Efficient Low-Dimensional Compression for Deep Overparameterized Learning and Fine-Tuning

Can Yaras, Soo Min Kwon, Peng Wang, **Laura Balzano**, Qing Qu

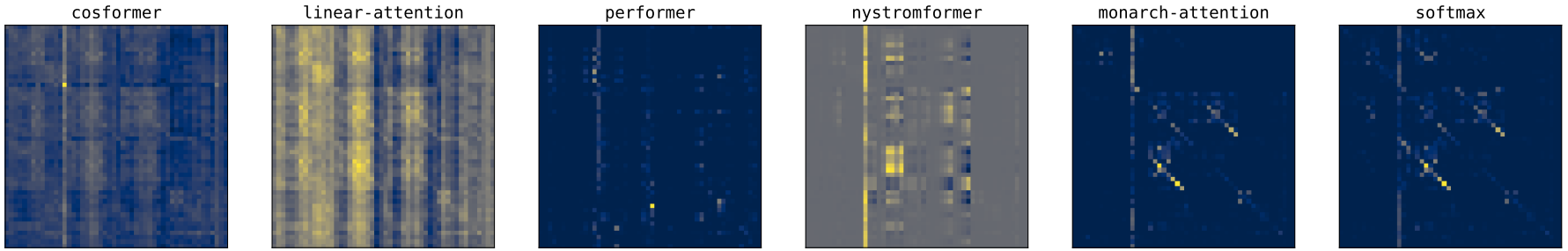
Electrical Engineering and Computer Science, University of Michigan

October 8, 2025

Institute for Mathematical and Statistical Innovation
Data Assimilation and Inverse Problems for Digital Twins

My Group's Work / Inverse Problems

- Optimization with structure and regularization
  - Low-rank structure, union of subspaces, low-dimensional varieties, sparsity structure, total variation, Monarch, and combinations



My Group's Work / Inverse Problems

- Optimization with structure and regularization
  - Low-rank structure, union of subspaces, low-dimensional varieties, sparsity structure, total variation, Monarch, and combinations
- Nonconvex optimization theory and practice
  - Algorithmic convergence behavior, when we need to initialize well and how, what induces implicit regularization and how to use it wisely

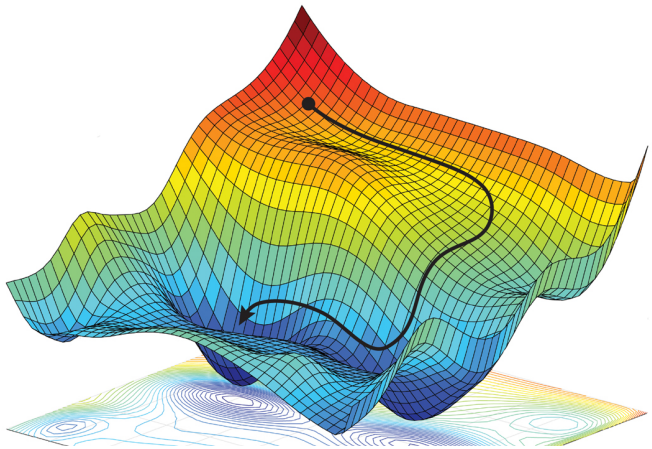
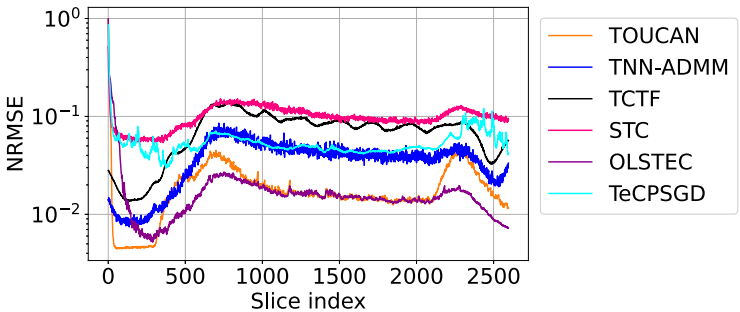


Figure courtesy Science Magazine

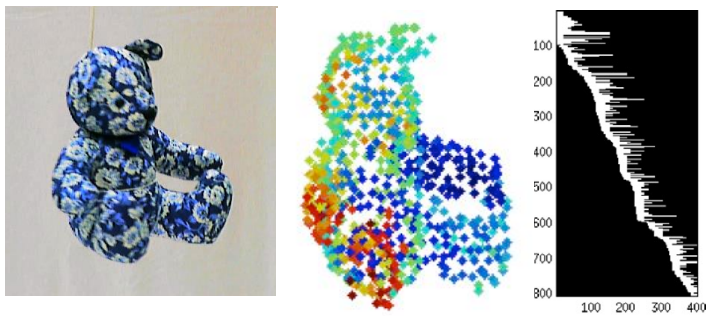
My Group's Work / Inverse Problems

- Optimization with structure and regularization
  - Low-rank structure, union of subspaces, low-dimensional varieties, sparsity structure, total variation, Monarch, and combinations
- Nonconvex optimization theory and practice
  - Algorithmic convergence behavior, when we need to initialize well and how, what induces implicit regularization and how to use it wisely
- Matrix Completion (recommender systems, environmental and chemical flow missing data imputation, hyperspectral video completion, rigid structure from motion, internet topology hopcount matrix estimation)



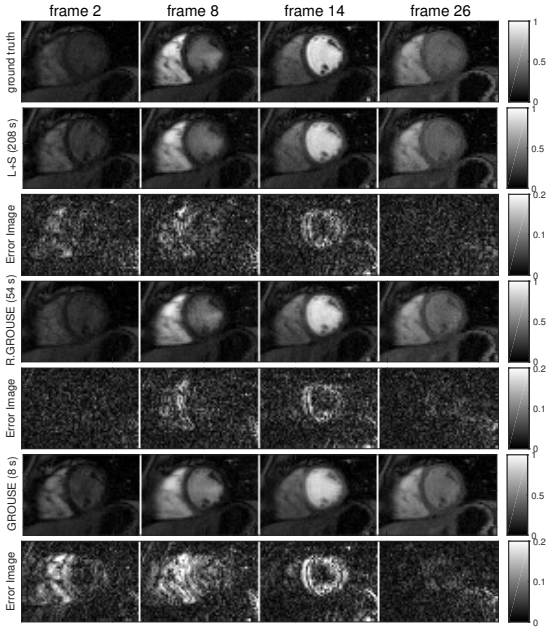
Algorithm	Time (s)
TOUCAN	30.81
TeCPSGD	137.31
STC	363.54
OLSTEC	552.53
TCTF	725.55
TNN-ADMM	879.78

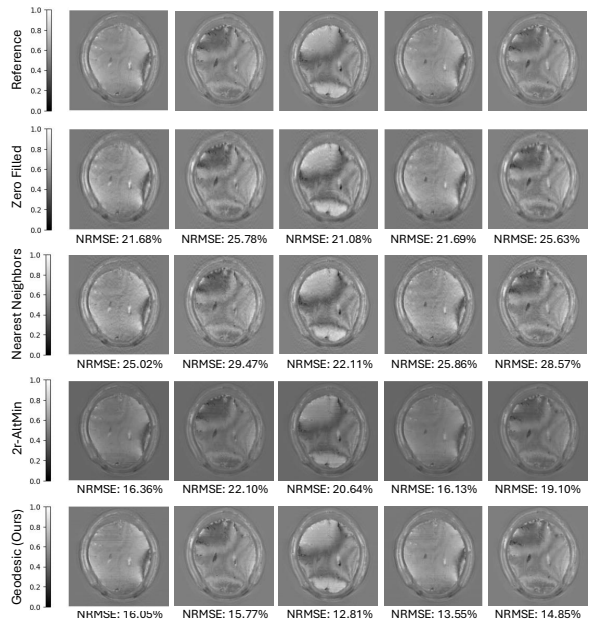
NRMSE of each recovered time slice for Toluene gas dataset from 25% samples, and wall time for the corresponding algorithms.



My Group's Work / Inverse Problems

- Optimization with structure and regularization
  - Low-rank structure, union of subspaces, low-dimensional varieties, sparsity structure, total variation, Monarch, and combinations
- Nonconvex optimization theory and practice
  - Algorithmic convergence behavior, when we need to initialize well and how, what induces implicit regularization and how to use it wisely
- Matrix Completion
- Dynamic Inverse Imaging (cardiac perfusion DMRI; dynamic fMRI OSSI)





My Group's Work / Inverse Problems

- Optimization with structure and regularization
  - Low-rank structure, union of subspaces, low-dimensional varieties, sparsity structure, total variation, Monarch, and combinations
- Nonconvex optimization theory and practice
  - Algorithmic convergence behavior, when we need to initialize well and how, what induces implicit regularization and how to use it wisely
- Matrix Completion
- Dynamic Inverse Imaging
- Learning with heterogeneous/heteroscedastic data, reduced order modeling and control, switched system identification, ...
- Today: Leveraging low-rank structure in deep learning

Most successful modern deep learning applications

text generation...

image generation...

... and image recognition.

How is modern ML achieved?

- **Training data** $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N \subset \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ with

$$\boldsymbol{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \dots \ \boldsymbol{x}_N] \in \mathbb{R}^{d_x \times N}, \quad \boldsymbol{Y} = [\boldsymbol{y}_1 \ \boldsymbol{y}_2 \ \dots \ \boldsymbol{y}_N] \in \mathbb{R}^{d_y \times N}$$

[1] This notation doesn't handle transformers or dimension-changing nonlinearities like pooling, but it's close enough for our purposes.

How is modern ML achieved?

- **Training data** $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N \subset \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ with

$$\boldsymbol{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \dots \ \boldsymbol{x}_N] \in \mathbb{R}^{d_x \times N}, \quad \boldsymbol{Y} = [\boldsymbol{y}_1 \ \boldsymbol{y}_2 \ \dots \ \boldsymbol{y}_N] \in \mathbb{R}^{d_y \times N}$$

and we wish to learn a prediction function

$$f_{\boldsymbol{\Theta}}(\boldsymbol{x}) \ := \ \sigma_L \left(\boldsymbol{W}_L \left(\sigma_{L-1} (\boldsymbol{W}_{L-1} \cdots \sigma_1 (\boldsymbol{W}_1 \boldsymbol{x})) \right) \right),$$

where $\boldsymbol{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$, $\boldsymbol{\Theta} = \{\boldsymbol{W}_l\}_{l=1}^L$, and $\{\sigma_l\}_{l=1}^L$ are known nonlinearities applied to each layer, usually related to ReLU for intermediate layers and softmax at the end[1].

[1] This notation doesn't handle transformers or dimension-changing nonlinearities like pooling, but it's close enough for our purposes.

How is modern ML achieved?

- **Training data** $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N \subset \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ with

$$\boldsymbol{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \dots \ \boldsymbol{x}_N] \in \mathbb{R}^{d_x \times N}, \quad \boldsymbol{Y} = [\boldsymbol{y}_1 \ \boldsymbol{y}_2 \ \dots \ \boldsymbol{y}_N] \in \mathbb{R}^{d_y \times N}$$

and we wish to learn a prediction function

$$f_{\boldsymbol{\Theta}}(\boldsymbol{x}) \ := \ \sigma_L \left(\boldsymbol{W}_L \left(\sigma_{L-1} (\boldsymbol{W}_{L-1} \cdots \sigma_1 (\boldsymbol{W}_1 \boldsymbol{x})) \right) \right),$$

where $\boldsymbol{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$, $\boldsymbol{\Theta} = \{\boldsymbol{W}_l\}_{l=1}^L$, and $\{\sigma_l\}_{l=1}^L$ are known nonlinearities applied to each layer, usually related to ReLU for intermediate layers and softmax at the end[1].

- **Optimize the loss function**:

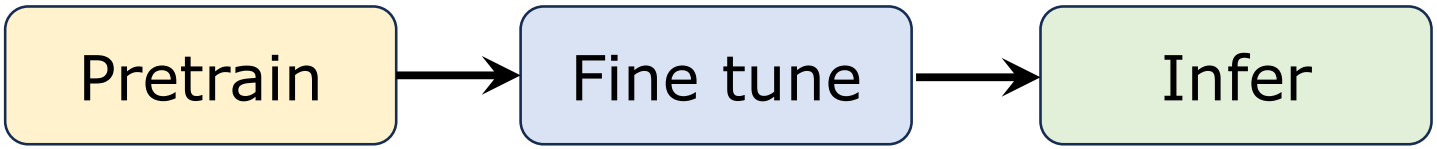
$$\min_{\boldsymbol{\Theta}} \ \ell(\boldsymbol{\Theta}) = \frac{1}{N} \sum_{i=1}^N \ell_i \left(f_{\boldsymbol{\Theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i \right)$$

Loss could be cross-entropy, MSE, or other error metrics.

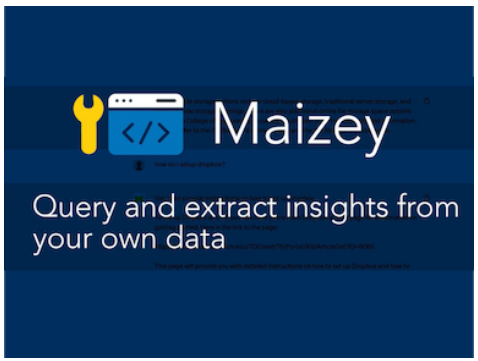
[1] This notation doesn't handle transformers or dimension-changing nonlinearities like pooling, but it's close enough for our purposes.

Pretrain, Fine tune, and Infer: The Modern ML Pipeline

- Estimates for param count and training costs:
  - GPT3 (175B params) $5M.
  - GPT4 (1.7T params) $50M.
  - GPT5 (2.0T params) $500M.



Making (training), fine-tuning, and inference more efficient has been one of the most important problems for widespread accessibility of generative AI.



Observation: deep layers have low-rank structure

Use gradient descent to optimize $\boldsymbol{\Theta} = \{\boldsymbol{W}_l\}_{l=1}^{L}$ and look at the weights over iterations t.

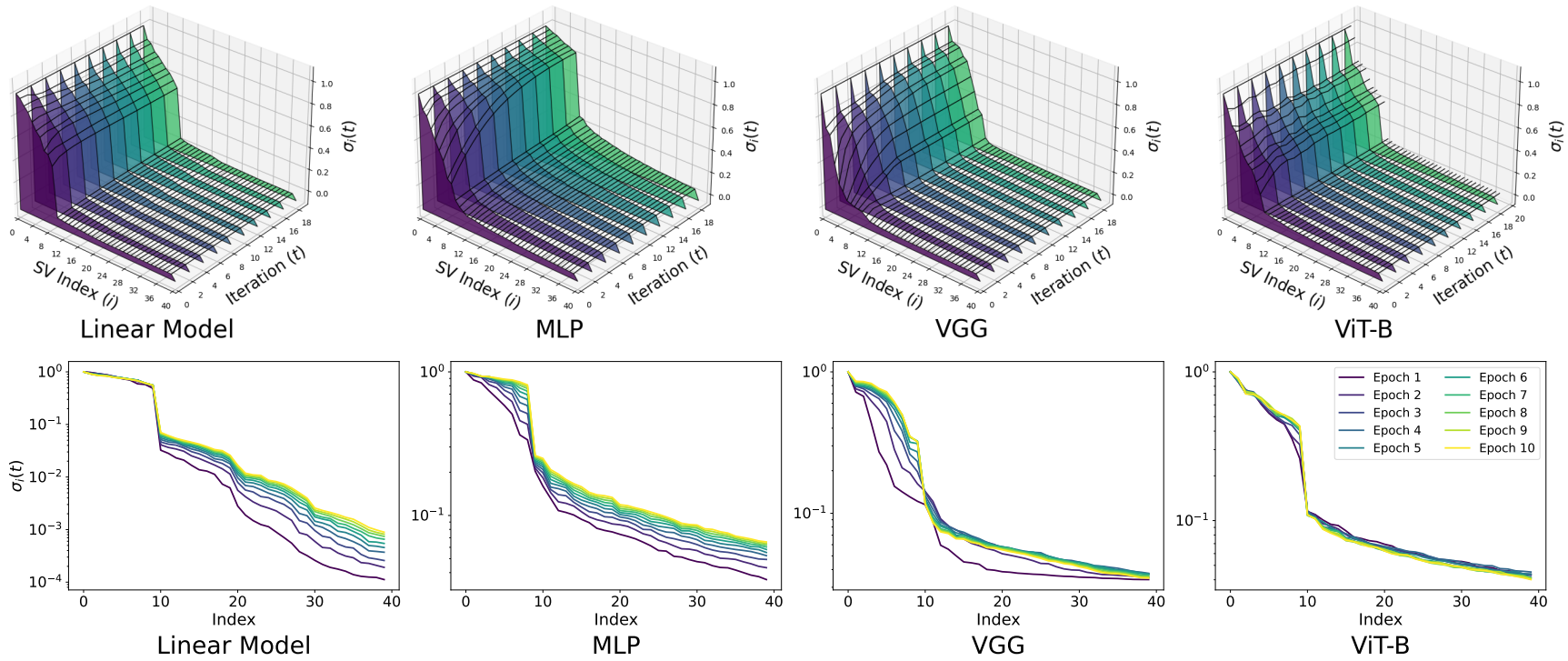


Figure: Singular values of $\Delta \boldsymbol{W}_{L-1}$ over iterations, where $\boldsymbol{W}_l(t) = \boldsymbol{W}_l(0) + \Delta \boldsymbol{W}_l$ for iteration t.

Deep linear network (DLN) with MSE loss, Multi-layer perception (MLP) with CE loss, trained on MNIST, VGG [Simonyan and Zisserman, 2015] and ViT-B [Dosovitskiy et al., 2020] trained with CE loss on CIFAR-10.

Observation: Success of Low-Rank Adaptation

- LoRA [Hu et al., 2021] is a parameter-efficient adaptation technique for large pretrained models that uses low-rank updates to pre-trained weights.
- Fine-tune dense weight matrix $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ as

$$\boldsymbol{W} = \overline{\boldsymbol{W}} + \boldsymbol{W}_2 \boldsymbol{W}_1$$

  where $\overline{\boldsymbol{W}}$ is frozen pretrained weight matrix and $\boldsymbol{W}_1, \boldsymbol{W}_2$ are trainable factors with $\boldsymbol{W}_1 \in \mathbb{R}^{r \times d}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d \times r}$, and $r \ll d$.
- LoRA achieves high accuracy with very low memory costs on many natural language and vision tasks.
- 19319 citations and counting

Goals: Understand and Leverage Low-Rank Structure

Goals:

- Understand how low-rank structure arises in deep networks.
- Leverage low-rank structure for more efficient deep learning optimization.
- Use it to improve LoRA.

Goals: Understand and Leverage Low-Rank Structure

Goals:

- Understand how low-rank structure arises in deep networks.

There is now a rich literature on how low-rank structure arises in *overparameterized deep linear networks*:

$$f_{\boldsymbol{\Theta}}(\boldsymbol{x}) \;:=\; \boldsymbol{W}_L \boldsymbol{W}_{L-1} \cdots \boldsymbol{W}_1 \boldsymbol{x}$$

- There is an implicit low-rank bias that gets stronger with the depth of the network.
- The gradients, weights, and activations all "share" rank.

Contributions

Theoretical: *For deep overparameterized matrix factorization,*

- We show every iteration of gradient descent occurs within an invariant low-dimensional subspace of the weights
- Weights are highly compressible – we construct near-identical factorizations at a fraction of the parameter count

Contributions

Theoretical: *For deep overparameterized matrix factorization,*

- We show every iteration of gradient descent occurs within an invariant low-dimensional subspace of the weights
- Weights are highly compressible – we construct near-identical factorizations at a fraction of the parameter count

Practical

- We validate our compression methodology for accelerating deep matrix completion
- We propose **Deep LoRA**: an efficient and robust method for fine-tuning language models based on compression

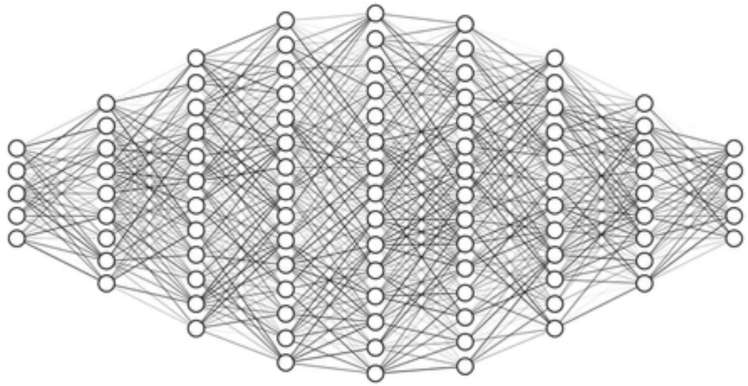
Outline

- Introduction
- Theory for Overparameterized Deep Linear Networks
- Matrix Completion
- Deep LoRA

Modern Machine Learning = Overparameterization

Success of overparameterized models attributed to

1. implicit algorithmic regularization
2. improved optimization landscape



But there are also costs:

Consider an *overparameterized* LoRA to bring these benefits to low-rank fine tuning:

$$\boldsymbol{W} = \overline{\boldsymbol{W}} + \boldsymbol{W}_3\boldsymbol{W}_2\boldsymbol{W}_1$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3 \in \mathbb{R}^{d\times d}$. This L layer factorization has

- Ld^2 parameters
- $O(Ld^3)$ gradient complexity

Much larger than $2rd$ parameters of LoRA! We are back to full training costs. **Can we get the benefits without the costs?**

Set-Up

Data. Target matrix $\mathbf{\Phi} \in \mathbb{R}^{d \times d}$ with $\text{rank}(\mathbf{\Phi}) = r^*$

Set-Up

Data. Target matrix $\mathbf{\Phi} \in \mathbb{R}^{d \times d}$ with $\text{rank}(\mathbf{\Phi}) = r^*$

Model. Deep matrix factorization

$$f(\mathbf{\Theta}) = \boldsymbol{W}_L \boldsymbol{W}_{L-1} \cdots \boldsymbol{W}_2 \boldsymbol{W}_1 = \boldsymbol{W}_{L:1}$$

with parameters $\mathbf{\Theta} = (\boldsymbol{W}_l)_{l=1}^L \subset \mathbb{R}^{d \times d}$

Set-Up

Data. Target matrix $\mathbf{\Phi} \in \mathbb{R}^{d \times d}$ with $\text{rank}(\mathbf{\Phi}) = r^*$

Model. Deep matrix factorization

$$f(\mathbf{\Theta}) = \boldsymbol{W}_L \boldsymbol{W}_{L-1} \cdots \boldsymbol{W}_2 \boldsymbol{W}_1 = \boldsymbol{W}_{L:1}$$

with parameters $\mathbf{\Theta} = (\boldsymbol{W}_l)_{l=1}^L \subset \mathbb{R}^{d \times d}$

Objective. Minimize $\ell(\mathbf{\Theta}) = \frac{1}{2}\|f(\mathbf{\Theta}) - \mathbf{\Phi}\|_F^2$ via Gradient Descent:

$$\boldsymbol{W}_l(t+1) = (1 - \eta\lambda)\boldsymbol{W}_l(t) - \eta \nabla_{\boldsymbol{W}_l} \ell(\mathbf{\Theta}(t)),\ l \in [L]$$

from *arbitrary* ϵ_l-scaled orthogonal initialization, i.e.,

$$\boldsymbol{W}_l(0)\boldsymbol{W}_l(0)^\top = \boldsymbol{W}_l(0)^\top \boldsymbol{W}_l(0) = \epsilon_l^2 \boldsymbol{I}_d,\ l \in [L]$$

Main Result

Theorem

Suppose $d - 2r^* > 0$. Then there exist orthogonal $(\boldsymbol{U}_l)_{l=1}^L \subset \mathbb{R}^{d\times d}$ and $(\boldsymbol{V}_l)_{l=1}^L \subset \mathbb{R}^{d\times d}$ with $\boldsymbol{V}_{l+1} = \boldsymbol{U}_l$ such that

$$\boldsymbol{W}_l(t) = \boldsymbol{U}_l \begin{bmatrix} \widetilde{\boldsymbol{W}}_l(t) & \boldsymbol{0} \\ \boldsymbol{0} & \rho_l(t)\boldsymbol{I}_{d-2r^*} \end{bmatrix} \boldsymbol{V}_l^\top, \ l \in [L]$$

for all $t \geq 0$ where $\widetilde{\boldsymbol{W}}_l(t) \in \mathbb{R}^{2r^*\times 2r^*}$ and

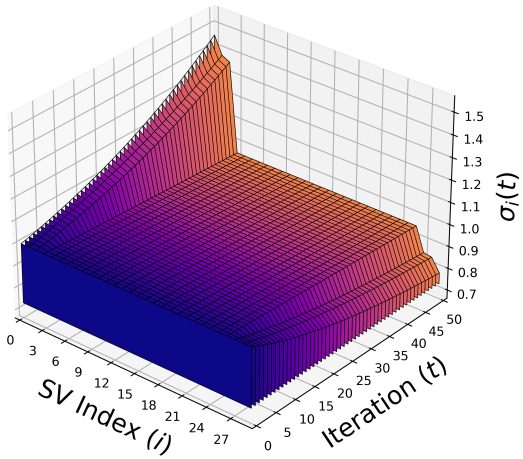
$$\rho_l(t) = \rho_l(t-1) \cdot (1 - \eta\lambda - \eta \cdot \prod_{k\neq l} \rho_k^2(t-1)) \tag{1}$$

for all $l \in [L]$ and $t \geq 1$ with $\rho_l(0) = \epsilon_l$. Note $\rho_l(t) = \rho(t)$ if they are all initialized at ϵ.

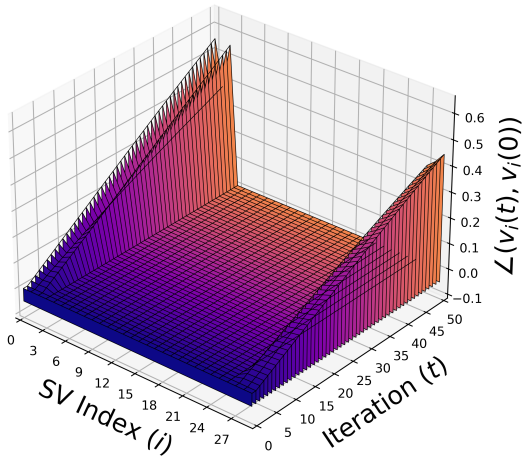
Proof Sketch

Illustration of the theory

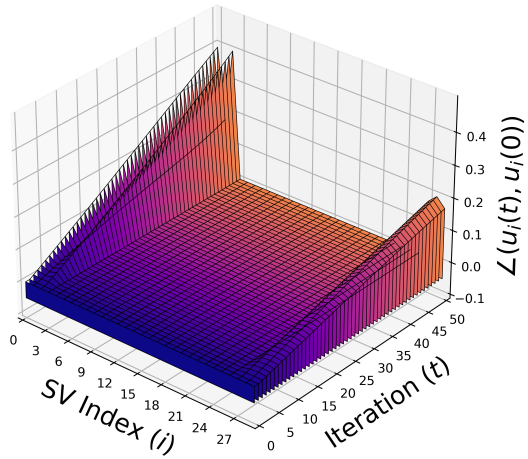
- Learning only occurs within a $2r^*$-dimensional *invariant* subspace of left/right spaces of weights (ρ is independent of $\mathbf{\Phi}$)
- The factors $\boldsymbol{U}_l, \boldsymbol{V}_l$ depend only on $\mathbf{\Phi}$ and $\mathbf{\Theta}(0) \rightarrow$ more generally the initial gradient $\nabla \ell(\mathbf{\Theta}(0))$



Singular Values



Right Singular Vectors



Left Singular Vectors

Insight at initialization

Let $\boldsymbol{Q} = \boldsymbol{W}_L \cdots \boldsymbol{W}_{l+1}$ and $\boldsymbol{R} = \boldsymbol{W}_{l-1} \cdots \boldsymbol{W}_1$, so from ϵ_l-scaled orthogonal initialization, we have the gradient

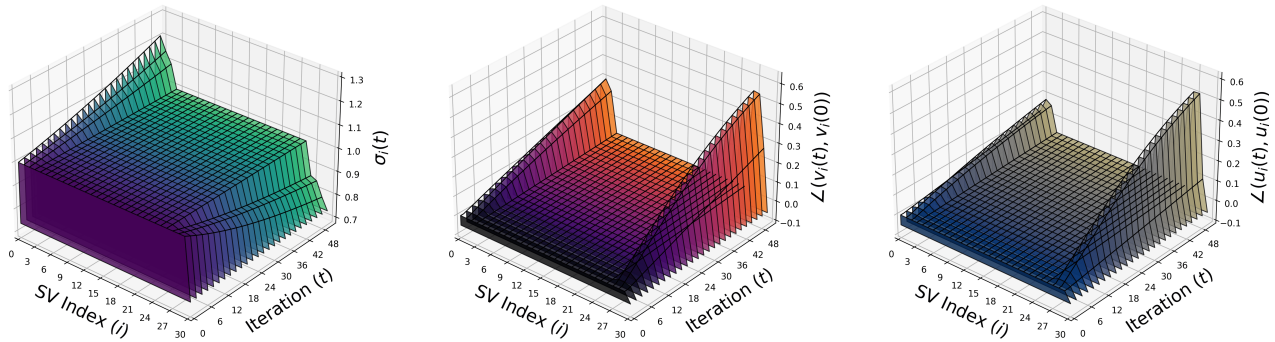
$$\nabla_{\boldsymbol{W}_l} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{Q}^\top (\boldsymbol{Q}\boldsymbol{W}_l\boldsymbol{R} - \boldsymbol{\Phi})\boldsymbol{R}^\top = \left(\prod_{k \neq l} \epsilon_k^2\right) \boldsymbol{W}_l - \underbrace{\boldsymbol{Q}^\top \boldsymbol{\Phi} \boldsymbol{R}^\top}_{\boldsymbol{A}}, \tag{2}$$

which gives the update

$$\boldsymbol{W}_l^+ = (1 - \eta\lambda)\boldsymbol{W}_l - \left(\prod_{k \neq l} \epsilon_k^2\right) \boldsymbol{W}_l + \boldsymbol{A} = \left(1 - \eta\lambda - \prod_{k \neq l} \epsilon_k^2\right) \boldsymbol{W}_l + \boldsymbol{A}. \tag{3}$$

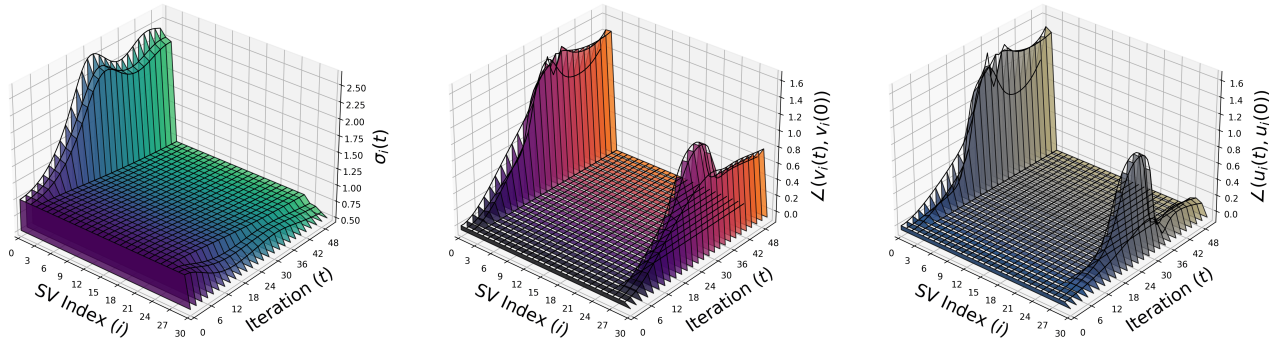
The $d - 2r$ identical singular values that satisfy (1) correspond to a construction of $d - 2r$ pairs of singular vectors $(\boldsymbol{u}, \boldsymbol{v})$ that are simultaneously annihilated on the left and right of $\boldsymbol{A}$ respectively.

The Evolution of Singular Spaces in More Generic Settings



Singular Values | Right Singular Vectors | Left Singular Vectors

Figure: **Evolution of SVD of weight matrices for $\ell(\boldsymbol{\Theta}) = \frac{1}{2}\|\boldsymbol{W}_{L:1}\boldsymbol{X} - \boldsymbol{Y}\|_F^2$ (for wide or low-rank $\boldsymbol{Y}$).**



Singular Values | Right Singular Vectors | Left Singular Vectors

Figure: **Evolution of SVD of weight matrices with momentum.**

Compressible Dynamics

From Theorem 1, we have that

$$f(\boldsymbol{\Theta}(t)) = \boldsymbol{U}_{L,1}\widetilde{\boldsymbol{W}}_{L:1}(t)\boldsymbol{V}_{1,1}^{\top} + \left(\prod_{l=1}^{L}\rho_l(t)^2\right)\cdot \boldsymbol{U}_{L,2}\boldsymbol{V}_{1,2}^{\top}$$
$$\approx \boldsymbol{U}_{L,1}\widetilde{\boldsymbol{W}}_{L:1}(t)\boldsymbol{V}_{1,1}^{\top} = f_C(\widetilde{\boldsymbol{\Theta}}(t), \boldsymbol{U}_{L,1}, \boldsymbol{V}_{1,1})$$

when ϵ_l are small (since $\rho_l \leq \epsilon_l$).

Compressible Dynamics

From Theorem 1, we have that

$$f(\boldsymbol{\Theta}(t)) = \boldsymbol{U}_{L,1}\widetilde{\boldsymbol{W}}_{L:1}(t)\boldsymbol{V}_{1,1}^\top + \left(\prod_{l=1}^{L}\rho_l(t)^2\right)\cdot \boldsymbol{U}_{L,2}\boldsymbol{V}_{1,2}^\top$$
$$\approx \boldsymbol{U}_{L,1}\widetilde{\boldsymbol{W}}_{L:1}(t)\boldsymbol{V}_{1,1}^\top = f_C(\widetilde{\boldsymbol{\Theta}}(t), \boldsymbol{U}_{L,1}, \boldsymbol{V}_{1,1})$$

when ϵ_l are small (since $\rho_l \leq \epsilon_l$).

Proposition 2

For $r \geq r^*$ such that $d - 2r > 0$, running GD on compressed weights $\widetilde{\boldsymbol{\Theta}}$ yields

$$\|f(\boldsymbol{\Theta}(t)) - f_C(\widetilde{\boldsymbol{\Theta}}(t), \boldsymbol{U}_{L,1}, \boldsymbol{V}_{1,1})\|_F^2 \leq (d-2r)\cdot\prod_{l=1}^{L}\epsilon_l^2.$$

Compressed Trajectory

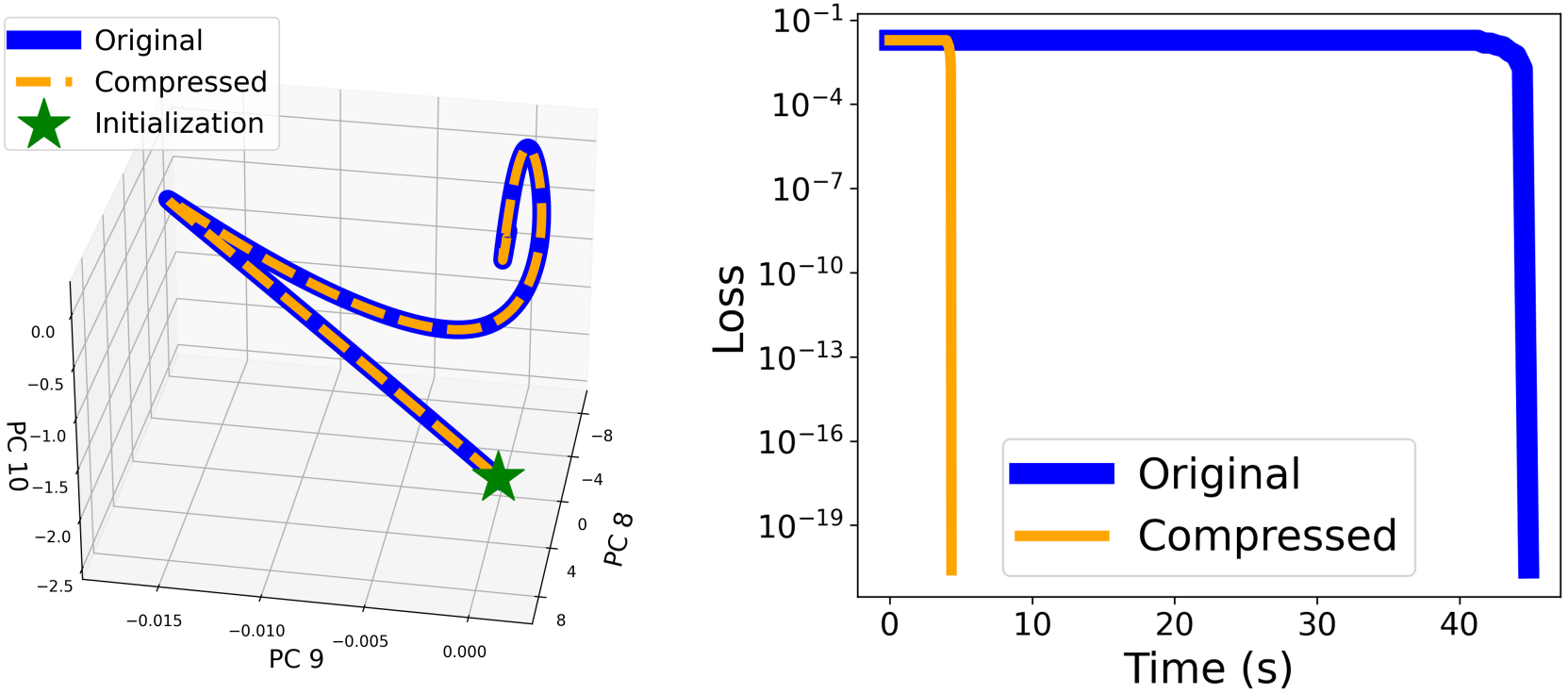
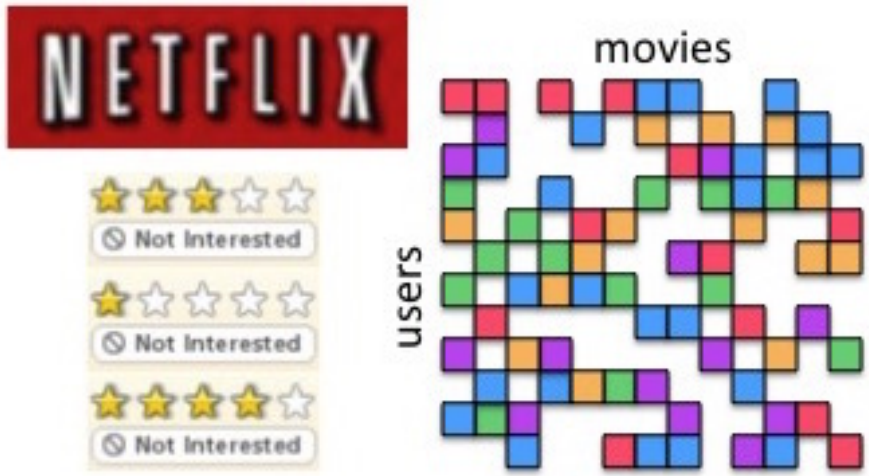


Figure: **Original vs. compressed trajectories for deep matrix factorization.**

- **Left**: Principal components of end-to-end GD trajectories.
- **Right**: Training loss vs. wall-time comparison.

Outline

- Introduction
- Theory for Overparameterized Deep Linear Networks
- Matrix Completion
- Deep LoRA



From Deep Matrix Factorization to Completion

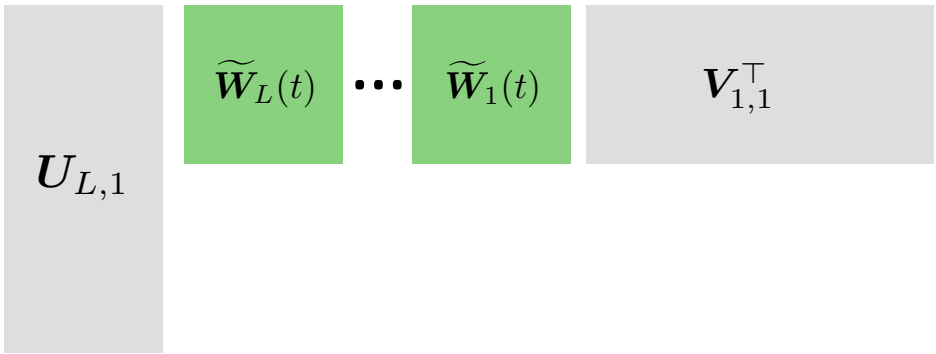
Let $\mathbf{\Omega} \subset \{1, \dots, d\} \times \{1, \dots, d\}$ be a subset of observed entries of $\mathbf{\Phi}$.

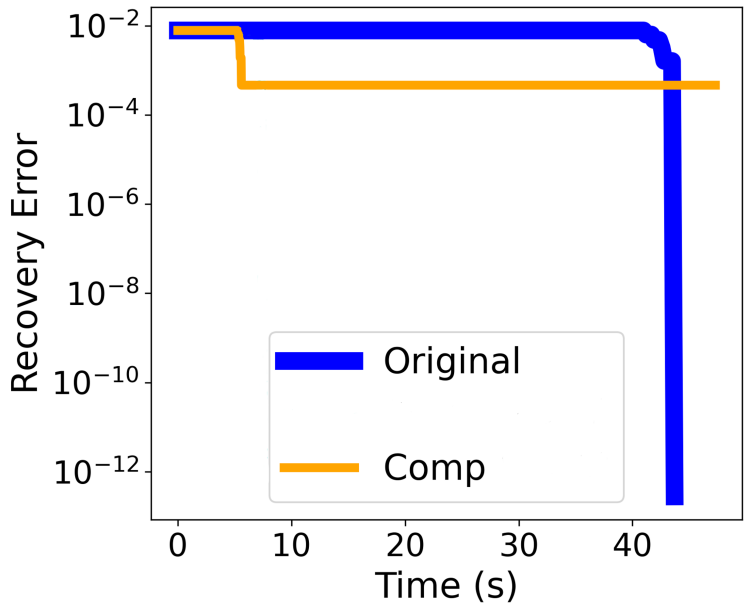
$$\min_{\mathbf{\Theta}} \ \frac{1}{2} \|\mathbf{\Omega} \odot (f(\mathbf{\Theta}) - \mathbf{\Phi})\|_F^2 \tag{4}$$

From Deep Matrix Factorization to Completion

Let $\mathbf{\Omega} \subset \{1, \dots, d\} \times \{1, \dots, d\}$ be a subset of observed entries of $\mathbf{\Phi}$.

$$\min_{\mathbf{\Theta}} \ \frac{1}{2} \|\mathbf{\Omega} \odot (f(\mathbf{\Theta}) - \mathbf{\Phi})\|_F^2 \tag{4}$$

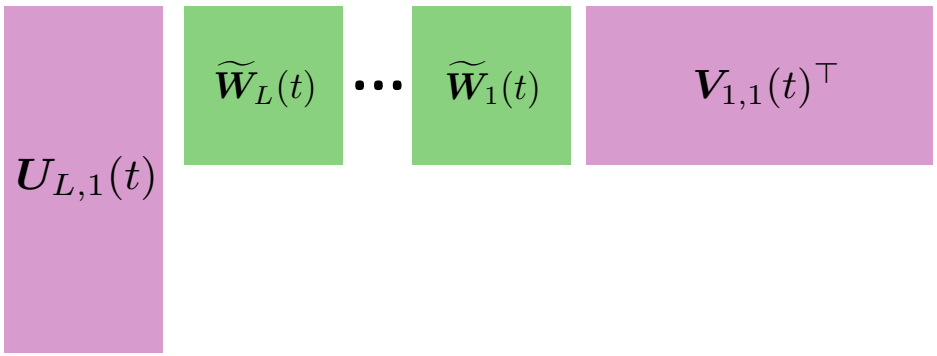




- Our theorem doesn't hold since $\mathbf{\Omega} \odot \mathbf{\Phi}$ is not low-rank for arbitrary $\mathbf{\Omega}$, and the method fails. $(d = 1000, r = 3, |\mathbf{\Omega}| = 20000)$

Compression for General Loss - Implemented Strategy

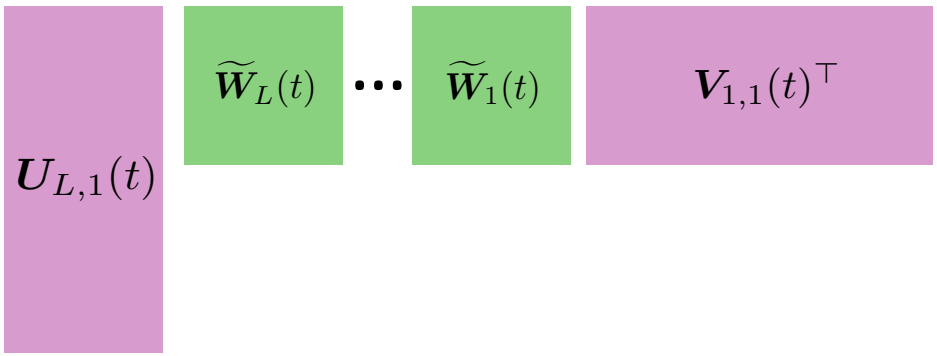
For a general loss ℓ, we make it work with a (currently, heuristic) modification:



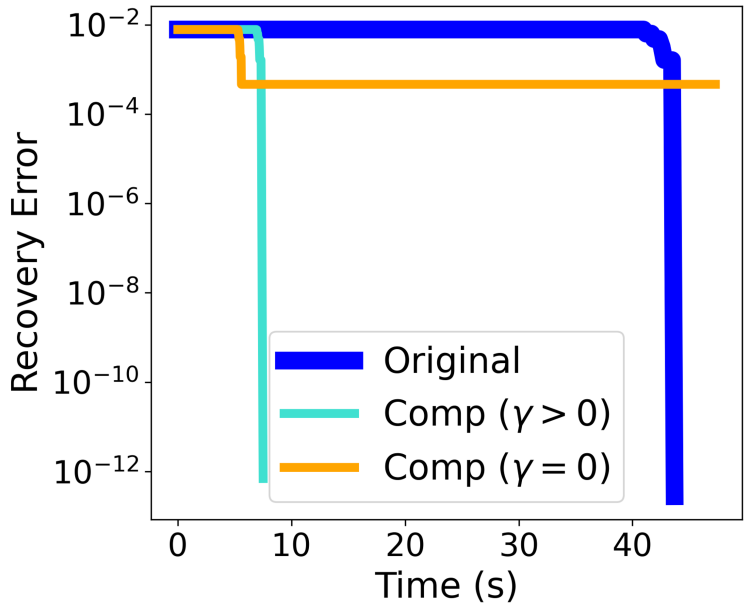
We make $\boldsymbol{U}_{L,1}, \boldsymbol{V}_{1,1} \in \mathbb{R}^{d\times 2r}$ *trainable* with a *discrepant* learning rate $\gamma\eta$.

Compression for General Loss - Implemented Strategy

For a general loss ℓ, we make it work with a (currently, heuristic) modification:



We make $\boldsymbol{U}_{L,1}, \boldsymbol{V}_{1,1} \in \mathbb{R}^{d\times 2r}$ *trainable* with a *discrepant* learning rate $\gamma\eta$.



We apply this to matrix completion as well as Deep LoRA.

Accelerating Matrix Completion

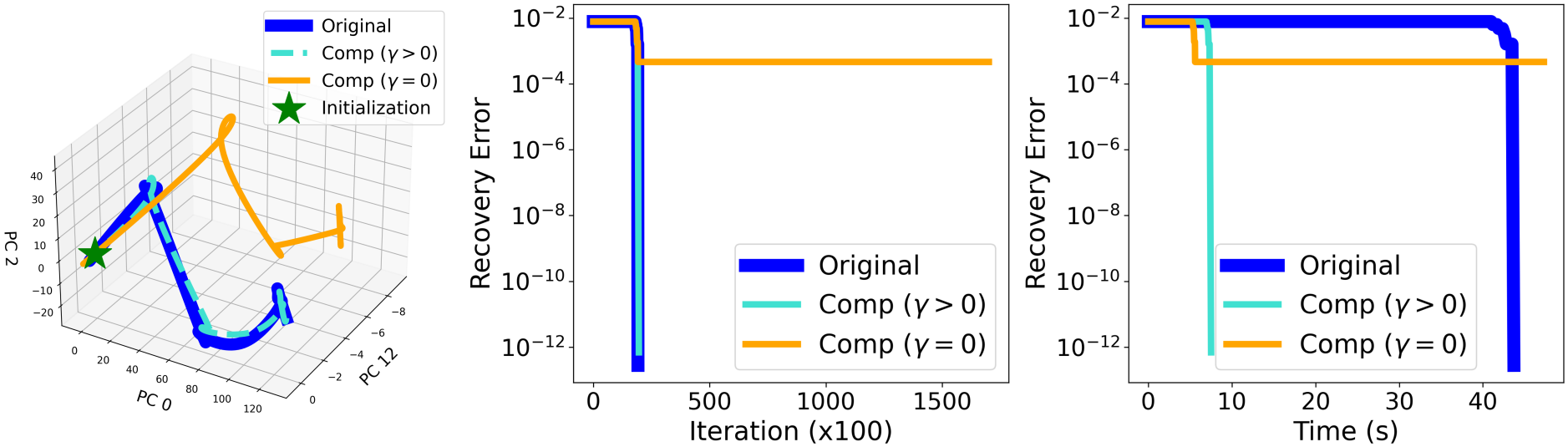


Figure: **Original vs. compressed trajectories with γ discrepant updates ($\gamma = 0.01$) and ablating γ ($\gamma = 0$).**

- **Left**: Principal components of end-to-end trajectories.
- **Middle**: Recovery error vs. iteration comparison.
- **Right**: Recovery error vs wall-time comparison.

Outline

- Introduction
- Theory for Overparameterized Deep Linear Networks
- Matrix Completion
- Deep LoRA

Success of Low-Rank Adaptation

Low-Rank Adaptation (LoRA) [Hu et al., 2021]

Fine-tune dense weight matrix $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ (e.g. of transformer) as

$$\boldsymbol{W} = \overline{\boldsymbol{W}} + \boldsymbol{W}_2 \boldsymbol{W}_1$$

where $\overline{\boldsymbol{W}}$ is frozen pretrained weight matrix and $\boldsymbol{W}_1, \boldsymbol{W}_2$ are trainable factors with $\boldsymbol{W}_1 \in \mathbb{R}^{r \times d}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d \times r}$, and $r \ll d$.

- LoRA achieves high accuracy with very low memory costs on many natural language and vision tasks.
- Its accuracy can exceed that of full fine-tuning!

Weaknesses of Low-Rank Adaptation

Low-Rank Adaptation (LoRA) [Hu et al., 2021]

Fine-tune dense weight matrix $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ of transformer as

$$\boldsymbol{W} = \overline{\boldsymbol{W}} + \boldsymbol{W}_2 \boldsymbol{W}_1$$

where $\overline{\boldsymbol{W}}$ is frozen pretrained weight matrix and $\boldsymbol{W}_1, \boldsymbol{W}_2$ are trainable factors with $\boldsymbol{W}_1 \in \mathbb{R}^{r \times d}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d \times r}$, and $r \ll d$.

- Prone to overfitting with limited data unless rank at each layer is carefully specified
- Choosing layer-specific update ranks using cross validation is intractable

LoRA Overfitting

Ex: LoRA is sensitive to the rank parameter.

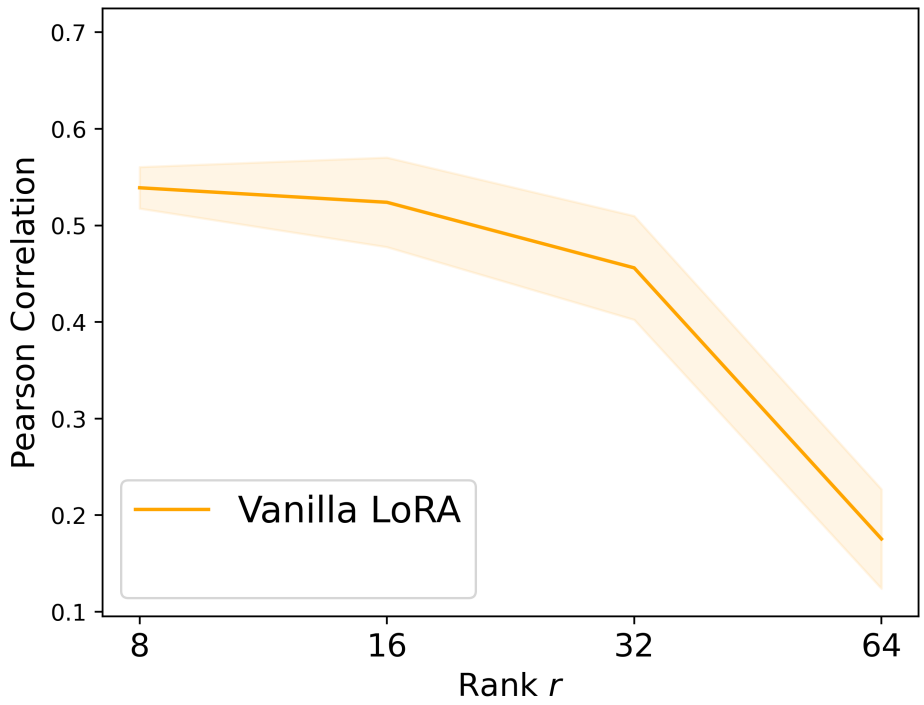


Figure: For fine-tuning BERT on STS-B, each choice of rank r, we draw 16 samples at random from STS-B over 5 trials with different seeds, and measure performance on the validation split.

LoRA Overfitting

Ex: LoRA is sensitive to the rank parameter.

An overparameterized factorization can solve this problem.

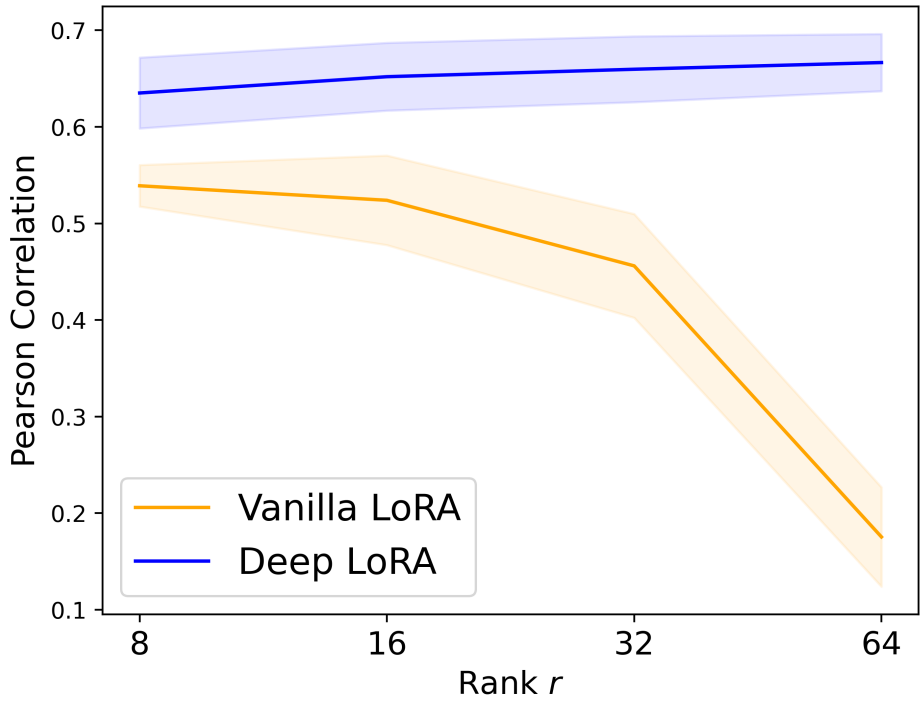


Figure: For fine-tuning BERT on STS-B, each choice of rank r, we draw 16 samples at random from STS-B over 5 trials with different seeds, and measure performance on the validation split.

Deep LoRA - Key Takeaway

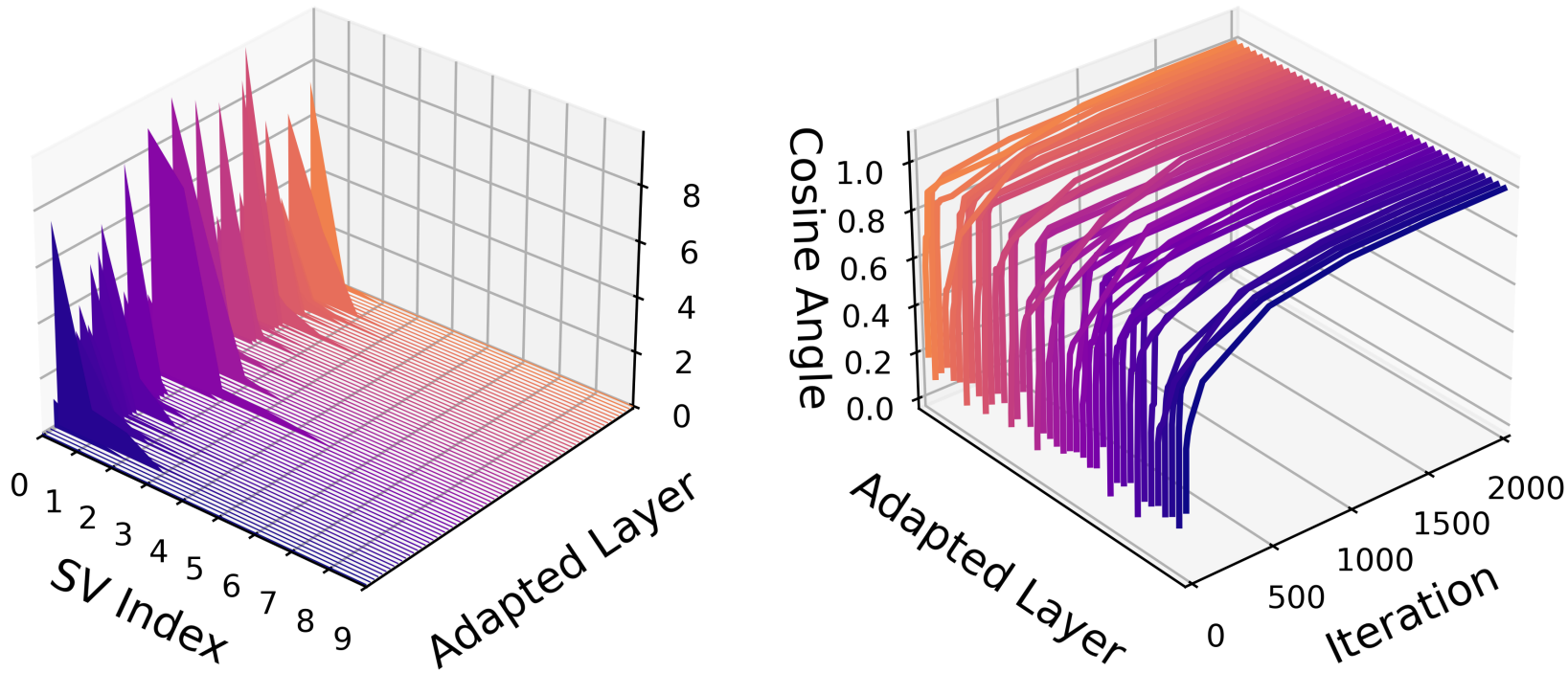


Figure: **Deep, wide** adaptation of BERT yields **simple weight updates** (left) whose directions **converge early** on in fine-tuning (right).

Early alignment to low-rank updates implies compressibility.
Can be compressed to only Lr^2 more parameters than vanilla LoRA.

figure details

Deep LoRA

Deep LoRA uses what we learned in our theorem.

- Step 1: Overparameterize each layer with three ϵ-orthogonal initialized matrices.

$$\boldsymbol{W} = \overline{\boldsymbol{W}} + \boldsymbol{W}_3\boldsymbol{W}_2\boldsymbol{W}_1$$

  where $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3 \in \mathbb{R}^{d\times d}$.

- Step 2: Compute one gradient (one full pass or one batch) and compute the initial "invariant" subspace
- Step 3: Initialize new W_3, W_1 to match these subspaces and W_2 to be random orthogonal, and run gradient descent with discrepant learning rates[2].

[2]The discrepant learning rate may not be necessary, according to more recent studies.

Improved Few-Shot Fine-Tuning with More Parsimony

Deep LoRA performs *better* with limited data

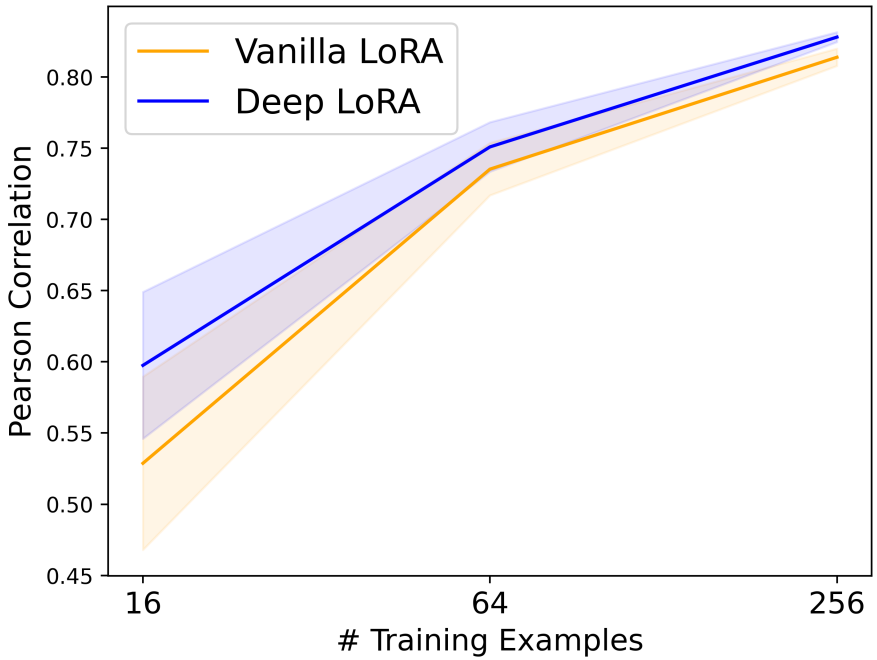


Figure: Fine-tune BERT on STS-B over 20 random trials; measure performance on the validation split of each method using the same train set.

Improved Few-Shot Fine-Tuning with More Parsimony

Deep LoRA performs *better* and *adaptively* chooses correct rank.

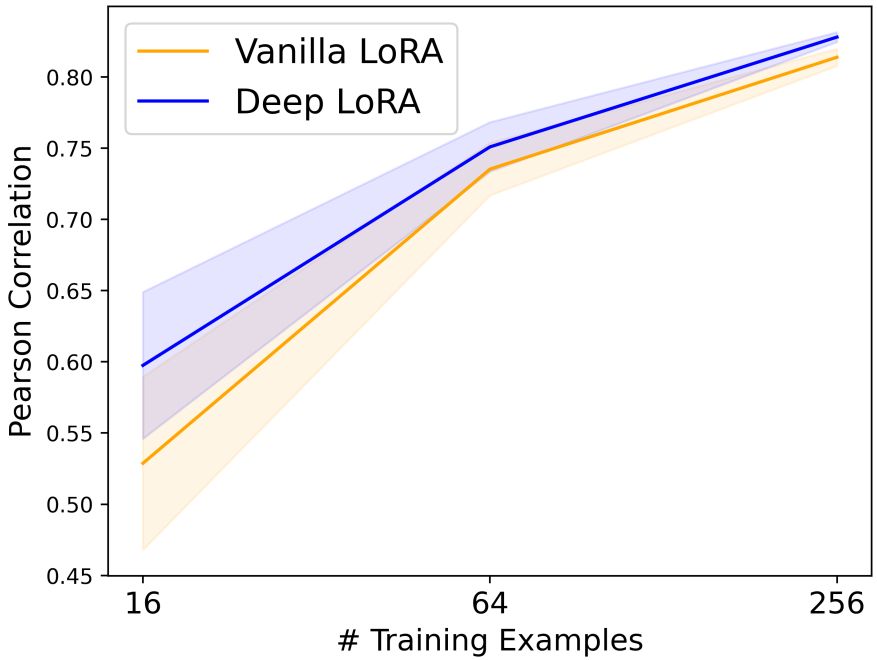


Figure: Fine-tune BERT on STS-B over 20 random trials; measure performance on the validation split of each method using the same train set.

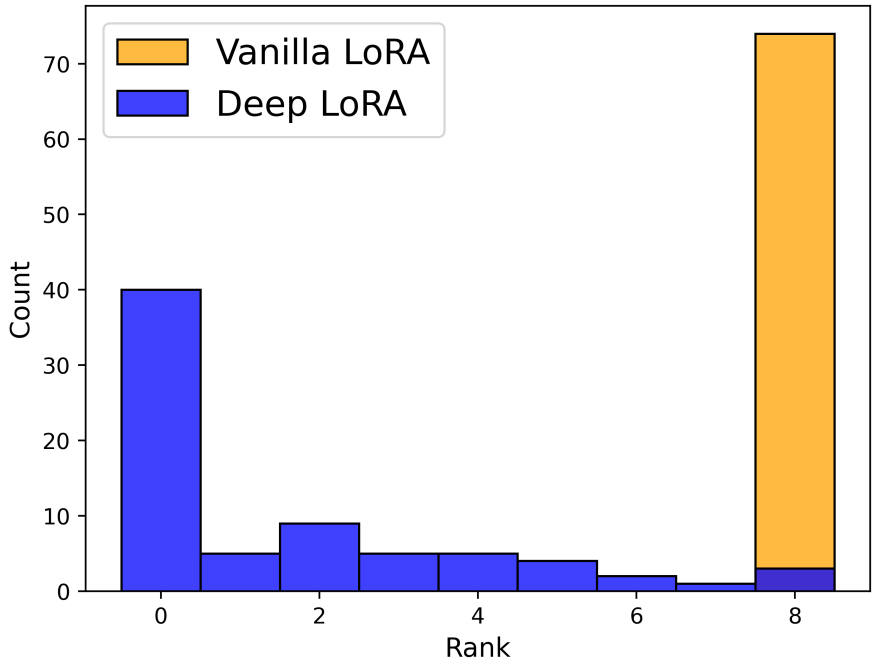


Figure: We plot a histogram of numerical ranks for Deep LoRA and vanilla LoRA with $r = 8$ after adapting to STS-B with 256 samples.

Limited-Data GLUE Benchmark

Table: Performance gap (with variance) for 1024 samples over 10 trials on the validation split between Deep LoRA and vanilla LoRA using the same train set. Metrics are normalized to 1.

	CoLA	**MNLI**	**MRPC**	**QNLI**	**QQP**
Δ	$+0.090_{\pm 0.002}$	$+0.011_{\pm 0.0005}$	$+0.0042_{\pm 0.001}$	$+0.048_{\pm 0.0009}$	$+0.005_{\pm 0.0002}$

	RTE	**SST-2**	**STS-B**	**Overall**
Δ	$+0.029_{\pm 0.002}$	$+0.019_{\pm 0.0006}$	$+0.018_{\pm 0.00006}$	$+0.028_{\pm 0.002}$

- **In practice: Deep LoRA.** Adaptive to the intrinsic rank of the problem with less overfitting.
- **In theory: compressible learning dynamics.** Gradient descent only happens within invariant subspaces of the weights.

- Yaras, Can, Peng Wang, Laura Balzano, and Qing Qu. "Compressible Dynamics in Deep Overparameterized Low-Rank Learning & Adaptation." In International Conference on Machine Learning, pp. 56946-56965. PMLR, 2024.
- Kwon, Soo Min, Zekai Zhang, Dogyoon Song, Laura Balzano, and Qing Qu. "Efficient Low-Dimensional Compression of Overparameterized Models." In Proceedings of Artificial Intelligence and Statistics, 2024.
- Wang, Peng, Xiao Li, Can Yaras, Zhihui Zhu, Laura Balzano, Wei Hu, and Qing Qu. "Understanding deep representation learning via layerwise feature compression and discrimination." Accepted to the Journal of Machine Learning, 2025.
- Yaras, Can, Alec S. Xu, Pierre Abillama, Changwoo Lee, and Laura Balzano. "MonarchAttention: Zero-Shot Conversion to Fast, Hardware-Aware Structured Attention." Accepted to Neural Information Processing Systems, 2025.
- Balzano, Laura, Tianjiao Ding, Benjamin D. Haeffele, Soo Min Kwon, Qing Qu, Peng Wang, Zhangyang Wang, and Can Yaras. "An overview of low-rank structures in the training and adaptation of large models." arXiv preprint arXiv:2503.19859 (2025).


Arora, S., Cohen, N., Hu, W., and Luo, Y. (2019).
Implicit regularization in deep matrix factorization.
Advances in Neural Information Processing Systems, 32.


Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020).
An image is worth 16x16 words: Transformers for image recognition at scale.
In *International Conference on Learning Representations*.


Guo, S., Alvarez, J. M., and Salzmann, M. (2020).
Expandnets: Linear over-parameterization to train compact convolutional networks.
Advances in Neural Information Processing Systems, 33:1298–1310.


Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2021).
Lora: Low-rank adaptation of large language models.
In *International Conference on Learning Representations*.


Huh, M., Mobahi, H., Zhang, R., Cheung, B., Agrawal, P., and Isola, P.
The low-rank simplicity bias in deep networks.
Transactions on Machine Learning Research.


Kwon, S. M., Zhang, Z., Song, D., Balzano, L., and Qu, Q. (2024).
Efficient low-dimensional compression of overparameterized models.
In *Proceedings of Artificial Intelligence and Statistics*.


Simonyan, K. and Zisserman, A. (2015).
Very deep convolutional networks for large-scale image recognition.
In *Proceedings of ICLR*.

Overparameterized Fine-Tuning: Deep LoRA

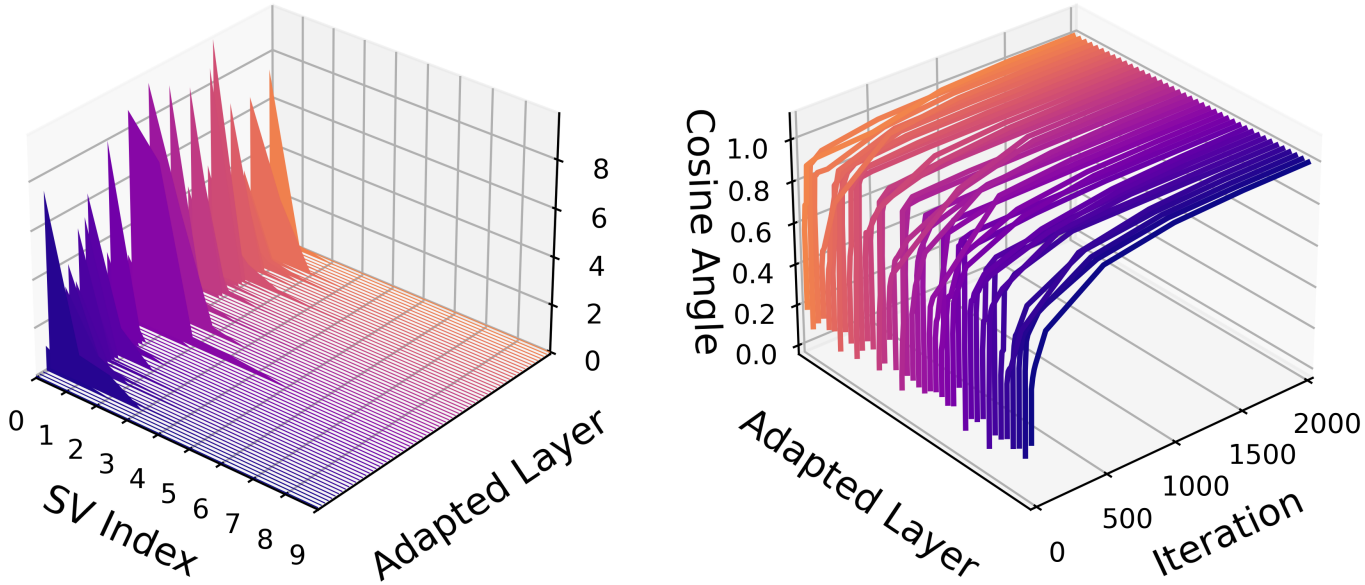


Figure: **Deep, wide** adaptation of BERT yields **simple weight updates** (left) whose directions **converge early** on in fine-tuning (right).

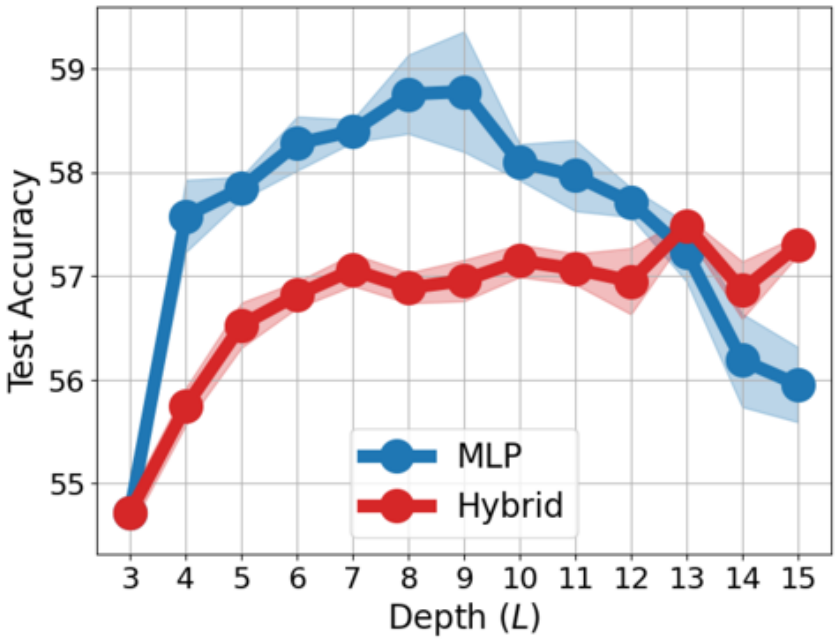
- **Left**: Final singular value spectra of $\boldsymbol{W} - \overline{\boldsymbol{W}}$.
- **Right**: Cosine alignment of subspace formed by top 8 right singular vectors between current and final iterations.

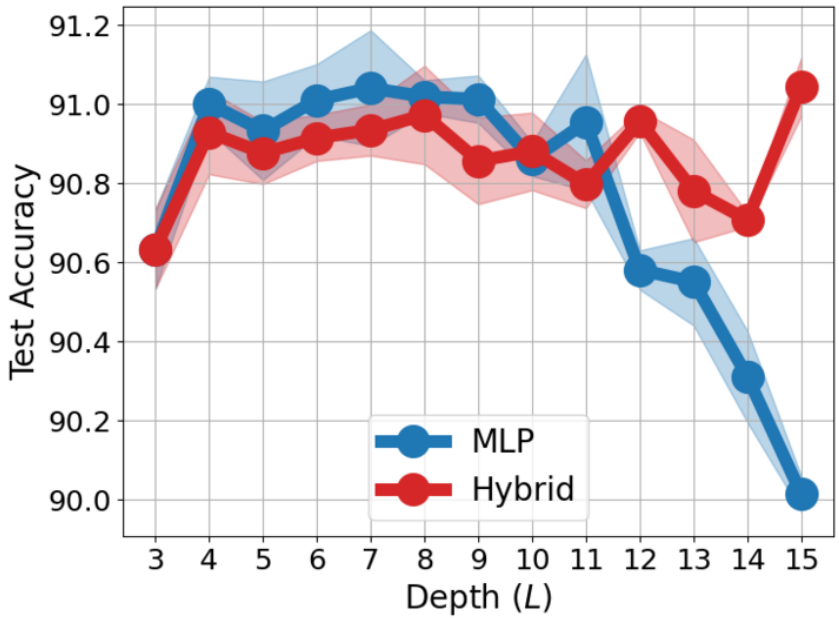
back to takeaway

Why do we need depth in linear networks?

Recall $\boldsymbol{W}_L \cdots \boldsymbol{W}_1 \boldsymbol{x} = \boldsymbol{W}_{L:1} \boldsymbol{x}$, so why not just learn the parameters of one matrix $\boldsymbol{W}_{L:1}$ to apply to $\boldsymbol{x}$? More linear depth related work

Training L-layer networks on F-MNIST (left) and CIFAR-10 (right)



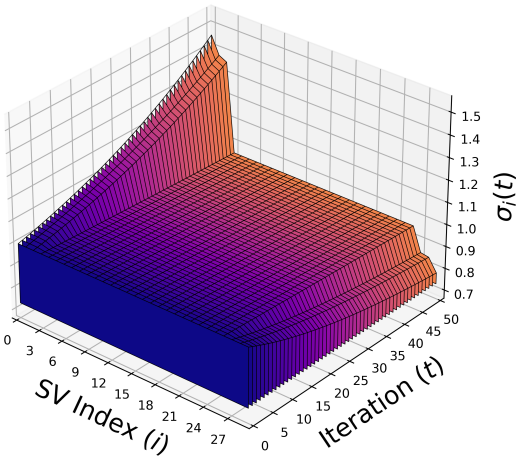


Why do we need depth in linear networks?

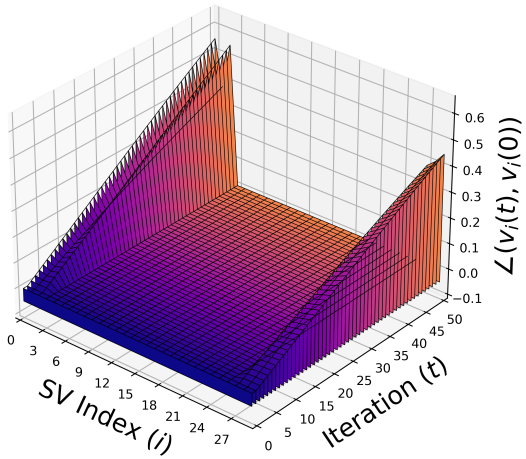
Recall $\boldsymbol{W}_L \cdots \boldsymbol{W}_1 \boldsymbol{x} = \boldsymbol{W}_{L:1} \boldsymbol{x}$, so why not just learn the parameters of one matrix $\boldsymbol{W}_{L:1}$ to apply to $\boldsymbol{x}$?

- Recent works demonstrated that linear over-parameterization by depth (i.e., expanding one linear layer into a composition of multiple linear layers) in deep nonlinear networks yields better generalization performance across different network architectures and datasets [Guo et al., 2020, Huh et al., , Kwon et al., 2024].
- This is also corroborated by our experiments, where increasing the depth of linear layers of a hybrid network leads to improved test accuracy and improved feature compression.
- Another work [Arora et al., 2019] shows that deeper overparameterized factorizations have implicit regularization towards low-rank solutions, and they can better handle ill-conditioned or low-sample settings. Our experiments also support this.

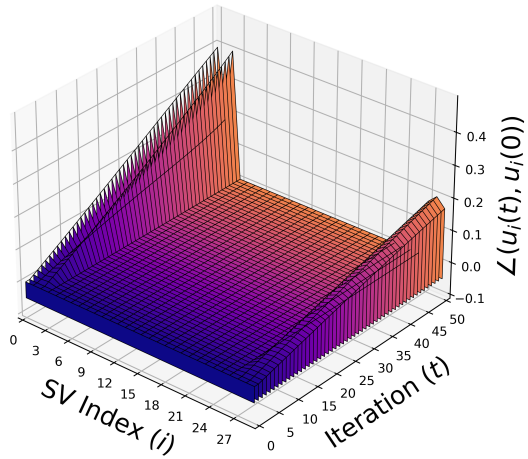
Learning Dynamics Visualized



Singular Values



Right Singular Vectors



Left Singular Vectors

Figure: **Evolution of SVD of the weight matrix $\boldsymbol{W}_1(t) = \boldsymbol{U}_1(t)\boldsymbol{\Sigma}_1(t)\boldsymbol{V}_1(t)^\top$.**

- **Left**: Evolution of singular values $\sigma_i(t)$ throughout training.
- **Middle**: Evolution of $\angle(\boldsymbol{v}_i(t), \boldsymbol{v}_i(0))$ throughout training.
- **Right**: Evolution of $\angle(\boldsymbol{u}_i(t), \boldsymbol{u}_i(0))$ throughout training.

Proof Ideas

- The analytic form of the gradient $\nabla_{\boldsymbol{\Theta}} \ell(\boldsymbol{\Theta})$ is given by

$$\nabla_{\boldsymbol{W}_l} \ell(\boldsymbol{\Theta}) = \boldsymbol{W}_{L:l+1}^\top \left(f(\boldsymbol{\Theta}) - \boldsymbol{\Phi} \right) \boldsymbol{W}_{l-1:1}^\top, \; l \in [L] \; ,$$

  which when substituted into the gradient descent update equation gives

$$\boldsymbol{W}_l(t+1) = (1 - \eta\lambda)\boldsymbol{W}_l(t) - \eta \boldsymbol{W}_{L:l+1}^\top(t) \left(\boldsymbol{W}_{L:1}(t) - \boldsymbol{\Phi} \right) \boldsymbol{W}_{l-1:1}^\top(t)$$

  for $l \in [L]$ and $t \geq 0$.

Proof Ideas

$$\boldsymbol{W}_l(t+1) = (1-\eta\lambda)\boldsymbol{W}_l(t) - \eta \boldsymbol{W}_{L:l+1}^\top(t)\left(\boldsymbol{W}_{L:1}(t) - \boldsymbol{\Phi}\right)\boldsymbol{W}_{l-1:1}^\top(t)$$

We proceed by establishing $\forall l$ the existence of $m := d - 2r^*$ vectors $\boldsymbol{v}_i^{(l)}, \boldsymbol{u}_i^{(l)}$, $i \in [m]$, so that $\forall t$:

$$\begin{aligned}
\mathcal{A}(t) &: \boldsymbol{W}_l(t)\boldsymbol{v}_i^{(l)} = \rho_l(t)\boldsymbol{u}_i^{(l)}, \\
\mathcal{B}(t) &: \boldsymbol{W}_l^\top(t)\boldsymbol{u}_i^{(l)} = \rho_l(t)\boldsymbol{v}_i^{(l)}, \\
\mathcal{C}(t) &: \boldsymbol{\Phi}^\top \boldsymbol{W}_{L:l+1}(t)\boldsymbol{u}_i^{(l)} = \boldsymbol{0}, \\
\mathcal{D}(t) &: \boldsymbol{\Phi}\boldsymbol{W}_{l-1:1}^\top(t)\boldsymbol{v}_i^{(l)} = \boldsymbol{0}.
\end{aligned}$$

If these four events hold, $\boldsymbol{\Phi}$ plays no role in the dynamics, and the subspace spanned by $\{\boldsymbol{v}_i^{(1)}\}_{i=1}^m$ "passes through" the update equation.

Proof Ideas

$$\boldsymbol{W}_1(1) = (1-\eta\lambda)\boldsymbol{W}_1(0) - \eta \boldsymbol{W}_{L:2}^\top(0)\left(\boldsymbol{W}_{L:1}(0) - \boldsymbol{\Phi}\right)\boldsymbol{W}_1^\top(0)$$

Define $\mathcal{S} := \mathcal{N}\left(\boldsymbol{W}_{L:2}^\top(0)\boldsymbol{\Phi}\right) \cap \mathcal{N}\left(\left(\boldsymbol{W}_{L:2}^\top(0)\boldsymbol{\Phi}\right)^\top \boldsymbol{W}_1(0)\right) \subset \mathbb{R}^d$ - notice it has $\dim(\mathcal{S}) \geq d - 2r^* = m$.

For any orthonormal set $\{\boldsymbol{v}_i^{(1)}\}_{i=1}^m \in \mathcal{S}$,

- Let $\boldsymbol{u}_i^{(1)} := \boldsymbol{W}_1(0)\boldsymbol{v}_i^{(1)}/\epsilon_1$ (also an orthonormal set)
- Then $\boldsymbol{W}_1(0)^\top \boldsymbol{u}_i^{(1)} = \epsilon_1 \boldsymbol{v}_i^{(1)}$
- and $\boldsymbol{W}_{L:2}^\top(0)\boldsymbol{\Phi}\boldsymbol{v}_i^{(1)} = \boldsymbol{0}$
- and $\boldsymbol{\Phi}^\top \boldsymbol{W}_{L:2}(0)\boldsymbol{W}_1(0)\boldsymbol{v}_i^{(1)} = \epsilon_1 \boldsymbol{\Phi}^\top \boldsymbol{W}_{L:2}(0)\boldsymbol{u}_i^{(1)} = \boldsymbol{0}$

The rest follows from induction on $l \in [L]$ and then induction on $t \geq 0$.

back to theorem

Ablating Compression for Deep LoRA