Why Transformers Need Adam: A Hessian Perspective

Yushun Zhang

The Chinese University of Hong Kong, Shenzhen, China

NeurIPS 2024

Presented at INFORMS Annual Meeting, Seattle, Oct, 2024

Motivation and Background

- **Adam** becomes the most popular algorithms in deep learning (DL). (>~~170,000 citations, by May 2024~~, >198k citations, by Oct 2024)

- **Default in** LLM (large language models)

```
optimizer = optim.Adam(net.parameters(), lr=args.lr, betas=(args.beta1, args.beta2), eps=1e-08,
                       weight_decay=args.weightdecay, amsgrad=False)
```

- **Empirical fact (sad?): Adam seems to be the only choice for ChatGPT**
  --Recent new algorithms (Sophia, Lion, etc.)
  cannot beat Adam on billion-parameter models.

Overview

In this talk, we discuss:

- Why LLM training requires Adam, not SGD?
- We explain it from Hessian spectrum perspective

- We will mostly present the numerical observations
- Will also show some initial theory

Let us start with SGD…

- Consider $\min_x f(x) := \sum_{i=1}^{n} f_i(x)$.

  n: number of samples (or mini-batches of samples)

  x: trainable parameters

- In the k-th iteration: Randomly sample τ_k from $\{1,2,\dots,n\}$

SGD (Stochastic gradient descent): $x_{k+1} = x_k - \eta_k \nabla f_{\tau_k}(x_k)$

SGD with momentum (SGDM):

$$m_k = (1-\beta_1)\nabla f_{\tau_k}(x_k) + \beta_1 m_{k-1}$$

$$x_{k+1} = x_k - \eta_k m_k$$

⇐ 1st order momentum

⇐ Iterate update

Adam

- $\min_x f(x) \coloneqq \sum_{i=1}^n f_i(x)$. In the k-th iteration: Randomly sample τ_k from $\{1,2,\dots,n\}$

- Adam (Kingma and Ba'15):
- $m_k = (1-\beta_1)\nabla f_{\tau_k}(x_k) + \beta_1 m_{k-1}$
- $v_k = (1-\beta_2)\nabla f_{\tau_k}(x_k) \circ \nabla f_{\tau_k}(x_k) + \beta_2 v_{k-1}$
- $x_{k+1} = x_k - \eta_k \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{m_k}{\sqrt{v_k}}$

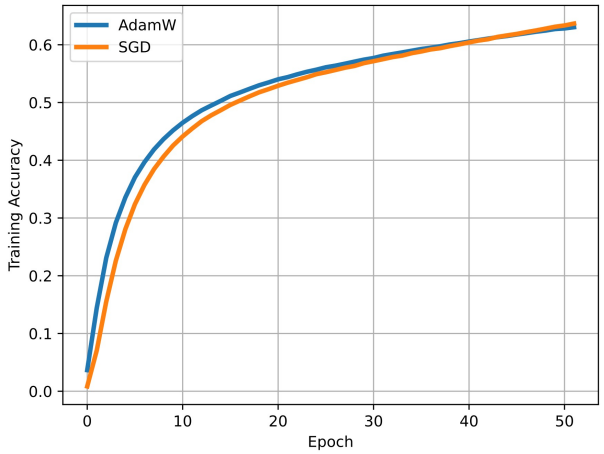
⇐ 1st order momentum

⇐ 2nd order momentum

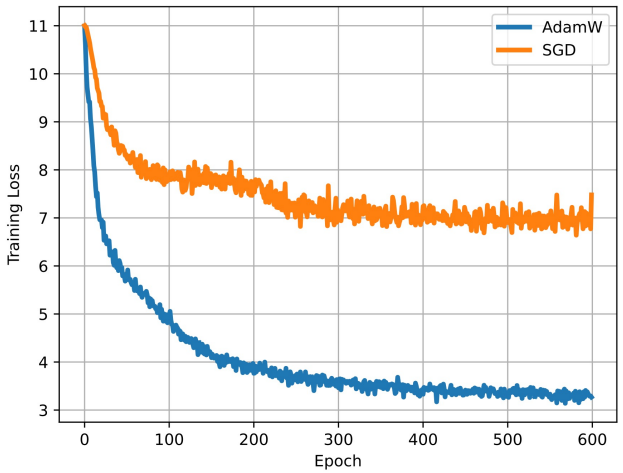
⇐ Iterate update

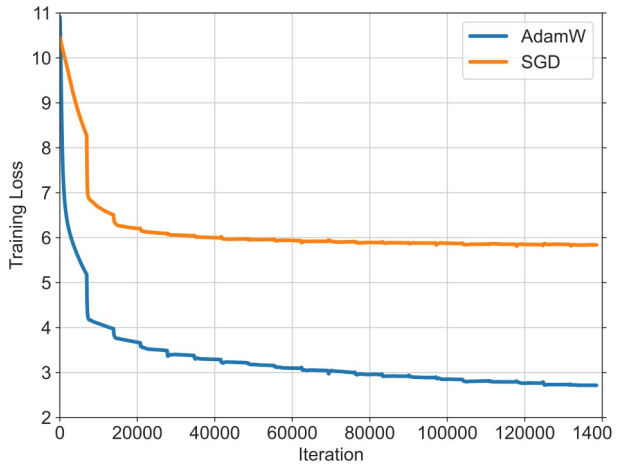
- β_1: Controls the 1st-order momentum m_k. Default setting: $\beta_1 = 0.9$
- β_2: Controls the 2nd-order momentum v_k. Default setting: $\beta_2 = 0.999$
- One important difference with SGD:
  - -- Adam use coordinate-wise lr $\frac{\eta}{v_i}$ ‘
  - -- SGD uses single lr η for all

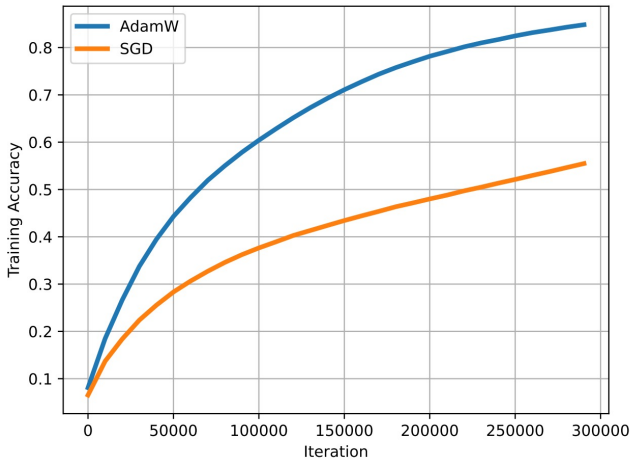
SGD works well on CNN, largely underperforms Adam on Transformers



VGG (CNN)







GPT2 BERT ViT

Transformers Need Adam, but why?

Literature on Why Adam better than SGD

- One perspective [J. Zhang et al. 19]
  NLP problems exhibit **heavy-tailed noise** in g → SGD fails

- However, [Chen, Kunstner, Schmidt'21] provides negative evidence:
  SGD is worse than Adam on Transformers, even in **full-batch case**
  (with no stochasticity)

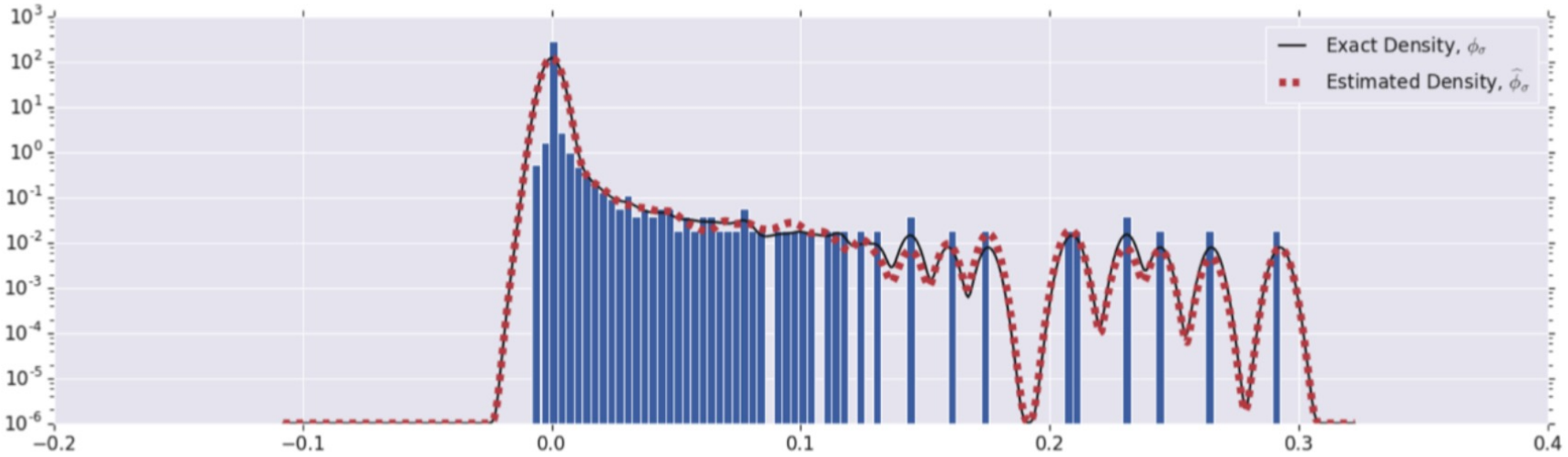
- So there shall be other reasons...

What problem structure might hamper SGD?

- **Hessian eigenvalues** largely decides behavior of gradient methods [Nocedal & Wright'99, Nesterov'13, Goh'17, Sun'2019]
- For instance: ill-conditioning slow down GD
- Can Hessian spectrum explain the gap of SGD and Adam?
- Unfortunately, no (not directly)…

Preliminary: Hessian spectrum

y-axis: frequency



x-axis: Eigenvalues.

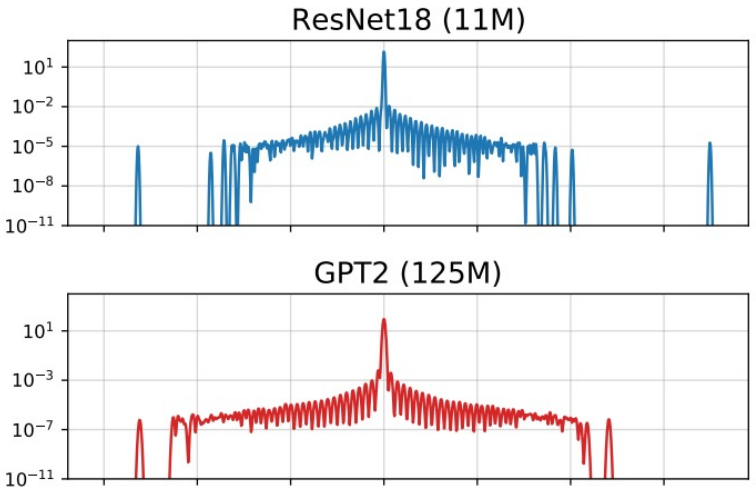
Figure from [Ghorbani et al. 19]

What is spectrum: histogram of eigenvalues

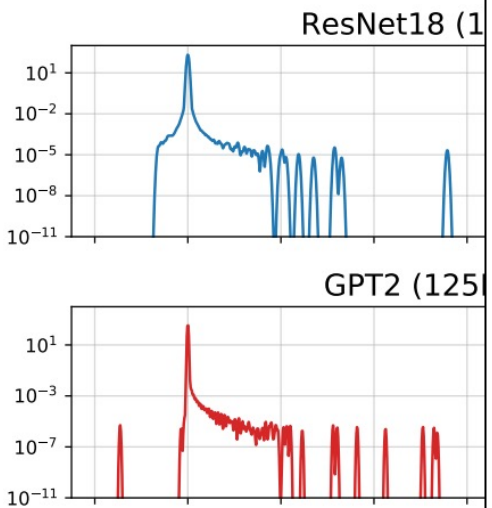
Remark: How to plot Hessian spectrum?
We use Stochastic Lanczos Quadrature (SQL)
[Bai, Fahey, and Golub 1996]
(will take >10 pages to explain, omitted today)

We will compute the Hessian spectrum for a wide range of neural networks

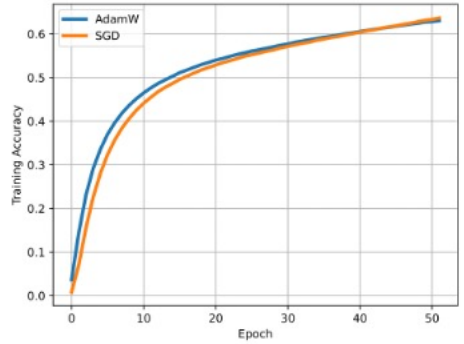
Hessian spectrum cannot explain the gap



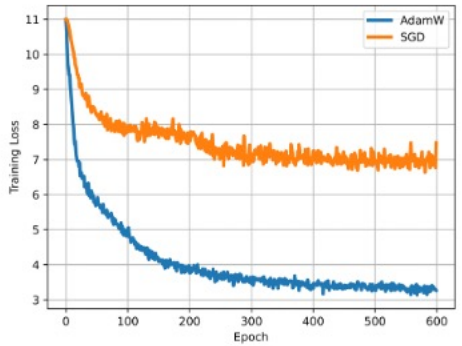
(a) Initialization



(c) 50% ste



SGD ≈ Adam on CNNs



SGD ≪ Adam on Transformers

CNN and Transformers: Spectrum looks quite similar!
(see more figures in the paper)

Something must be overlooked…

- Full hessian spectrum does not seem informative enough

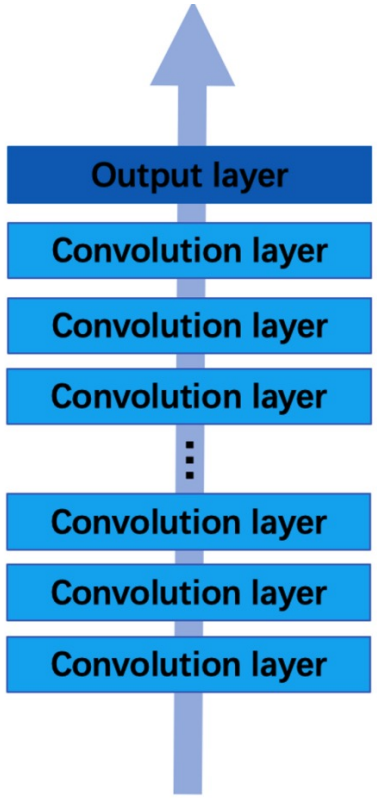
- What else?

- We find one important features that are overlooked:

  The build-up rules of the architecture

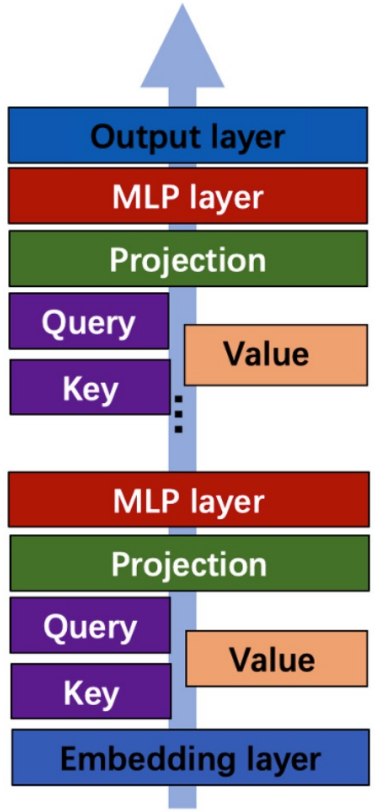
  -- Transformers are stacked up by different kinds of layers

Build-up rules of architectures



CNNs:

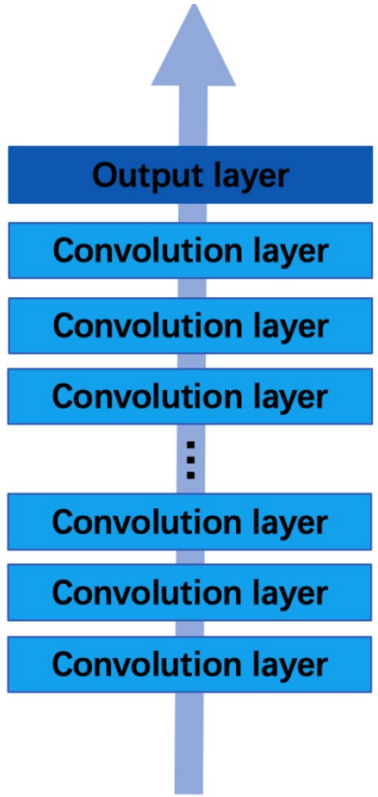
repetitive stack of similar layers



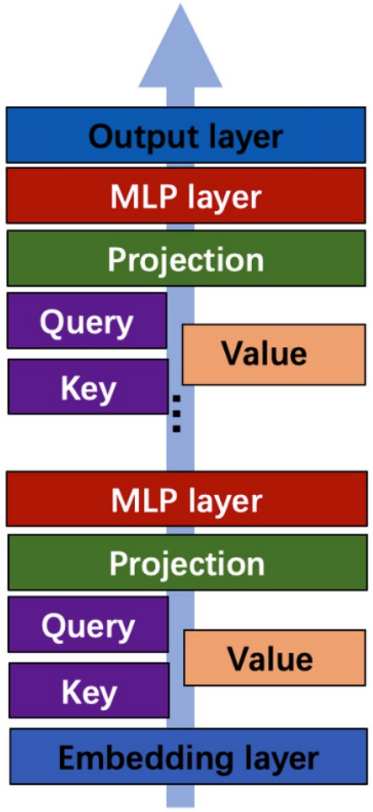
Transformers:

each block follow different design (e.g., Q, K, V, MLP)

Build-up rules of architectures



CNNs



Transformers

We hypothesize:

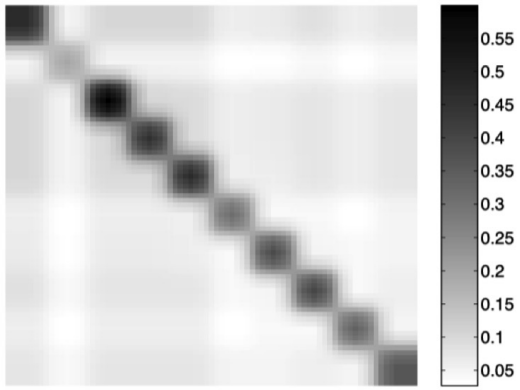
Different designs among parameter blocks

will affect

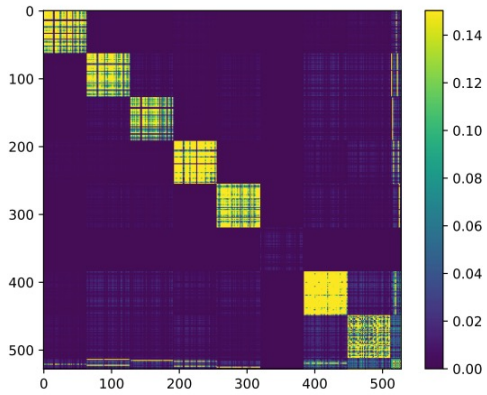
Hessian of these parameter blocks.

This inspires us to investigate the **blockwise Hessian spectra** (i.e., principle diag blocks in Hessian)

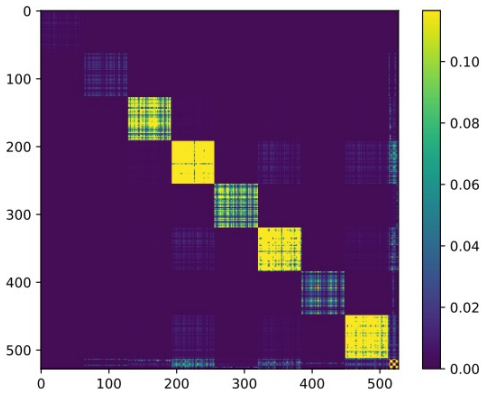
Another reason for studying blockwise Hessian **near Block Diagonal structure**



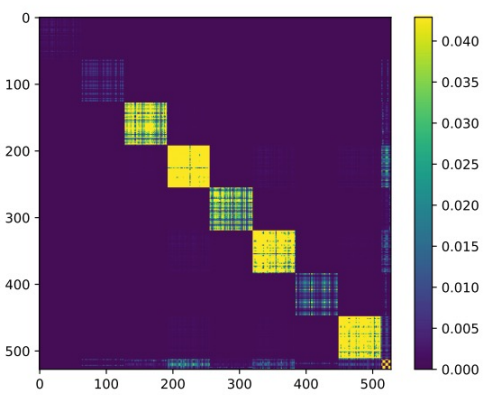
(a) Exact Hessian of a MLP (Collobert, 2004)



(b) Hessian of an MLP at 1% step



(c) Hessian of an MLP at 50% step



(d) Hessian of an MLP at 100% step

Proof (from Collobert 2004): for 1-hidden-layer network $f(\theta, x)$ + **Cross Entropy loss**, we have

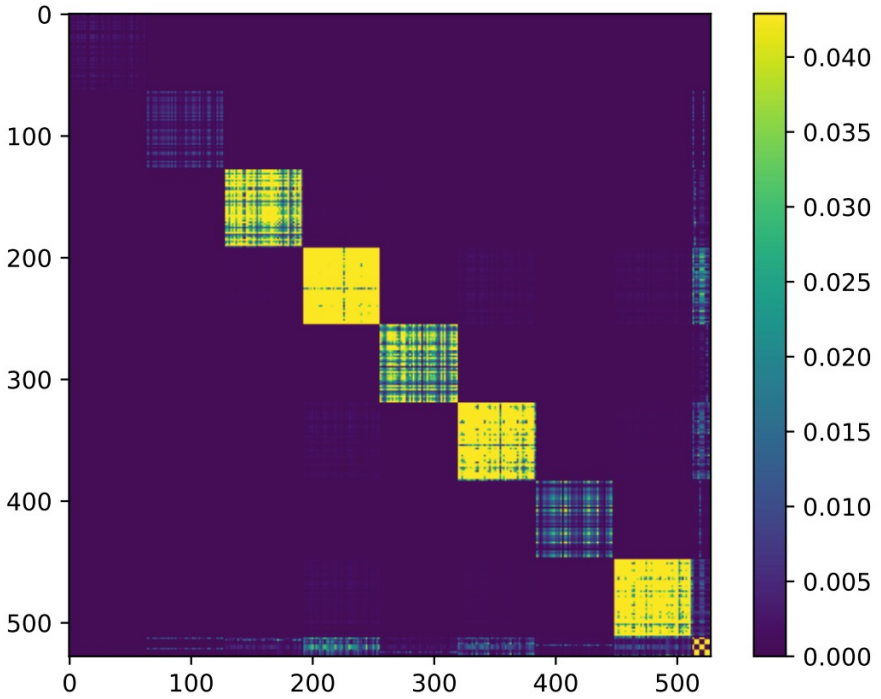
$$\frac{\partial^2 \ell(f(\theta,x),y)}{\partial w_i \partial w_j} = p_\theta(y|x)\left(1-p_\theta(y|x)\right) v_i v_j \phi'\left(w_i^\top x\right)\phi'\left(w_j^\top x\right) xx^\top \quad \text{for } i \neq j,$$

where $w_i \in R^{\,data\ dimension}$: the weight associated with the i-th output neuron

When we maximize $p_\theta(y|x)$, $p_\theta(y|x)\left(1-p_\theta(y|x)\right)$ will quickly shrink to zero

But this structure is largely overlooked for both Opt & DL community (sadly…)

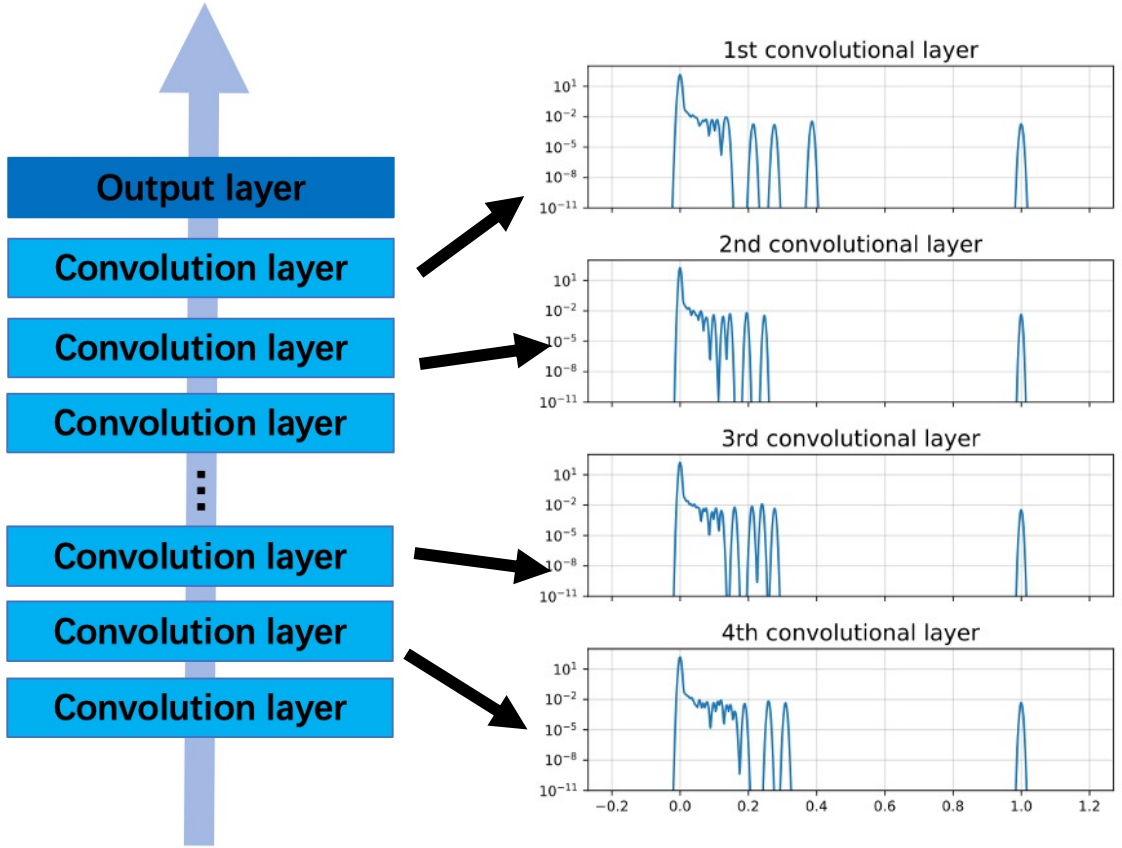
Blockwise Hessian spectrum might matters



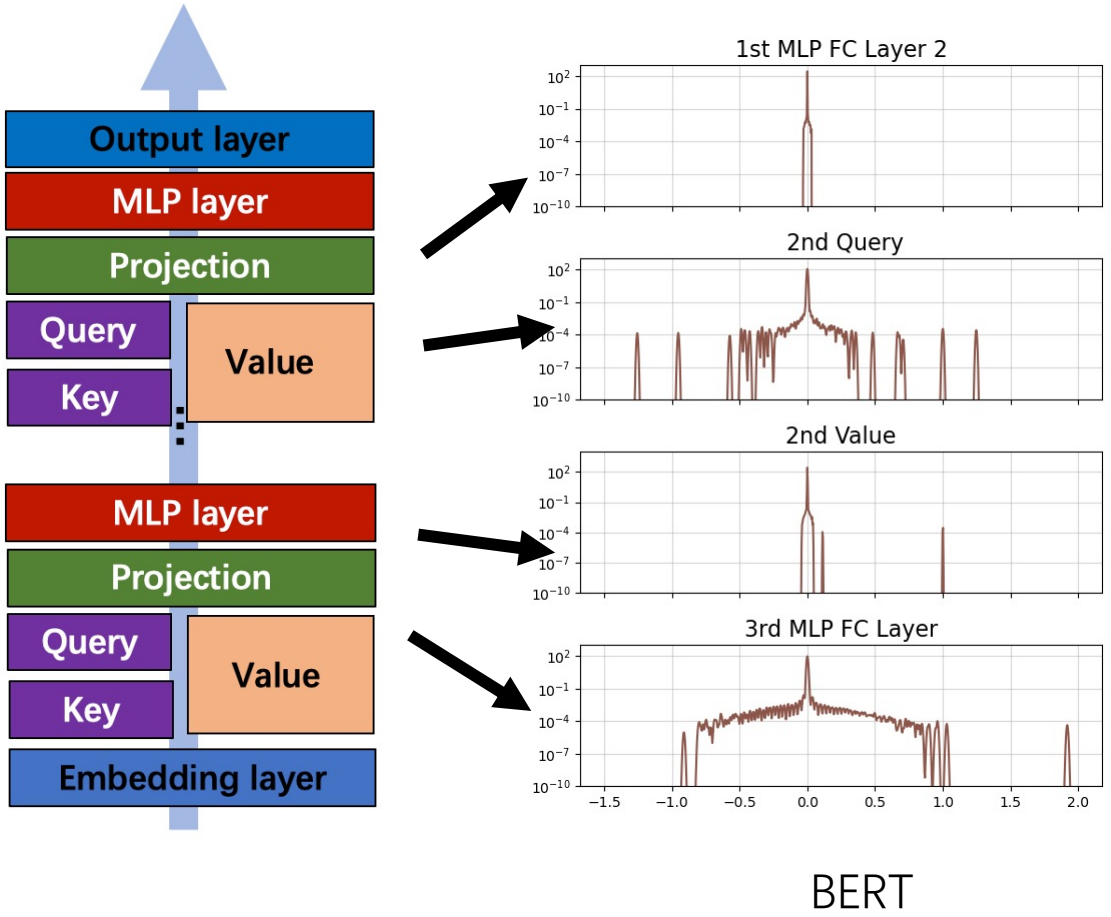
Conjecture: Eigenvalues in **each block (e.g., Q, K, V)** could be important

- What extra info over full spectrum?
- By linear algebra: **location** of eigenvalues

Blockwise Hessian spectrum



ResNet18



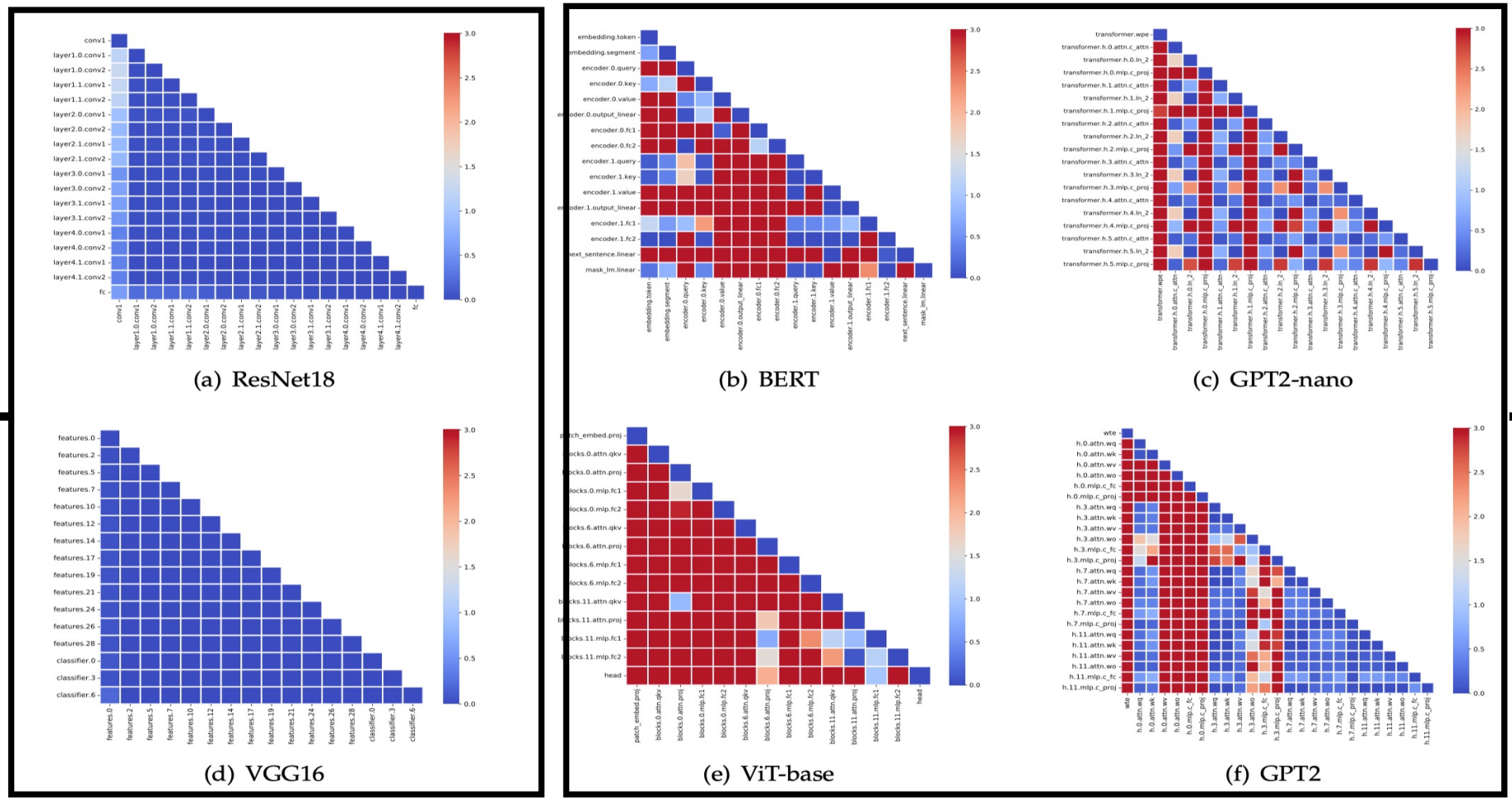
BERT

CNNs: blockwise spectrum are quite **similar**
We call it ``**homogeneity**”

Transformers: blockwise spectrum are largely **different**
We call it ``**heterogeneity**”

JS-distance among blocks

CNNs ←



→ Transformers

Figure 4: The JS distance among blockwise Hessian spectra for different models at initialization.

Observation 1: Heterogeneity is widely observed in Transformers, but not on CNNs!

Our Explanation: Why Transformer Needs Adam

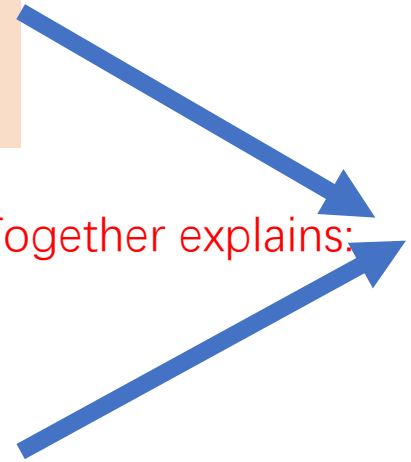
Observation 1:
Transformer's block Hessian are heterogeneous

Previous part

Observation 2:
If block Hessian are heterogeneous,
then SGD worse than Adam.

Next part:

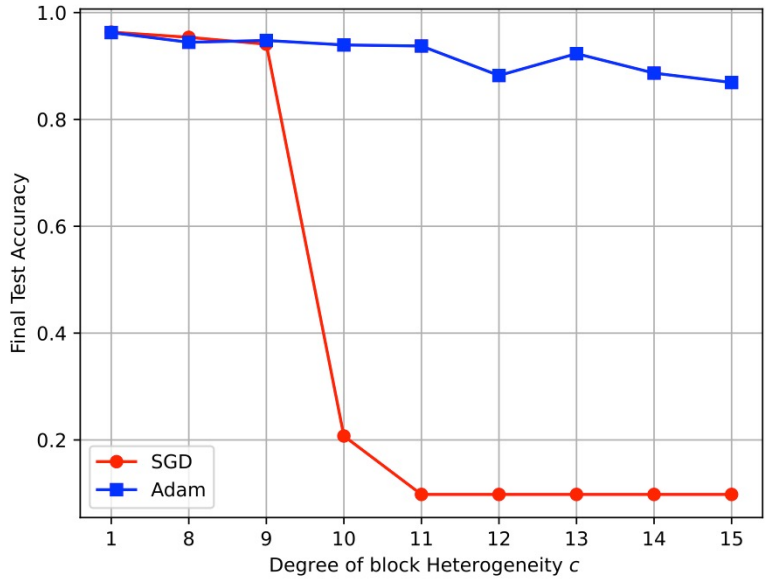
Claim 2: Hetereogeneity causes SGD worse than Adam.



Original claim:
Transformer:
SGD worse than Adam.

Hetereogeneity makes SGD worse: Example 2 (pure MLP)

- **Q:** Is Transformer the only architecture that is heterogeneous & SGD worse?
- **A:** No! We provide a few more heterogeneous examples that SGD is worse.



x-axis: degree of Heterogeneity
y-axis: final converged accuracy

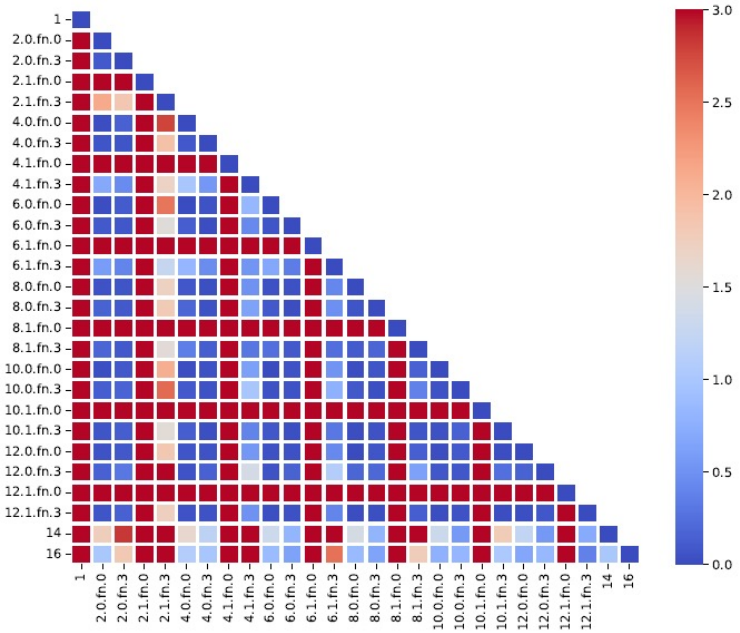
Example 2:
A man-made MLP on MNIST:
We exert heterogeneity by scaling each layer differently

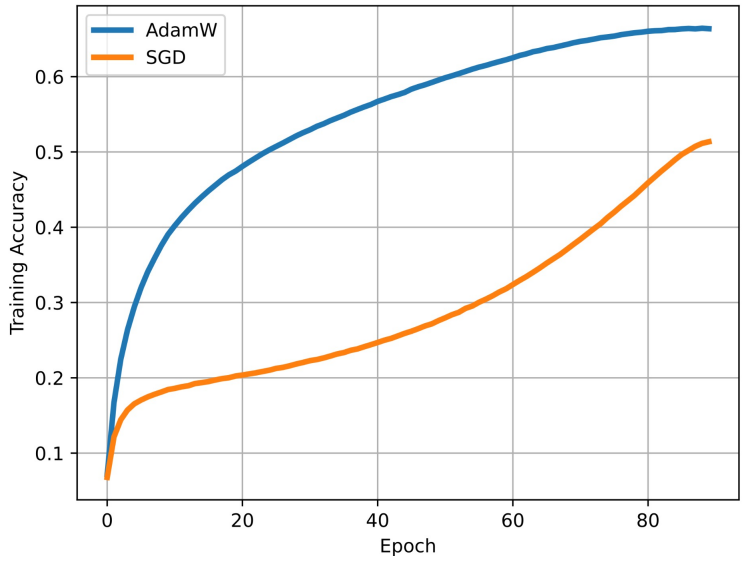
Observation:
SGD fails as heterogeneity grows
while Adam remains unaffected

Hetereogeneity makes SGD worse: Example 3 (MLP-mixer)

Example 3: MLP-mixer [1]

A pure MLP architecture that outperforms CNNs on ImageNet





Observation:

--MLP mixer is heterogeneous

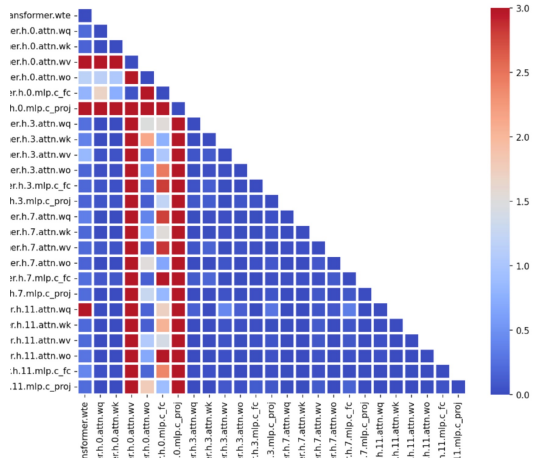
--SGD performs worse than Adam

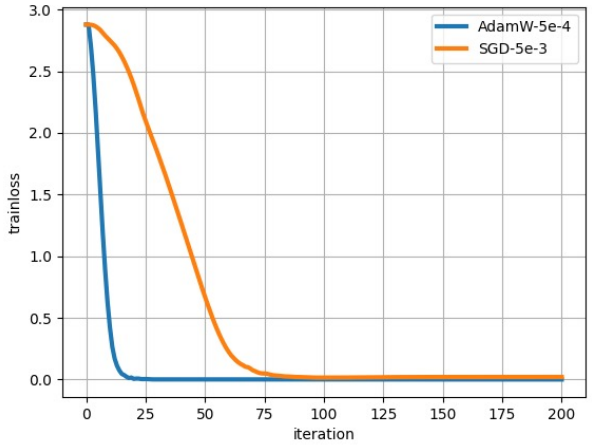
[1] MLP-mixer: An All-MLP Architecture for Vision. Tolstikhin et al. , NeurIPS 2021

Heterogeneity is redued after training

- We find **pretrained Transformers** suffer less heterogeneity

  → may explain why fine-tuning is easier
- SGD could work here: still slower, but can reach similar loss as Adam
- Similar phenomena also holds for ViT-base

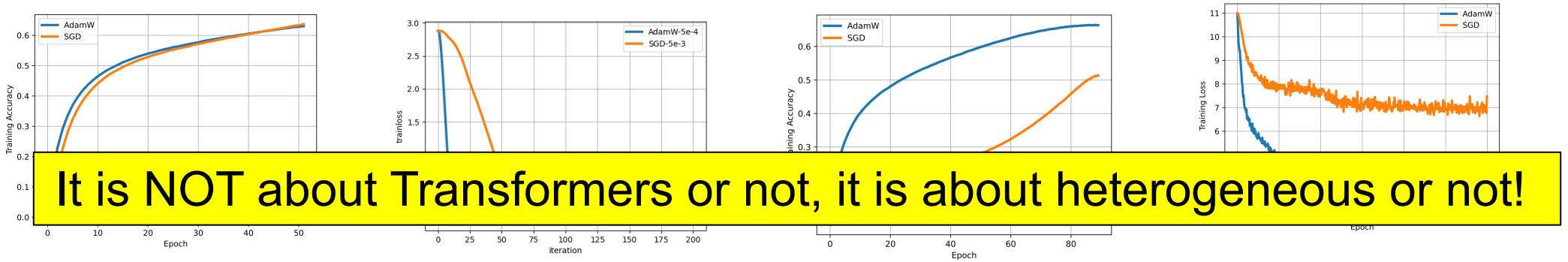




GPT2-125M (pretrained on 25B tokens): finetuning on a subset of Alpaca

Observation 3: Heterogeneity tends to reduce after (pre)-training

SGD performance v.s. Heterogeneity in Hessian



It is NOT about Transformers or not, it is about heterogeneous or not!

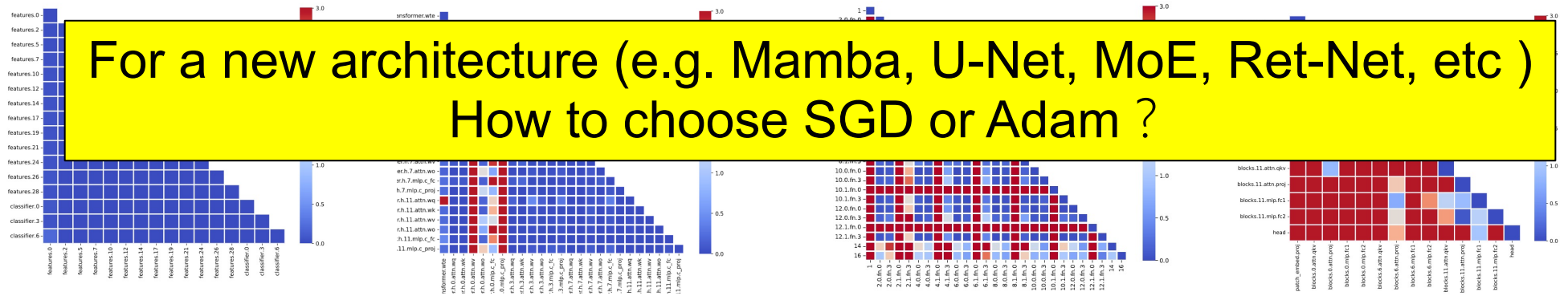
Easy for SGD

Difficult for SGD

Homogeneity in Hessian

Heterogeneity in Hessian

For a new architecture (e.g. Mamba, U-Net, MoE, Ret-Net, etc)
How to choose SGD or Adam ?



CNN | Transformer (pretrained) | MLP-mixer | Transformer

Empirical guidance: choose SGD or Adam?

- Introduce metric to **predict** the failure of SGD **before launching the training**

- **Our metric:** average JS distance of spectrum among blocks at step = 0, called JS^0
- This metric could be efficiently computed using Stochastic Lanczos Quadrature
  Our **PyTorch implementation**: https://github.com/zyushun/hessian-spectrum

Model	ResNet18	VGG16	GPT2 (pretrained)	MLP-mixer	BERT	GPT2	ViT-base
JS^0	0.10	0.09	18.84	34.90	53.38	83.23	286.41

For CNNs: 100x smaller than Transformers!

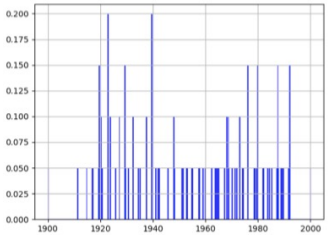
Initial theory

Hetereogeneity makes SGD worse: Quadratic Prob

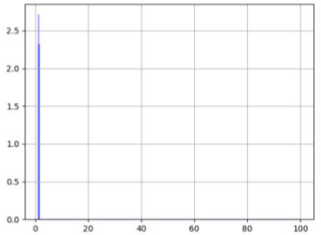
- Case study on quadratic models

$$\min_w \frac{1}{2} w^T H w, H = diag(H1, H2, H3, H4), H \text{ is PD}$$

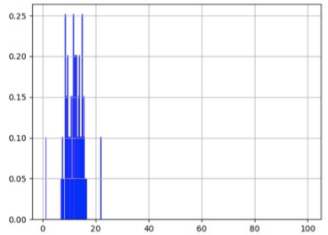
Case 1: H with Transformer-type spectra: sampled from GPT2



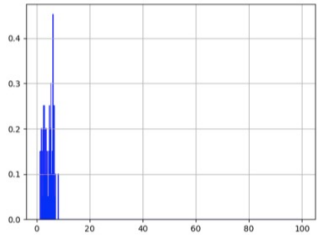
(a) Spectrum of H_1



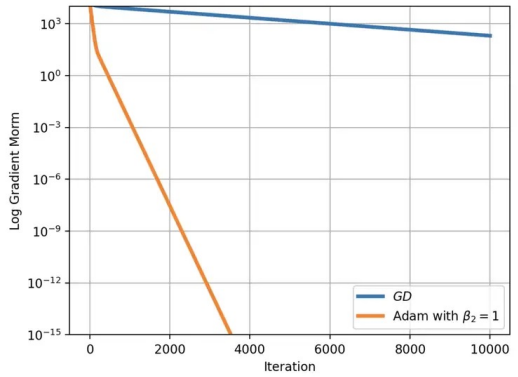
(b) Spectrum of H_2



(c) Spectrum of H_3



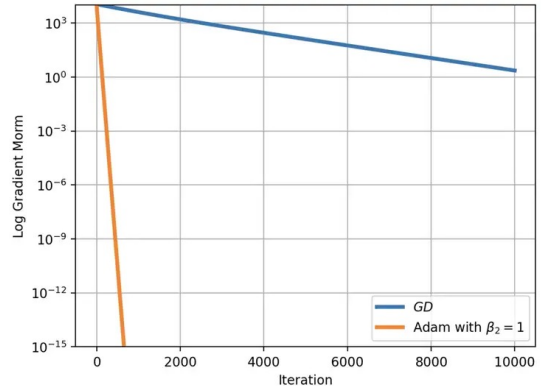
(d) Spectrum of H_4



GD is slower than Adam

Case 2: H with simplified **heterogeneous** spectra

Eigenvalues of H_l : { 1, 2, 3 } , { 99, 100, 101 } , { 1998, 1999, 2000 }



GD is slower than Adam

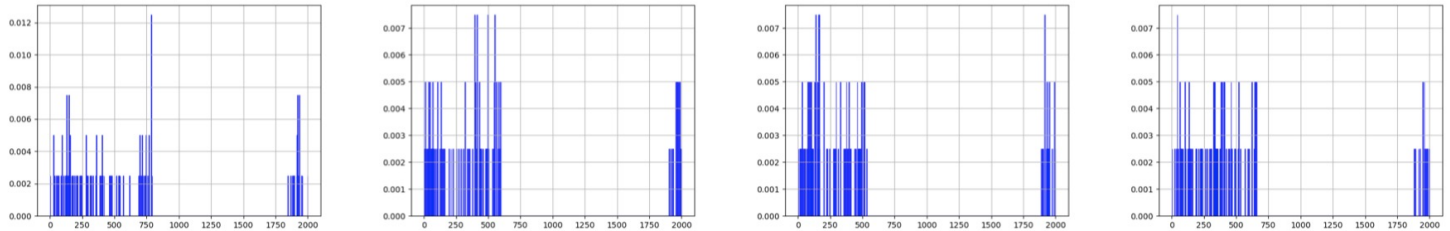
Remark:
Same condition number for case 1 & 2
but the performance is different, due to **homo & heterogeneity**

Homogeneity can make SGD work: Example 2 (quadratic prob)

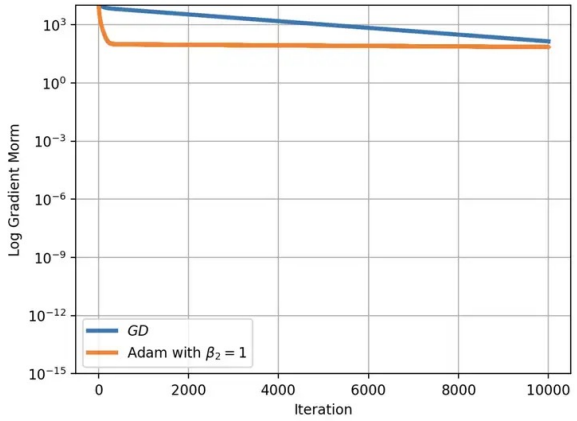
- Case study on quadratic models

$$\min_w \frac{1}{2} w^T H w, H = diag(H1, H2, H3), H \text{ is PD}, \quad H_l \in R^{3\times 3}, l = 1,2,3$$

Case 3: H with CNN-type spectra: sampled from ResNet18



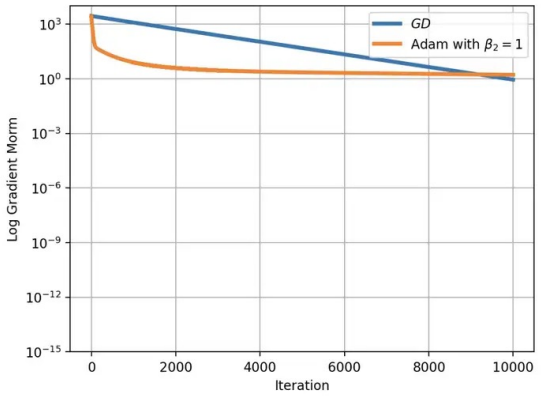
(a) Spectrum of H_1 (b) Spectrum of H_2 (c) Spectrum of H_3 (d) Spectrum of H_4



GD is similar as Adam

Case 4: H with simplified **homogeneous** spectra

Eigenvalues of H_l : { 1, 99, 1998 } , { 2, 100, 1999 } , { 3, 101, 2000 }



GD is similar as Adam

Remark:
Same condition number for case 1 & 3
All eigenvalues are the same for case 2 & 4
but the performance is different, due to **homo & heterogeneity**

Theoretical results

A well-known result for GD: Consider $\min_w f(w) = \frac{1}{2} w^T H w$, where H is PD, then there exits a H and w^0 :

$$f\left(w_{GD}^{t+1}\right) - f^* \geq \left(1 - \frac{2}{\kappa}\right)\left(f(w_{GD}^t) - f^*\right)$$

where $\boldsymbol{\kappa}$ is the condition number of H

Theorem 1(Adam): Consider $\min_w f(w) = \frac{1}{2} w^T H w$, where H a block-diagonal PD matrix with L blocks, then:

$$f\left(w_{Adam}^{t+1}\right) - f^* \leq \max_{l \in [L]} \left(1 - \frac{1}{\kappa_{Adam,l}}\right)\left(f\left(w_{Adam}^{t+1}\right) - f^*\right)$$

where $\kappa_{Adam,l} = r\,\boldsymbol{\kappa_l}$, $\boldsymbol{\kappa_l}$ is the condition number of H_l; r is a constant related to initialization w^0

Comparing Two Results

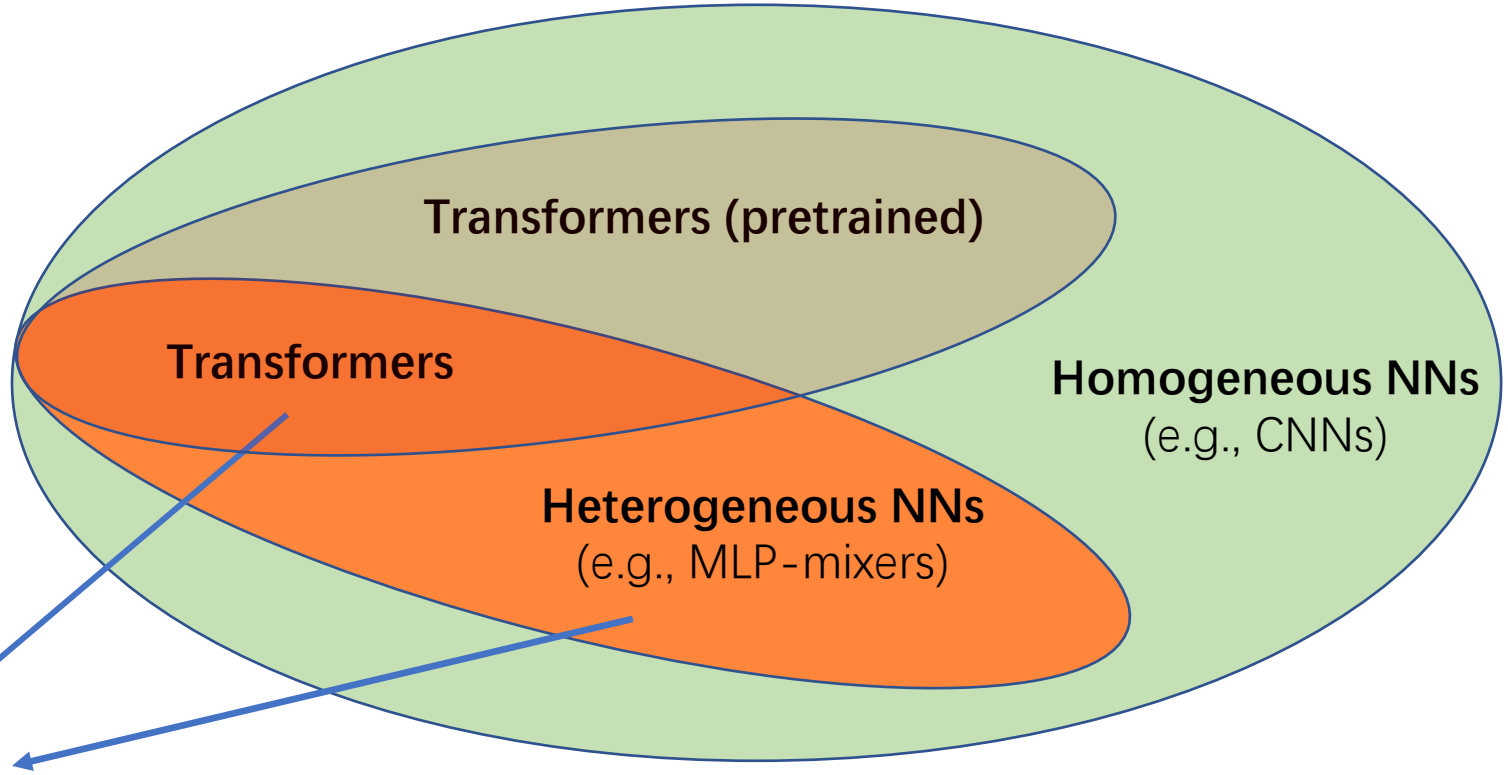
- **Compare GD vs Adam:**
  -Adam is faster than GD when $r \, \boldsymbol{max}_l \, \boldsymbol{\kappa}_l \leq \boldsymbol{\kappa}$,

Happens in the heterogeneous quadratic examples 1 & 2

Quantity of Adam ~20x smaller, and Adam is also 20x faster

- **This provides a partial theoretical explanation why Adam works better than SGD on Hetergeneous case**

Summarize in one figure



SGD < Adam here!

Why is SGD slow?
-- SGD assigns **one lr for all** parameter blocks
-- Cannot handle heterogeneity across blocks

Why Adam?
--Each block needs (at least) one customized lr
-- could be provided by v

Try our code

Github repo for Hessian spectrum calculation

This talk also motivate the design of our new method **Adam-mini (after understand spectrum, need to think for one more step)**

-- we remove 99% Adam's v, reduce **50% memory** of Adam

-- same performance on **13B models**

Thanks to all the collaborators!