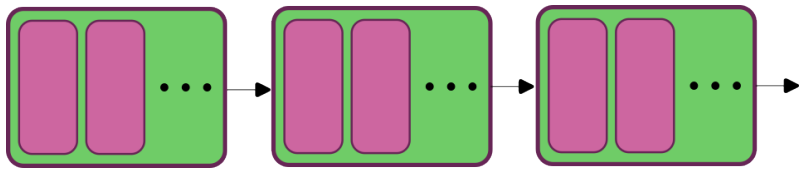


Figure: Example of a cell-based architecture.

Definition

- The dataset $\mathcal{D} := \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^n$
- the neural network or predictor f
- the class of functions $\mathcal{C} := \{f \text{ with architecture } \mathcal{A}\}$

Expr. & Compl. : noise fitting

$$\mathcal{C} := \{f \text{ for } f \in \mathcal{F}\} \quad \text{and} \quad f(\boldsymbol{x}) \in \{a, b\}$$

The Vapnik–Chervonenkis (VC) dimension

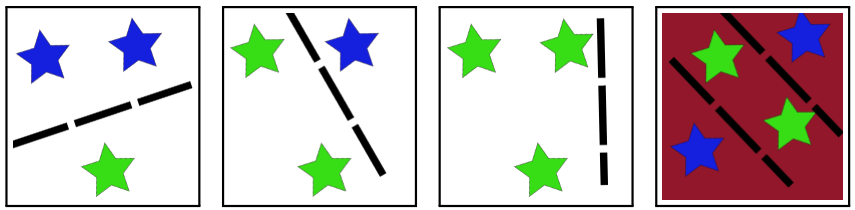


Figure: Example of VC-dimension when $\mathcal{F}$ is the class of linear separator in a two-dimensional plane.

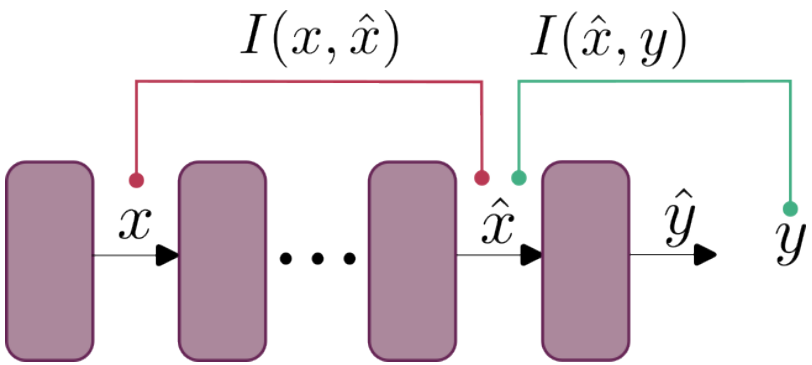
Rademacher complexity

Let $\sigma \sim \mathcal{B}(\frac{1}{2})$, then :

$$\mathbb{E}_{\mathcal{D}}\left[\frac{1}{n}\widehat{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{f\in\mathcal{C}} \boldsymbol{\sigma}^T f(\boldsymbol{X}))\middle|\mathcal{D}\right]\right]$$

Expr. & Compl. : minimization criterion

the Information Bottleneck



$$IB(f) := I(\boldsymbol{x}, \hat{\boldsymbol{x}}) - \beta I(\hat{\boldsymbol{x}}, \boldsymbol{y}).$$

the Kolmogorov Complexity

$$0\,1\,1\,0\,0\,0\,\ldots\,1\,1$$

shortest binary program generating f_θ

$H(\boldsymbol{x})$ the Shannon entropy, then $I(\boldsymbol{x}; \hat{\boldsymbol{x}}) := H(\boldsymbol{x}) - H(\boldsymbol{x}|\hat{\boldsymbol{x}})$.

From TINY to GradMax

First order development of the loss :

$$\mathcal{L}(f_{\theta\oplus(\boldsymbol{A},\boldsymbol{\Omega})}) := \frac{1}{n}\sum_i \ell(f_\theta(\boldsymbol{x}_i))$$

$$= \mathcal{L}(f_\theta) - \frac{1}{n}\langle \boldsymbol{V}, \boldsymbol{V}_{\nabla}\rangle_{\mathrm{Tr}} + o(\frac{1}{n}\|\boldsymbol{V}\|)$$

The expressivity bottleneck :

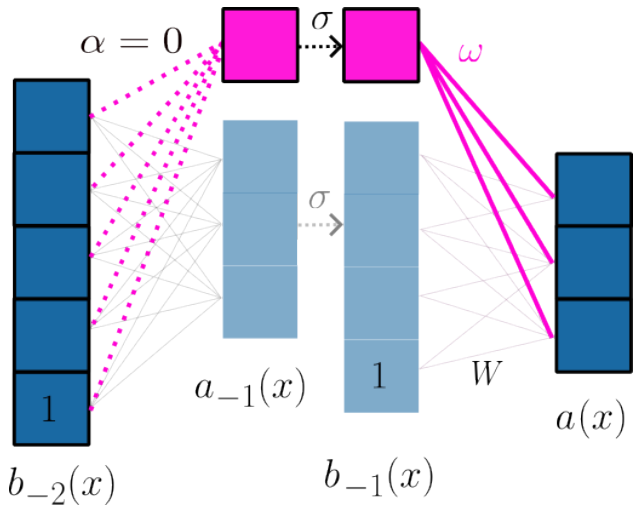
$$\Psi = \min_{\delta\theta} \frac{1}{n}\|\boldsymbol{V}((\boldsymbol{A},\boldsymbol{\Omega})) - \boldsymbol{V}_\perp\|^2_{\mathrm{Tr}}$$

Proposition

$\forall \boldsymbol{D}\in\mathbb{R}^{p,q}, \boldsymbol{B}\in\mathbb{R}^{k,q},$

$$\exists \tilde{c}\in\mathbb{R} \quad \text{s.t,} \quad \arg\min_{\boldsymbol{H}} \|\boldsymbol{D}-\boldsymbol{H}\boldsymbol{B}\|^2 = \arg\max_{\boldsymbol{H},\ \|\boldsymbol{H}\boldsymbol{B}\|^2\leq\tilde{c}} \langle \boldsymbol{D}, \boldsymbol{H}\boldsymbol{B}\rangle$$

GradMax initialization



$$\boldsymbol{S} := \frac{1}{n}\boldsymbol{B}_{-2}\boldsymbol{B}_{-2}^T$$
$$\boldsymbol{N}_{\perp} := \frac{1}{n}\boldsymbol{B}_{-2}\boldsymbol{V}_{\perp}^T$$
$$\boldsymbol{N}_{\nabla} := \frac{1}{n}\boldsymbol{B}_{-2}\boldsymbol{V}_{\nabla}^T$$

GradMax initialization :

$$\boldsymbol{\Omega}^* := \arg\max_{\boldsymbol{\Omega}} \|\boldsymbol{N}_{\nabla}\boldsymbol{\Omega}\|^2 \quad s.t. \quad \|\boldsymbol{\Omega}\|^2 \leq c \tag{4}$$

TINY initialization :

$$\exists \tilde{c} \qquad \boldsymbol{\Omega}^* := \arg\max_{\boldsymbol{\Omega}} \|\boldsymbol{N}_{\perp}\boldsymbol{\Omega}\|_{\boldsymbol{S}^{-1}}^2 \quad s.t. \|\boldsymbol{\Omega}\| \leq \tilde{c} \tag{5}$$

Experiment : Algorithm (1)

Algorithm 3: Grow a network with *InitStrat*.

Data: $InitStrat =$ Initialization strategy
Result: Neural Network N
Initialize a *thin* network with *few* neurons per layer;
for t *in* $[1, ..., T]$ **do**
    $depth = np.mod(t, deep) + 1$;
    Using InitStrat, compute the first neuron α_1, ω_1 of at $depth$;
    Compute the amplitude factor γ^* for this new neuron;
    Multiply the neuron with its amplitude factor;
    Add it to the architecture;
    Evaluate the performances of the network;
end

} Grow

Experiemtn (1) : Random vs TINY

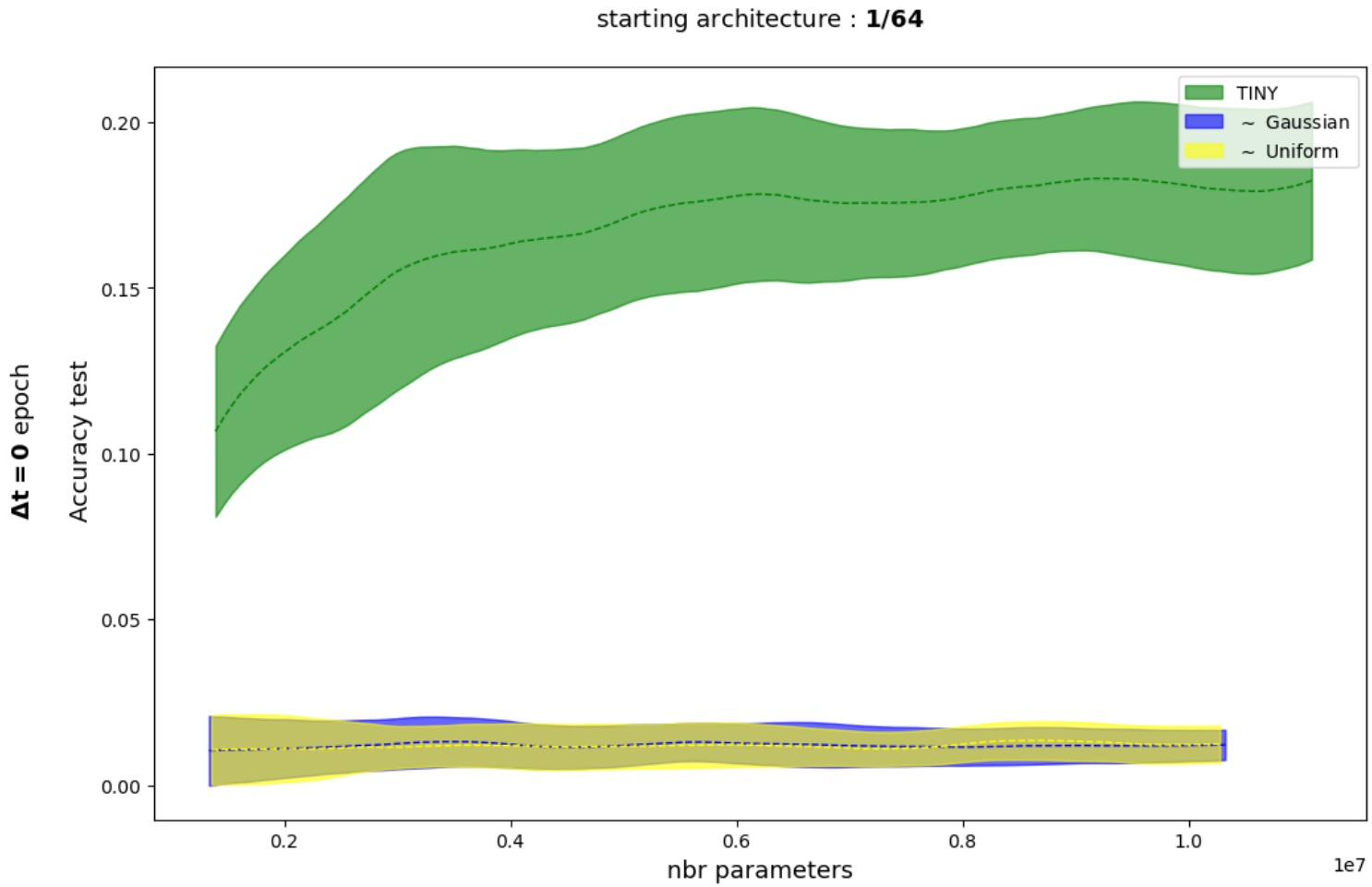


Figure: Test accuracy as a function of the number of parameters during pure architecture growth (no gradient steps performed, only neuron addition with a given initialization) from $\text{ResNet}_{1/64}$ to ResNet_{18}, averaged over four independent runs.

Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019. doi: 10.1109/TC.2019.2914438.

Stella Douka, Manon Verbockhaven, Théo Rudkiewicz, Stéphane Rivaud, François P Landes, Sylvain Chevallier, and Guillaume Charpiat. Growth strategies for arbitrary dag neural architectures, 2025. URL https://arxiv.org/abs/2501.12690.

Utku Evci, Bart van Merrienboer, Thomas Unterthiner, Fabian Pedregosa, and Max Vladymyrov. Gradmax: Growing neural networks using gradient information. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=qjN4h_wwUO.

Kaitlin Maile, Emmanuel Rachelson, Hervé Luga, and Dennis George Wilson. When, where, and how to add new neurons to ANNs. In *First Conference on Automated Machine Learning (Main Track)*, 2022. URL https://openreview.net/forum?id=SW0g-arIg9.

Manon Verbockhaven, Théo Rudkiewicz, Guillaume Charpiat, and Sylvain Chevallier. Growing tiny networks: Spotting expressivity bottlenecks and fixing them optimally. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=hbtG6s6e7r.

Lemeng Wu, Bo Liu, Peter Stone, and Qiang Liu. Firefly Neural Architecture Descent: a General Approach for Growing Neural Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 22373–22383. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fdbe012e2e11314b96402b32c0df26b7-Abstract.html.