Train Faster, Perform Better: Modular Adaptive Training in Over-Parameterized Models

Yubin Shi Yixuan Chen Mingzhi Dong Xiaochen Yang
Dongsheng Li Yujiang Wang Robert Dick Qin Lv
Yingying Zhao Fan Yang Tun Lu Ning Gu Li Shang

ybshi21@m.fudan.edu.cn

NeurIPS '23

Motivation

Modern deep models represented by the Transformer-based large language models (LLM) has two important properties:

- **Over-parameterization:** Over-parameterized models are potential in optimization and generalization, but expensive to train.
- **Modular Structure:** Deep models are structured by various functioning modules connected layer-wisely, e.g., attention heads in Transformer.

This module-based architecture inevitably leads to a question: *Has the computational resource been spent efficiently and effectively across modules during model optimization?*

Motivation

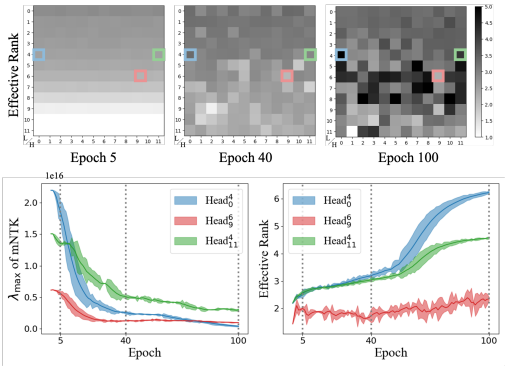

Figure 1: Characteristics of training BERT on WikiText-2, which demonstrates the joint variation of effective rank and $\lambda_{\max}$ across the attention heads.

$\lambda_{\max}$ of mNTK

The degree of consistency in the gradient direction for learning the most common features in data.

Effective Rank

The effective dimensionality of a feature matrix like an attention matrix.

Such asynchronous modularly and temporally training dynamics result in inefficient resource utilization by existing training schemes.

Motivation

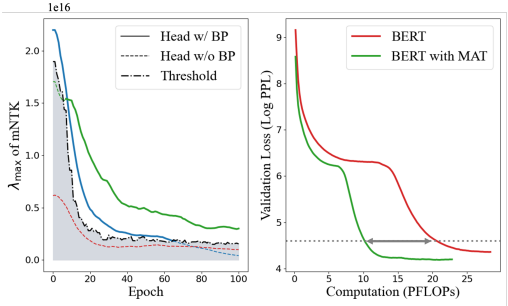

Figure 2: The idea of MAT, which governs the heads training by a dynamic threshold. Using MAT speeds up convergence and achieves lower validation loss.

We design **Modular Adaptive Training** (MAT):

- MAT only back-propagates the parameters of those modules whose $\lambda_{\max}$ is larger than the threshold.
- This forces the network to concentrate on learning the common features and ignore the inconsistent ones, preventing it from fitting superfluous features or noises.

Analysis of Modular Neural Tangent Kernel

Modular Neural Tangent Kernel (mNTK)

We define mNTK as a matrix by $\mathbf{\Theta}^l(\mathcal{X}, \mathcal{X}) = J_{\boldsymbol{\theta}^l}(\mathcal{X}) J_{\boldsymbol{\theta}^l}(\mathcal{X})^\top$, where $J_{\boldsymbol{\theta}^l} = \nabla_{\boldsymbol{\theta}^l} f(\mathcal{X}; \boldsymbol{\theta}^l)$ denotes the Jacobian of the function f at the points $\mathcal{X}$ with respect to the l^{th} module's parameters $\boldsymbol{\theta}^l$.

$\mathbf{\Theta}^l$ is a positive semi-definite real symmetric matrix and can be eigen-decomposed with nk non-negative eigenvalues $\boldsymbol{\lambda}(\mathbf{\Theta}^l) = \{\lambda_1^l, \lambda_2^l, ..., \lambda_{nk}^l\}$.

Since each module has its own mNTK, the integral NTK of a model can be computed as the sum of mNTKs of each module:

$$\mathbf{\Theta}(\mathcal{X}, \mathcal{X}) \overset{(i)}{=} \sum_{p=1}^{m} J_{\boldsymbol{\theta}_p}(\mathcal{X}) J_{\boldsymbol{\theta}_p}(\mathcal{X})^\top \overset{(ii)}{=} \sum_{l=1}^{L} \sum_{\boldsymbol{\theta}_p \in \boldsymbol{\theta}^l} J_{\boldsymbol{\theta}^l}(\mathcal{X}) J_{\boldsymbol{\theta}^l}(\mathcal{X})^\top \overset{(iii)}{=} \sum_{l=1}^{L} \mathbf{\Theta}^l(\mathcal{X}, \mathcal{X}) \tag{1}$$

Empirical Analysis of mNTK

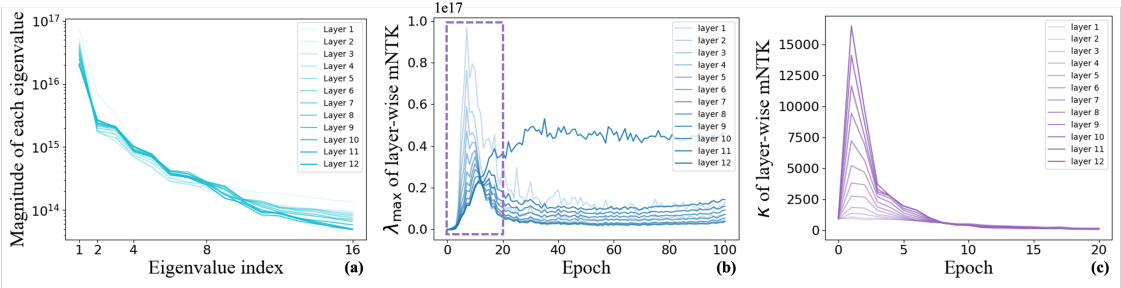

Figure 3: Training dynamics characterized by layer-wise mNTK of BERT trained on WikiText-2.

- The principal eigenvalue of mNTK dominates the eigenspectrum of mNTK.
- The variation of principal eigenvalues of mNTKs is highly asynchronous across modules.

Empirical Analysis of mNTK

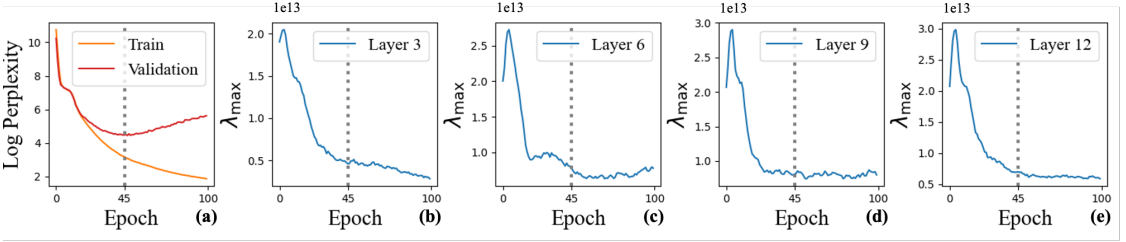

Figure 4: Training dynamics in the overfitting case of 4-layer BERT trained by 64 token MLM task.

- The temporal variation of the principal eigenvalue of mNTK indicates the generalization ability.

Analysis of Trainability and Generalization

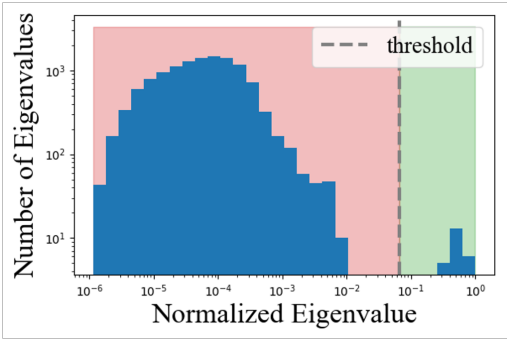

Figure 5: The normalized eigen-spectrum distribution exhibits two distinct regions, termed information space and nuisance space.

- **Trainability**:

$$\Delta\mathcal{L} \approx \sum_{l=1}^{L} \lambda_{\max}^{l} \left({\mathbf{u}_1^l}^{\top} \mathcal{Y} \right)^2 . \tag{2}$$

The loss decreases faster along the eigenspaces that correspond to larger eigenvalues.

- **Generalization**:

$$\mathcal{R} \propto \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_F \tag{3}$$

Those modules whose $\lambda_{\max}$ below the threshold, are prone to learn superfluous features, which significantly increase the Rademacher complexity, and deteriorate generalization ability.

Modular Adaptive Training

During back propagation, MAT quantifies module parameters in terms of **modular policy** and **temporal policy** such that only subsets of modules are updated.

- **Modular Policy**:

$$\lambda_\alpha = \widetilde{\lambda}_{\min} + (\widetilde{\lambda}_{\max} - \widetilde{\lambda}_{\min}) \times \alpha. \tag{4}$$

$$\mathcal{I}_t = \{\boldsymbol{\theta}^l | \lambda_{\max}(\boldsymbol{\Theta}_t^l) \geq \lambda_\alpha\}, \mathcal{N}_t = \{\boldsymbol{\theta}^l | \lambda_{\max}(\boldsymbol{\Theta}_t^l) < \lambda_\alpha\}. \tag{5}$$

- **Temporal Policy**:

$$\Delta_t^l = |\lambda_{\max}(\boldsymbol{\Theta}_t^l) - \lambda_{\max}(\boldsymbol{\Theta}_0^l)| \tag{6}$$

$$\frac{|\Delta_t^l - \Delta_{t-1}^l|}{\Delta_1^l} < \beta. \tag{7}$$

Experiments

- Attention heads in BERT (Transformer)

Table 1: Results of BERT on WikiText-2. FLOPs is measured per GPU without embedding. Computation refers to the FLOPs model used until achieving best test loss. Best results are in boldface.

Method	Validation Loss (Log PPL) @ 10 PFLOPs	Validation Loss (Log PPL) @ 15 PFLOPs	Validation Loss (Log PPL) @ 20 PFLOPs	Test Loss (Log PPL) @ Convergence	Computation (PFLOPs) @ Convergence
BERT	5.39 ± 0.15	4.75 ± 0.06	4.48 ± 0.03	4.41 ± 0.05	28.70
BERT-Rand	5.65 ± 0.18	5.11 ± 0.16	4.96 ± 0.11	4.82 ± 0.06	21.53
Multirate	4.93 ± 0.10	4.57 ± 0.03	4.52 ± 0.02	4.48 ± 0.03	19.46
MAT	**4.46 ± 0.04**	**4.41 ± 0.02**	**4.35 ± 0.02**	**4.27 ± 0.03**	**16.50**

Experiments

- Experts in Switch-Transformer (Mixture-of-Experts)

Table 2: Results of Switch-Transformer on WikiText-103. FLOPs is measured per GPU without embedding. Best results are in boldface.

Method	Validation Loss (Log PPL) @ 200 PFLOPs	@ 400 PFLOPs	@ 600 PFLOPs	Test Loss (Log PPL) @ Convergence	Computation (PFLOPs) @ Convergence
Switch	3.16 ± 0.12	2.31 ± 0.05	1.93 ± 0.03	1.74 ± 0.02	1155.2
Switch-Rand	2.92 ± 0.15	2.15 ± 0.08	2.01 ± 0.07	1.93 ± 0.05	837.5
Multirate	2.67 ± 0.09	2.02 ± 0.08	1.83 ± 0.05	1.77 ± 0.04	740.4
MAT	**2.34 ± 0.06**	**1.92 ± 0.03**	**1.69 ± 0.02**	**1.68 ± 0.02**	**614.8**

Experiments

- Filters in VGG16 (Convolutional Network)

Table 3: Results of VGG16 on CIFAR-10. Best results are in boldface.

Method	Computation until Train Acc = n (PFLOPs)		Test Accuracy (Top-1, %)	Computation (PFLOPs)
	@ n = 95%	@ n = 99%	@ Convergence	
VGG16	8.85	12.06	93.77 ± 0.08	17.14
VGG16-Rand	9.96	13.35	92.71 ± 0.08	18.84
Multirate	7.19	9.75	93.43 ± 0.14	13.35
MAT	**5.31**	**7.21**	**93.86 ± 0.05**	**9.03**

Thanks

ybshi21@m.fudan.edu.cn

NeurIPS ’23