

Learning rate transfer and feature learning across depth for constrained spectral optimizers: Complete Scion

Eline Pot, PhD Student

L. Lefebvre, M. Segond (Framatome)

A. Silveti-Falls (CVN, CentraleSupélec)

DSAM CC



Confidentiality



This document contains *Framatome's know-how*
When the box is checked, add the information in the 'Footer'

Export Control COEDM

EU EC country code = FR marking code = 0E001	US EC marking code = N	OTH EC country code = N marking code = N
--	---------------------------	--

Goods marked EU EC different to "N" are subject to export authorization of the States members of the European Union (EU) in case of export outside EU, and also for certain sensitive goods (e. g. goods marked "EU EC": "0E001") in case of transfer within UE ; they cannot be reexported nor transferred back without a prior agreement from the State of the first export. Goods marked US EC different to "N" are subject to export, re-export or transfer authorization of the United States of America. Goods marked OTH EC (country code(s)) different to "N" are subject to export, re-export or transfer authorization of the mentioned country(ies). Even for goods marked "EU EC: N" and "US EC: N" and "OTH EC: N", an export authorization may be required depending on their final destination or end-use.

Export Control Other

AL = N	ECCN = N
--------	----------

Goods labeled with "AL not equal to N" are subject to European or German export authorization when being exported within or out of the EU.
Goods labeled with "ECCN not equal to N or EAR99" are subject to U.S. reexport authorization. Even without a label, or with label: "AL: N" or "ECCN: N" or "ECCN: EAR99," authorization may be required due to the final whereabouts and purpose for which the goods are to be used.

This document and any and all information contained therein and/or disclosed in discussions supported by this document, are confidential, protected by applicable intellectual property regulations and contain data subject to trade secrets regulations. Any reproduction, alteration, disclosure to any third party and/or publication in whole or in part of this document and/or its content is strictly prohibited without prior written express approval of Framatome. This document and any information it contains shall not be used for any other purpose than the one for which they were provided. Legal and disciplinary actions may be taken against any infringer and/or any person breaching the aforementioned obligations.

Framatome's information protection rules



C1 : This document and any and all information contained therein and/or disclosed in discussions supported by this document are **restricted**.



C2 : This document and any and all information contained therein and/or disclosed in discussions supported by this document are sensitive and **Framatome confidential**, such as its disclosure, alteration or loss are detrimental with a significant-to-high impact for Framatome.

The document, if disclosed, and any information it contains are intended for the sole attendees. The disclosure or reference to such information or document shall be made only on a proper judgment basis and by mentioning expressly "this information shall not be disclosed / transferred without prior consent".

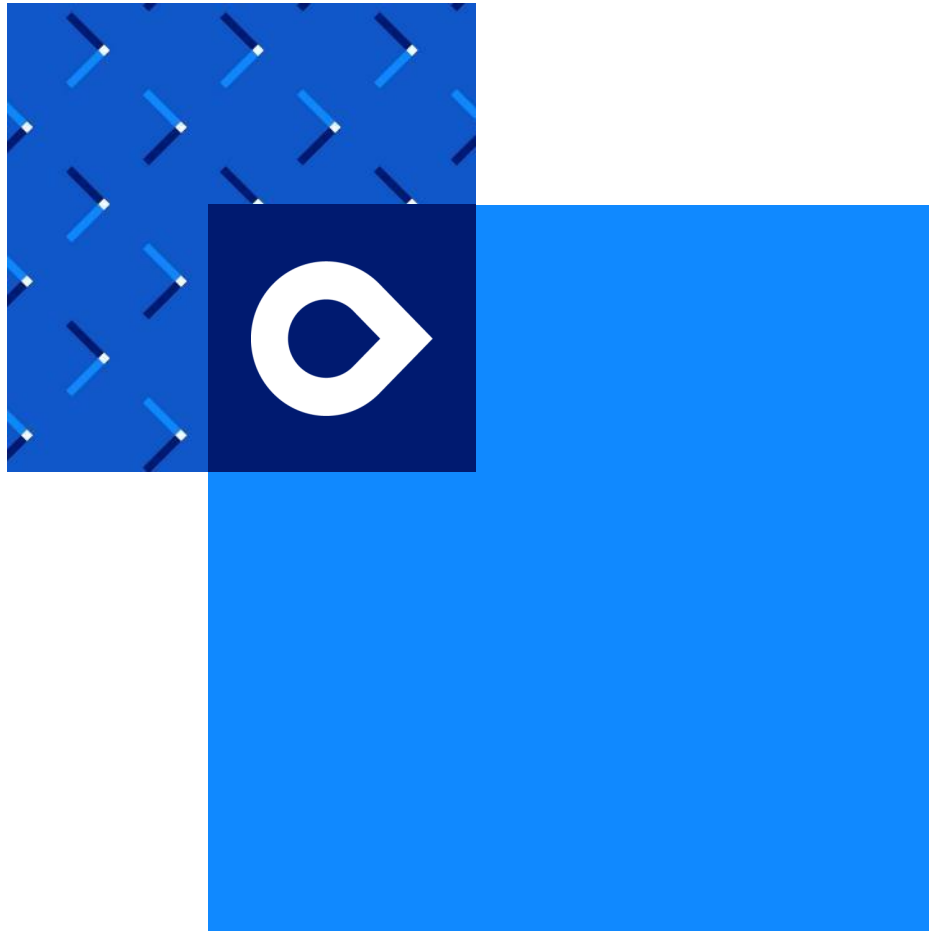


C3 : This document and any and all information contained therein and/or disclosed in discussions supported by this document are classified **Framatome Secret**. Each one must commit to keep secret any written or oral information disclosed during the meeting. It is forbidden to disclose it to any legal entity and any individual (including within Framatome) without prior consent of the meeting chairman.

Dissemination perimeter: specify a perimeter as narrow as possible.

Content

1. Introduction
2. Robustness & Lipschitz Neural Networks
3. SCION: Stochastic Conditional Gradient with Operator Norm
4. Hyperparameter Transfer
5. Conclusion & Perspectives



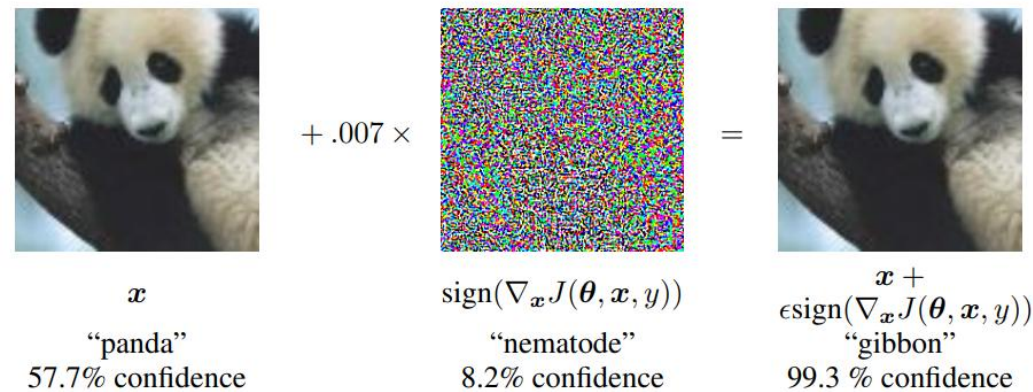
1.

Introduction

Introduction

Advances on Deep Neural Networks (DNNs)

- Over the last decade, neural networks have expanded across science and real-world applications, making confidence in their predictions increasingly important
- DNNs lack inherent **uncertainty quantification**
- Ensuring **robustness** in DNNs is a critical challenge
 - In industrial settings, models can face operation conditions not covered in training data
 - DNNs can suffer from **adversarial attacks**



[Ian J. Goodfellow, et al. "Explaining and Harnessing Adversarial Examples". In: 3rd International Conference on Learning Representations, ICLR 2015]



2.

Robustness & Lipschitz Continuous Neural Networks

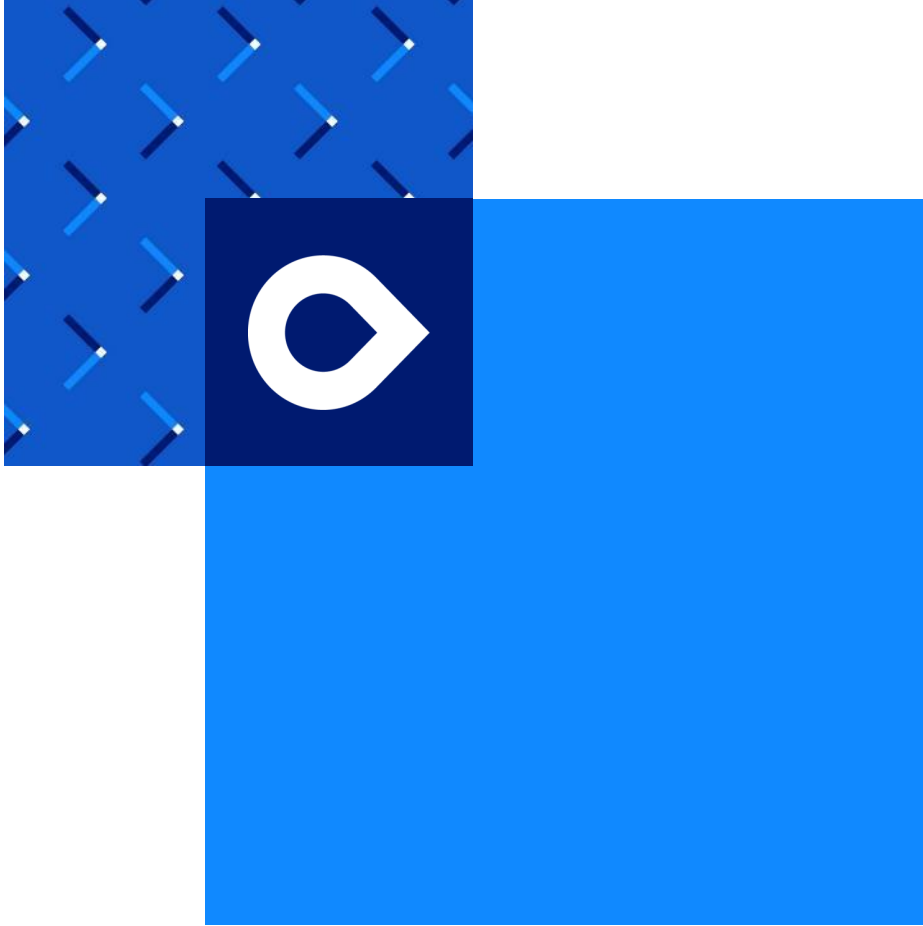
Lipschitz Continuous Neural Networks

- **Robustness:** A model is said to be robust if it is stable against light perturbations on the data or on the parameters
 - › Compromise generalization and robustness
- **Lipschitz continuous:** Given two metrics space $\mathcal{X}$ and $\mathcal{Y}$, a function $f: \mathcal{X} \rightarrow \mathcal{Y}$ is Lipschitz continuous if there exists $L \in \mathbb{R}$ such that for all x_1, x_2 in $\mathcal{X}$: $d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq L d_{\mathcal{X}}(x_1, x_2)$
 - › The smallest such L is the Lipschitz constant of f
 - › Finding the Lipschitz constant of a DNN is a NP-hard problem
- **Naive Upper Bound:** Let $f: \Theta \times \mathcal{X} \rightarrow \mathbb{R}^r$ where $\mathcal{X} \subset \mathbb{R}^n$ and $\Theta \subset \mathbb{R}^p$ is the space of parameters. We denote its weights $W_1, W_2, \dots, W_d$ and $\Phi: \mathbb{R} \rightarrow \mathbb{R}$ its element-wise activation function that is 1-Lipschitz:

$$f_1(\theta, x) = W_1 x + b_1 \quad ; \quad f_i(\theta, x) = W_i \Phi(f_{i-1}(\theta, x)) + b_i \quad \forall i = 1 \dots d \quad ; \quad f(\theta, x) = f_d(\theta, x)$$

› Then, the naive upper bound on the Lipschitz constant of f is: $\overline{L}_d = \|W_d\|_{op} \|W_{d-1}\|_{op} \dots \|W_1\|_{op}$

- Imposing the Lipschitz continuity:
 - › Constraints within the neural network
 - › Regularization in the loss function
 - › Projection methods



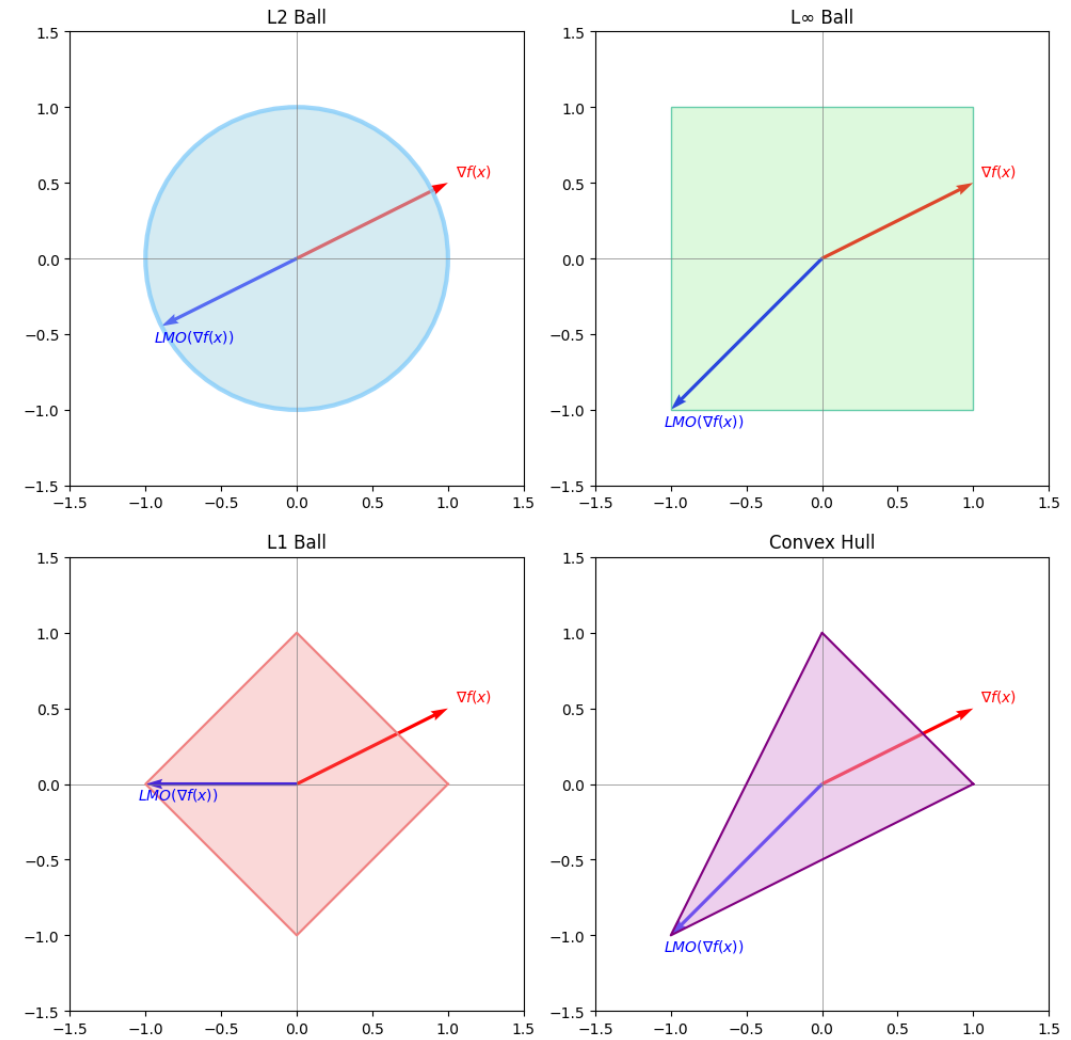
3.

SCION: Stochastic Conditional Gradient with Operator Norm

LMO: Linear Minimization Oracle

Given a norm $\|\cdot\|$, the associated lmo gives the direction the least aligned with the input.

- $\text{lmo}(\nabla f(x)) \in \arg \min_{v \in D} \langle \nabla f(x), v \rangle$
 - › minimizes a linear approximation of the function f
- Gives the steepest descent direction while staying within the constraint set.
- It is always on the boundary of the ball.
- It is scale invariant.



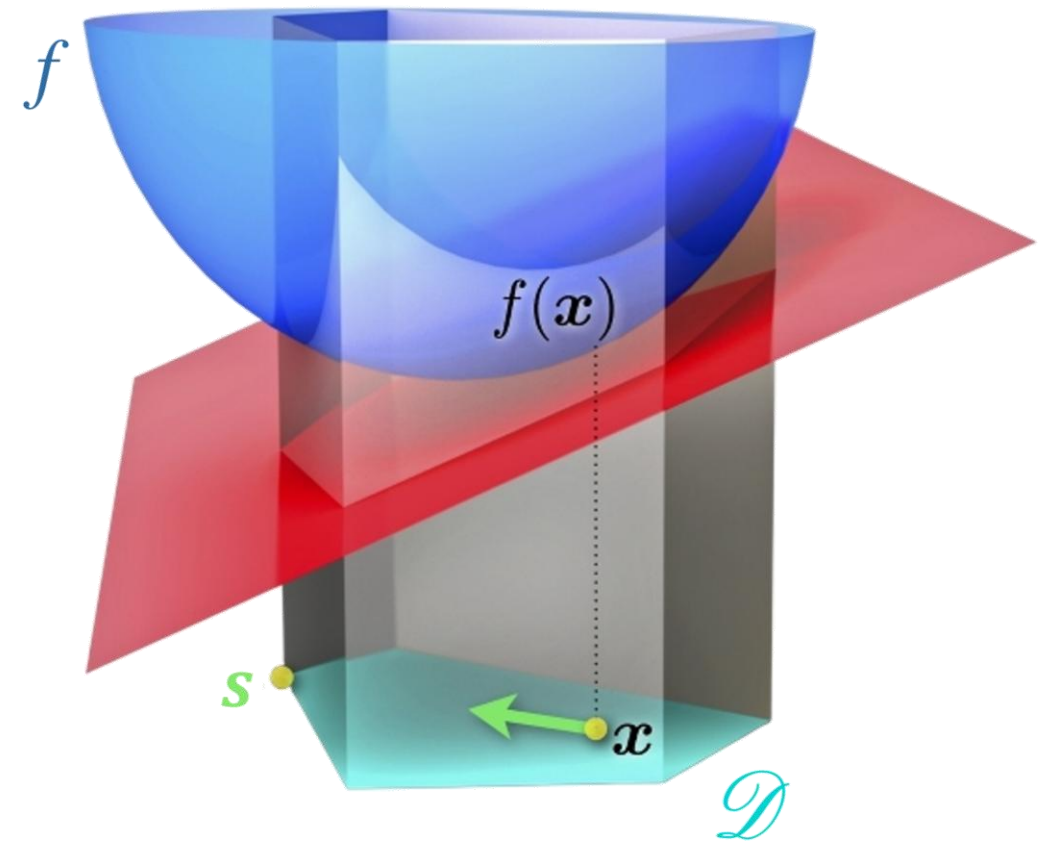
Conditional Gradient Algorithm

The conditional gradient algorithm (or Frank-Wolfe algorithm) solves constrained optimization problems

$$\min_{x \in \mathcal{D}} f(x)$$

With $\mathcal{D} = \{x \mid \|x\| \leq \rho\}$ a norm ball

$$\begin{cases} \text{lmo}(\nabla f(x)) \in \arg \min_{v \in \mathcal{D}} \langle \nabla f(x), v \rangle \\ x^{k+1} = (1 - \gamma_k)x^k + \gamma_k \text{lmo}(\nabla f(x)) \end{cases}$$



(Unconstrained) Stochastic Conditional Gradient

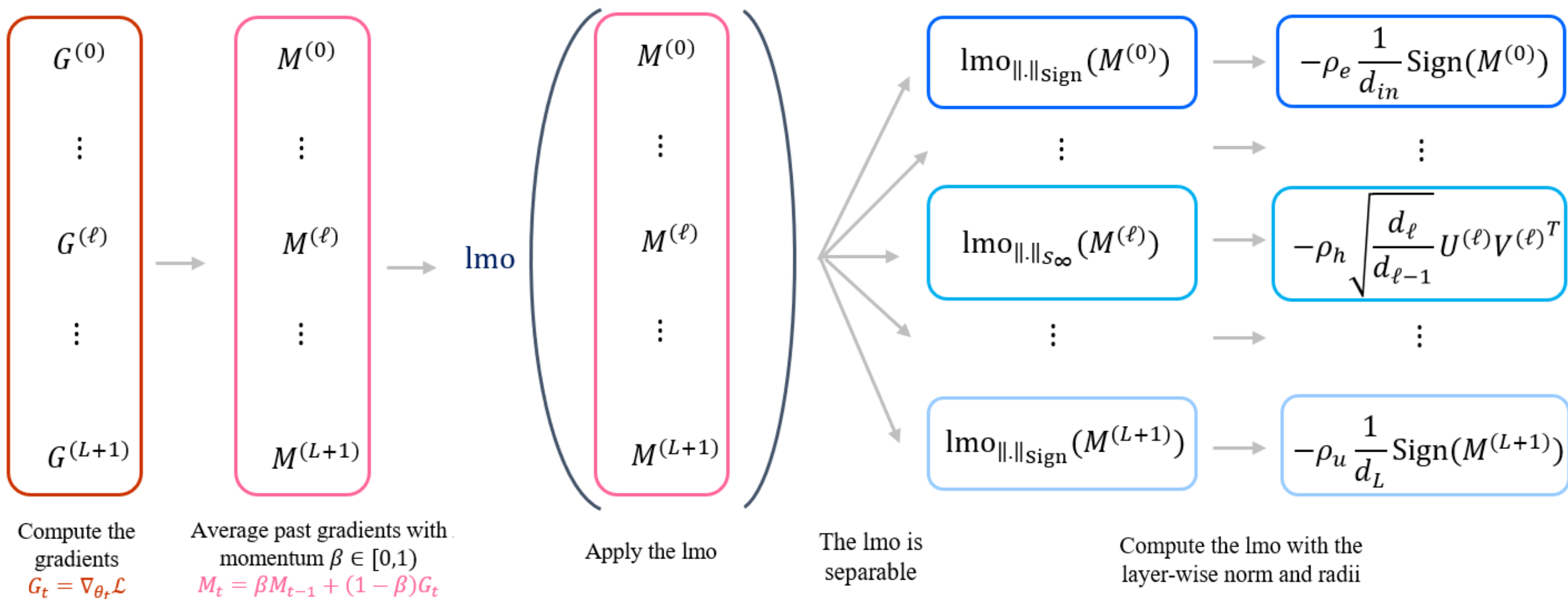
Algorithm 2 (Unconstrained) Stochastic Conditional Gradient

Input: Horizon n , initialization $\theta_1 \in \mathcal{D}$, $m_0 = 0$, momentum $\beta_t \in [0, 1)$, and stepsize $\gamma_t \in (0, 1)$

```
1: for  $t = 1, \dots, n$  do
2:   Sample  $\xi_t$ 
3:    $g_t = \nabla f(\theta_t, \xi_t)$ 
4:    $m_t = (1 - \beta_t)g_t + \beta_t m_{t-1}$ 
5:    $\theta_{t+1} = \begin{cases} (1 - \gamma_t)\theta_t + \gamma_t \text{lmo}(m_t) & \text{constrained} \\ \theta_t + \gamma_t \text{lmo}(m_t) & \text{unconstrained} \end{cases}$ 
6: end for
```

- Momentum: reduces variance
- SCG solves $\min_{x \in \mathcal{D}} f(x)$ with $\mathcal{D}$ a norm ball and uSCG solves $\min_{x \in \mathbb{R}^d} f(x)$.

The lmo is separable



How to chose the norm ?

It depends !

The operator norms for the initial and output layers are different depending on the task and the assumptions on the input.

$$\|W\|_{\alpha \rightarrow \beta} := \max_{z \in \mathbb{R}^d, z \neq 0} \frac{\|Wz\|_{\beta}}{\|z\|_{\alpha}} = \sup_{\|z\|_{\alpha}=1} \|Wz\|_{\beta}.$$



For image domain, the input is usually rescaled pixel-wise:

RMS $\rightarrow$ RMS norm

$$\|x\|_{\text{RMS}} = \frac{\|x\|_2}{\sqrt{d}}, \quad x \in \mathbb{R}^d$$

To be, or not to be, that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles
And by opposing end them?
To die: to sleep;
No more; and, by a sleep to say we end
The heart-ache and the thousand natural shocks
That flesh is heir to, 'tis a consummation
Devoutly to be wish'd. To die, to sleep;
To sleep: perchance to dream: ay, there's the rub;
For in that sleep of death what dreams may come
When we have shuffled off this mortal coil,
Must give us pause.

For language text, the input is usually a 1-hot encoded vector, so $\|z\|_1 = \|z\|_2 = \|z\|_{\infty}$:

1 $\rightarrow$ RMS, 2 $\rightarrow$ RMS , or $\infty \rightarrow$ RMS norm



4.

Hyperparameter Transfer

Hyperparameter tuning for Industrial-Scale Deep Learning

Hyperparameter tuning is essential

- Direct impact on model quality
 - Final task **performance**
 - Generalization, robustness, stability
- Optimization dynamics
 - How the model learns: learning rate, warm-up, decay
 - Gradient noise and convergence; batch size, gradient accumulation
 - Smoothness and generalization; regularization parameters
- Critical in industrial settings
 - Ensures the best performances
 - Reproducibility and reliability
 - Scalability of AI / metamodel based method

... but expensive

- Large models with billions of parameters
 - Large Language Models (LLM)
 - Computer vision
- Huge and interdependent search space
 - Model depth
 - Model width
 - Learning rate
 - Batch size
 - ...
- Computational **cost**
 - Time consuming training phases
 - Costly GPU resources
 - Energy cost

Hyperparameter Transfer method in 3 steps

How to transfer optimal hyperparameters rather than what the large model optimal hyperparameters are

1. **Parametrization of the model** : find the initialization of weights and learning rates dependency as a function of width and depth (solver dependent) of the neural network's architecture
2. **Hyperparameter tuning on a smaller** “proxy” model (small width and depth) than the large targeted one
3. **Transfer** (copy) optimal hyperparameters to the **large** targeted model



[G. Yang et al. "Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer", 2022]
[G. Yang et al. "Feature learning in infinite-width neural networks". In: International Conference on Machine Learning, 2022]

ETICS, 27/09/26-02/10/26 © Framatome – All rights reserved
C0 - Free distribution / Framatome know-how: No
Export Control Other Marking – AL: N / ECCN: N

Feature Learning Desiderata

We are in the **feature learning regime** if, for all layers ℓ :

- The hidden states are stable with respect to the width (stable forward pass):

$$\|h^{(\ell)}\|_{\text{RMS}} = \Theta(1)$$

- The update of the feature is stable with respect to the width (not vanishing, not exploding):

$$\|\Delta h^{(\ell)}\|_{\text{RMS}} = \Theta(1)$$

Spectral condition for feature learning (Yang et al 2023)

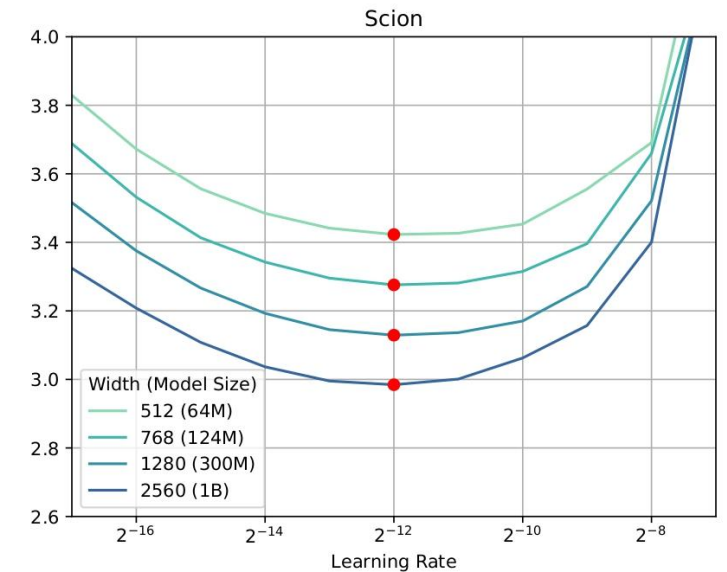
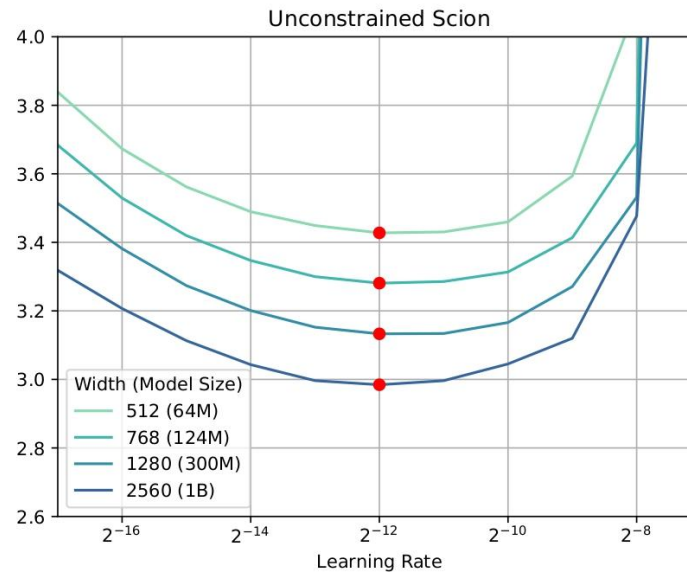
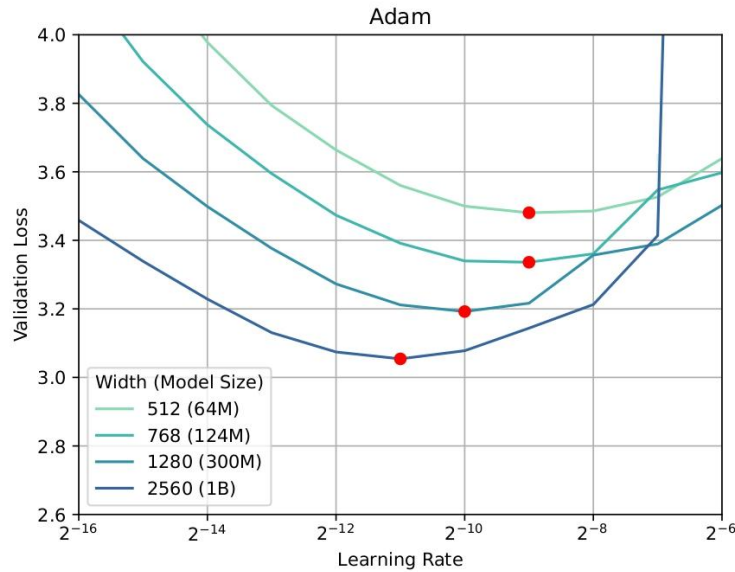
Given a L -layers neural network, with weight matrices W_ℓ , if, for every layer ℓ , we have

$$\|W^{(\ell)}\|_{\text{RMS} \rightarrow \text{RMS}} = \Theta(1) \quad \text{and} \quad \|\Delta W^{(\ell)}\|_{\text{RMS} \rightarrow \text{RMS}} = \Theta(1),$$

then we have feature learning.

[G. Yang et al. "A Spectral condition for feature learning", 2023]

Width Transfer



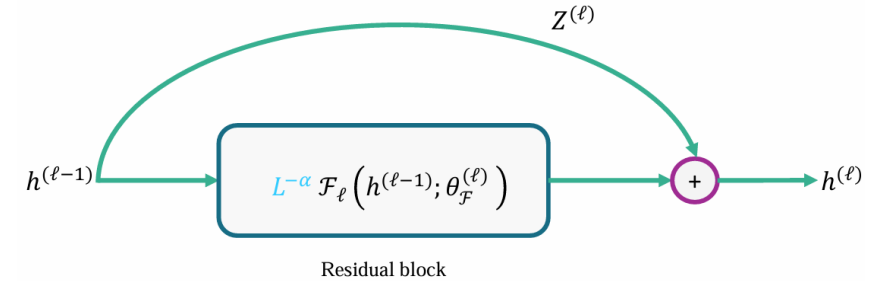
Performance on NanoGPT with between 64M and 1B parameters. The optimal learning rate of SCION is invariant to width

Depth Transfer Desiderata

$$h^{(0)} = W^{(0)}x$$

$$h^{(\ell)} = Z^{(\ell)}h^{(\ell-1)} + L^{-\alpha}\mathcal{F}_{\ell}(h^{(\ell-1)}; \theta_{\mathcal{F}}^{(\ell)}) \quad (\forall \ell \in \{1, \dots, L\})$$

$$f = W^{(L+1)}\phi(h^{(L)}), \quad \alpha \in [1/2, 1]$$



- **Stable Initialization.** The hidden layers and the output remain stable at initialization when both the depth and width become large.

$$\frac{1}{\sqrt{d_{\ell}}} \|h^{(\ell)}\|_2 = \Theta(1) \quad \text{and} \quad \|f\|_2 = \mathcal{O}(1)$$

- **Stable Feature Update.** Each residual block should contribute with order $1/L$ to feature movements, while non-residual layers should contribute with constant order.

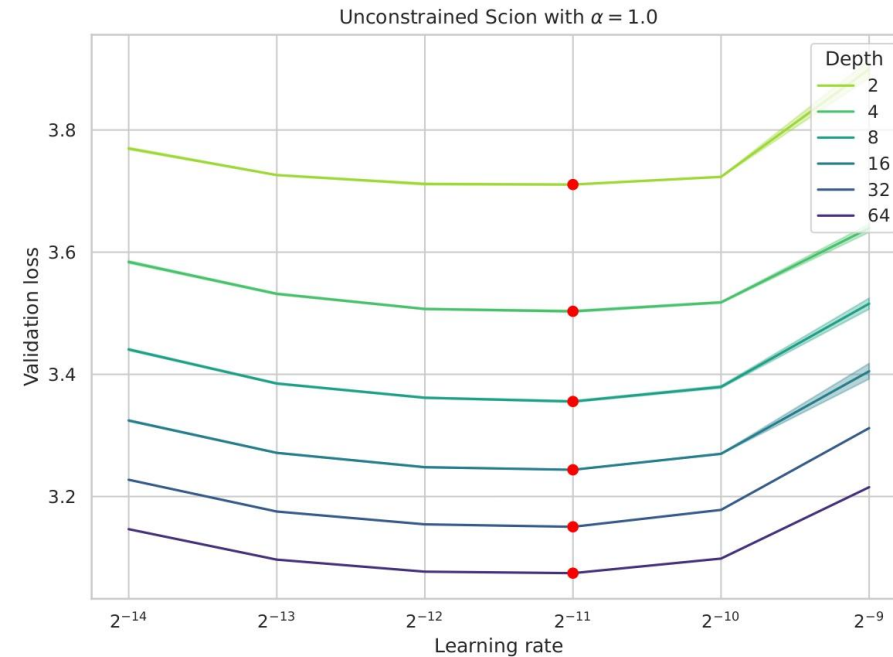
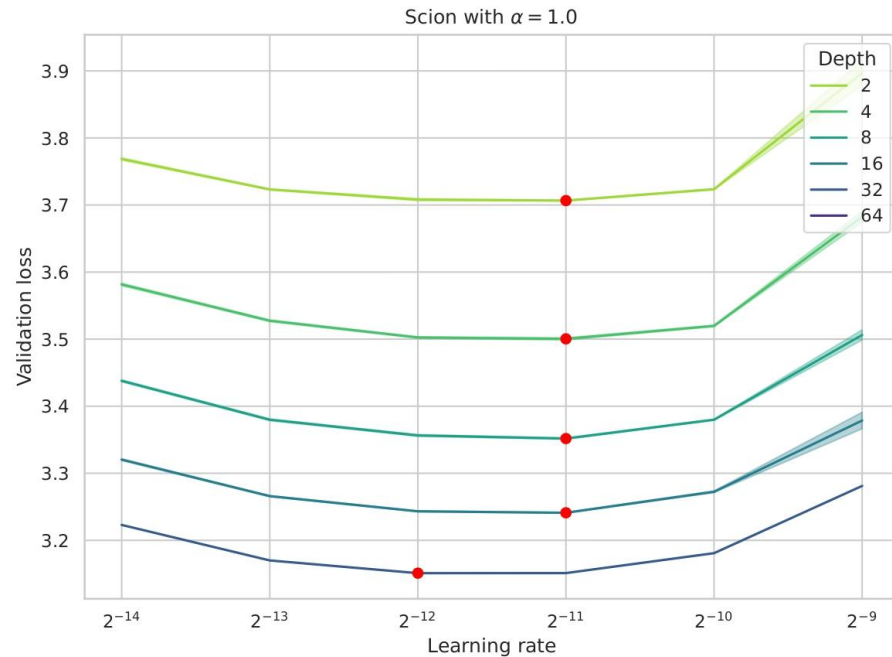
$$\frac{1}{\sqrt{d_{\ell}}} \left\| \Delta_{\theta_{\mathcal{F}}^{(\ell)}, Z^{(\ell)}} h^{(\ell)} \right\|_2 = \Theta\left(\frac{1}{L}\right), \quad \frac{1}{\sqrt{d_0}} \|\Delta W^{(0)}x\|_2 = \Theta(1), \quad \frac{1}{\sqrt{d_{\text{out}}}} \|\Delta W^{(L+1)}h^{(L)}\|_2 = \Theta(1)$$

SCION and uSCION satisfy the depth transfer desiderata with the scaled learning rates:

$$\gamma = \gamma_{\text{base}} L^{\alpha-1}$$

$$\gamma_Z = \gamma_{Z, \text{base}} L^{-1}$$

Hyperparameter Transfer in Depth - NanoGPT

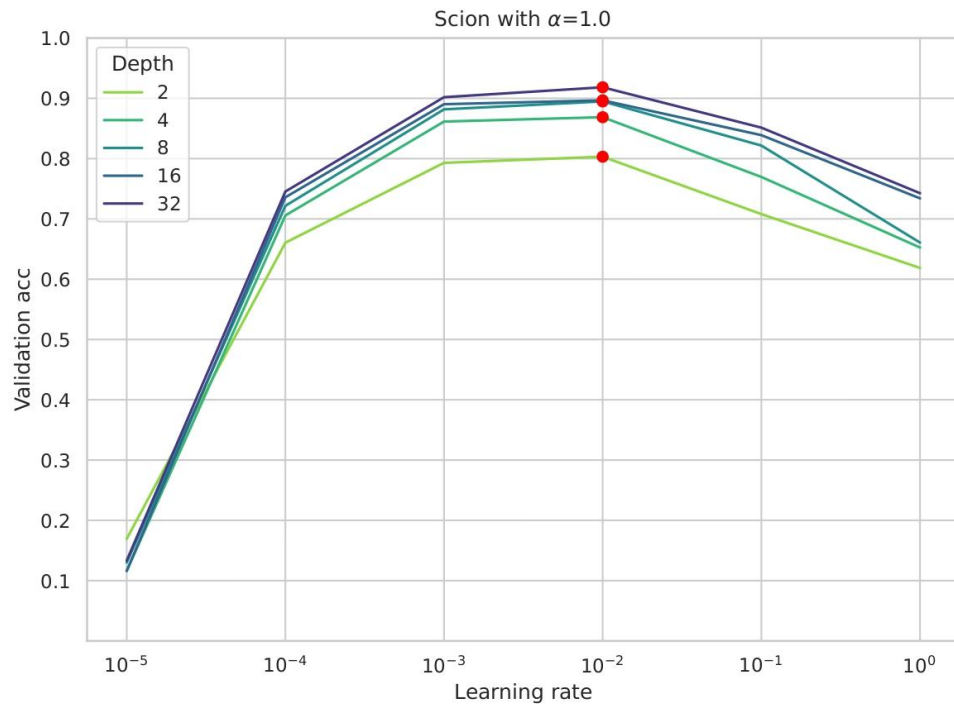


Performance on NanoGPT with between 53M and 500M parameters. The optimal learning rate of SCION is invariant to depth

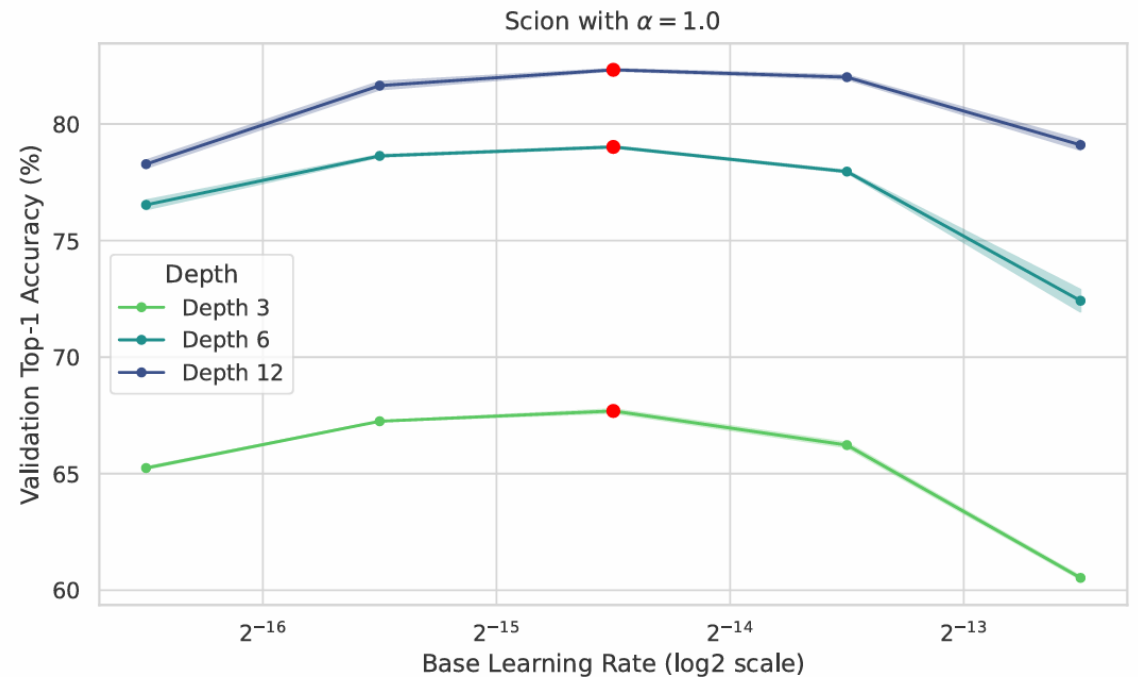
[K. Jordan et al. "modded nanogpt: Speedrunning the nanogpt baseline", 2024]

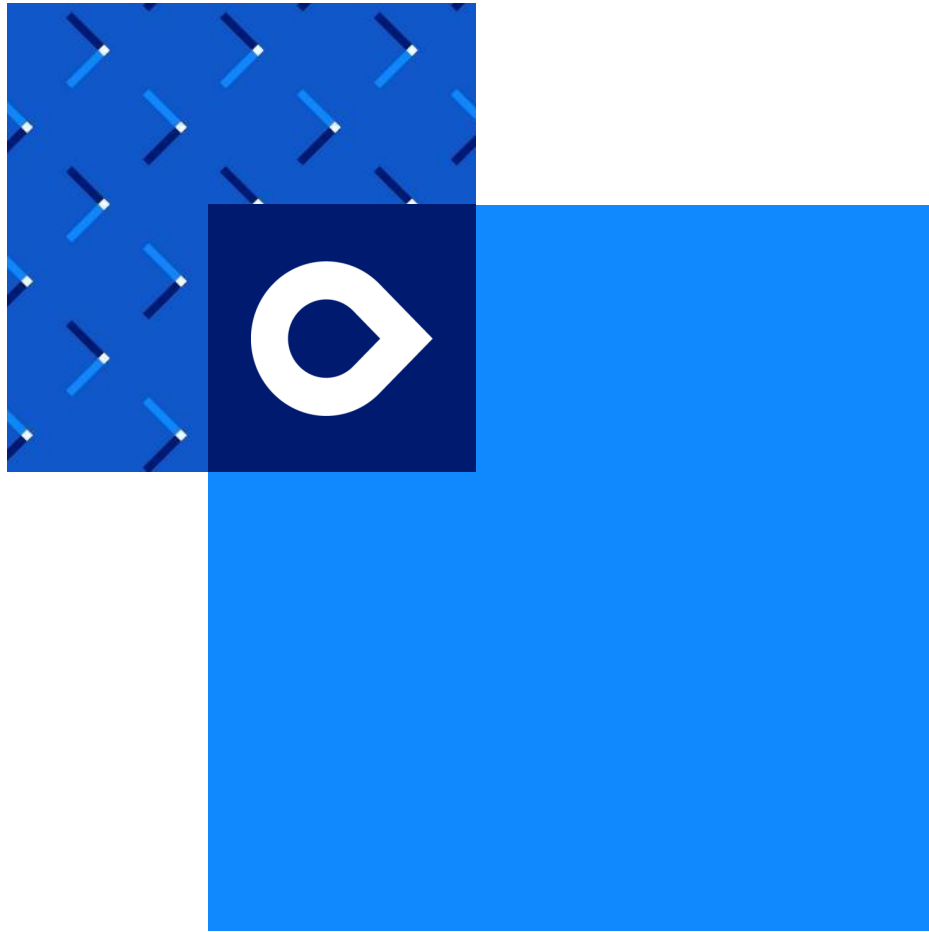
Hyperparameter Transfer in Depth- Computer Vision

- MLP-Mixer: An all MLP architecture for vision tasks.
 - [Ilya Tolstikhin et al. "MLP-Mixer: An all-MLP Architecture for Vision". 2021]
- On CIFAR10



- DeiT.
 - [H. Touvron et al. "Training data-efficient image transformers & distillation through attention". 2021]
- On ImageNet





5.

Conclusion & Perspectives

Conclusion & Perspectives

Study SCION properties:

- › Hyperparameter transfer method: [paper submitted to TMLR](#) (Transactions on Machine Learning Research)
- › Layer-wise constraints
- › Projected version of SCION: clipping the singular values of the weight matrices to a certain range
- › Reduce the gap between the Naïve Upper Bound and the estimation of the Lipschitz constant with more sophisticated relaxations
- › Explore other constraints

Use and leverage the hyperparameter transfer method for Framatome use cases:

- › Models with billions of parameters
- › Language models (embedding models, RAG)
- › Computer vision & Segmentation (Forged Steel & Zirconium ingots)
- › Box series (DNBR, LPD, PCI) [Segond, M., et al. DNBR Box: Enhancing the Thermal-Hydraulic Design of the Pressurized Water Reactor Protection and Surveillance Systems Using Machine Learning and Uncertainty Quantification. Nuclear Science and Engineering 2026]

Finalize and publish the paper on UQ methods for the Box series



References

1. Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Ed. by Yoshua Bengio and Yann LeCun. 2015
2. Takeru Miyato et al. “Spectral Normalization for Generative Adversarial Networks”. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018
3. Cem Anil, James Lucas, and Roger Grosse. “Sorting Out Lipschitz Function Approximation”. In: Proceedings of the 36th International Conference on Machine Learning. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 291–301.
4. Jean-Christophe Pesquet et al. “Learning Maximally Monotone Operators for Image Recovery”. In: SIAM Journal on Imaging Sciences 14.3 (2021), pp. 1206–1237.
5. Jean-Christophe Pesquet, Corneliu Burileanu, and Ana-Antonia Neacsu. “EMG-Based Automatic Gesture Recognition Using Lipschitz-Regularized Neural Networks”. In: ACM Transactions on Intelligent Systems and Technology 15.2 (Feb. 2024), pp. 1–25.
6. Marguerite Frank and Philip Wolfe. “An algorithm for quadratic programming”. In: Naval Research Logistics Quarterly (1956).
7. Thomas Pethick et al. “Training Deep Learning Models with Norm-Constrained LMOs”. In: Forty-second International Conference on Machine Learning, ICML 2025. 2025.
8. Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. “Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. In Advances in Neural Information Processing Systems (NeurIPS), 2021
9. Greg Yang and Edward J. Hu. Tensor programs IV: Feature learning in infinite-width neural networks. In International Conference on Machine Learning, pp. 11727–11737. PMLR, 2021.
10. Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning. arXiv preprint arXiv:2310.17813, 2023.
11. Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don’t be lazy: Completem enables compute-efficient deep transformers, 2026. URL <https://arxiv.org/abs/2505.01618>

References

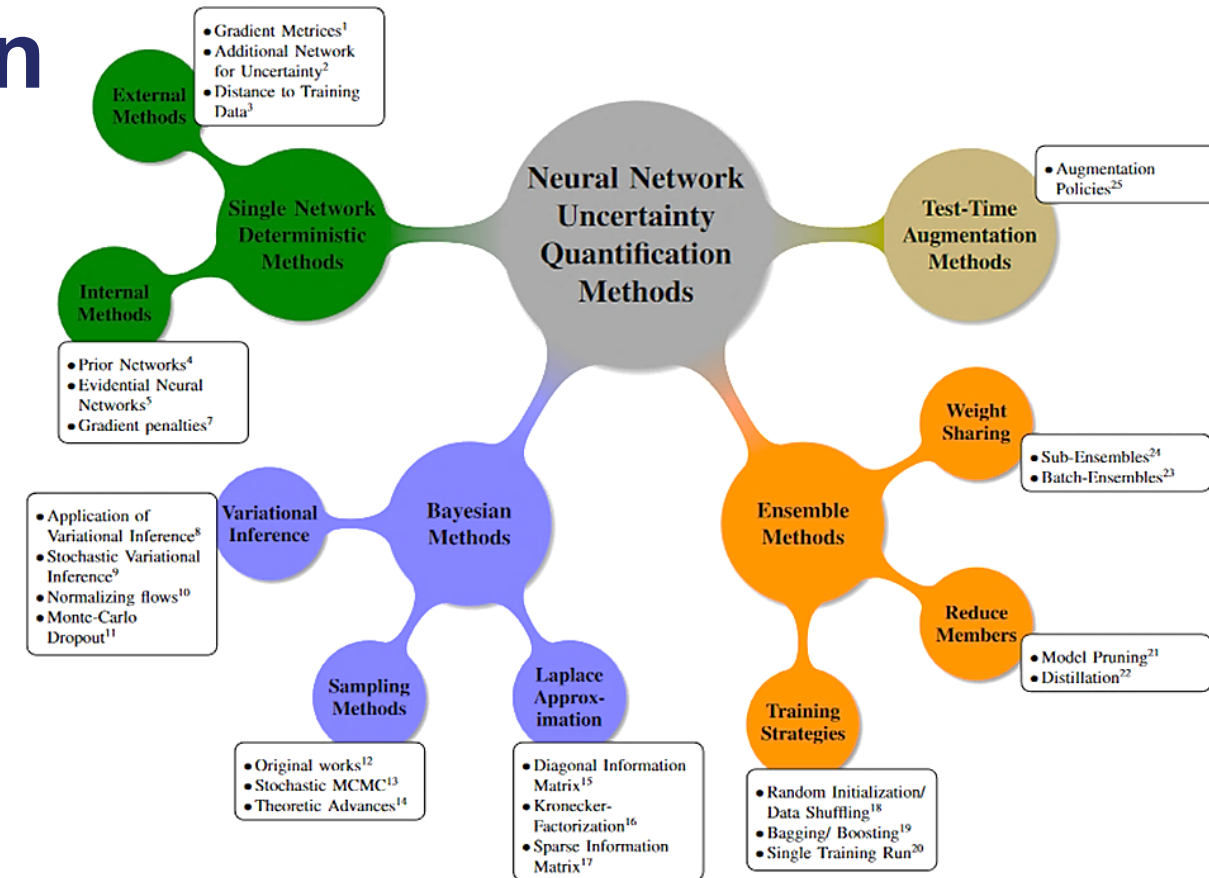
12. Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. Modded-NanoGPT: Speedrunning the NanoGPT baseline, 2024. URL <https://github.com/KellerJordan/modded-nanogpt>
13. Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jes sica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP Mixer: An all-MLP architecture for vision, 2021.
14. Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In International Conference on Machine Learning, volume 139, pp. 10347–10357, July 2021.
15. Segond, M., et al. DNBR Box: Enhancing the Thermal-Hydraulic Design of the Pressurized Water Reactor Protection and Surveillance Systems Using Machine Learning and Uncertainty Quantification. Nuclear Science and Engineering 2026
16. Jakob Gawlikowski et al. “A survey of uncertainty in deep neural networks”. In: Artificial Intelligence Review 56 (2021), pp. 1513–1589.
17. Loshchilov, Ilya and Frank Hutter. “Decoupled Weight Decay Regularization.” International Conference on Learning Representations (2017).
18. Takeru Miyato et al. “Spectral Normalization for Generative Adversarial Networks”. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30- May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.
19. Qiyang Li, Saminul Haque, Cem Anil, James Lucas, Roger Grosse, and Jörn-Henrik Jacobsen. 2019. Preventing gradient attenuation in lipschitz constrained convolutional networks. Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, Article 1379, 15390–15402.
20. Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Semidefinite relaxations for certifying robustness to adversarial examples. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS’18). Curran Associates Inc., Red Hook, NY, USA, 10900–10910.

Uncertainty Quantification Overview

- **Critical** for neural network metamodels in safety-related industrial applications
 - Framatome use case DNBR Box
 - Ensure reliable predictions
 - Controlled risk in operational decision-making
- Various Uncertainty Quantification (UQ) methods, depending on :
 - the type of the model (deterministic or stochastic)
 - the source of the uncertainty we want to capture (aleatoric or epistemic)
- Predictive Bayesian distribution for a prediction y given an input x , a dataset D , and model parameters :

$$p(y|x, D) = \int \underbrace{p(y|x, \theta)}_{\text{data}} \underbrace{p(\theta|D)}_{\text{model}} d\theta$$

- First results obtained in 2025 on DNBR Box Framatome's use case leveraging N4 NPP operating dataset, to be published
- **No consensus:** different theoretical guarantees, computation costs, constraints depending on the application cases, etc.



[Jakob Gawlikowski et al. "A survey of uncertainty in deep neural networks". In: Artificial Intelligence Review 56, 2021]

Residual Networks

We consider models with depth L , with residual connections, defined as

$$\begin{aligned} h^0 &= W^0 x \\ h^{(\ell)} &= h^{(\ell-1)} + L^{-\alpha} \mathcal{F}_\ell(h^{(\ell-1)}; \theta^\ell) \quad \text{for } \ell \in \{1, \dots, L\} \\ f &= W^{L+1} \phi(h^{(L)}) \end{aligned} \tag{2}$$

where $x \in \mathbb{R}^{d_{\text{in}}}$ is the input data, $h^{(\ell)} \in \mathbb{R}^N, \ell \in \{0, \dots, L\}$ are the hidden states, and $f \in \mathbb{R}^{d_{\text{out}}}$ is the output. The embedding and un-embedding matrices are respectively $W^0 \in \mathbb{R}^{N \times d_{\text{in}}}$ and $W^{L+1} \in \mathbb{R}^{d_{\text{out}} \times N}$. The residual block $\mathcal{F}_\ell(h^{(\ell-1)}), \ell \in \{1, \dots, L\}$ can be an attention block, or an MLP. θ^ℓ is the trainable parameter of the layer ℓ . The value of α determines how much the residual block contributes to the residual stream as a function of the depth ℓ .

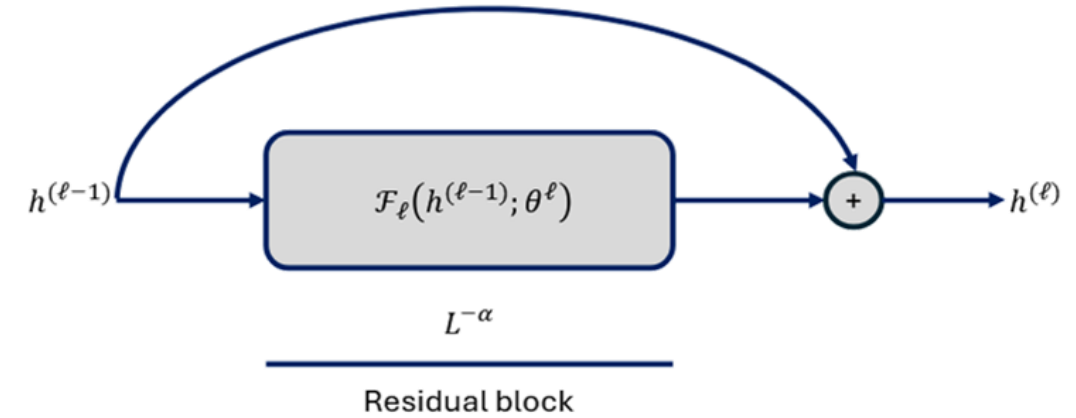


Fig. 1. Residual block mechanism.

Conditions for Depth wise transfer

Condition 1: Stable initialization The hidden layers and the output remain stable (preventing signals from exploding or vanishing as they pass through many layers) at the initialization when $L, N \rightarrow \infty$:

$$\forall \ell \in [L], \|h^{(\ell)}\|_{\text{RMS}}^2 \in \Theta(1), \text{ and } f \in \mathcal{O}(1), \text{ as } N \rightarrow \infty, L \rightarrow \infty. \quad (4)$$

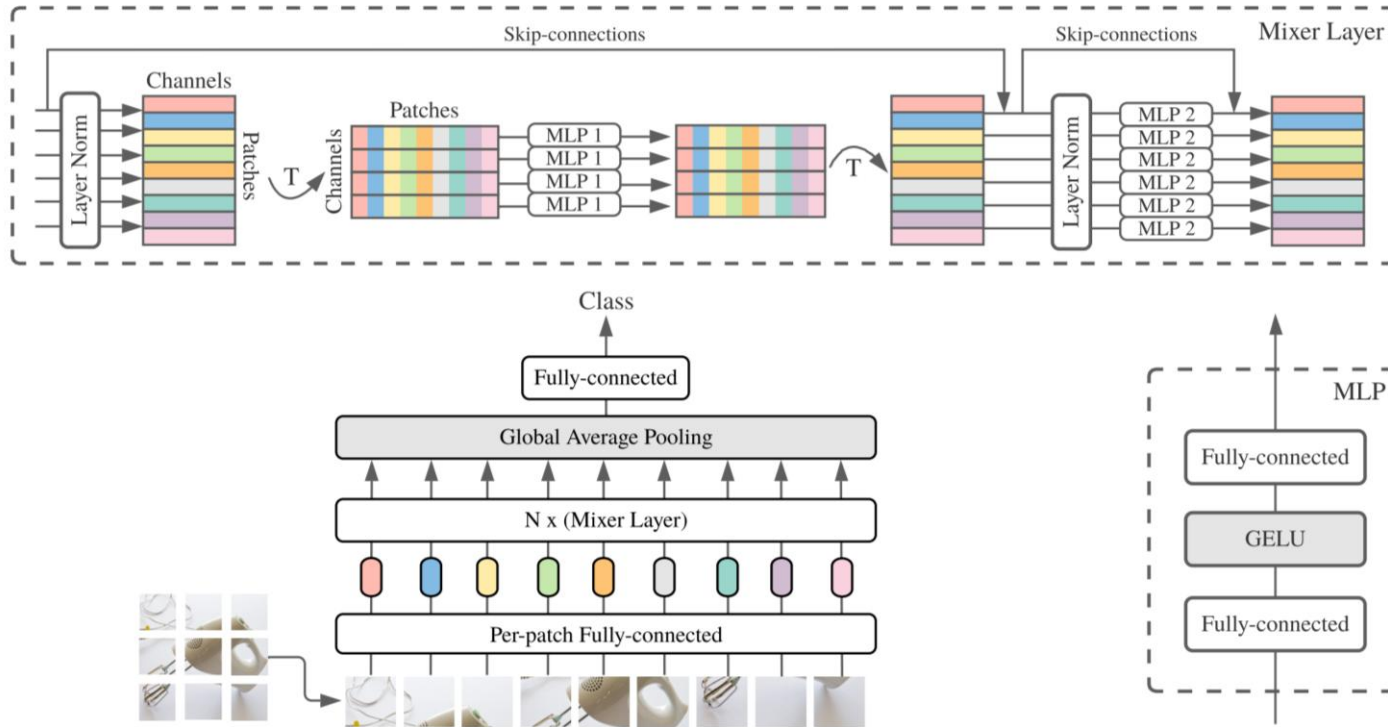
Condition 2: Stable Feature Update The weights of each block should contribute order $1/L$ to feature movements, and each non-residual block should contribute constant order (e.g. each non-residual block has a meaningful impact on the residual stream):

$$\begin{aligned} \frac{1}{\sqrt{N}} \|\Delta_{\theta^\ell} h^{\ell+1}\|_2 &\in \Theta\left(\frac{1}{L}\right) \quad \forall \ell \in [L-1], \\ \frac{1}{\sqrt{N}} \|\Delta W^0 x\|_2 &\in \Theta(1), \\ \frac{1}{\sqrt{N}} \|\Delta W^L h^L\|_2 &\in \Theta(1). \end{aligned} \quad (5)$$

$$\gamma = \gamma_{\text{base}} L^{\alpha-1}.$$

[N. Dey et al. “Don’t be lazy: CompleteP enables compute-efficient deep transformers”, 2025]

MLP-Mixer



Dataset	Acc.(%)	Time(hh:mm:ss)	Steps
CIFAR-10	90.83%	3:34.31	117.3k
CIFAR-100	67.51%	3:35.26	117.3k
SVHN	97.63%	5:23.26	171.9k

- Number of Parameters: 0.67M
- Device: P100 (single GPU)

MLP-Mixer: An all-MLP Architecture for Vision

Ilya Tolstikhin*, Neil Houlsby*, Alexander Kolesnikov*, Lucas Beyer*,
Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner,
Daniel Keysers, Jakob Uszkoreit, Mario Lucic, Alexey Dosovitskiy

2 types of MLPs :

- Channel-mixing MLP :
 - communication between different channels
 - operate on each patch independently
- Token-mixing MLP :
 - communication between different spatial location
 - operate on each channel independently

[Ilya Tolstikhin et al. "MLP-Mixer: An all-MLP Architecture for Vision". 2021.]

Experimental details

Table 7. NanoGPT hyperparameters.

Hyperparameter	AdamW	Muon	UNCONSTRAINED SCION	SCION
Layers	12			
Head dim	128			
Activation function	ReLU ²		ScaledReLU ² (see Appendix E.3)	
Vocabulary size	50304			
Dataset	FineWeb			
batch size	512			
block size	1024			
Iterations n	5100			
Warmdown	28.5%			
Stepsize schedule	Constant then linear decay $\gamma_k = \begin{cases} \gamma & \text{if } k < n - m \\ \gamma \cdot (\frac{n-k}{m}) & \text{if } k \geq n - m \end{cases}$			
Warmup	5%	0		
Gradient clipping	Yes	No		
Momentum β_1 / β_2	0.9 / 0.95	-		
Averaging parameter α	-	0.1		
Muon stepsize multiplier ¹	-	0.1	-	
Nesterov	-	Yes	-	
Boundary init.	-		No	
Radius $\rho_1 / \rho_\ell / \rho_L$	-		- / 50 / 3000	

¹ Muon uses Adam for the first and last layer. The stepsize for the intermediary layers is multiplied by a constant.

Table 1. NanoGPT hyperparameters.

Hyperparameter	UNCONSTRAINED SCION and SCION
Width	768
Head dim	128
Activation function	ScaledReLU ²
Vocabulary size	50304
Dataset	FineWeb
Batch size	512
Block size	1024
Iterations n	5100
Warmdown	28.5%
Stepsize schedule	Constant then linear decay
Averaging parameter β	0.9
Radii	3000 / 50 / 3000

Table 3. MLP-Mixer hyperparameters.

Hyperparameter	UNCONSTRAINED SCION and SCION
Dataset	CIFAR10
Embedding dimension	128
Token-mixing MLP hidden width	64
Channel-mixing MLP hidden width	512
Activation	ReLU
Batch size	2048
Learning rate schedule	Cosine decay with linear warm-up
Training epochs n	300
Averaging parameter β	0.9
Radii	25/25/500

Operator norms & lmos

Table 2. Example operator norms and the associated lmos of a matrix $A \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. The reduced SVD is given as $A = U \text{diag}(\sigma) V^\top$, sign acts elementwise, $\text{col}_j(A) := A_{:,j}$ and $\text{row}_i(A) := A_{i,:}$. Note that this table is not exhaustive.

	$1 \rightarrow \text{RMS}$ (ColNorm)	$1 \rightarrow \infty$ (Sign)	$\text{RMS} \rightarrow \text{RMS}$ (Spectral)	$\text{RMS} \rightarrow \infty$ (RowNorm)
Norm	$\max_j \frac{1}{\sqrt{d_{\text{out}}}} \ \text{col}_j(A)\ _2$	$\max_{i,j} A_{i,j} $	$\sqrt{d_{\text{in}}/d_{\text{out}}} \ A\ _{\mathcal{S}_\infty}$	$\max_i \sqrt{d_{\text{in}}} \ \text{row}_i(A)\ _2$
LMO	$\text{col}_j(A) \mapsto -\sqrt{d_{\text{out}}} \frac{\text{col}_j(A)}{\ \text{col}_j(A)\ _2}$	$A \mapsto -\text{sign}(A)$	$A \mapsto -\sqrt{d_{\text{out}}/d_{\text{in}}} UV^\top$	$\text{row}_i(A) \mapsto -\frac{1}{\sqrt{d_{\text{in}}}} \frac{\text{row}_i(A)}{\ \text{row}_i(A)\ _2}$

Table 3. The choice of lmo can be different between layers and can depend on the assumptions on the input. For simplicity we overload notation and write the reduced SVD as $W_\ell = U \text{diag}(\sigma) V^\top \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ for all $\ell \in [L]$.

Parameter	W_1 (image domain)	$\{W_\ell\}_{\ell \in [2, \dots, L-1]}$	W_L			b_ℓ
Norm	$\text{RMS} \rightarrow \text{RMS}$	$\text{RMS} \rightarrow \text{RMS}$	$\text{RMS} \rightarrow \text{RMS}$	$\text{RMS} \rightarrow \infty$	$1 \rightarrow \infty$	RMS
LMO	$-\max(1, \sqrt{d_{\text{out}}/d_{\text{in}}}) UV^\top$	$-\sqrt{d_{\text{out}}/d_{\text{in}}} UV^\top$	$-\sqrt{d_{\text{out}}/d_{\text{in}}} UV^\top$	$\text{row}_i(W_L) \mapsto -\frac{1}{\sqrt{d_{\text{in}}}} \frac{\text{row}_i(W_L)}{\ \text{row}_i(W_L)\ _2}$	$-\frac{1}{d_{\text{in}}} \text{sign}(W_L)$	$-\frac{b_\ell}{\ b_\ell\ _{\text{RMS}}}$
Init.	Semi-orthogonal	Semi-orthogonal	Semi-orthogonal	Row-wise normalized Gaussian	Random sign	0

Operator norms & lmos

Insight 3.2.

- (i) For 1-hot encoded input, ColNorm and Spectral are equivalent for the first layer, in which case ColNorm is favored since the lmo can be computed exactly.
- (ii) Sign can be used both for the first and last layer which is crucial for weight sharing.
- (iii) To transfer learning rate from proxy models when the width is smaller than the input dimension it is important to rescale the lmo as $\max(1, \sqrt{d_{\text{out}}/d_{\text{in}}})$.

- (i) Initial layer $h_1(z)$: $\|W_1\|_{\alpha_1 \rightarrow \text{RMS}} \leq 1$.
- (ii) Intermediary layers $h_\ell(z)$: $\|W_\ell\|_{\text{RMS} \rightarrow \text{RMS}} \leq 1$
 $\forall \ell \in \{2, \dots, L-1\}$.
- (iii) Last layer $h_L(z)$: $\|W_L\|_{\text{RMS} \rightarrow \beta_L} \leq 1$.

$$\|A\|_{\text{RMS} \rightarrow \infty} \leq \frac{1}{d_{\text{in}}} \|A\|_{1 \rightarrow \infty},$$

Recommendation 3.1. We refer to the instantiation of **uSCG** and **SCG** using operator norms as **UNCONSTRAINED SCION** and **SCION** respectively (*cf.* Algorithm 3), which stands for **Stochastic Conditional Gradient with Operator Norms**. We recommend the following configurations of the layer norms (First layer $\rightarrow$ Intermediary layers $\rightarrow$ Last layer):

- (i) image domains: Spectral $\rightarrow$ Spectral $\rightarrow$ Sign
- (ii) 1-hot input: ColNorm $\rightarrow$ Spectral $\rightarrow$ Sign
- (iii) weight sharing: Sign $\rightarrow$ Spectral $\rightarrow$ Sign

The lmo names are defined in Table 2 and weight sharing refers to parameter sharing between the first and last layer. Each layer should be scaled appropriately according to Tables 3 and 4.

Optimization Algorithms

- **Stochastic Gradient Descent :**

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} f(\theta; x_i, y_i)$$

- **RMSProp** ; uses an exponentially weighted moving average (EWMA) to smooth the cumulated gradients [3].

$$\begin{aligned} m_t &= \beta m_{t-1} + (1 - \beta) \nabla_{\theta} f(\theta_t) \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{\nu_t} + \epsilon} \nabla_{\theta} f(\theta_t) \end{aligned}$$

- **Adam** [5]

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) \nabla_{\theta} f(\theta_t) \\ \nu_t &= \beta_2 \nu_{t-1} + (1 - \beta_2) \nabla_{\theta} f(\theta_t)^2 \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \hat{\nu}_t &= \frac{\nu_t}{1 - \beta_2^t} \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{\nu}_t} + \epsilon} \hat{m}_t \end{aligned}$$

Algorithm 2 Muon

Require: Learning rate η , momentum μ

```
1: Initialize  $B_0 \leftarrow 0$ 
2: for  $t = 1, \dots$  do
3:   Compute gradient  $G_t \leftarrow \nabla_{\theta} \mathcal{L}_t(\theta_{t-1})$ 
4:    $B_t \leftarrow \mu B_{t-1} + G_t$ 
5:    $O_t \leftarrow \text{NewtonSchulz5}(B_t)$ 
6:   Update parameters  $\theta_t \leftarrow \theta_{t-1} - \eta O_t$ 
7: end for
8: return  $\theta_t$ 
```

SGD & Adam with Weight decay

Algorithm 1 SGD with L₂ regularization and SGD with decoupled weight decay (SGDW), both with momentum

```

1: given initial learning rate  $\alpha \in \mathbb{R}$ , momentum factor  $\beta_1 \in \mathbb{R}$ , weight decay/L2 regularization factor  $\lambda \in \mathbb{R}$ 
2: initialize time step  $t \leftarrow 0$ , parameter vector  $\theta_{t=0} \in \mathbb{R}^n$ , first moment vector  $m_{t=0} \leftarrow \mathbf{0}$ , schedule multiplier  $\eta_{t=0} \in \mathbb{R}$ 
3: repeat
4:    $t \leftarrow t + 1$ 
5:    $\nabla f_t(\theta_{t-1}) \leftarrow \text{SelectBatch}(\theta_{t-1})$  ▷ select batch and return the corresponding gradient
6:    $g_t \leftarrow \nabla f_t(\theta_{t-1}) + \lambda \theta_{t-1}$ 
7:    $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$  ▷ can be fixed, decay, be used for warm restarts
8:    $m_t \leftarrow \beta_1 m_{t-1} + \eta_t \alpha g_t$ 
9:    $\theta_t \leftarrow \theta_{t-1} - m_t - \eta_t \lambda \theta_{t-1}$ 
10: until stopping criterion is met
11: return optimized parameters  $\theta_t$ 

```

Algorithm 2 Adam with L₂ regularization and Adam with decoupled weight decay (AdamW)

```

1: given  $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$ 
2: initialize time step  $t \leftarrow 0$ , parameter vector  $\theta_{t=0} \in \mathbb{R}^n$ , first moment vector  $m_{t=0} \leftarrow \mathbf{0}$ , second moment vector  $v_{t=0} \leftarrow \mathbf{0}$ , schedule multiplier  $\eta_{t=0} \in \mathbb{R}$ 
3: repeat
4:    $t \leftarrow t + 1$ 
5:    $\nabla f_t(\theta_{t-1}) \leftarrow \text{SelectBatch}(\theta_{t-1})$  ▷ select batch and return the corresponding gradient
6:    $g_t \leftarrow \nabla f_t(\theta_{t-1}) + \lambda \theta_{t-1}$ 
7:    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$  ▷ here and below all operations are element-wise
8:    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
9:    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$  ▷  $\beta_1$  is taken to the power of  $t$ 
10:   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$  ▷  $\beta_2$  is taken to the power of  $t$ 
11:   $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$  ▷ can be fixed, decay, or also be used for warm restarts
12:   $\theta_t \leftarrow \theta_{t-1} - \eta_t \left( \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) + \lambda \theta_{t-1} \right)$ 
13: until stopping criterion is met
14: return optimized parameters  $\theta_t$ 

```

[Ilya Loshchilov, Frank Hutter, Decoupled Weight Decay Regularization, 2017]

Spectral Normalization

Algorithm 1 SGD with spectral normalization

- Initialize $\tilde{\mathbf{u}}_l \in \mathcal{R}^{d_l}$ for $l = 1, \dots, L$ with a random vector (sampled from isotropic distribution).
- For each update and each layer l :
 1. Apply power iteration method to a unnormalized weight W^l :

$$\tilde{\mathbf{v}}_l \leftarrow (W^l)^T \tilde{\mathbf{u}}_l / \|(W^l)^T \tilde{\mathbf{u}}_l\|_2 \quad (20)$$

$$\tilde{\mathbf{u}}_l \leftarrow W^l \tilde{\mathbf{v}}_l / \|W^l \tilde{\mathbf{v}}_l\|_2 \quad (21)$$

2. Calculate $\bar{W}_{\text{SN}}$ with the spectral norm:

$$\bar{W}_{\text{SN}}^l(W^l) = W^l / \sigma(W^l), \text{ where } \sigma(W^l) = \tilde{\mathbf{u}}_l^T W^l \tilde{\mathbf{v}}_l \quad (22)$$

3. Update W^l with SGD on mini-batch dataset $\mathcal{D}_M$ with a learning rate α :

$$W^l \leftarrow W^l - \alpha \nabla_{W^l} \ell(\bar{W}_{\text{SN}}^l(W^l), \mathcal{D}_M) \quad (23)$$

[Takeru Miyato et al. Spectral Normalization for Generative Adversarial Networks, 2018.]

Björck Algorithm

G Orthogonalization Procedure

Several ways have been proposed in the literature to orthogonalize an unconstrained matrix in a differentiable manner [32, 5]. In this work, we adopt Björck orthogonalization algorithm from Björck and Bowie [5].

The original Björck paper proposes the following iterative procedure to find the closest matrix under the metric of the Frobenius norm of the difference matrix:

$$\text{Björck}(A) = A \left(I + \frac{1}{2}Q + \frac{3}{8}Q^2 + \cdots + (-1)^p \binom{-\frac{1}{2}}{p} Q^p \right)$$

where $p \geq 1$ and $Q = I - A^T A$. This function can be iterated arbitrarily to get tighter estimates of orthogonal matrices. In all our experiments, we use $p = 1$ and iteratively apply this function 20 times.

[Q Li et al. Preventing Gradient Attenuation in Lipschitz Constrained Convolutional Networks, 2019]

Semi Definite Programming

Optimization objective :
maximizing the worst case
margin

$$f(x^0)_y - f(x^0)_{\bar{y}} = (c_y - c_{\bar{y}})^\top x^L$$

where x_0 is an adversarially
perturbed input

Quadratically Constrained
Quadratic Program (QCQP)

By reformulating the ReLU

Relaxation of the non-
convex (QCQP) to a convex
SDP, by introducing a new
set of variables representing
all linear and quadratic
monomials x and z

$$\ell_y^*(\bar{x}, \bar{y}) = \max_{x^0, \dots, x^L} (c_y - c_{\bar{y}})^\top x^L \quad (2)$$

subject to $x^i = \text{ReLU}(W^{i-1} x^{i-1})$ for $i=1, 2, \dots, L$ (Neural network constraints)

$\|x_j^0 - \bar{x}_j\|_\infty \leq \epsilon$ for $j=1, 2, \dots, d$ (Attack model constraints)

Computing ℓ_y^* is computationally hard in general. In the following sections, we present how to relax this objective to a convex semidefinite program and discuss some properties of this relaxation.

We use the key insight that the ReLU constraints can be written as linear and quadratic constraints, allowing us to embed these constraints into a QCQP. We can also express the input constraint $l_j \leq x_j \leq u_j$ as a quadratic constraint, which will be useful later. In particular, $l_j \leq x_j \leq u_j$ if and only if $(x_j - l_j)(x_j - u_j) \leq 0$, thereby yielding the quadratic constraint $x_j^2 \leq (l_j + u_j)x_j - l_j u_j$. This gives us the final QCQP below:

$$\ell_y^*(\bar{x}, \bar{y}) = f_{\text{QCQP}} = \max_{x, z} c^\top z \quad (3)$$

s.t. $z \geq 0$, $z \geq Wx$, $z^2 = z \odot (Wx)$ (ReLU constraints)

$x^2 \leq (l+u) \odot x - l \odot u$ (Input constraints)

In particular, let $v \stackrel{\text{def}}{=} \begin{bmatrix} 1 \\ x \\ z \end{bmatrix}$. We define a matrix $P \stackrel{\text{def}}{=} vv^\top$ and use symbolic indexing $P[\cdot]$ to index

$$\text{the elements of } P, \text{ i.e } P = \begin{bmatrix} P[1] & P[x^\top] & P[z^\top] \\ P[x] & P[xx^\top] & P[xz^\top] \\ P[z] & P[zx^\top] & P[zz^\top] \end{bmatrix}.$$

The SDP relaxation of (3) can be written in terms of the matrix P as follows.

$$f_{\text{SDP}} = \max_P c^\top P[z] \quad (4)$$

s.t. $P[z] \geq 0$, $P[z] \geq WP[x]$, $\text{diag}(P[zz^\top]) = \text{diag}(WP[xz^\top])$ (ReLU constraints)

$\text{diag}(P[xx^\top]) \leq (l+u) \odot P[x] - l \odot u$ (Input constraints)

$P[1] = 1$, $P \succeq 0$ (Matrix constraints).

[Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples, 2018]