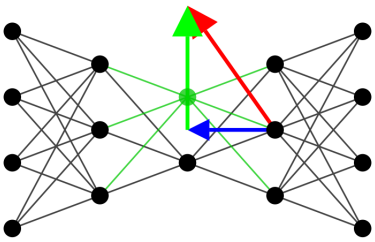
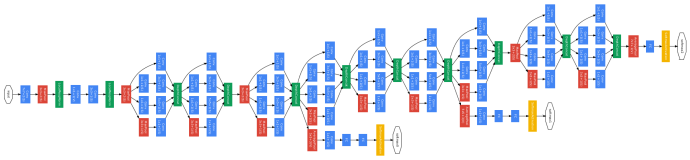
Neural Architecture Growth for Frugal Learning



Punchline: We aim at training tiny neural networks with a flexible architecture that adapts and grows on the fly while training, in order to reduce AI environmental footprint.



Where is a supplementary block most needed to increase the performance?

Keywords: neural networks, frugal AI, differential calculus, optimization, expressive power

université PARIS-SACLAY · LISN LABORATOIRE INTERDISCIPLINAIRE DES SCIENCES DU NUMÉRIQUE · Inria · τ · MANOLO

1 Internship context

Research lab: INRIA TAU team (joint team between INRIA, CNRS and LISN of Université Paris-Saclay)
Location: LISN (building 660 "Digiteo", at Université Paris-Saclay)
Supervision team: Guillaume Charpiat, Sylvain Chevallier, Alex Davey, Stella Douka, François Landes, Stéphane Rivaud, Théo Rudkiewicz and Alena Shilova
Contact: tau-frugal@inria.fr
Funding: European project MANOLO

2 Problem statement

Thematic context Deep learning has shown impressive, highly-mediatized results on various applications (the game Go, StarCraft, translation, object detection in images, high-resolution image generation, text generation...), obtained at the cost of training huge neural network architectures, which therefore also takes time and money (for instance, GPT-3 has 10^{11} parameters and might cost millions of dollars to be trained), both at training and exploitation times. Frugal learning, on the opposite, consists in training with as few samples or as little computational power (Green AI) as possible. We will focus on the latter here.

Large vs. small One advantage of having many neurons per layer is that it is known (experimentally and theoretically [GJS+20]) to facilitate optimization during training, thus yielding better results. However, trained neural networks show high internal redundancy, and various techniques have been developed to squeeze them into smaller networks with comparable accuracy. For instance [LUW17] manages to divide by 100 the number of neurons in order to run online object recognition in videos on a smartphone. On the other extreme, training and applying "tiny" models (with "only" 10^5 parameters) will be much faster (as each test of the network is of much smaller complexity) but they might suffer from a lack of expressivity, preventing them from fitting data accurately.

Change of paradigm In our work published in TMLR [VRCC24], we propose to start with the simplest possible neural network (i.e., one neuron, a.k.a., linear regression) and make the network grow according to the information brought by the backpropagation pass. Such information can indeed be used to go beyond usual limitations in small network training, to explicitly

tackle potential optimization and expressivity issues. To do this, we locate precisely, while training, the learning bottlenecks of a neural network, i.e., the layers which lack expressive power, in order to boost them by adding neurons or new layers appropriately where and when it is needed. With such an iterative architecture refinement scheme, important gains in architecture search (auto-DL) are expected. Indeed current Neural Architecture Search methods consist in trying many different architectures, while with our approach a single training progressively grows the architecture.

3 State of the art

Work done on this topic in our team so far A PhD student (Manon Verbockhaven) formalized mathematically the concepts required to spot and fix expressivity bottlenecks. She also implemented layer growth of fixed convolutional and fully-connected architectures. More details concerning this methodology can be found in [VRCC24]. A master 2 intern (Barbara Hajdarevic) extended this work to layer graph growth, i.e. adding new layers to the computational graph. This is a necessary step to unleash the power of the approach and to check how it performs. Currently, 2 PhD students (Stella Douka and Théo Rudkiewicz) and 2 Post-Doctoral researchers (Stéphane Rivaud and Alex Davey) are extending neural network growth to convolutions, ResNet blocks, attention mechanisms, and studying growth strategies. Our networks are growing and so is our team.

State of the art and other methods The main approaches to full neural network architecture optimization are based on automatic hyper-parameter tuning (auto-DL), but they are computationally extremely demanding, as they run many tries on many architecture variations. Some approaches incorporate architecture flexibility in their design [LSY19]; they can however only suppress connections between existing blocks, or suppress blocks, but not add new ones. Only a few approaches try to grow architectures (such as [MRLW22]); unfortunately they are most often based on ad-hoc criteria which are not mathematically justified.

4 Scientific proposal

There are many open axes of research that we would like to tackle. We therefore offer multiple internship positions and research tracks, described below.

Note that the precise internship topics will be tailored to the candidates' skills and preferences.

- **How much training and growth?**
  - **When to add neurons? Training/growth compromise.** We want to **study the trade-off** between **training by gradient descent** and **architecture growth** (neuron or layer additions). In practice indeed, we approximate the problem to solve (optimal move of existing parameters and optimal neuron addition) at first order, and as the solutions found are not perfect for the real, non-linear initial problem, we complete training with a standard gradient descent. The less we train between two neuron additions, the higher quality the new neurons found have to be, and reciprocally. How to choose a good trade-off ?
  - **Complexity/performance compromise.** Ideally, to decide whether to add the best possible neuron (or layer), we could consider a trade-off (or the Pareto front) between **performance gain** and **additional computational complexity**, inspired by Information Theory (Minimum Descriptor Length paradigm, Kolmogorov complexity, BIC), as well as by statistical significance.
- **Where to add neurons?**
  - **Algorithmics for expressivity bottlenecks localization**. Backpropagation properties might be better exploited to design new algorithms to identify expressivity bottlenecks faster. Ideally, predict the location where the expressivity bottleneck is maximized **without having to calculate it** at each location.
  - **Reinforcement Learning approach.** Define features to efficiently describe the state of the model (size, expressivity bottlenecks, architecture, complexity, performance, etc.). This can then be used by Reinforcement Learning algorithms (or other exploration tools) to define a growth strategy to develop the architecture, instead of the current greedy approach (adding where the expressivity bottleneck is the highest, based on first-order approximations), which experimentally is not optimal.
- **Introducing stochastics.**
  - **Monte Carlo approach.** The current algorithm is a greedy one, attempting to increase performance at each addition, adding

always where and how it is optimal to do so. An alternative approach is to **propose random additions** (random location, random initial weights, etc) and **accept them with some probability** related to the performance gain, in a Monte-Carlo way $p = \max(1, e^{\text{Gain}/T})$ (*i.e.* allow for detrimental additions, with Gain < 0). The randomness may help better explore architectures, and can be controlled by the "temperature" T.

- **Robustness of statistical estimators.** The expressivity bottleneck is estimated using a mini batch of data of a given size. We want to study the **statistical robustness** of this estimation and provide a theoretical bound, in order to consider as few samples as possible, hence speeding up computations, while thereby adding a reasonable amount of noise.

- **Applications to Reinforcement Learning.** There has been a recent surge of interest in studying the problem of *plasticity loss* in Reinforcement Learning: the policy quickly overfits to initial observations and fails to adapt to new data as training progresses. Proposed solutions include periodically resetting the neural network or using various forms of regularization (see e.g. [NSD+22]). Most recently, dynamic growth methods have been explored [LOCP25], but these are heuristic and lack a clear theoretical understanding. We want to study more principled growth criteria such as the expressivity bottleneck [VRCC24] in this RL context, as well as their impact on neural network plasticity. Extensions of this work could involve Continual Learning, where a network must perform well on a series of (potentially unrelated) new tasks without forgetting old ones.

- **Dynamic architectures.** (One of the tasks of) Frugal AI consists in searching for neural networks that will consume as few computational resources as possible at run time. We are currently searching for small architectures (in terms of flops), but we could consider **larger architectures, of which only a part is executed for each sample** at run time, that is, only the branches of the network that are needed for that particular sample are run. An example is the mixture of experts in the MLP part of Transformers: modern versions select only the most suitable experts independently for each token. How to grow such a dynamic architecture ?

Note that we already have a well-defined **theoretical framework** [VRCC24], applicative results [DVR+25] and a **pytorch library**

(`https://github.com/growingnet/gromo`) actively developed and have defined **several experimental test cases**.

5 Expected results

Depending of the subject, the results can be of theoretical nature, or empirical, or both.
A theoretical result could for instance provide new guarantees to a used algorithm. It could also give useful insight on the growing process which could later be used to improve the method.
Regarding empirical results, the method developed would be tested on various standard image classifications benchmarks (CIFAR, ImageNet), NAS Challenge datasets and should be compared with existing methods from the team and the literature. In addition, the experiments should be reproducible and the new code should be documented to be reusable.

6 Expected skills

The required skills are the central ones for any ML researcher both on the theoretical and technical sides :

- Python, PyTorch, Git (could be learned on the fly)
- maths: linear algebra (SVD...), differential calculus, statistics...

Many other skills can be used and learned during the internship (see footnotes for some of the MVA related courses):

- Optimisation techniques [1]
- Properties of small or big networks [2]
- Expressivity of neural networks and general notions of expressivity (VC-dimension, Rademacher complexity) [3]
- Estimation of carbon footprint of a computation [4]

[1] Convex optimisation, Computational Statistics
[2] Deep learning in practice
[3] Fondements Théoriques du deep learning, Introduction to statistical learning
[4] Intelligence Artificielle et Environnement

- Deep understanding of computational cost of neural network training and linear algebra operation on GPU [5]
- Many software development skills (continuous integration, documentation, use of cluster (slurm), ...) to improve our open-source implementation. [6]

References

[DVR+25] Stella Douka, Manon Verbockhaven, Théo Rudkiewicz, Stéphane Rivaud, François Landes, Sylvain Chevallier, and Guillaume Charpiat. Growth strategies for arbitrary dag neural architectures. *European Symposium on Artificial Neural Networks*, 01 2025.

[GJS+20] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, February 2020.

[LOCP25] Jiashun Liu, Johan Obando-Ceron, Aaron Courville, and Ling Pan. Neuroplastic Expansion in Deep Reinforcement Learning, June 2025.

[LSY19] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search, April 2019.

[LUW17] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian Compression for Deep Learning, November 2017.

[MRLW22] Kaitlin Maile, Emmanuel Rachelson, Hervé Luga, and Dennis G. Wilson. When, where, and how to add new neurons to ANNs, May 2022.

[NSD+22] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The Primacy Bias in Deep Reinforcement Learning, May 2022.

[5] Geometric data analysis

[6] Reproducible research project, Fundamentals of reproducible research and free software

[VRCC24] Manon Verbockhaven, Théo Rudkiewicz, Sylvain Chevallier, and Guillaume Charpiat. Growing tiny networks: Spotting expressivity bottlenecks and fixing them optimally. *Transactions on Machine Learning Research*, 2024.