# Towards Quantifying the Hessian Structure of Neural Networks

**Yushun Zhang, June, 2026**

**Presented at OP2026, Edinburgh, UK**

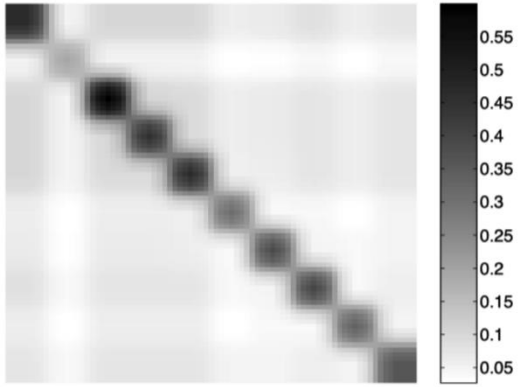
**School of Data Science,**

**The Chinese University of Hong Kong, Shenzhen**

# Contents

# Classical Results in 2004

- **Hessian of NNs are numerically observed to be near-block-diagonal**




(a) Hessian of an MLP [18] after 1 step

**Hessian of a 2-layer NN**

One block corresponds to? Not discussed
Why so? Not discussed

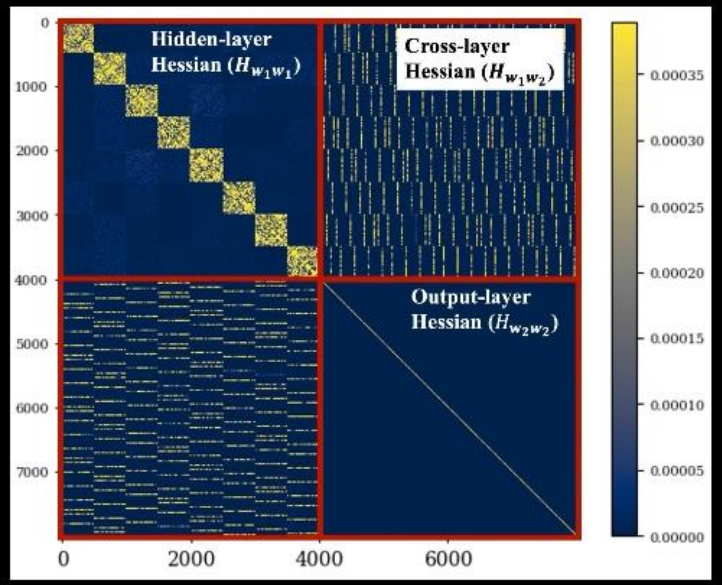
Figure from: Large Scale Machine Learning, Collobert, thesis, 2004

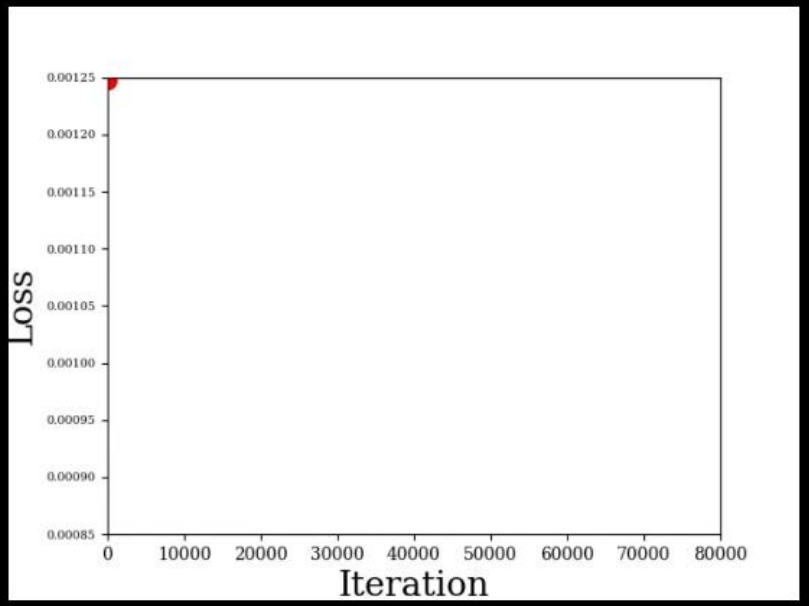
# A closer look of Hessian for 2-layer models

$y = W_2 ReLU(W_1)x$

- $W_1 \in R^{8 \times 500}$
- $W_2 \in R^{500 \times 8}$
- Gaussian data
- CE loss

Hessian of a 2-layer relu NN, input dim = # classes = 500, width = 8, CE loss +Adam, Gaussian data + random label, sample size = 5000



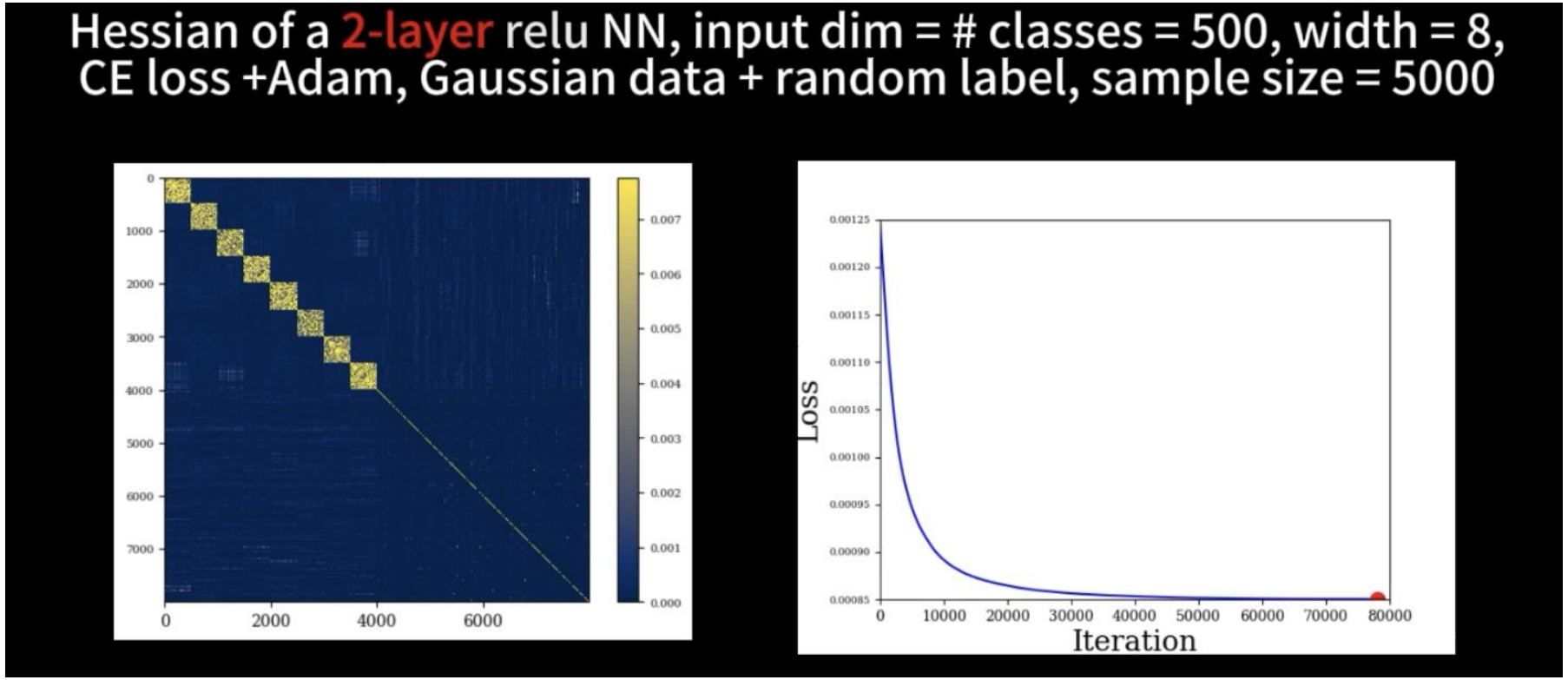

(i) **block-circulant-block-diagonal** structure at initialization
(ii) The **near-block-diagonal** pattern along training

**One block = one output neuron (or one row in a weight matrix)**

# A closer look of Hessian for 2-layer models

$y = W_2 ReLU(W_1)x$

- $W_1 \in R^{8 \times 500}$
- $W_2 \in R^{500 \times 8}$
- Gaussian data
- CE loss




(i) **block-circulant-block-diagonal** structure at initialization
(ii) The **near-block-diagonal** pattern along training

**One block = one output neuron (or one row in a weight matrix)**

# Hessian of Transformers?

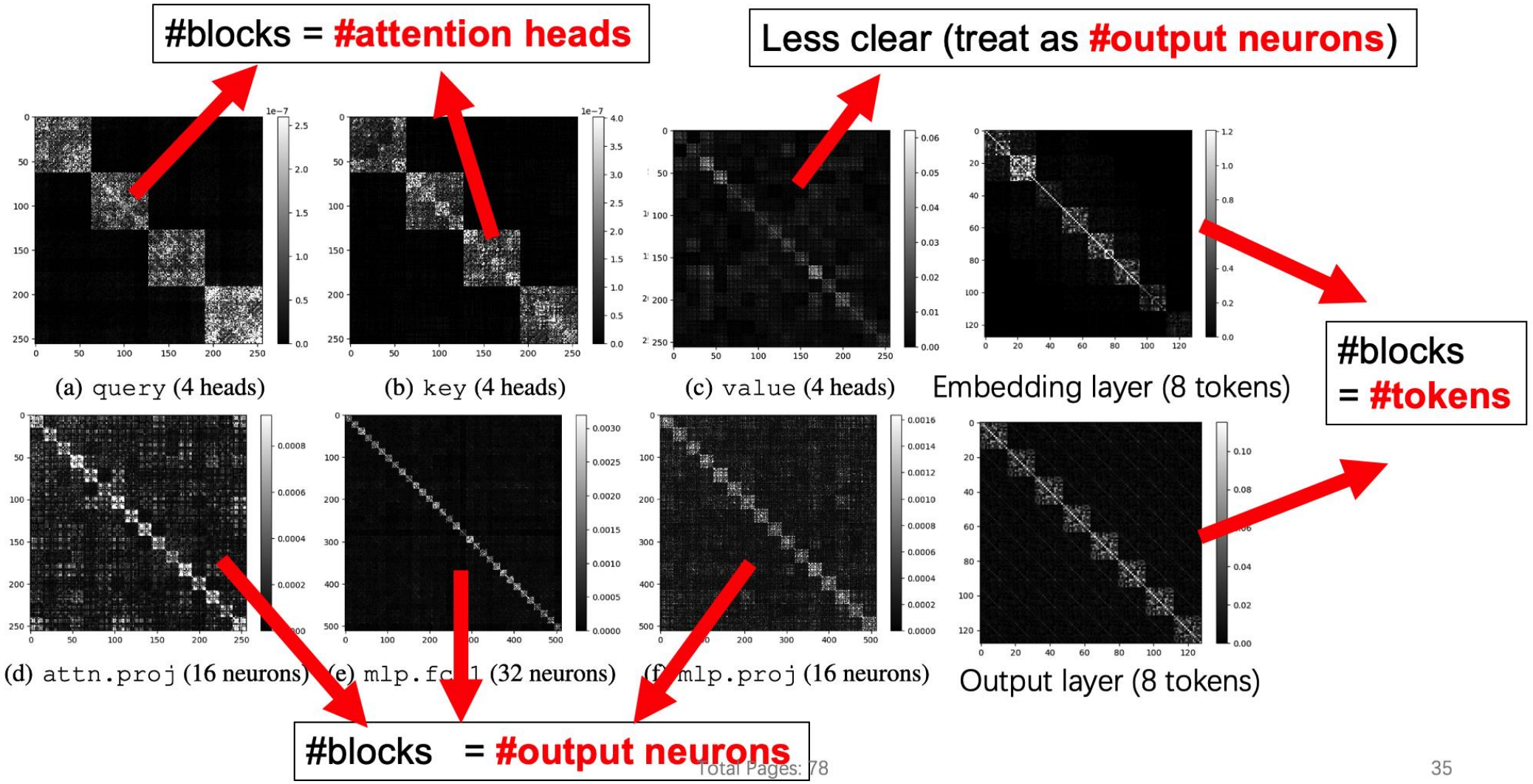
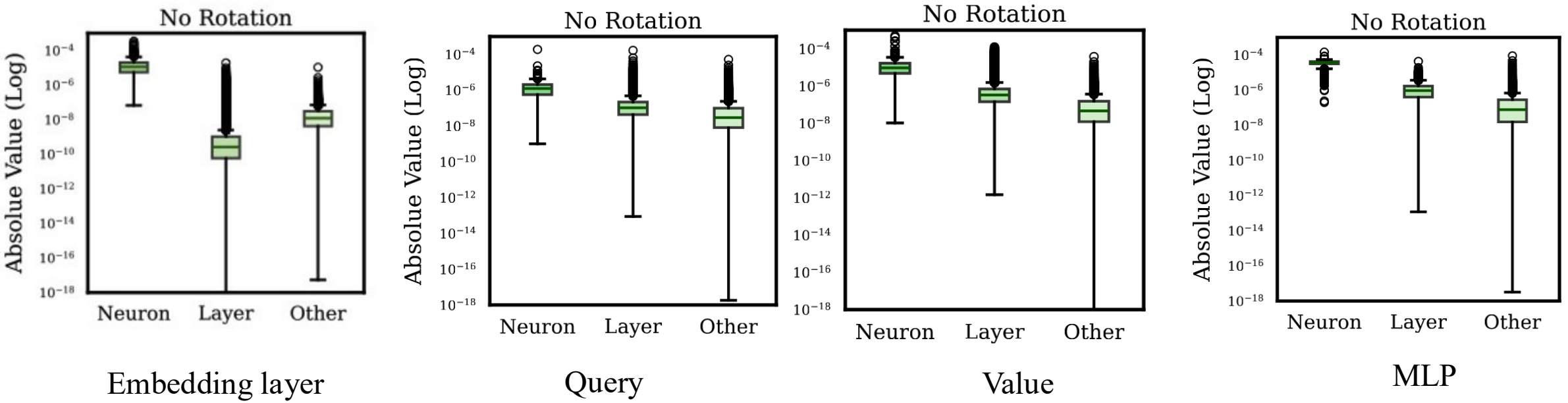





Figure from: Adam-mini: Use Fewer Learning Rates To Gain More, **Zhang*,** Chen*, …, Luo, Sun, ICLR 2025

# Follow-up observations from community




Embedding layer | Query | Value | MLP

**Hessian sub-blocks sampled from GPT2-125M (diag-blocks > 10^4 off-diag-blocks)**

Figure from: Understanding Adam Requires Better Rotation Dependent Assumptions, Maes, et al., 2024

# Motivation: Why Studying Hessian Structure?

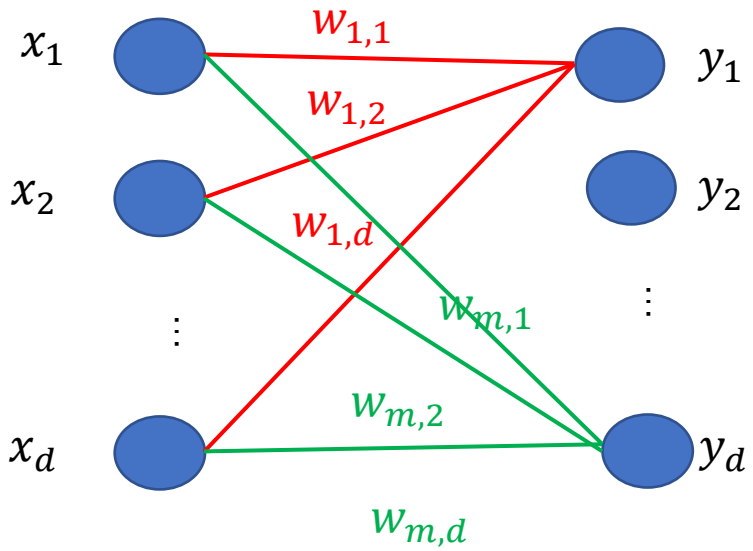
- **1. Hessian structure is crucial for understanding NN training**
  - The effectiveness of Adam
    (Zhang et al 24a, Kunstner et al. 24)
  - The effectiveness of general diagonal-preconditioned methods
    (Sun and Ye, 21, Qu et al. 22, Das et al. 24)
  - The effectiveness of recent block-diagonal-preconditioned methods
    (Shampoo, Muon)
- **2. Hessian structure can help design new training methods for NNs**
- **3. Offering a new function class for optimization community**
  - Typical problems do NOT have such structure:
    In classical non-linear programming dataset (Lavezzi et al 22), all problems have non-block-diag Hessian
  - Motivate new study into this specific class of problems

# Today, we focus on…

- **Why do Hessian matrices look like this? Is it trivial?**
  - Why one block = one neuron?
  - What is the fundamental reason for this structure?
- **Any more structure missed in the previous experiments?**
- **Relation to Adam and Muon?**

# Contents

# Intuition: Example 1: 1-layer model




$$y = W\,x = \begin{pmatrix} w_1^T x \\ \vdots \\ w_d^T x \end{pmatrix}$$

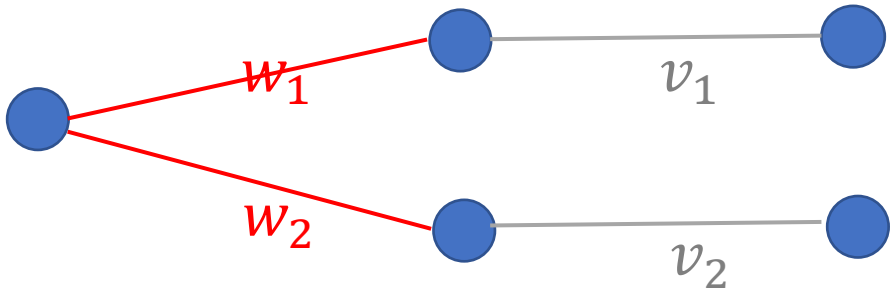
**Linear algebra fact:** $y_i$ only depends on the i–th row of W, but not other rows

**So one block = one output neuron (or one row in a weight matrix)**

**Difficulties in the proof:**
Nonlinear activation + CE loss

# Intuition: Example 2: 2-layer model

- **Example 1: Single-input-multi-output (SIMO):**
  (this is not a standard NN, but is good for understanding)




**check the links!**

Multiplicative relation

Gives non-zero Hessian entry

Input data x = 1. No activation, label = 0, MSE loss: $\ell(w_1, w_2, v_1, v_2) = \frac{1}{2}(w_1 v_1)^2 + \frac{1}{2}(w_2 v_2)^2$

Hessian:

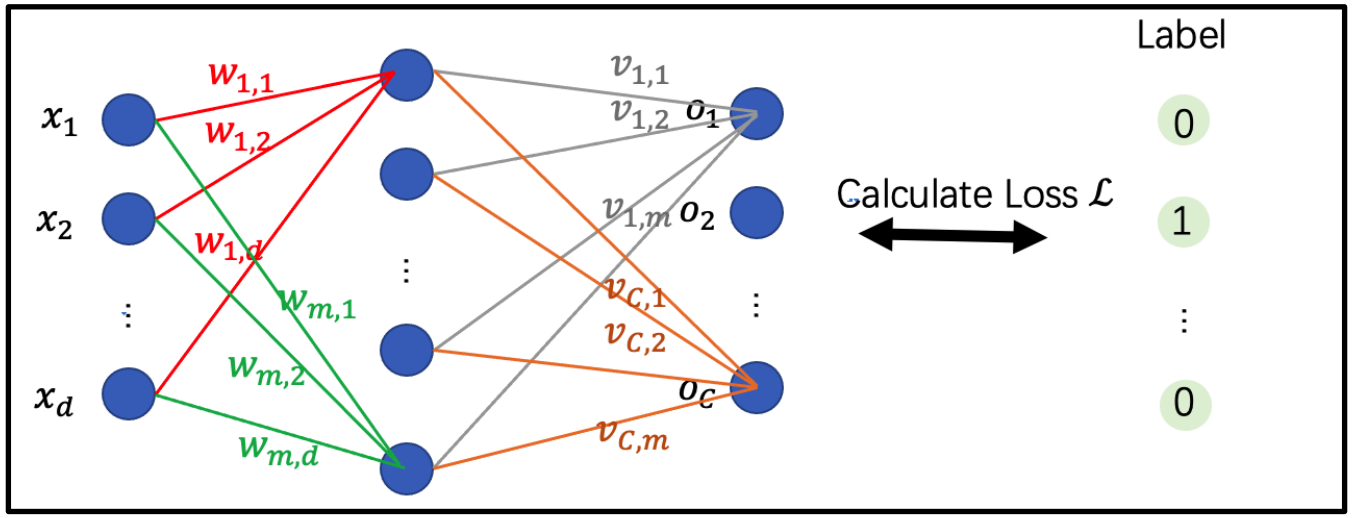
| | $w_1$ | $w_2$ | $v_1$ | $v_2$ |
|---|---|---|---|---|
| $w_1$ | | 0 | | 0 |
| $w_2$ | 0 | | 0 | |
| $v_1$ | | 0 | | 0 |
| $v_2$ | 0 | | 0 | |

$$\frac{\partial^2 \ell}{\partial w_1 \partial w_1} = v_1^2 \qquad \frac{\partial^2 \ell}{\partial w_1 \partial w_2} = 0 \qquad \frac{\partial^2 \ell}{\partial w_1 \partial v_1} = 2 w_1 v_1 \qquad \frac{\partial^2 \ell}{\partial w_1 \partial v_2} = 0$$

This is a most simple block-circulant-block-diagonal matrix

Some Hessian entries are naturally 0

# Why does the block-circulant $H_{wv}$ vanishes?




$$H_{w_i v_j} = \frac{\partial^2 \ell}{\partial w_i \, \partial v_j^T} = \begin{bmatrix} 0 & \cdots & a_{i,1} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{i,d} & 0 & \cdots & 0 \end{bmatrix} + O\left(\frac{1}{c}\right) \in R^{d \times m}, \text{ where } a_{i,d'} = -\frac{1}{N} \sum_n \sum_c \left(\delta_{y_n,c} - p_{n,c}\right) v_{c,i} \mathbb{I}\left(w_i^T x_n \geq 0\right) x_{n,d'}$$

- **Linear algebra perspective (like previous part):**
  from computation graph, only $v_{1,i}, \cdots, v_{C,i}$ are linked to $w_i$. So only $i$-th column in $H_{w_i v_j}$ is non-zero

**Key take-away:** $H_{wv} \approx O(optimality\ gap)$, which are expected to vanish (experiments: vanishes quickly as training begins)

# Rigorous Theory: 2-layer ReLU NNs + CE loss

- A1 is restricted
- A2 is widely used

**Assumption 1** *The entries of the data matrix* $X_N = (x_1, \cdots, x_N) \in \mathbb{R}^{d\times N}$ *are i.i.d.* $\mathcal{N}(0,1)$.

**Assumption 2** *The model weights in* $W$ *and* $V$ *are initialized by LeCun initialization. That is: for the linear model,* $V_{i,j} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{1}{d})$, $i \in [C], j \in [d]$; *for 1-hidden-layer network,* $W_{i,j} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{1}{d})$, $i \in [m], j \in [d]$, $V_{i,j} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{1}{m})$, $i \in [C], j \in [m]$.

**Theorem 2** **(1-hidden-layer networks.)** *Consider the Hessian expressions in (8) to (13), and assume Assumptions 1 and 2 hold. Then for any fixed* $m \geq 3$, *suppose* $d, N \to \infty$, $\frac{d}{N} \to \gamma \in (0, +\infty)$, *it holds that*

$$\lim_{d,N\to\infty} \frac{\mathbf{E}\left[\left\|\frac{\partial^2 \ell_{\text{MSE}}(W,V)}{\partial w_i \partial w_j^\top}\right\|_{\text{F}}^2\right]}{\mathbf{E}\left[\left\|\frac{\partial^2 \ell_{\text{MSE}}(W,V)}{\partial w_i \partial w_i^\top}\right\|_{\text{F}}^2\right]}, \quad \lim_{d,N\to\infty} \frac{\mathbf{E}\left[\left\|\frac{\partial^2 \ell_{\text{CE}}(W,V)}{\partial v_i \partial v_j^\top}\right\|_{\text{F}}^2\right]}{\mathbf{E}\left[\left\|\frac{\partial^2 \ell_{\text{CE}}(W,V)}{\partial v_i \partial v_i^\top}\right\|_{\text{F}}^2\right]} \tag{28}$$

vanish at the rate $\mathcal{O}(1/C)$, $\mathcal{O}(1/C^2)$, respectively, and the block-diagonal structure also emerges as $C$ increases.

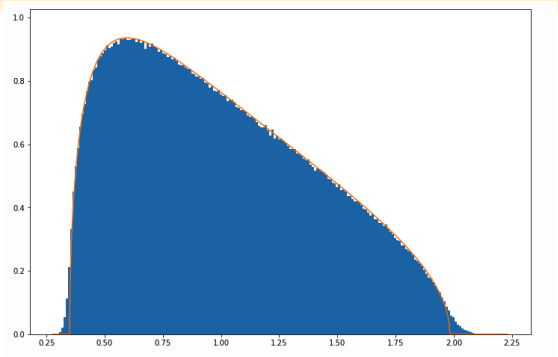
**Key messages:**
the block-diagonal structure arises when
# classes C (i.e., NN output dim) → ∞

# Key Challenges in the Proof

Diagonal Hessian block: $$\frac{\partial^2 \ell_{\mathrm{CE}}(V)}{\partial v_i \partial v_i^\top} \overset{(5)}{=} \frac{1}{N}\sum_{n=1}^{N} p_{n,i}(1-p_{n,i}) x_n x_n^\top := \frac{1}{N} X_N \Lambda_N X_N^\top \in \mathbb{R}^{d\times d},$$

Q: How to characterize the Hessian block $\mathrm{E}||\frac{1}{N} X_N \Lambda_N X_N^T||_F$?

Just 2nd-order moment of the eigenvalue distribution



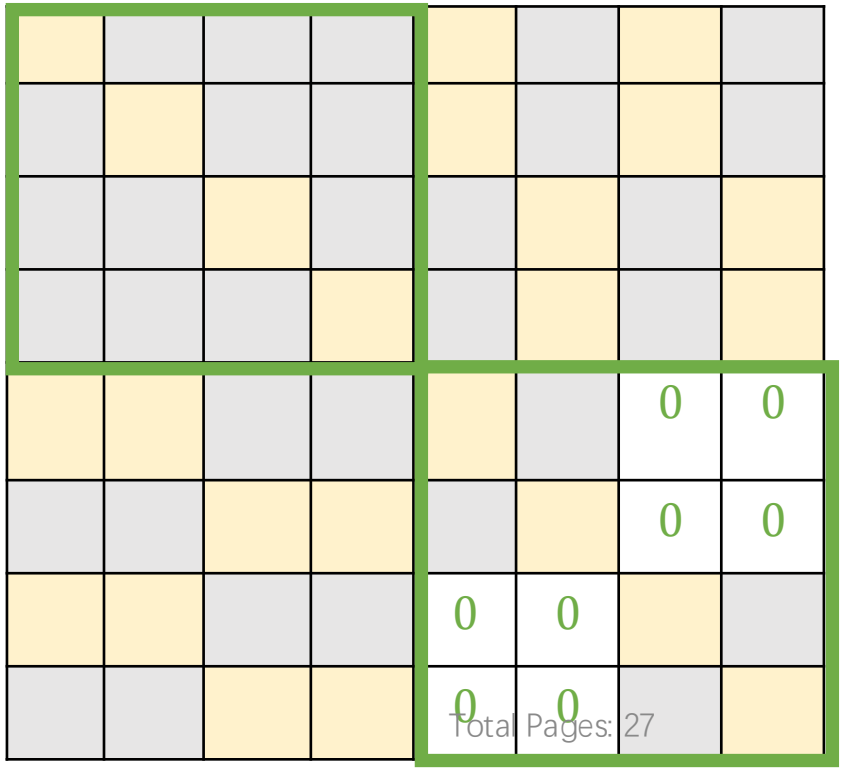

Eigenvalue histogram of $A_n = \frac{1}{n} X\Lambda X^T \in R^{d\times d}, \Lambda = I, n = 50, d = 300$, 1000 samples of $A_n$

**We will use Random Matrix Theory (RMT), but classical methods cannot be directly applied:**

- If $X_N, \Lambda_N$ are **independent**: can use **Generalized Marchenko–Pastur Theorem (1967)**
- In our $\frac{1}{N} X_N \Lambda_N X_N^T$, $X_N, \Lambda_N$ are **clearly NOT independent**, so MP theorem cannot be applied
- Dependent matrix product is a difficult topic in RMT
- **Our solution:** a new method built upon **the Lindeberg principle** (originally proposed to prove CLT)

# Summary: 3-level sources of block-diag structure

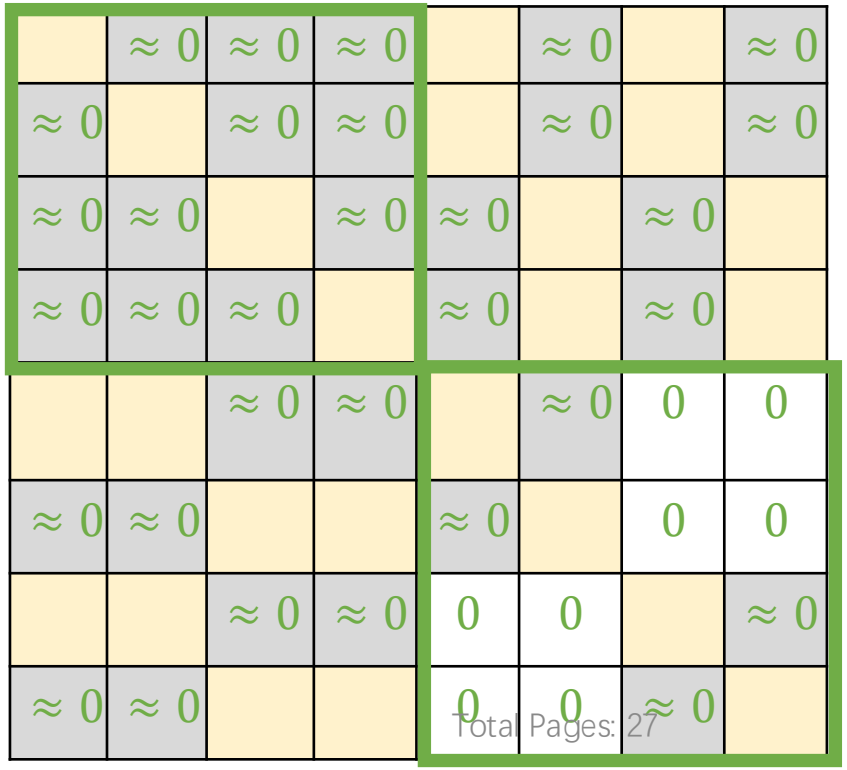
- **Level 1: definition of matrix product: some zeros, no links**



# Summary: 3-level sources of block-diag structure

- **Level 1: definition of matrix product: some zeros, no links**
- **Level 2: #Class C goes to infinity: weaken many links in $H_{ww}, H_{vv}$**

**Static force**
**(Proved by RMT at random initialization)**



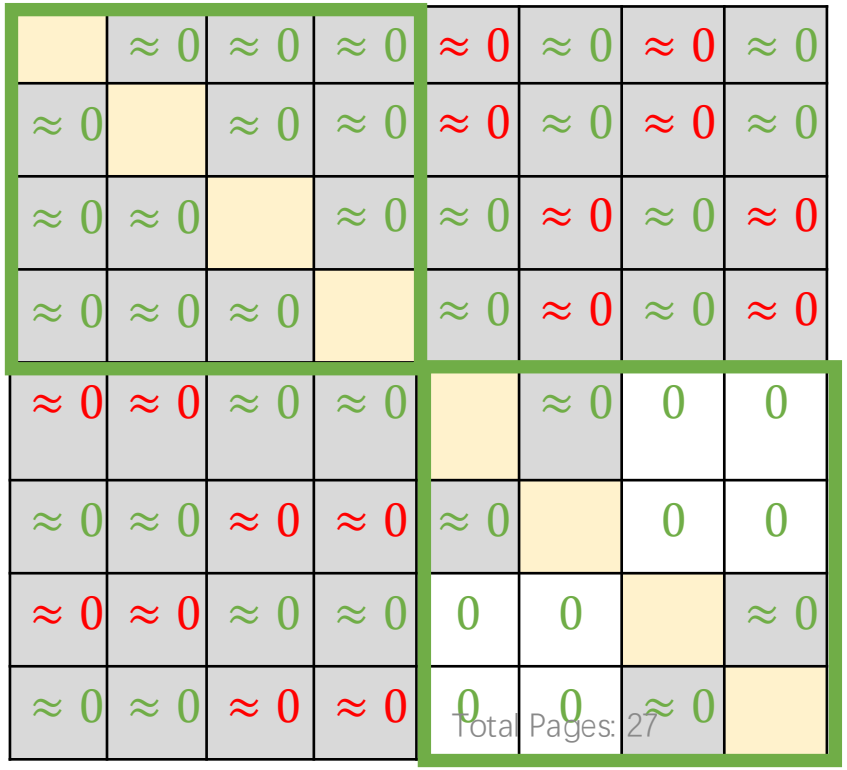
# Summary: 3-level sources of block-diag structure

- **Level 1: definition of matrix product: some zeros, no links**
- **Level 2: #Class C goes to infinity: weaken many links in $H_{ww}, H_{vv}$**

} **Static force**
**(Proved by RMT at random initialization)**

- **Level 3: Training: eliminates strong links in $H_{wv}$**

} **Dynamic force**
**(See from direct calculation)**

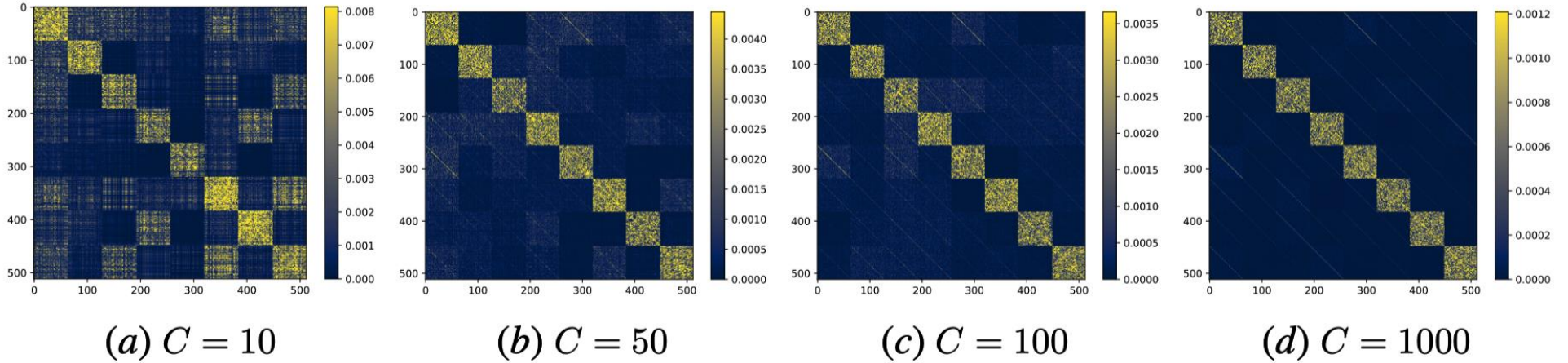



**Limitations:**
**Current RMT** does not consider training dynamics

Still incomplete, has a long way to go…

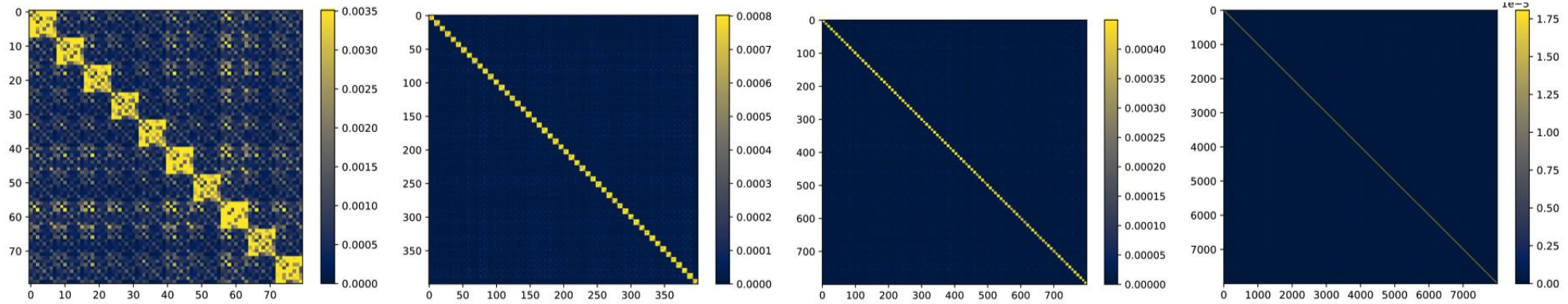
# Experiments: The effect of Increasing matrix size

Hessian of hidden weights



$(a)\ C = 10$ $(b)\ C = 50$ $(c)\ C = 100$ $(d)\ C = 1000$

Hessian of output weights



$(a)\ C = 10$ $(b)\ C = 50$ $(c)\ C = 100$ $(d)\ C = 1000$

- The Hessian blocks of **1-hidden-laye NN** with 8 hidden neurons + #class C at random init.
- The block-diag structure **becomes clearer as C increases**

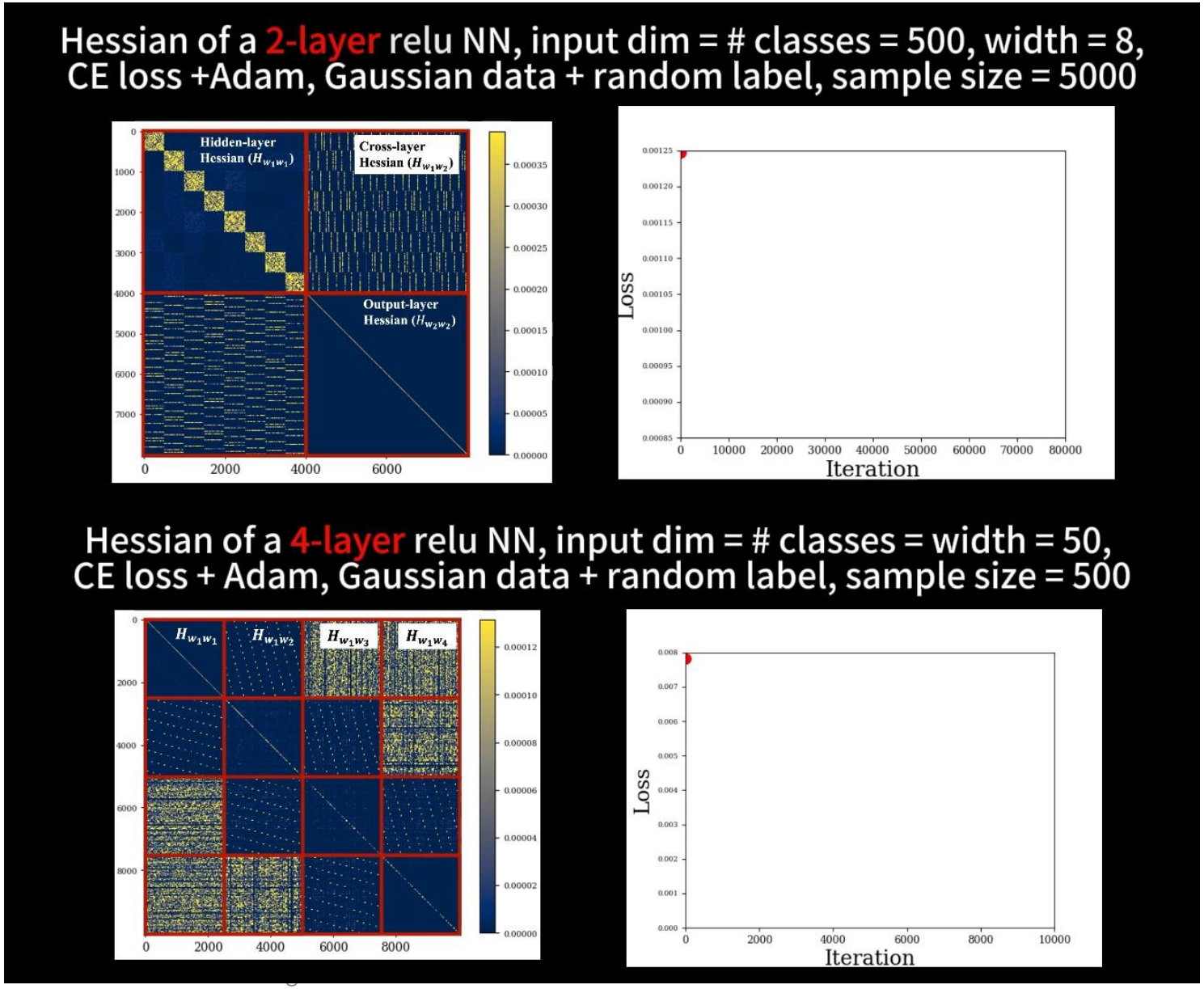
# Hessian for Deep NNs?

**For a rough estimate:**
just check the links in the computational graph:

Check if there exists
a connected path between two links

**Source of special Hessian:**
consecutive multiplication of
large and well-shaped matrix variables

$\ldots W_1 W_2 W_3 W_4 \ldots$



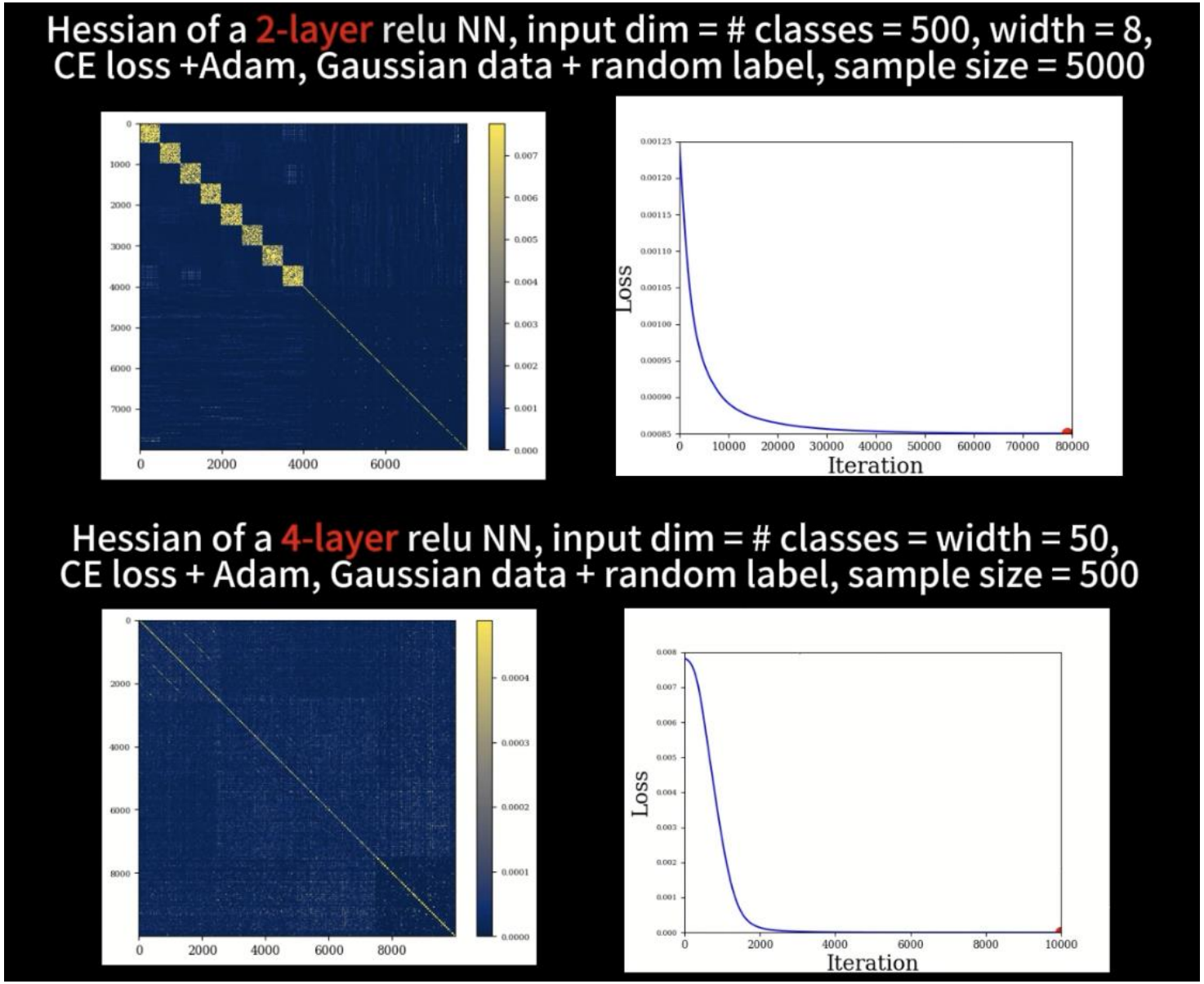
# Hessian for Deep NNs?

**For a rough estimate:**
just check the links in the computational graph:

Check if there exists
a connected path between two links

**Source of special Hessian:**
consecutive multiplication of
large and well-shaped matrix variables

$\ldots W_1 W_2 W_3 W_4 \ldots$



# Contents

- Part I: Empirical observations
- Part II: Intuitions and Theory
- **Part III: Implications to Muon**

# Why Adam < Muon? A preconditioner perspective

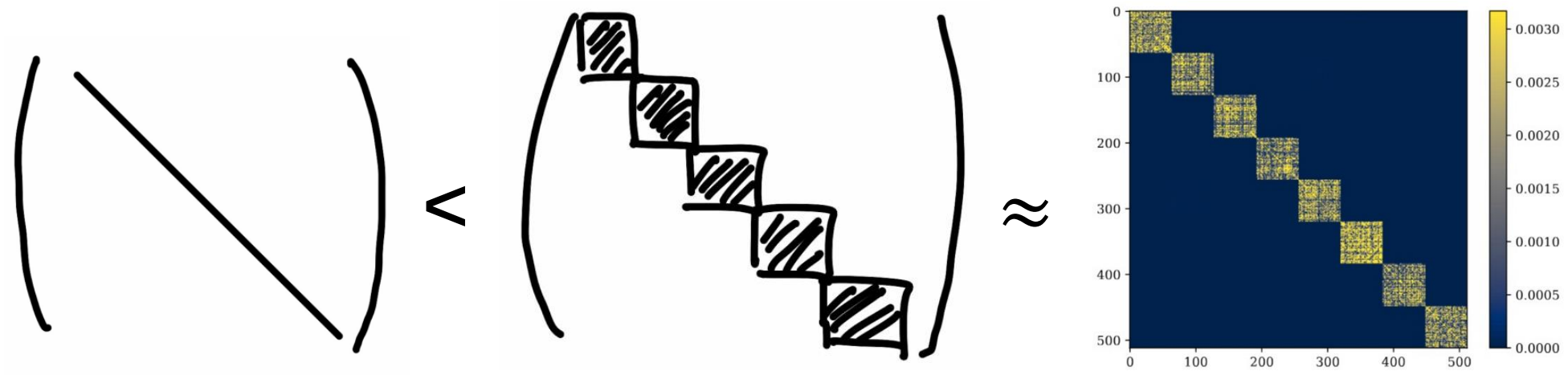
- **Matrix form of Muon:** $W \in R^{m \times n}$

$$W_{k+1} = W_k - \eta_k \left(M_k M_k^T\right)^{-\frac{1}{2}} M_k \; = W_k - \eta_k \, I_m \; M_k \left(M_k^T M_k\right)^{-\frac{1}{2}}$$

- **Flattened vector form of Muon:**

$$\mathrm{vec}(W_{k+1}) = vec(W_k) \; - \eta_k \left( I_m \bigotimes \left(M_k^T \, M_k\right) \right)^{-\frac{1}{2}} \; vec(M_k)$$

**we used the proposition of Kronecker product:** $\mathbf{A} \otimes B \; vec\,(C) = ACB$




Adam preconditioner

Muon preconditioner
(1 block = 1 neuron )

Hessian structure
(1 block = 1 neuron)

# How to improve Muon?

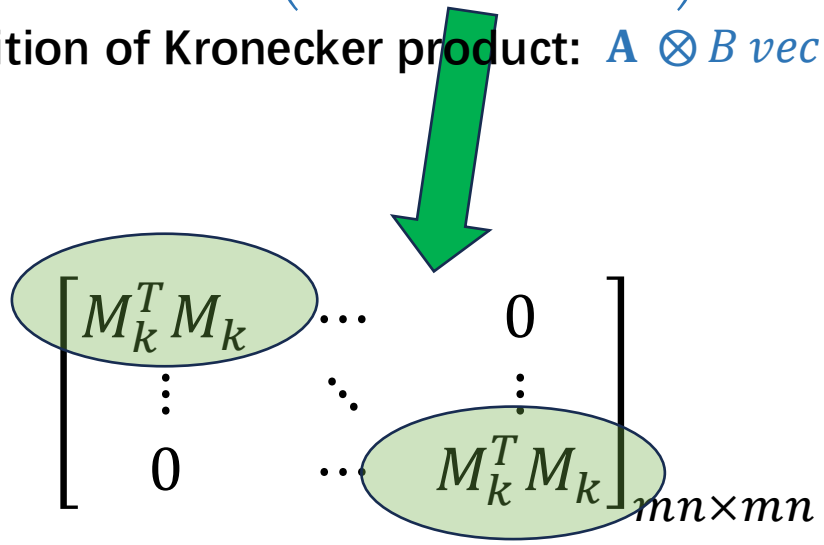
- Matrix form of Muon: $W \in R^{m \times n}$

$$W_{k+1} = W_k - \eta_k \left(M_k M_k^T\right)^{-\frac{1}{2}} M_k = W_k - \eta_k \, I_m \; M_k \left(M_k^T M_k\right)^{-\frac{1}{2}}$$

- Flattened vector form of Muon:

$$\mathrm{vec}(W_{k+1}) = vec(W_k) - \eta_k \left( I_m \bigotimes \left(M_k^T M_k\right) \right)^{-\frac{1}{2}} vec(M_k)$$

We used the proposition of Kronecker product: $\mathbf{A} \otimes B \; vec\,(C) = ACB$

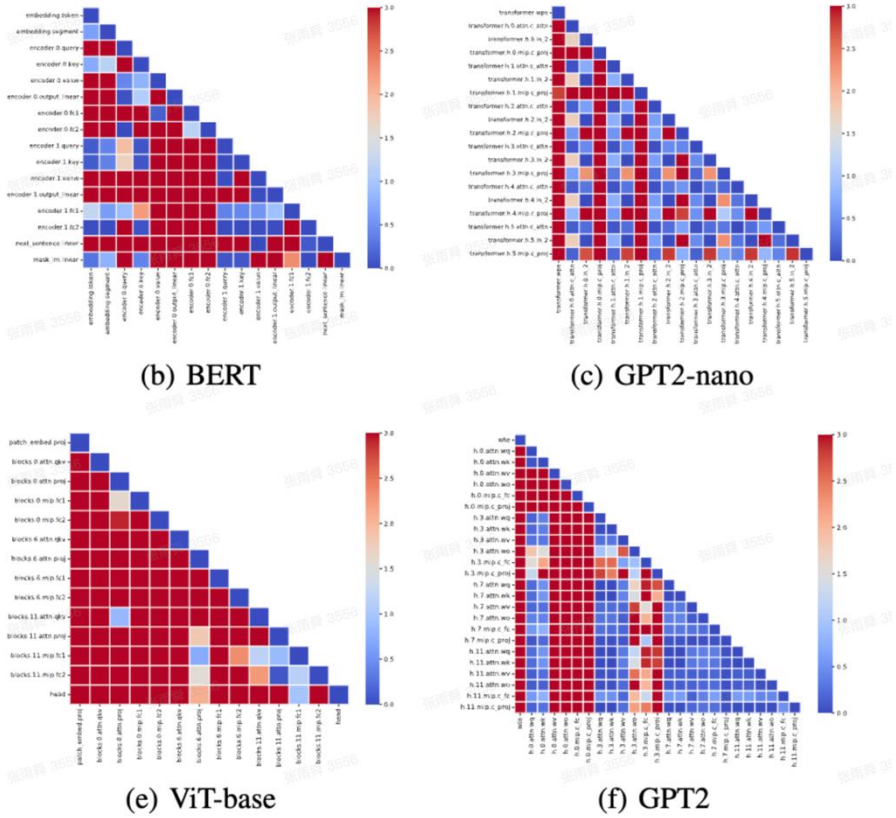



**Key observation:**
Muon use the **same** preconditioner across neurons …

**But neurons in Transformers are heterogeneous… [1, 2]**
**How to improve Muon?**


[1] **Why Transformers Need Adam: A Hessian Perspective**, NeurIPS 2024
[2] Technical report of Aurora optimizer, May, 2026

# LLM neurons are heterogeneous



(b) BERT (c) GPT2-nano

(e) ViT-base (f) GPT2

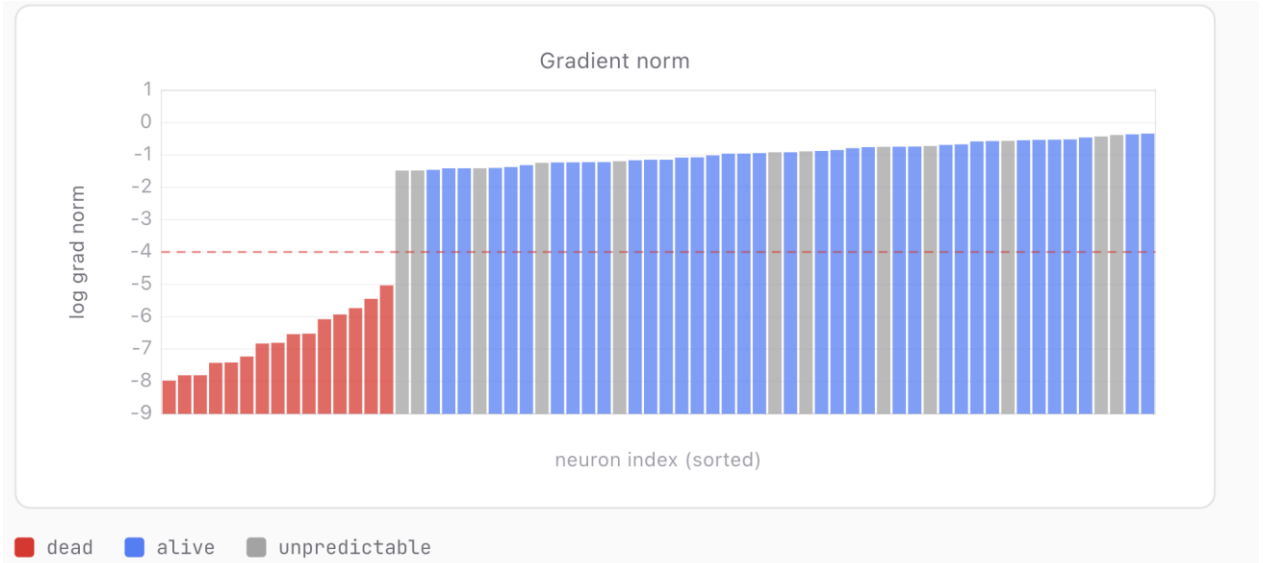



Figure 5. Visualizing each criterion for identifying dead neurons: low effective gradient norm, low effective update norm, and persistence of both over training. We identify a neuron as dead if it satisfies all three criteria.

**Finding 1:** In LLMs, the Hessian spectrum of different **weight matrices** are heterogeneous [1]

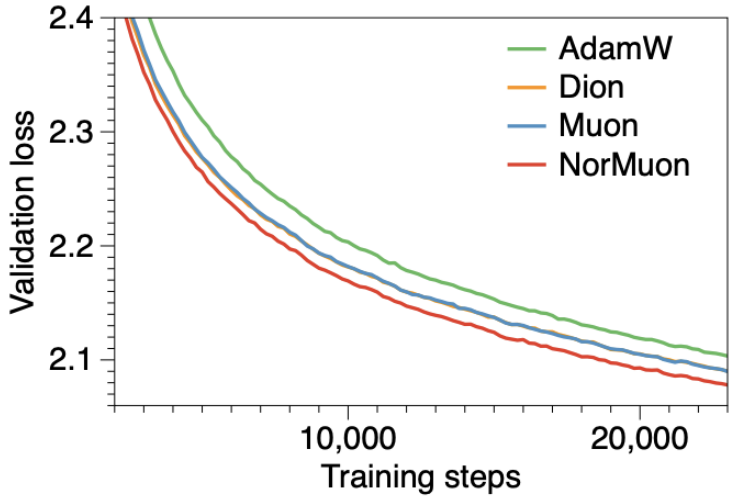
[1] **Why Transformers Need Adam: A Hessian Perspective**, NeurIPS 2024

**Finding 2:** In LLMs, the gradient of different **rows in the same matrix** are heterogeneous [2]

[2] **Aurora optimizer blog**, May, 2026

# Solution: Adam-mini + Muon

- **NorMuon:** add neuronwise lr to Muon
- by Microsoft and GaTech, including an author of LoRA




(a) Pretraining results of 1.1B model.

**Algorithm 1** NorMuon

**Input:** Initial weights $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, loss $L$, learning rate $\eta$, momentum parameters $(\beta_1, \beta_2)$, perturbation parameter $\varepsilon$, weight decay $\lambda$.
Initialize $\mathbf{M}_0 \in \mathbb{R}^{m\times n} \leftarrow \mathbf{0}$, $\mathbf{v}_0 \in \mathbb{R}^{m} \leftarrow \mathbf{0}$
**for** $t = 1, 2, \ldots$ **do**
  $\mathbf{G}_t \leftarrow \nabla_{\mathbf{W}} L(\mathbf{W}_t)$
  $\mathbf{M}_t \leftarrow \beta_1 \mathbf{M}_{t-1} + (1-\beta_1)\mathbf{G}_t$
  $\mathbf{O}_t \leftarrow \mathrm{NS5}(\mathbf{M}_t)$
  $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathrm{mean}_{\mathrm{cols}}(\mathbf{O}_t \odot \mathbf{O}_t)$
  $\mathbf{V}_t \leftarrow \mathrm{ExpandRows}(\mathbf{v}_t)$
  $\widehat{\mathbf{O}}_t \leftarrow \mathbf{O}_t \oslash \left(\sqrt{\mathbf{V}_t} + \varepsilon\right)$
  $\hat{\eta} = 0.2\eta\sqrt{mn}/\|\widehat{\mathbf{O}}_t\|_F$
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\lambda\mathbf{W}_t - \hat{\eta}\widehat{\mathbf{O}}_t$
**end for**

## NorMuon: Making Muon more efficient and scalable

Zichong Li [*1], Liming Liu [*1 †], Chen Liang [2], Weizhu Chen [2], Tuo Zhao [1]
[1]Georgia Tech [2]Microsoft

This observation motivates our key insight: while Muon's orthogonalization effectively reduces the condition number of updates, the remaining high variance in neuron norms still creates an imbalanced learning dynamic, potentially leading to inefficient parameter usage. Drawing inspiration from Adam-mini's success Zhang et al. (2025) with per-neuron adaptive learning rates, we propose to incorporate second-order momentum to normalize these disparate scales and ensure more balanced parameter updates. Our method, **NorMuon**, augments Muon's orthogonalization with neuron-wise adaptive learning rates computed from accumulated second-order statistics.

# Solution: Adam-mini + Muon

**NorMuon** has won the world record of fastest NanoGPT speedrun optimizer, twice!
(2025 Nov; 2026 May)

**Lasted tweet by Keller Jordan in May 3rd 2026 (1st author of Muon, OpenAI):**
"Three new modded-NanoGPT optimization benchmark results, **all of them using NorMuon**"

**Keller Jordan**
@kellerjordan0

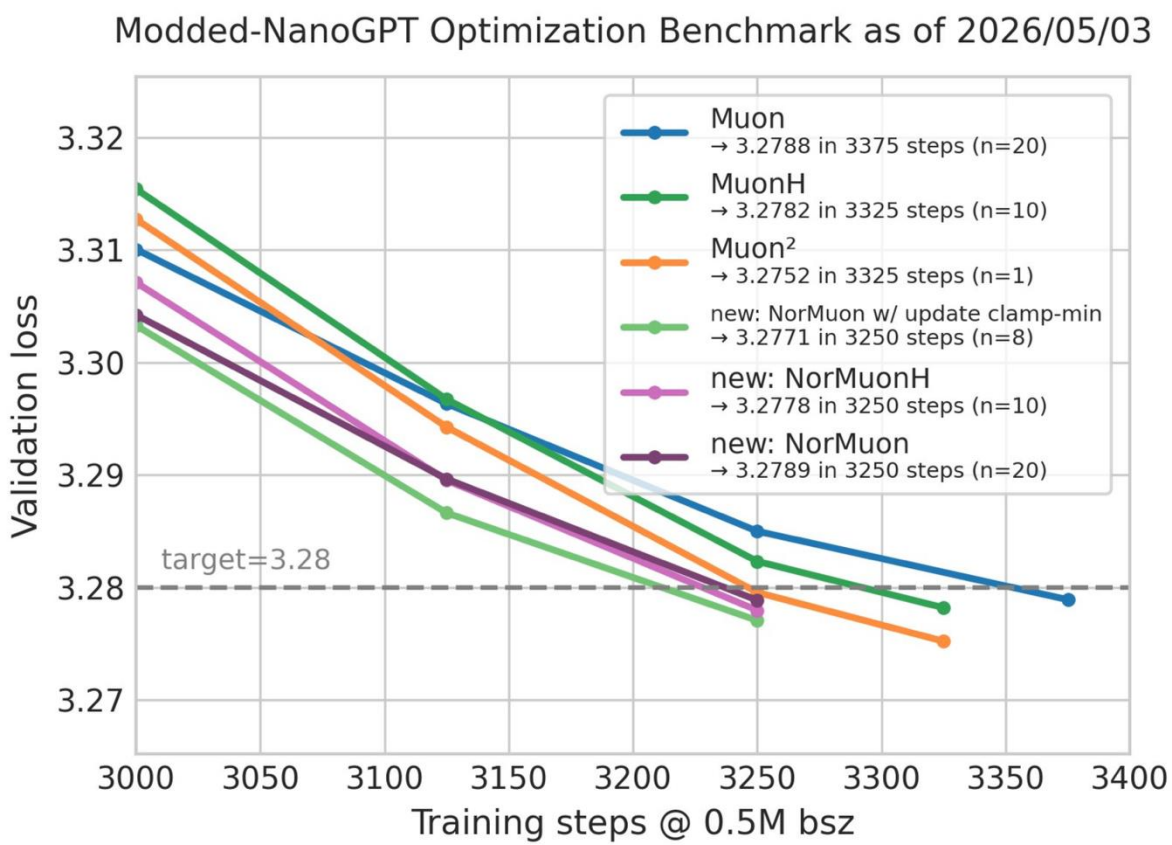
显示翻译

Three new modded-NanoGPT optimization benchmark results, all of them using NorMuon, have near-concurrently improved the benchmark record from 3325 to 3250 steps.

1) Kumar Krishna Agrawal (gh:kumarkrishna) used NorMuon with an update-clamping strategy (no weight decay, like hyperball) to reach the target loss in 3250 steps.
2) @wen_kaiyue used NorMuon with hyperball optimization.
3) Subsequently, Liming Liu (gh:lliu606) (one of the NorMuon authors!) used standard NorMuon with weight decay.

NorMuon is also used in the main NanoGPT speedrun track. Ali Naeimi did the earliest NorMuon runs on the optimization benchmark, albeit below stat sig.

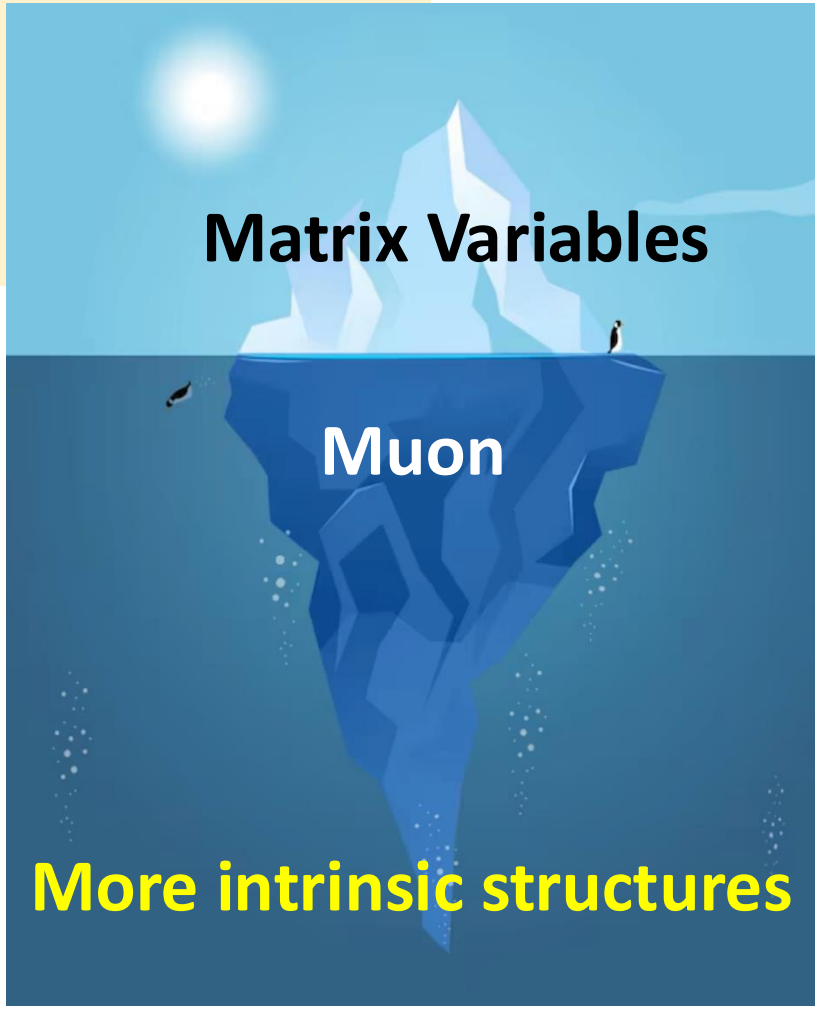
NorMuon authors: @li_zichong, Liming Liu, @chenliang1_, @WeizhuChen, and @tourzhao



# Summary

- **Source of special Hessian:**
  consecutive multiplication of **large** and **well-shaped** matrix variables
  - Neural nets, Transformers
  - Matrix factorization
  - Matrix completion
  - …
- **Implications to optimizers:** Adam, Muon and NorMuon

$$y = W_3 W_2 W_1 x$$



# Thanks for listening!