Incremental Low-Rank Learning

Jiawei Zhao [*1] **Yifei Zhang** [*2] **Beidi Chen** [3] **Florian Schäfer** [4] **Anima Anandkumar** [1 5]

Abstract

The theory of greedy low-rank learning (GLRL) aims to explain the impressive generalization capabilities of deep learning. It proves that stochastic gradient-based training implicitly regularizes neural networks towards low-rank solutions through a gradual increase of the rank during training. However, there is a gap between theory and practice since GLRL requires an infinitesimal initialization of the weights, which is not practical due to the fact that it is a saddle point. In this work, we remove the assumption of infinitesimal initialization by focusing on *cumulative weight updates*. We prove the cumulative weight updates follow an incremental low-rank trajectory for arbitrary orthogonal initialization of weights in a three-layer linear network. Empirically, we demonstrate that our theory holds on a broad range of neural networks (e.g., transformers) and standard training algorithms (e.g., SGD, Adam). However, existing training algorithms do not exploit the low-rank property to improve computational efficiency as the networks are not parameterized in low-rank. To remedy this, we design a new training algorithm *Incremental Low-Rank Learning* (InRank), which explicitly expresses cumulative weight updates as low-rank matrices while incrementally augmenting their ranks during training. We evaluate InRank on GPT-2, and our results indicate that InRank achieves comparable prediction performance as the full-rank counterpart while requiring at most 33% of the total ranks throughout training. We also propose an efficient version of InRank that achieves a reduction of 20% in total training time and 37% in memory usage when training GPT-medium on WikiText-103 from scratch.

[*]Equal contribution [1]California Institute of Technology [2]University of Wisconsin-Madison [3]Carnegie Mellon University [4]Georgia Institute of Technology [5]NVIDIA. Correspondence to: Jiawei Zhao <jiawei@caltech.edu>.

Work presented at the ES-FoMo Workshop at ICML 2023.

1. Introduction

The generalization ability of deep neural networks continues to intrigue researchers since the classical theory is not applicable in the over-parameterized regime, where there are more learnable parameters than training samples. Instead, efforts to understand this puzzle are based on the belief that first-order learning algorithms (e.g., stochastic gradient descent) implicitly bias the neural networks toward simple solutions.

For instance, it has been shown that stochastic gradient descent implicitly minimizes the rank of solutions during training (Arora et al., 2019). Recent theoretical studies have further demonstrated one of its training characterizations - Greedy Low-Rank Learning (GLRL) (Li et al., 2021; Jacot et al., 2022). GLRL characterizes the trajectory of stochastic gradient descent, which performs a rank-constrained optimization and greedily increases the rank whenever it fails to reach a global minimizer.

However, one major drawback is that the GLRL theory requires the assumption of infinitesimal initialization, which is impractical as gradient descent cannot effortlessly escape from the saddle point at zero, unless the noise is large enough. Therefore, a generalized notion of GLRL under practical initialization is needed to bridge the gap between theory and practice.

In this work, we generalize the theory of GLRL by removing the requirement of infinitesimal initialization. To do this, we focus on characterizing the trajectories of a new set of quantities, *cumulative weight updates*, instead of weight matrices. Cumulative weight updates do not include the initialization values, and only incorporate the rest of the updates of the weight matrices during training. This allows us to remove the requirement of infinitesimal initialization in GLRL.

We establish incremental rank augmentation of cumulative weight updates during training under arbitrary orthogonal initialization of the weights. This new formulation proves that low-rank learning can be extended to non-zero initialization, where the singular vector with a larger value in the associated target matrix is learned exponentially faster. We prove this relationship by following the work of Saxe et al. (2014) to analyze the evolution of each mode (singular vector) independently, which can be achieved by ensuring orthogonality over the weights matrices and inputs in a three-layer linear network.

Empirically, we further demonstrate that standard networks (e.g., transformers) and training algorithms (e.g., SGD, Adam) follow low-rank learning trajectories on the cumulative weight updates, under standard weight initialization. However, current algorithms can not exploit the low-rank property to improve computational efficiency as the networks are not parameterized in low-rank.

To address this, we propose *Incremental Low-Rank Learning* (InRank), which parameterizes the cumulative weight updates in low-rank while incrementally augmenting its rank during training, as illustrated in Figure 1. InRank adds a new batch of modes whenever a certain quantity, known as the explained ratio, exceeds a certain threshold. The explained ratio represents the amount of information in the underlying spectrum that the current rank can explain. A low explained ratio indicates that the current rank is inadequate to represent the spectrum, necessitating the addition of more modes.

InRank is capable of identifying *intrinsic rank* of networks during training. The intrinsic rank of a neural network is defined as the minimum sufficient rank that trains the network from scratch without sacrificing performance. The capability of finding the intrinsic rank addresses the challenge of pre-defining the fixed ranks in training low-rank neural networks, which requires expensive hyperparameter tuning. An inappropriate selection of rank may either limit model capacity, hinder the learning process, or result in excessive memory usage and computation, thereby negating the benefits of low-rank factorization. We further improve computational efficiency by applying InRank only in the initial phase of training. This approach mitigates the computational cost induced by expensive SVD operations in InRank, while maintaining comparable accuracy as the full-rank models.

Our contributions are summarized as follows:

1. We generalize the theory of GLRL to arbitrary orthogonal initialization by establishing incremental rank augmentation of cumulative weight updates.

2. We propose InRank, which can find the intrinsic rank of networks during training. It maintains prediction performance equivalent to full-rank counterparts but requires a maximum of 33% total ranks when evaluating InRank on WikiText-103 using GPT-2 models.

3. We further enhance the training efficiency of InRank. This efficient variant decreases the training time by 20% and reduces memory usage by 37% when training GPT-medium from scratch.

2. Cumulative Weight Updates follow Low-Rank Learning Trajectory

In order to generalize GLRL beyond infinitesimal initialization, we focus on *cumulative weight updates* that characterize GLRL for any regular initializations (details of GLRL can be found in the appendix).

We wish to train the function $\mathcal{F}(x)$ to learn a particular input-output map given a set of P training samples $(x_\mu, y_\mu) \in \mathbb{R}^{N_x \times N_y}$, where $\mu = 1, ..., P$. Training is accomplished by minimizing the squared error $\mathcal{L} = \frac{1}{2}\sum_{\mu=1}^{P} \|y_\mu - \mathcal{F}(x_\mu)\|_2^2$ using gradient descent with a step size η. We first model $\mathcal{F}(x)$ to be a deep linear network: $\mathcal{F}(x) = W^L \dots W^1 x$, where $W^l \in \mathbb{R}^{N_h \times N_h}$ for $l \in 1, ..., L$. w_t denotes the whole parameter vector at iteration t.

We let $A_\theta = W^L \dots W^1$ denote the product matrix of the network, and θ denote the whole parameter vector. Thus, we also denote the training error as $C(A_\theta)$ where C is a convex error (e.g., the squared error). We define the cumulative weight updates as follows:

Definition 1 (Cumulative Weight Updates)**.** The *cumulative weight updates* d_t at iteration t is defined as the difference between the current parameterization w_t and initialization w_0 in the parameter space, such that

$$d_t = w_t - w_0 = \sum_{i=1}^{t} \Delta w_i. \tag{1}$$

The cumulative weight updates d_t have been widely studied in the literature, especially in the field of distributed training (Vogels et al., 2019), as it is known to exhibit low-rank properties during training.

This is attributed to the fact that d_t is a summation of updates to the weights Δw_i, with each update being determined by the learning algorithm and current gradient g_t. Gradient g_t has been shown to possess low-rank properties, which has been exploited to reduce communication costs in distributed training through low-rank compression (Vogels et al., 2019; Wang et al.).

We theoretically prove that the cumulative weight updates d_t follow a low-rank learning trajectory, even when the initialization is not infinitesimal. We continue to focus on a linear network and analyze the difference of the product matrix $D_t = A_t - A_0$ (which can be viewed as the cumulative weight updates of the product matrix). Our goal is to demonstrate that D_t exhibits an exponential rank increase even when the initial weights are not close to zero. Our analysis builds upon the work of Saxe et al. (2014), which studies training dynamics under orthogonal inputs.

Assumption 1 (Orthogonal Inputs)**.** We assume the inputs are orthogonal, i.e., $x_i^T x_j = 0$ for $i \neq j$.

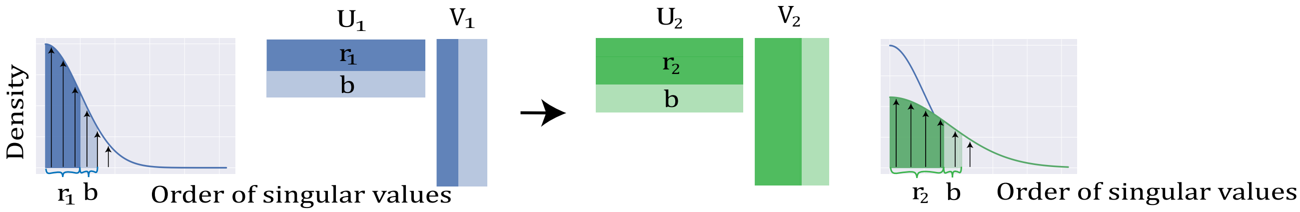

Figure 1. **Incremental Low-Rank Learning from iteration t_1 to t_2.** U and V represent any factorized layer. Density plots indicate the strength of each singular vector (normalized by the total strengths). Solid areas represent how much information in the spectrum is explained by the current rank r_t at iteration t. From iteration t_1 to t_2, InRank adds $r_2 - r_1$ additional ranks to ensure the ratio of the explained information is greater than a certain threshold α.

Consider the input-output correlation matrix:

$$\Sigma^{yx} = \sum_{\mu=1}^{P} y_\mu x_\mu^T = U^{yy} S^{yx} V^{xx} = \sum_{\alpha=1}^{N_x} s_\alpha u_\alpha v_\alpha^T, \quad (2)$$

where U^{yy} and V^{xx} represent the left and right singular vectors of Σ^{yx}, and S^{yx} denotes its singular value matrix. The singular values are ordered such that $s_1 \geq s_2 \geq \cdots \geq s_{N_x}$.

We analyze a 3-layer linear network where $y = W^2 W^1 x$, $W^1 \in \mathbb{R}^{N_h \times N_x}$ and $W^2 \in \mathbb{R}^{N_y \times N_h}$ are the weight matrices of the first and second layers, respectively, and $N_h < N_x, N_y$. After training, the converged network should satisfy:

$$W^2 W^1 = \sum_{\alpha=1}^{N_h} s_\alpha u_\alpha v_\alpha^T, \quad (3)$$

which is the closest rank-N_h approximation to Σ^{yx}. To further analyze its trajectory, we assume that the weights are initialized as $W_0^2 = U^{yy} M^2 O^T, W_0^1 = O M^1 V^{xx^T}$, where M^2, M^1 are diagonal matrices, and O is an arbitrary orthogonal matrix. We have the following theorem for the training evolution of D_t:

Theorem 1. *For any orthogonal matrix O and scaled diagonal matrices M^2 and M^1, each singular value $u_f(t)$ in D_t at iteration t follows the trajectory:*

$$u_f(t) = \frac{s e^{2st/\tau}}{e^{2st/\tau} - 1 + s/u_0} - u_0, \quad (4)$$

where s is the target singular value in Σ^{yx}, u_0 is the initial value determined by M^2 and M^1, and τ is a constant.

$W_0^2 W_0^1$ ensures that each mode α is learned independently right from the beginning of training, enabling us to analyze the learning trajectory of each mode separately. The diagonal matrices M^2 and M^1 control the scale of the initial weights, i.e., the initial value u_0 of each mode α. Consequently, a larger u_0 that is closer to s accelerates the learning speed. We provide comprehensive proof of the theorem in the appendix for further clarity.

The sigmoid function in Theorem 1 exhibits a sharp transition from a state of no learning to full learning, with the transition point determined by the initial value u_0 and s. This indicates that if the target singular values s are distinct enough (given $s >> u_0$), each $u_f(t)$ will follow an independent sigmoid trajectory, permitting ranks to be learned sequentially and independently.

Shifting our focus to practical non-linear networks, we analyze the difference of layer-wise weight matrix $D_t = W_t^l - W_0^l$ for $l = 1, ..., L$ instead of the product matrix $D_t = W_t^L ... W_t^1 - W_0^L ... W_0^1$. We also extend our evaluation to more practical cases with modern weight initialization methods.

As shown in the appendix, cumulative weight updates D_t follow the greedy low-rank learning trajectory even under regular initializations, including Orthogonal, ZerO, and Kaiming methods (Saxe et al., 2014; He et al., 2015; Zhao et al., 2021). We further verify our theory on a broad range of neural networks (e.g., transformers) and standard training algorithms (e.g., SGD, Adam). This observation motivates us to design an efficient incremental learning algorithm that leverages the properties of cumulative weight updates.

3. Incremental Learning

Motivated by the previous findings, we propose an incremental low-rank learning algorithm that leverages the implicit low-rank learning trajectory in practice. To explicitly represent the cumulative weight updates, we parametrize the weight matrix W^l at any layer l as follows:

$$W^l = W_0^l + D^l, \quad (5)$$

where W_0^l is the initial matrix and D^l is the summation of

Algorithm 1 Incremental Low-Rank Learning (InRank)

Require: $L(W_t)$ is the cost of total weights $W = (W^1, ..., W^L)$ at iteration t, $W_t^l = W_0^l + U_t^l V_t^l$ for each layer l (W_0^l is not trainable) and let $r, b, \alpha, \epsilon, \eta, T > 0$

1: Initialize W_0^l using standard initialization, and set U_0^l, V_0^l to 0
2: Compute the top $(1+b)$ singular vectors: $u^l, s^l, v^l \leftarrow \mathrm{SVD}_{(1+b)}(\frac{\partial L(W_0)}{\partial W_0^l})$
3: Initialize factorized weights with small ϵ : $U_0^l \leftarrow -\epsilon v^l$, $V_0^l \leftarrow \epsilon u^l$, and $r_0^l \leftarrow 1$
4: **for** $t = 1, 2, 3, ..., T$ **do**
5: Train low-rank network and update $U_t^l V_t^l$ using SGD with learning rate η
6: Compute the top $(r^l + b)$ singular vectors: $u^l, s^l, v^l \leftarrow \mathrm{SVD}_{(r^l+b)}(U_t^l V_t^l)$
7: Increment r_t^l to r_{t+1}^l until the explained ratio $g(U_t^l V_t^l, r_{t+1}^l, b) \geq \alpha$
8: Initialize additional parameters: $U_{t+1}^l \leftarrow [U_t^l, U^*]$, $V_{t+1}^l \leftarrow [V_t^l, V^*]$, where $U^* \in \mathbb{R}^{p^l \times (r_{t+1}^l - r_t^l)}$ and $V^* \in \mathbb{R}^{(r_{t+1}^l - r_t^l) \times q^l}$ are randomly initialized with small values
9: **end for**

weight updates. Since D^l exhibits low-rank properties, we can factorize it as $D^l = U^l V^l$, resulting in:

$$W^l = W_0^l + U^l V^l, \tag{6}$$

where $U^l \in \mathbb{R}^{p^l \times r^l}$ and $V^l \in \mathbb{R}^{r^l \times q^l}$ are the factorized matrices, and r^l is the rank of D^l.

To emulate the implicit low-rank learning, we train factorized matrices $U^l V^l$ with an initially small rank r^l, subsequently increasing the rank (i.e., the matrix size) throughout the training process.

A crucial challenge lies in determining how to increase the rank r^l during training. An inappropriate choice of rank may either lead to insufficient model capacity, hinder the learning process, or result in excessive memory usage and computation, negating the benefits of low-rank factorization.

To address this, we propose a novel method for dynamically identifying when a rank increase is necessary, based on measuring the representability of the current rank r^l. Inspired by Zhao et al. (2022), we define *explained ratio*:

$$g(M, r^l, b) = \frac{\sum_{i=1}^{r^l} s_i^l}{\sum_{i=1}^{r^l+b} s_i^l}, \tag{7}$$

where s_i^l is the i-th singular value of a matrix M, and b is a buffer size used to encompass a broader spectrum for determination. The explained ratio g quantifies the representability of the current rank r^l in the truncated spectrum (of size $(r^l + b)$) of M. A low explained ratio g indicates that the existing rank r^l cannot sufficiently represent the truncated spectrum, necessitating an increase in r^l to incorporate more useful modes.

We let $M = U^l V^l$ for each layer l and compute the explained ratio $g(U^l V^l, r^l, b)$ at each iteration (can be relaxed each k iterations in practice). By predefining an appropriate threshold α and ensuring that $g(U^l V^l, r^l, b)$ remains larger than α during training, the rank r^l can automatically increase when needed. This process is illustrated in Figure 1.

It is worth noting that b buffer ranks serve to provide a wider spectrum range, but their corresponding singular vectors may be less useful. These buffer ranks can be discarded by fine-tuning in the post-training stage. The full algorithm is detailed in Algorithm 1.

4. Evaluation

In this section, we conduct a comprehensive evaluation of our proposed InRank algorithm on GPT-2 [1]. Our method particularly focuses on the fully-connected layers in the models, where we substitute the conventional weight parameterization with our relative parameterization as described in Equation 6. This operation involves applying InRank to the resulting low-rank factorized matrices. Notably, our approach is not exclusively limited to fully-connected layers. It bears the flexibility to be extended to various types of layers, including convolution and self-attention layers. However, to maintain the focus on our current research, we leave this promising exploration for future work. We further improve the efficiency of InRank by proposing an efficient variant, InRank-Efficient, which only applies InRank during the initial training stage. Details of InRank-Efficient and hyperparameters configuration are provided in the appendix for further reference.

We compare InRank and InRank-Efficient with a full-rank baseline using different sizes of GPT models on the WikiText-103 dataset. All methods are trained with the same hyperparameters, including the learning rate, weight decay, and the number of epochs. We use the Adam optimizer to train for 100 epochs with an initial learning rate of 0.001. All experiments are run using the same computational setting with 8 NVIDIA® Tesla® V100 GPUs.

[1]The code for our algorithm is available at this link

Table 1. Performance comparison of different methods.

Model	Method	PPL	Rank	Runtime	Memory	Params	FLOPs
GPT-small	Baseline	**18.5**	768	24.5h	248Mb	124M	292G
	InRank	18.6	**254**	23.4h	295Mb	147.7M	348G
	InRank-Efficient	18.9	**254**	**22.2h**	**182Mb**	**91.2M**	**178G**
GPT-medium	Baseline	**19.5**	1024	60.5h	709Mb	355M	828G
	InRank	19.6	**286**	57.4h	850Mb	424M	993G
	InRank-Efficient	19.9	**286**	**48.6h**	**447Mb**	**223M**	**428G**

As shown in Table 1, both InRank and InRank-Efficient achieve validation perplexity comparable to the full-rank baseline while requiring at most 33% of the total rank. The rank is calculated as the average rank across all weight matrices in the model. We observed that InRank outperforms InRank-Efficient, even though they find the same rank. This can be attributed to the fact that InRank-Efficient discards the parameterization of W_0 during the training process.

We also measure several efficiency metrics to compare the computational efficiency of different methods. Specifically, we measure the total training time, memory usage, number of parameters, and the number of floating point operations (FLOPs) required for training.

Notably, InRank-Efficient significantly reduces both computational cost and memory usage. For instance, when compared to the baseline on GPT-medium, InRank-Efficient reduces the total training time by 20% and memory usage by 37%. On the other hand, throughout the entire training process, InRank-Efficient requires a maximum of 63% memory usage, enabling the training of large language models from scratch on memory-constrained devices.

Moreover, InRank-Efficient demonstrates even greater efficiency benefits with larger models. In the case of GPT-large, InRank-Efficient reduces 75% of the total rank, resulting in a reduction of 30% in training time and 42% in memory usage when measured over a single epoch. Unfortunately, due to our limited computational resources, we were unable to report its performance over a full training run. Additional results and discussions are provided in the appendix for further reference.

5. Conclusion

In this work, we generalize the Greedy Low-Rank Learning (GLRL) to arbitrary orthogonal initialization, leading to the development of Incremental Low-Rank Learning (InRank). Our method is capable of discovering the intrinsic rank of networks and has demonstrated comparable performance to full-rank counterparts on training GPT-2, while utilizing a maximum of 33% of total ranks throughout training. The efficient variant of InRank also achieves a significant reduction of 20% in total training time and 37% in memory usage when training GPT-medium on WikiText-103.

We believe our work offers a novel approach to training low-rank networks through automatic rank determination. In the future, we aim to expand our method to encompass various network architectures and datasets. Additionally, we intend to optimize our algorithm implementation to further improve its computational efficiency.

References

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit Regularization in Deep Matrix Factorization. *arXiv:1905.13655 [cs, stat]*, October 2019. URL `http://arxiv.org/abs/1905.13655`. arXiv: 1905.13655.

Dao, T., Chen, B., Sohoni, N., Desai, A., Poli, M., Grogan, J., Liu, A., Rao, A., Rudra, A., and Ré, C. Monarch: Expressive Structured Matrices for Efficient and Accurate Training, April 2022. URL `http://arxiv.org/abs/2204.00595`. arXiv:2204.00595 [cs].

Gissin, D., Shalev-Shwartz, S., and Daniely, A. The Implicit Bias of Depth: How Incremental Learning Drives Generalization, December 2019. URL `http://arxiv.org/abs/1909.12051`. arXiv:1909.12051 [cs, stat].

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Characterizing Implicit Bias in Terms of Optimization Geometry. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1832–1841. PMLR, July 2018. URL `https://proceedings.mlr.press/v80/gunasekar18a.html`. ISSN: 2640-3498.

He, K., Zhang, X., Ren, S., and Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, Santiago, Chile, December 2015. IEEE. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.123. URL `http://ieeexplore.ieee.org/document/7410480/`.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-

Rank Adaptation of Large Language Models, October 2021. URL http://arxiv.org/abs/2106.09685. arXiv:2106.09685 [cs].

Huang, X. and Alkhalifah, T. PINNup: Robust neural network wavefield solutions using frequency upscaling and neuron splitting. *arXiv:2109.14536 [physics]*, September 2021. URL http://arxiv.org/abs/2109.14536. arXiv: 2109.14536.

Idelbayev, Y. and Carreira-Perpinan, M. A. Low-Rank Compression of Neural Nets: Learning the Rank of Each Layer. pp. 8049–8059, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Idelbayev_Low-Rank_Compression_of_Neural_Nets_Learning_the_Rank_of_Each_CVPR_2020_paper.html.

Ioannou, Y., Robertson, D., Shotton, J., Cipolla, R., and Criminisi, A. Training CNNs with Low-Rank Filters for Efficient Image Classification, February 2016. URL http://arxiv.org/abs/1511.06744. arXiv:1511.06744 [cs].

Jacot, A., Ged, F., Şimşek, B., Hongler, C., and Gabriel, F. Saddle-to-Saddle Dynamics in Deep Linear Networks: Small Initialization Training, Symmetry, and Sparsity. *arXiv:2106.15933 [cs, stat]*, January 2022. URL http://arxiv.org/abs/2106.15933. arXiv: 2106.15933.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. Measuring the Intrinsic Dimension of Objective Landscapes. In *International Conference on Learning Representations*, May 2023.

Li, Z., Luo, Y., and Lyu, K. Towards Resolving the Implicit Bias of Gradient Descent for Matrix Factorization: Greedy Low-Rank Learning. *arXiv:2012.09839 [cs, stat]*, April 2021. URL http://arxiv.org/abs/2012.09839. arXiv: 2012.09839.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. On the Spectral Bias of Neural Networks. Technical Report arXiv:1806.08734, arXiv, May 2019. URL http://arxiv.org/abs/1806.08734. arXiv:1806.08734 [cs, stat] type: article.

Razin, N., Maman, A., and Cohen, N. Implicit Regularization in Tensor Factorization. *arXiv:2102.09972 [cs, stat]*, June 2021. URL http://arxiv.org/abs/2102.09972. arXiv: 2102.09972.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120 [cond-mat, q-bio, stat]*, February 2014. URL http://arxiv.org/abs/1312.6120. arXiv: 1312.6120.

Schotthöfer, S., Zangrando, E., Kusch, J., Ceruti, G., and Tudisco, F. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations, May 2022. arXiv:2205.13571 [cs, math, stat].

Vogels, T., Karimireddy, S. P., and Jaggi, M. PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d9fbed9da256e344c1fa46bb46c34c5f-Abstract.html.

Wang, H., Agarwal, S., and Papailiopoulos, D. Pufferfish: Communication-efficient Models At No Extra Cost. pp. 22.

Wang, H., Agarwal, S., U-chupala, P., Tanaka, Y., Xing, E. P., and Papailiopoulos, D. Cuttlefish: Low-rank Model Training without All The Tuning, May 2023. URL http://arxiv.org/abs/2305.02538. arXiv:2305.02538 [cs].

Yang, H., Tang, M., Wen, W., Yan, F., Hu, D., Li, A., Li, H., and Chen, Y. Learning Low-rank Deep Neural Networks via Singular Vector Orthogonality Regularization and Singular Value Sparsification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2899–2908, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72819-360-1. doi: 10.1109/CVPRW50498.2020.00347. URL https://ieeexplore.ieee.org/document/9150852/.

You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R. G., Wang, Z., and Lin, Y. Drawing Early-Bird Tickets: Towards More Efficient Training of Deep Networks, February 2022. URL http://arxiv.org/abs/1909.11957. arXiv:1909.11957 [cs, stat].

Zhao, J., Schäfer, F., and Anandkumar, A. ZerO Initialization: Initializing Residual Networks with only Zeros and Ones, October 2021. URL http://arxiv.org/abs/2110.12661. arXiv:2110.12661 [cs].

Zhao, J., George, R. J., Li, Z., and Anandkumar, A. Incremental Spectral Learning in Fourier Neural Operator, November 2022. URL https://arxiv.org/abs/2211.15188v3.

A. Related Work

Implicit regularization has been well studied to explain excellent generalization in neural networks (Gunasekar et al., 2018; Rahaman et al., 2019). Implicit rank regularization stands out among the diverse aspects of implicit regularization, which demonstrates that a neural network minimizes its rank implicitly during training (Arora et al., 2019; Gissin et al., 2019). Further research has corroborated that neural networks pursue a greedy low-rank learning strategy under infinitesimal initialization (Razin et al., 2021; Jacot et al., 2022; Li et al., 2021). However, the practical advantages of such an approach remain unexplored, predominantly due to the challenges of deviating from the infinitesimal initialization assumption.

Low-rank training and other structured pruning methods aim to promote structured sparsity within neural networks (NNs) throughout the training process, enabling substantial computational acceleration (You et al., 2022; Dao et al., 2022). The low-rank training technique has proven effective for training low-rank neural networks from scratch (Ioannou et al., 2016; Yang et al., 2020; Schotthöfer et al., 2022). Nonetheless, this method often necessitates extra hyperparameters, such as the rank of the factorization, which can be challenging to determine accurately, and thus it requires careful tuning.

Idelbayev & Carreira-Perpinan (2020) propose the LC compression method that explicitly integrates the learning of low-rank factors into the training process, despite its computational intensity. More recently, Wang et al. (2023) introduce Cuttlefish, a low-rank training method capable of automatically determining the factorization rank in the early stages of training. However, Cuttlefish requires a pre-set full-rank initialization and lacks a theoretical comprehension of its low-rank behavior, unlike our proposed InRank.

Moreover, low-rank training has been employed for fine-tuning large-scale pre-trained models (Hu et al., 2021), and for reducing communication overhead in distributed training (Vogels et al., 2019; Wang et al.). Li et al. (2023) adopt the low-rankness in cumulative weight updates to measure the intrinsic dimension of objective landscapes. The concept of incremental learning has been examined within the context of learning partial differential equations using neural networks, such as parameter expansion in the frequency domain (Zhao et al., 2022), and increasing the complexity of the underlying PDE problem (Huang & Alkhalifah, 2021).

B. Greedy Low-Rank Learning

In this section, we first introduce greedy low-rank learning (GLRL) and its practical limitations.

We have the following theorem that characterizes the implicit rank regularization behavior of gradient descent under infinitesimal initialization.

Theorem 2 (Greedy Low-Rank Learning, informal)**.** *If initialize $W^1, ..., W^L$ to be infinitesimal, then the product matrix A_θ follows a greedy low-rank learning trajectory, such that the gradient descent first searches over a rank-1 subspace of A_θ, and then greedily increases the rank by one whenever it fails to reach a global minimizer.*

Theorem 2 characterizes the trajectory of gradient descent, which performs a rank-constrained optimization and greedily relaxes the rank restriction by one whenever it fails to reach a global minimizer.

Inspired by this implicit low-rank trajectory, the greedy low-rank learning (GLRL) algorithm is proposed to capture this implicit behavior explicitly (Li et al., 2021). As shown in Algorithm 2, GLRL incrementally increases the rank of the weight matrices in a deep linear network and initializes the additional rows and columns based on the top singular vector of the current matrix derivative.

Although the GLRL algorithm provides a theoretical understanding of implicit rank regularization, it has some practical drawbacks. One notable limitation is the infinitesimally small initialization, which leads to slow convergence and makes it difficult to apply the algorithm in large-scale settings. In addition, GLRL is only applicable to linear networks as it highly relies on the product matrix A_θ. This makes it inapplicable to practical neural networks with nonlinear activation functions.

C. Details of Evaluation

We benchmark the effectiveness of our method mainly on Generative Pre-trained Transformer 2 (GPT-2), a model widely used in language tasks. In our experiment, we apply InRank to all the MLP layers in GPT-2 and assess the training of GPT-2 from scratch on the WikiText-103 dataset.

We fix the hyperparameters of InRank across all experiments and different models, including an initial rank of $r_0 = 1$, a buffer size of $b = 100$, and a threshold of $\alpha = 0.9$. We find both values r_0 and b are insensitive to the performance of InRank, and we will discuss the selection of the threshold α in the following section.

C.1. Automatic Rank Determination

A key finding from our evaluation is that InRank can automatically find the intrinsic rank of the model during training, facilitated by the automatic rank determination feature in cumulative weight updates. Figure 2 demonstrates that the rank identified by InRank aligns with the intrinsic rank discovered by costly sweeping across a wide range of ranks. This capability could potentially eliminate the need for the

Algorithm 2 Greedy Low-Rank Learning (GLRL)

Require: C is a convex cost, A_θ is the product matrix: $A_\theta = W^1...W^L$, and let $\epsilon, \eta, T > 0$

1: Compute the first singular vector of $\nabla C(0)$: $u, s, v \leftarrow \mathrm{SVD}_1(\nabla C(0))$
2: Initialize parameters and network width: $\theta \leftarrow (-\epsilon v^T, \epsilon, \dots, \epsilon u)$, $w \leftarrow 1$
3: **while** $C(A_\theta) < C_{min} + \epsilon$ **do**
4: Train width-w deep linear network for T steps using SGD with learning rate η
5: Compute the first singular vectors of $\nabla C(A_\theta)$: $u, s, v \leftarrow \mathrm{SVD}_1(\nabla C(A_\theta))$
6: Expand network width: $w \leftarrow w + 1$
7: Initialize additional parameters:

$$\theta \leftarrow \left(\begin{pmatrix} W^1 \\ -\epsilon v^T \end{pmatrix}, \begin{pmatrix} W^2 & 0 \\ 0 & \epsilon \end{pmatrix}, \dots, \begin{pmatrix} W^L & \epsilon u \end{pmatrix} \right)$$

8: **end while**

laborious and time-intensive process of tuning the rank hyperparameters for training low-rank networks.

C.2. InRank-Efficient

We aim to improve the efficiency of InRank. We find the rank increment mostly occurs during the early stages of training, remaining relatively stable thereafter. This observation suggests that the initial training phase can sufficiently infer the intrinsic rank of the model, corroborating the findings of previous work (Wang et al., 2023).

This motivates us to apply InRank only in the early stages of training to determine an appropriate rank for low-rank training, fixing its rank afterward. We denote this variant as InRank-Efficient. The conventional InRank is computationally expensive due to the $O(n^3)$ cost of the SVD operation for a matrix of size $n \times n$. On the other hand, InRank-Efficient reduces the computational burden by only applying InRank during the initial training stage. In the remaining evaluation on GPT-2, InRank-Efficient is only applied for the first epoch.

In the InRank-Efficient approach, once we determine the optimal rank r^* for UV using InRank, we parameterize W as a rank-r^* factorization of UV only, eliminating the need for representing a separated W_0. By removing W_0, we can reduce both memory usage and computational costs as it avoids additional matrix multiplication. Moreover, we can enhance efficiency by discarding the buffer size b once the optimal rank has been determined. The full algorithm is detailed in Algorithm C.2.

C.3. Selection of Threshold α

To determine the optimal configuration for InRank, we conduct evaluations using various values of threshold α. Table 2 demonstrates that the performance of each threshold value is consistent across different model sizes. Taking both prediction performance and efficiency into consideration, we have selected $\alpha = 0.9$ as the default value for all experiments. The stable choice of α ensures that InRank can automatically identify the optimal rank for new tasks and models without the need for extensive tuning, thereby minimizing the associated costs.

Algorithm 3 InRank-Efficient

1: **Rank determination:**
2: Run InRank (Algorithm 1) for T iterations and get the final rank r^l_T for each layer l
3: **Efficient training:**
4: Reparametrize $W^l_t = U^l_t V^l_t$ where $U^l_t \in \mathbb{R}^{p^l \times r^l_T}$ and $V^l_t \in \mathbb{R}^{r^l_T \times q^l}$
5: Perform standard training over fixed low-rank network until convergence

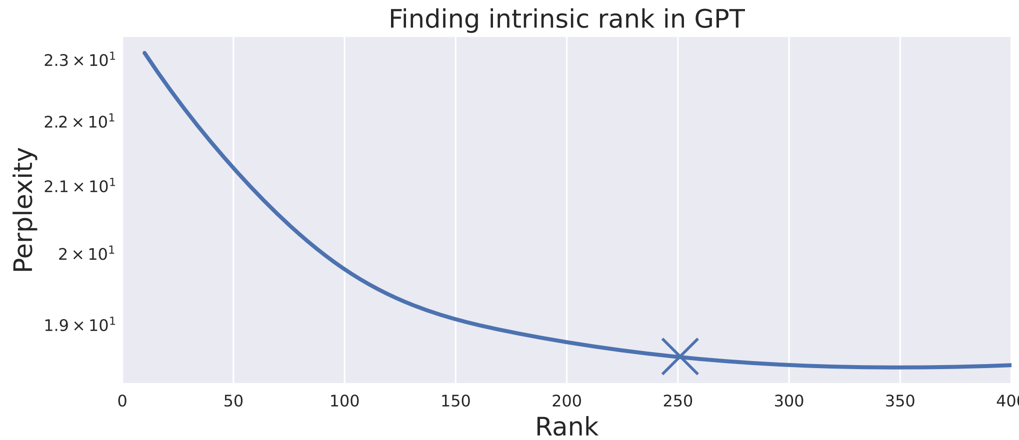

Figure 2. Identifying intrinsic rank in GPT-small on WikiText-103. The cross marker signifies the rank determined by InRank. The rank varies from 10 to 400.

Table 2. Varying threshold α in InRank-Efficient.

Threshold α	**GPT-small**					
	PPL	**Rank**	**Runtime**	**Memory**	**Parameters**	**FLOPs**
Baseline	18.5	768	24.5h	248Mb	124M	292G
0.8	19.4	152	20.2h	163Mb	81.8M	156G
0.85	19.1	193	21.9h	171Mb	85.6M	165G
0.9	18.9	254	22.2h	182Mb	91.2M	178G

Threshold α	**GPT-medium**					
	PPL	**Rank**	**Runtime**	**Memory**	**Parameters**	**FLOPs**
Baseline	19.5	1024	60.5h	709Mb	355M	828G
0.8	20.6	168	45.9h	389Mb	194M	363G
0.85	20.2	213	48.1h	411Mb	205M	389G
0.9	19.9	286	48.6h	447Mb	223M	428G

D. Proof

In this section, we present the proof of our main analysis, as shown in Theorem 1.

As introduced in the main text, we analyze the learning trajectory of a 3-layer linear network where $y = W^2 W^1 x$, $W^1 \in \mathbb{R}^{N_h \times N_x}$ and $W^2 \in \mathbb{R}^{N_y \times N_h}$ are the weight matrices of the first and second layers, respectively, and $N_h < N_x, N_y$.

We assume the inputs are orthogonal, i.e., $x_i^T x_j = 0$ for $i \neq j$. In this case, the continuous gradient flow follows the following differential equations:

$$\frac{\partial}{\partial t} W^1 = W^{2^T} \left(\Sigma^{yx} - W^2 W^1 \Sigma^{xx} \right), \quad \frac{\partial}{\partial t} W^2 = \left(\Sigma^{yx} - W^2 W^1 \Sigma^{xx} \right) W^{21^T}. \tag{8}$$

Since the inputs are orthogonal $\Sigma^{xx} = I$, the input-output correlation matrix Σ^{yx} contains all information we need to learn the network. We decompose Σ^{yx} using SVD as follows:

$$\Sigma^{yx} = U^{yy} S^{yx} V^{xx^T} = \sum_{\alpha=1}^{N_x} s_\alpha u_\alpha v_\alpha^T. \tag{9}$$

Learning the direction and strength of each mode α is crucial to interpolate the input-output correlation matrix Σ^{yx}.

To analyze the evolution of each mode independently, we let a^α be the α^{th} column of $\bar{W}^1$, and let $b^{\alpha T}$ be the α^{th} row of $\bar{W}^2$, where $W^1 = \bar{W}^1 V^{xx^T}, W^2 = U^{yy} \bar{W}^2$. Based on Equation 8 we can characterize the evolution of each mode using a^α and b^α:

$$\frac{\partial}{\partial t} a^\alpha = \left(s_\alpha - a^\alpha \cdot b^\alpha \right) b^\alpha - \sum_{\gamma \neq \alpha} b^\gamma \left(a^\alpha \cdot b^\gamma \right), \quad \frac{\partial}{\partial t} b^\alpha = \left(s_\alpha - a^\alpha \cdot b^\alpha \right) a^\alpha - \sum_{\gamma \neq \alpha} a^\gamma \left(b^\alpha \cdot a^\gamma \right). \tag{10}$$

For both $\frac{\partial}{\partial t}a^\alpha$ and $\frac{\partial}{\partial t}b^\alpha$, the first term characterizes the cooperative learning of the strength s_α using the a^α and b^α. The second term characterizes the competitive learning of the direction a^α and b^α given the distraction from other directions a^γ and b^γ.

It is difficult to solve Equation 10 given arbitrary weight initialization due to complex competitive interaction between modes. Therefore, we assume the weight initialization follows $W_0^2 = U^{yy}M^2O^T, W_0^1 = OM^1V^{xx^T}$, where M^2, M^1 are diagonal matrices, and O is an arbitrary orthogonal matrix. $W_0^2W_0^1$ ensures that each mode α is learned independently right from the beginning of training, enabling us to analyze the learning trajectory of each mode separately. a^α and b^α will remain parallel to a certain direction r^α throughout the learning process, and we can rewrite Equation 10 as follows:

$$\frac{\partial}{\partial t}a = b(s - ab), \quad \frac{\partial}{\partial t}b = a(s - ab), \tag{11}$$

where we let $a = a^\alpha \cdot r^\alpha, b = b^\alpha \cdot r^\alpha$, and $s = s^\alpha$. By further assuming $a = b$ and $u = ab$, we obtain:

$$\frac{\partial}{\partial t}u = 2u(s - u). \tag{12}$$

Integrate the above equation to obtain:

$$t = \tau \int_{u_0}^{u_f} \frac{du}{2u(s-u)} = \frac{\tau}{2s} \ln \frac{u_f\,(s - u_0)}{u_0\,(s - u_f)}, \tag{13}$$

where u_0 is the initial value determined by M^2 and M^1, u_f is the target value of strength, and τ is a constant. t is the time it takes for u to travel from u_0 to u_f.

As we analyze the difference of the product matrix $D_t = A_t - A_0$, it is equivalent to analyzing the residual of each mode: $u_t - u_0$. To analyze the entire evolution ($u_f \approx s$) of u over time, we yield the following equation:

$$u_f(t) = \frac{se^{2st/\tau}}{e^{2st/\tau} - 1 + s/u_0} - u_0. \tag{14}$$

E. Clarification on Greedy Low-Rank Learning

In this section, we additional clarification on the greedy low-rank learning hypothesis, which is presented in Theorem 2.

Several works have demonstrated the greedy low-rank learning behavior under various settings and assumptions. Li et al. (2021) prove it under matrix factorization setting for deep linear network by analyzing the asymptotic behavior of gradient flow under infinitesimal initialization. Jacot et al. (2022) also demonstrate the saddle-to-saddle learning behavior for deep linear networks, although they prove the rank-one case only. Razin et al. (2021) further extend the discussion to the setting of tensor factorization.

A formal description of Theorem 2 is given below:

Theorem 3. *Let $\tilde{W}_r$ be the r-th critical point of a rank-r subspace of W, and let $\tilde{W}_0 = 0$ be the saddle point at zero. From an infinitesimal initialization ($W_0 \approx \tilde{W}_0$), the gradient flow $G(W)$ first visits the critical point $\tilde{W}_1$. If $\tilde{W}_1$ is not a minimizer, $G(W)$ will expand the searching space to a rank-2 subspace and converge to the critical point $\tilde{W}_2$. If $\tilde{W}_2$ is also not a minimizer, this process continues until $G(W)$ reaches $\tilde{W}_{r^*}$ in a rank-r^* subspace that minimizes the objective function, provided that $r^* < \mathrm{rank}(W)$.*

The theorem implies the greedy low-rank learning trajectory, such that the gradient descent first searches over a rank-1 subspace of A_θ, and then greedily increases the rank by one whenever it fails to reach the minimizer.

Proving this requires the analysis of the limiting flow $G_{r\to r+1}(W)$, which is the gradient flow between two critical points $\tilde{W}_r$ and $\tilde{W}_{r+1}$. Theorem 3 holds by showing that the flows $G_{0\to 1}(W)$, $G_{1\to 2}(W)$, ..., $G_{r^*-1\to r^*}(W)$ all exist during learning, which is a general proving direction adopted by recent works. The details of the proof can be found in Li et al. (2021); Jacot et al. (2022).

F. Low-Rank Learning in Practice

In this section, we provide additional results demonstrating that the cumulative weight updates follow the low-rank learning trajectory over a broad range of network architectures and learning algorithms.

F.1. Evaluation on Theorem 1

To validate this, we carry out an empirical simulation using different sets of u_0 and s. As illustrated in Figure 3, under various scales of initialization, the evolution of $u_f(t)$ consistently adheres to the low-rank learning trajectory. We note that analyzing weights W^2W^1 directly under infinitesimal initialization in Li et al. (2021) can be viewed as a special case of analyzing D_t here.

F.2. Low-Rank learning under different initialization

As shown in Figure 4, cumulative weight updates D_t follow the greedy low-rank learning trajectory even under regular initializations, including Orthogonal, ZerO, and Kaiming methods (Saxe et al., 2014; He et al., 2015; Zhao et al., 2021).

F.3. Low-Rank learning under different architectures

We further verify our theory on different architectures, including LSTM and Transformer.

F.3.1. LSTM

As shown in Figure 5, the cumulative weight updates D_t of LSTM follow the low-rank learning trajectory.

F.3.2. Transformer

As shown in Figure 6, the cumulative weight updates D_t of Transformer follow the low-rank learning trajectory.

F.4. Low-Rank learning under different learning algorithms

We further verify our theory on different learning algorithms, including SGD and Adam. As shown in Figure 7, the cumulative weight updates D_t of MLP follow the low-rank learning trajectory under both SGD and Adam.

G. Rank Evolution during Training

We present the rank evolution in various MLP layers when applying InRank on GPT-small model. As shown in Figure 8, we visualize the rank evolution over the first 5% of the total training iterations. The figure indicates that the increment of rank mostly happens in the early stage of training.

H. Repeated Experiment on different GPT models

We report the evaluation results of InRank and InRank-Efficient on different sizes of GPT models in Table 3. All experiments are repeated 3 times. We also report testing perplexity instead of validation perplexity.

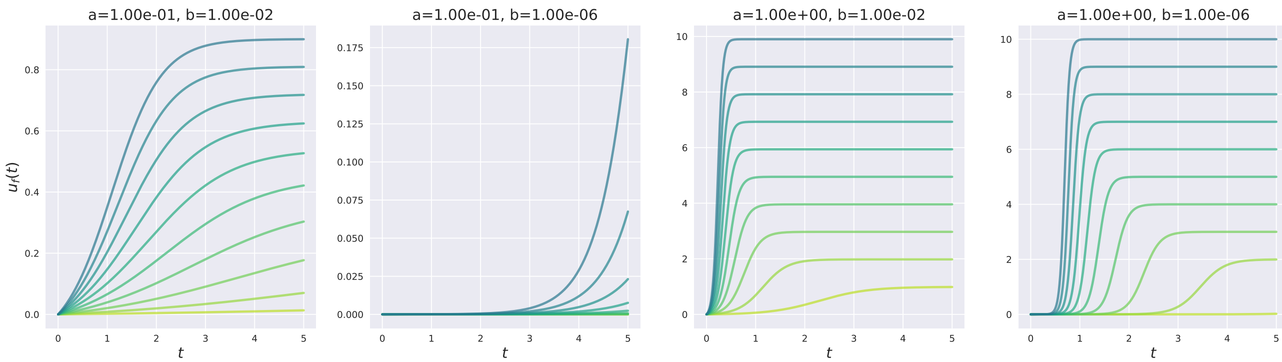

Figure 3. $u_f(t)$ follows low-rank learning trajectory regardless of s and u_0. We generate a set of s given $s_i = a \times i, i = 1, ..., 10$ while varying a from 0.1 to 1.0. We also generate a set of u_0 given $u_0 \sim \mathcal{N}(0, b^2)$. Darker colors indicate singular vectors with higher strengths.

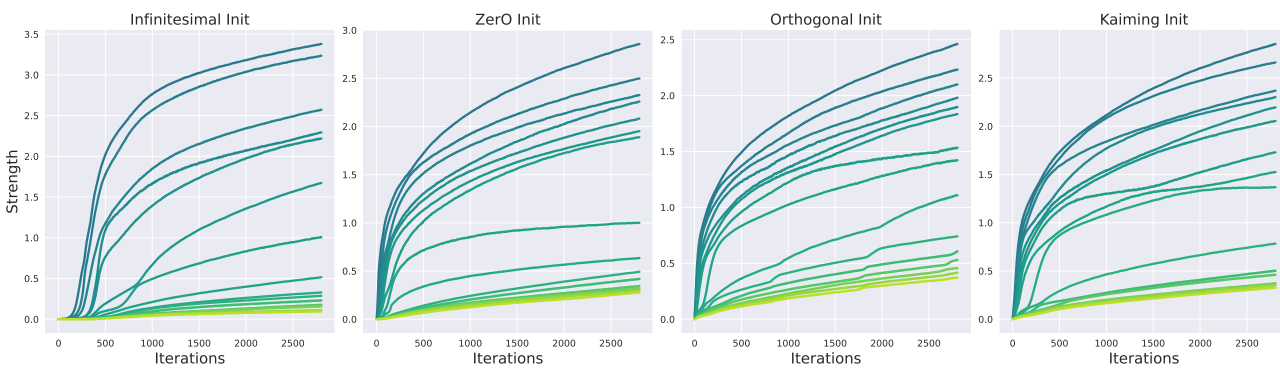

Figure 4. The evolutions of top 20 singular vectors of cumulative weight updates D_t over training under different initializations. They are evaluated on the training of a 3-layer perceptron on Fashion MNIST. Darker colors indicate singular vectors with higher strengths.

Table 3. Evaluating InRank across different sizes of GPT models. All experiments are repeated 3 times.

Model	**Method**	**Test Perplexity**
GPT-small	InRank	19.96 ± 0.10
	InRank-Efficient	20.07 ± 0.14
GPT-medium	InRank	22.14 ± 0.07
	InRank-Efficient	21.23 ± 0.05
GPT-large	InRank	22.63 ± 0.09
	InRank-Efficient	21.49 ± 0.07

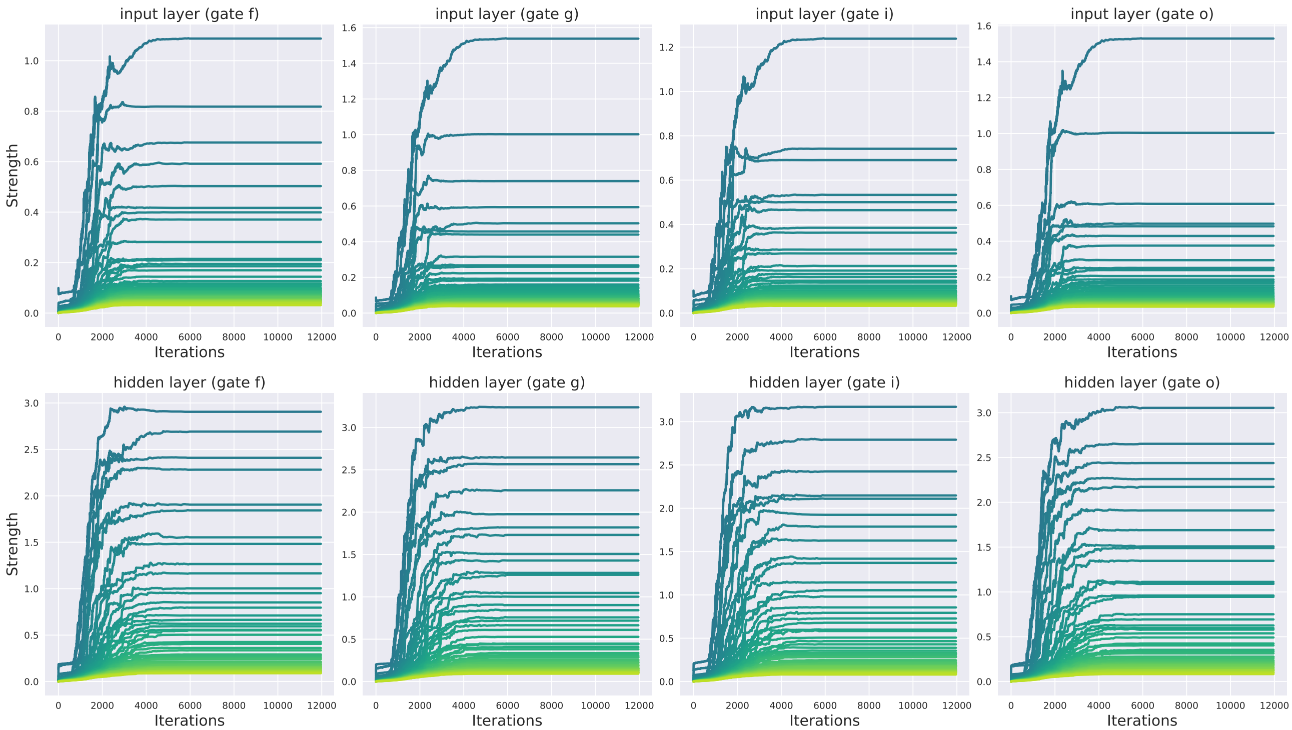

Figure 5. The evolutions of all singular vectors of cumulative weight updates D_t over the training of LSTM. The top row shows the input-to-hidden weight matrix W_{ih}, and the bottom row shows the hidden-to-hidden weight matrix W_{hh}. Darker colors indicate singular vectors with higher strengths.

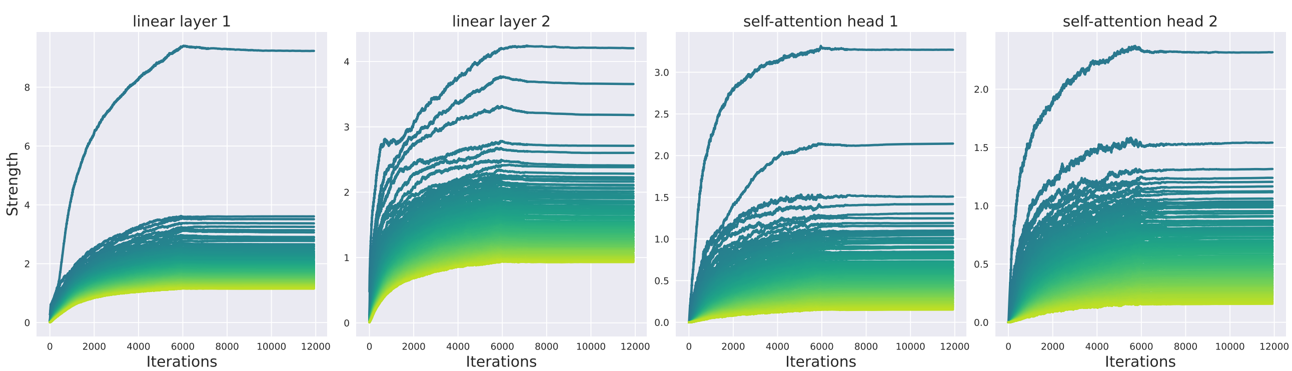

Figure 6. The evolutions of all singular vectors of cumulative weight updates D_t over the training of Transformer. In a single layer, we visualize two weight matrices in MLP and two K matrices in self-attention. Darker colors indicate singular vectors with higher strengths.

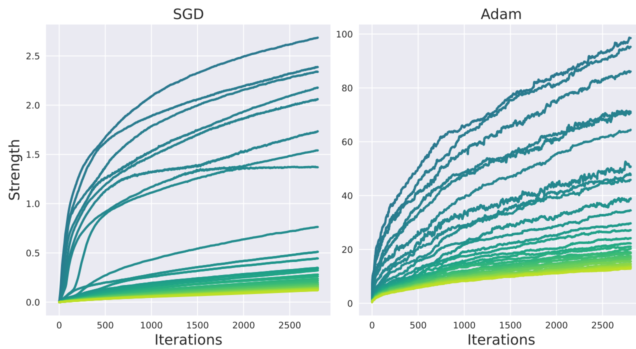

Figure 7. The evolutions of all singular vectors of cumulative weight updates D_t over the training of MLP using SGD and Adam optimizers. Darker colors indicate singular vectors with higher strengths.

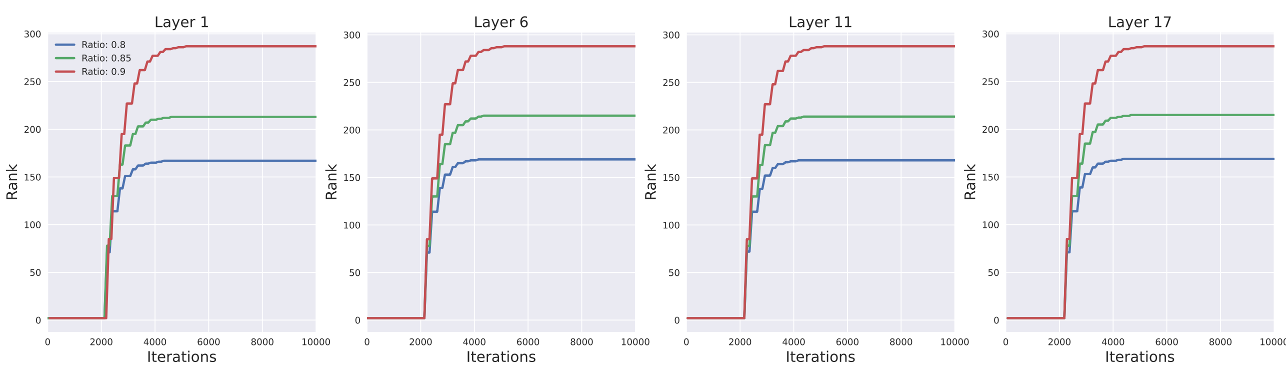

Figure 8. The rank evolution in various MLP layers when applying InRank on GPT-small model.