Rank Minimization, Alignment and Weight Decay in Neural Networks

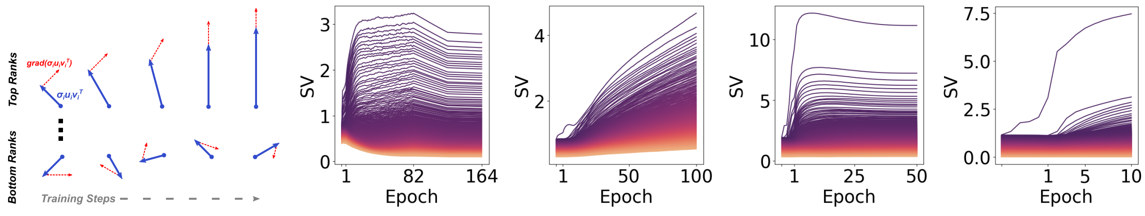
David Yunis DYUNIS@TTIC.EDU
Kumar Kshitij Patel KKPATEL@TTIC.EDU
Pedro Henrique Pamplona Savarese SAVARESE@TTIC.EDU
Gal Vardi GALVARDI@TTIC.EDU
Karen Livescu KLIVESCU@TTIC.EDU
Matthew Walter MWALTER@TTIC.EDU
Toyota Technological Institute at Chicago, Chicago, IL, USA

Samuel Wheeler SWHEELER@ANL.GOV
Argonne National Laboratory, Lemont, IL, USA

Michael Maire MMAIRE@UCHICAGO.EDU
University of Chicago, Chicago, IL, USA

Abstract

We empirically study the evolution of the singular values and vectors of neural network weights across a wide variety of practical architectures and domains, including CNNs for image classification, LSTMs for speech recognition, and Transformers for language modeling. Across these settings, we observe that (i) large singular values grow much faster, decreasing the effective ranks of weight matrices, (ii) this growth occurs despite weak alignment between neighboring layers' singular vectors, a common assumption in prior theoretical work, and (iii) weight decay promotes both rank minimization, and neighboring layer alignment. Since these architectures are far from idealized linear neural networks, our observations extend the predictions of existing theory to more practical settings.



(a) Schematic (b) VGG (c) UNet (d) LSTM (e) Transformer

Figure 1: Singular value evolution for a single weight matrix in the middle of each network across different architectures. Each line tracks the evolution of a single singular value over time, whereas darker lines indicate larger singular values. Notice the disparate dynamics, where larger values grow at a disproportionate rate over the course of training. Schematic (a) provides a possible explanation for this behavior.

1. Introduction

After many years of dizzying empirical advancements, a large number of fundamental questions about neural networks remain unresolved. For instance, despite extensive research, we still lack a complete understanding of the implicit biases of neural networks trained via stochastic optimization. Even basic questions like the role of weight decay [10, 18, 44], have only partial answers [1, 36, 41].

A recent line of theoretical work [3, 23, 32, 41] reveals that the evolution of singular values in deep linear models is quite structured and results in rank minimization with depth. One standard assumption in these works is that of "balanced initialization", which requires that at initialization for two consecutive matrices W_i and W_{i+1} in a product matrix $\prod_j W_j$, we have $W_{i+1}^\top W_{i+1} = W_i W_i^\top$. When substituting the SVDs of these matrices and simplifying through orthogonality, this results in the condition $V_{i+1}\Sigma_{i+1}^2 V_{i+1}^\top = U_i \Sigma_i^2 U_i^\top$ where U_i and V_{i+1} are orthogonal matrices. From this point, one can derive that, due to the alignment, the product of the diagonals will evolve in a closed-form fashion, which results in a rank-minimizing behavior with depth [3]. Similar results are derived in tensor products and other structured settings [32, 40].

If such behavior were true for general neural networks, it might have wide-ranging implications from generalization to compression. Inspired by these works, we empirically investigate the dynamics of singular values and singular vectors of neural network weight matrices without restrictive assumptions made by contemporary methods like small and orthogonal initialization, small learning rates, bounded norms, or linearity. To improve generality, we examine these behaviors across diverse architectures and tasks. Our findings reveal effective rank minimization in weight matrices, where larger singular values grow more rapidly (Fig. 1), which occurs despite very weak alignment of the singular vectors in consecutive layers.

Our specific contributions are:

- In Section 3, we provide an empirical overview of the training dynamics of neural network layers through the lens of the SVD. We demonstrate effective rank minimization across various practical neural networks in complex settings. We also see only a very weak alignment of singular vectors in consecutive layers, which disagrees with a common theoretical assumption.
- In Section 4, we show that weight decay promotes rank minimization and alignment in consecutive layers, extending theoretical work on the topic. Small [8] to large [4] amounts of weight decay are commonly used to improve generalization, so this suggests a connection between rank and generalization.

2. Methodology

In all our experiments, we aim to study reasonably sized neural networks across a variety of tasks. We choose models and tasks to represent current applications. In particular, we select image classification with CNNs (VGG-16 [33]) on CIFAR10 [17], image generation through diffusion with UNets [30] on MNIST [19], speech recognition with LSTMs [14] on LibriSpeech [25], and language modeling with Transformers [37] on Wikitext-103 [22]. We primarily take hyperparameters from existing settings in the literature, making small modifications for simplicity. Thus, we intend that any correlations between settings will be a reflection of common practice as opposed to introduced bias on our part.

The bulk of the evidence presented comes from computing singular value decompositions (SVDs) of weight matrices in models. As there are many matrices in the models we study, for conciseness of presentation, we provide plots of individual layers' matrix parameters and statistics summarizing

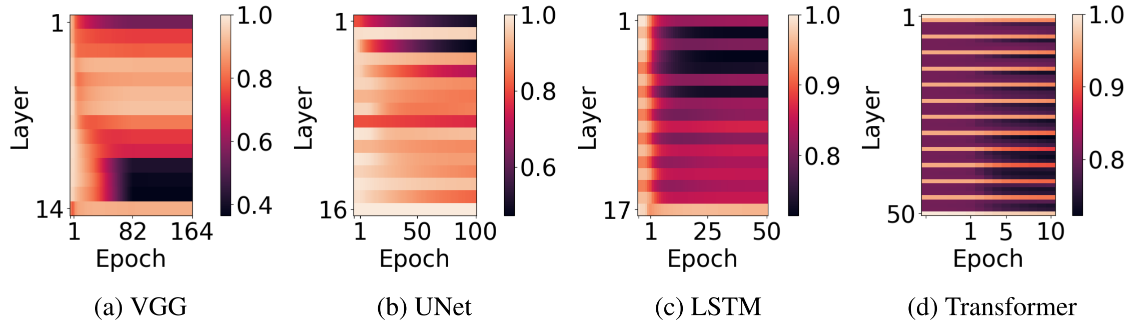


Figure 2: Normalized effective rank (Eqn. 1) evolution visualized in color for different matrices across architectures and time. As we move down the y-axis, the depth of the parameters in the model increases, while the x-axis tracks training time. Notice decreasing effective rank across nearly all parameters, though the magnitude differs across layers. Block-like patterns in the VGG case are due to different channel dimensions. Banding in the UNet, LSTM, and Transformer are due to differences between convolutional and linear layers, input and hidden parameters, and attention and fully connected layers, respectively.

behavior across layers. Please see Appendix C for more experiment details, including hyperparameters. We will release code for all experiments.

3. Spectral Dynamics

3.1. Effective Rank Minimization

To track rank minimization, we compute the (normalized) effective rank of a matrix W [31] with rank R as

$$\text{EffRank}(W) = -\sum_{i=1}^{R} \frac{\sigma_i}{\sum_j \sigma_j} \log \frac{\sigma_i}{\sum_j \sigma_j} \quad \text{and} \quad \text{NormEffRank}(W) = \frac{\text{EffRank}(W)}{R} \ , \qquad (1)$$

where σ_i are the singular values of matrix W and $\text{EffRank}(W)$ is the entropy of the normalized singular value distribution. As the probability mass concentrates, the effective rank decreases. We choose to plot $\text{NormEffRank}(W)$ to compare across layers and time in Figure 2, where we see that effective rank tends to decrease as training proceeds, regardless of the parameter or network. This echoes predictions made in simple models.

3.2. Alignment of Singular Vectors Between Layers

To quantify alignment between consecutive matrices in a neural network $W_i = \sum_{j=1}^{R} \sigma_j(t) u_j(t) v_j(t)^\top$, $W_{i+1} = \sum_{k=1}^{R} \sigma'_k(t) u'_k(t) v'_k(t)^\top$ at training time t, we compute for all $j, k \in [R]$,

$$A(t)_{jk} = |\langle u_j(t), v'_k(t) \rangle| \ , \qquad (2)$$

where the absolute value is taken to ignore sign flips in the SVD computation. We then plot the diagonal of this matrix $A(t)_{ii}\ \forall\ i \leq 100$ over time for a single pair of layers in Figure 3, as well as

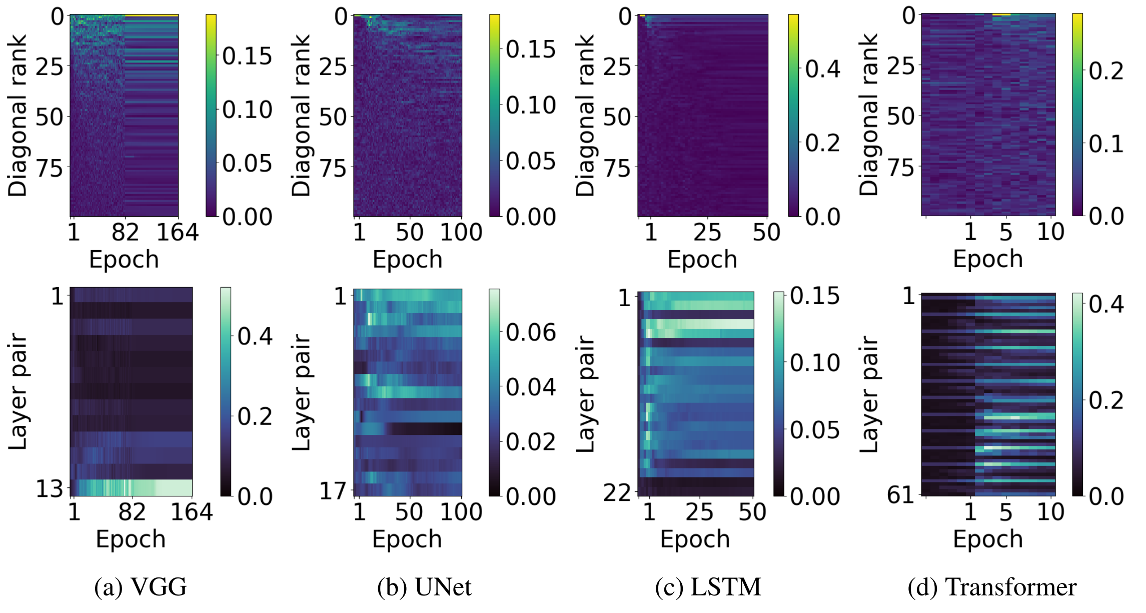


Figure 3: Neighboring layer alignment of singular vectors. **Top row:** The diagonal of the alignment matrix $A(t)_{ii}$ (Eqn. 2) vs. training time for a single pair of matrices in the middle of each model. We see a small amount of alignment in the top ranks between layers shortly after training begins, but this becomes more diffuse over time. **Bottom row:** Alignment metric (Eqn. 3) for pairs of matrices for depth vs. training time. It is hard to make out a global trend across models. However, the LSTM shows a weak signal around Epoch 1 when the initial alignment occurs, and the Transformer case has a banding pattern with depth due to alignment between the query and key matrices that have no nonlinearity in between.

a scalar measure of alignment in the top ranks:

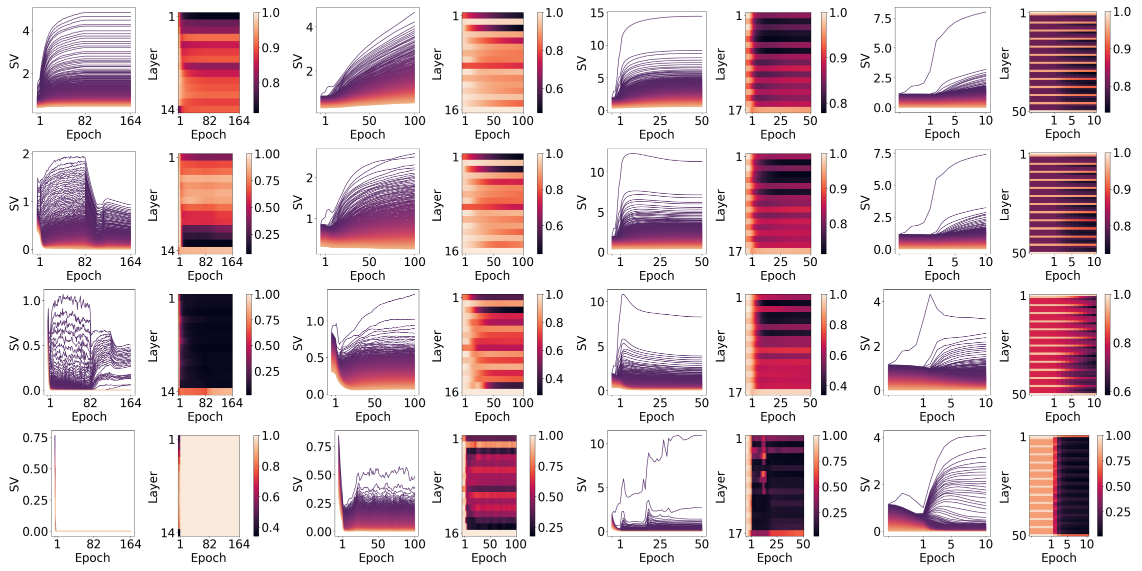
$$a(t) = \frac{1}{10} \sum_{i=1}^{10} A(t)_{ii} \quad . \tag{3}$$

For exact details on how this is computed for different architectures and layers that are more complex than the fully-connected case, please see Appendix C.

Figure 3 establishes that the theoretical assumption of balanced initialization [2, 32], which assumes aligned SVDs between weight matrices, is not valid at the beginning of training. Nor does it appear that alignment is static, like the linear case discussed by Du et al. [7]. Still, there does seem to be a somewhat weak signal in the top ranks.

4. The Effect of Weight Decay

In light of the previously observed evolution of singular values, we investigate a proposed effect of weight decay. Though weight decay explicitly penalizes the norm of weights, there is empirical evidence discarding the connection between norm and generalization for neural networks [1, 29].



(a) VGG (b) UNet (c) LSTM (d) Transformer

Figure 4: SV evolution for a single matrix and normalized effective rank (Eqn. 1) across matrices over time, where the rows use differing amounts of weight decay. From top to bottom, for VGG we use coefficients $\{0, 0.001, 0.01, 0.1\}$, while for other networks we use coefficients $\{0, 0.1, 1, 10\}$. Higher weight decay coefficients promote more aggressive rank minimization. VGG uses SGD and momentum, while the rest use AdamW [21].

Some theoretical [6, 9, 24, 29, 35, 40, 42] and empirical works [6, 9] propose a connection with the rank of matrices in constrained settings. Still, a comprehensive connection to larger empirical networks has not yet been demonstrated.

Notice in its simplest form that weight decay asks for $\arg\min_W \mathcal{L}(W) + \lambda\|W\|_F^2$, where $\|W\|_F^2 = \sum_{i=1}^R \sigma_i^2$ with singular values σ_i of weight matrix W with rank R. We saw that larger singular values of neural networks grow faster (Fig. 1) and that the top singular vectors are much more useful for minimizing task loss than the bottom ones (Fig. 2). Thus, with minor weight decay regularization, a straightforward solution for the network may be to minimize the rank of a given weight matrix while preserving the top singular values to minimize $\mathcal{L}(W)$.

Figure 4 shows that adding weight decay indeed produces this exact behavior, while too much weight decay leads to complete norm collapse in the VGG case. Even more surprisingly, in Figure 5 (in Appendix A), large weight decay promotes a tighter alignment in the top singular vectors of consecutive layers. This behavior is quite reminiscent of balancedness [2, 3, 7], though these networks have nonlinearities and much more complex structures. We provide additional evidence in Appendix C, Figure 6 that the solutions with very high weight decay are still performant, even though they are much lower rank. We further investigate the connection between rank and generalization in Appendix B, where we find memorizing networks have higher rank parameters.

5. Discussion

We saw that, regardless of architecture or task, neural network weights diminish in rank over training. We also provide firmer evidence for the effect of weight decay in diminishing the rank of these matrices and aligning intermediate layers. These observations extend existing theory on the subject, and may provide pointers as to how to understand neural network generalization more precisely.

References

[1] Maksym Andriushchenko, Francesco D'Angelo, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning? *arXiv preprint arXiv:2310.04415*, 2023.

[2] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 244–253, 2018.

[3] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[4] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2397–2430, 2023.

[5] Lukas Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

[6] Enric Boix-Adserà, Etai Littwin, Emmanuel Abbe, Samy Bengio, and Joshua M Susskind. Transformers learn through gradual rank increase. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[7] Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[8] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3259–3269, 2020.

[9] Tomer Galanti, Zachary S Siegel, Aparna Gupte, and Tomaso Poggio. SGD and weight decay provably induce a low-rank bias in neural networks. *arXiv preprint arXiv:2206.05794*, 2022.

[10] Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1988.

[11] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

[15] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.

[18] Anders Krogh and John Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1991.

[19] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[20] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. Datasets: A community library for natural language processing. *arXiv preprint arXiv:2109.02846*, 2021.

[21] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018.

[22] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[23] Paolo Milanesi, Hachem Kadri, Stéphane Ayache, and Thierry Artières. Implicit regularization in deep tensor factorization. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[24] Greg Ongie and Rebecca Willett. The role of linear layers in nonlinear interpolating networks. *arXiv preprint arXiv:2202.00856*, 2022.

[25] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[27] Brenda Praggastis, Davis Brown, Carlos Ortiz Marrero, Emilie Purvine, Madelyn Shapiro, and Bei Wang. The svd of convolutional weights: a cnn interpretability framework. *arXiv preprint arXiv:2208.06894*, 2022.

[28] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[29] Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 21174–21187, 2020.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241, 2015.

[31] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 606–610, 2007.

[32] A Saxe, J McClelland, and S Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[35] Nadav Timor, Gal Vardi, and Ohad Shamir. Implicit regularization towards rank minimization in relu networks. In *Proceedings of the International Conference on Algorithmic Learning Theory*, pages 1429–1459, 2023.

[36] Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[38] Binxu Wang and John Vastola. Ml from scratch: Stable diffusion, day 2, 2022. URL https://colab.research.google.com/drive/1Y5wr91g5jmpCDiX-RLfWL1eSBWoSuLqO?usp=sharing#scrollTo=9is-DXZYwIIi.

[39] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

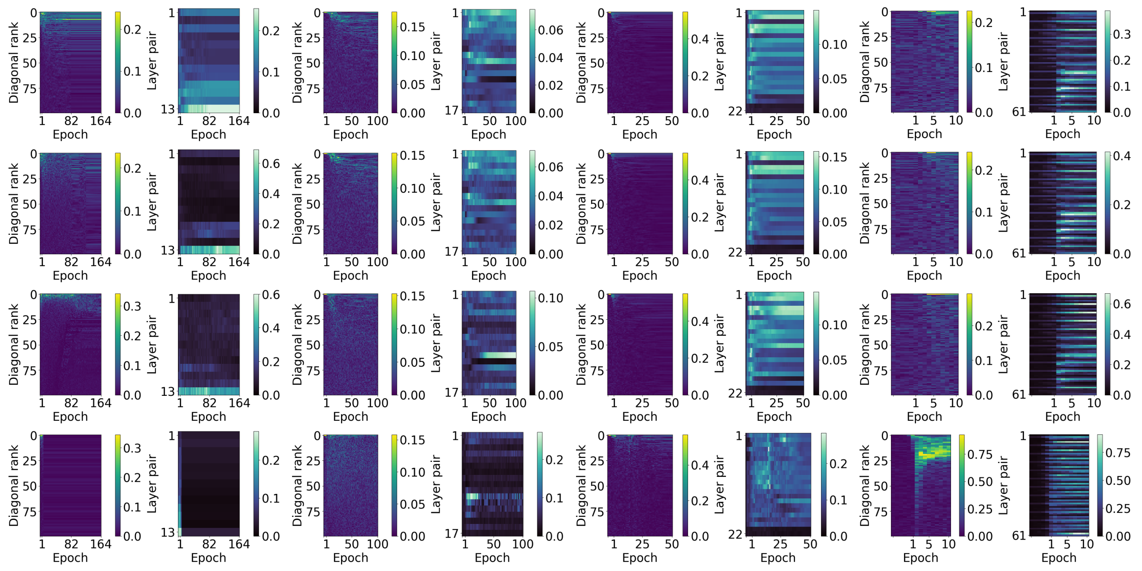
[40] Can Yaras, Peng Wang, Wei Hu, Zhihui Zhu, Laura Balzano, and Qing Qu. Invariant low-dimensional subspaces in gradient descent for learning deep matrix factorizations. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023.

[41] Can Yaras, Peng Wang, Wei Hu, Zhihui Zhu, Laura Balzano, and Qing Qu. The law of parsimony in gradient descent for learning deep linear networks. *arXiv preprint arXiv:2306.01154*, 2023.

[42] Emanuele Zangrando, Piero Deidda, Simone Brugiapaglia, Nicola Guglielmi, and Francesco Tudisco. Neural rank collapse: Weight decay and small within-class variability yield low-rank bias. *arXiv preprint arXiv:2402.03991*, 2024.

[43] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[44] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*, 2018.



(a) VGG (b) UNet (c) LSTM (d) Transformer

Figure 5: Diagonal of alignment for a single pair over time (Eqn. 2) and alignment metric across pairs of matrices over time (Eqn. 3) where the y-axis represents depth. From top to bottom, for VGG we use coefficients $\{0, 0.001, 0.01, 0.1\}$, while for other networks we use coefficients $\{0, 0.1, 1, 10\}$. We see that the alignment magnitude is much higher with higher weight decay, and in particular, the Transformer has the strongest alignment even when nonlinearities separate the MLP layers.

Acknowledgements

DY would like to thank the following people (in alphabetical order by surname) for helpful discussions throughout the process of this paper: Sudarshan Babu, Mikhail Belkin, Sam Buchanan, Jiading Fang, Jonathan Frankle, Wei Hu, Anmol Kabra, Tengyu Ma, David McAllester, Rob Nowak, Shane Settle, Shubham Toshniwal, Haochen Wang, Takuma Yoneda, and Xiao Zhang. DY would also like to thank Adam Bohlander for his work in maintaining the TTIC cluster, which made the many experiments in this paper possible. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1754881. KKP was supported through the NSF TRIPOD Institute on Data, Economics, Algorithms and Learning (IDEAL) and other awards from DARPA and NSF.

Appendix A. Additional Plots for Weight Decay Experiments

All tasks are trained in exactly the same fashion as mentioned, but with weight decay coefficient of 10. To additionally validate the claim that weight decay makes the parameters lower-rank, we save intermediate checkpoints throughout training and evaluate the original model, the model that results from approximating all parameters except the last linear layer by the top half of the SVD, and the same model resulting from the bottom half of the SVD. We see in Figure 6 that, like one

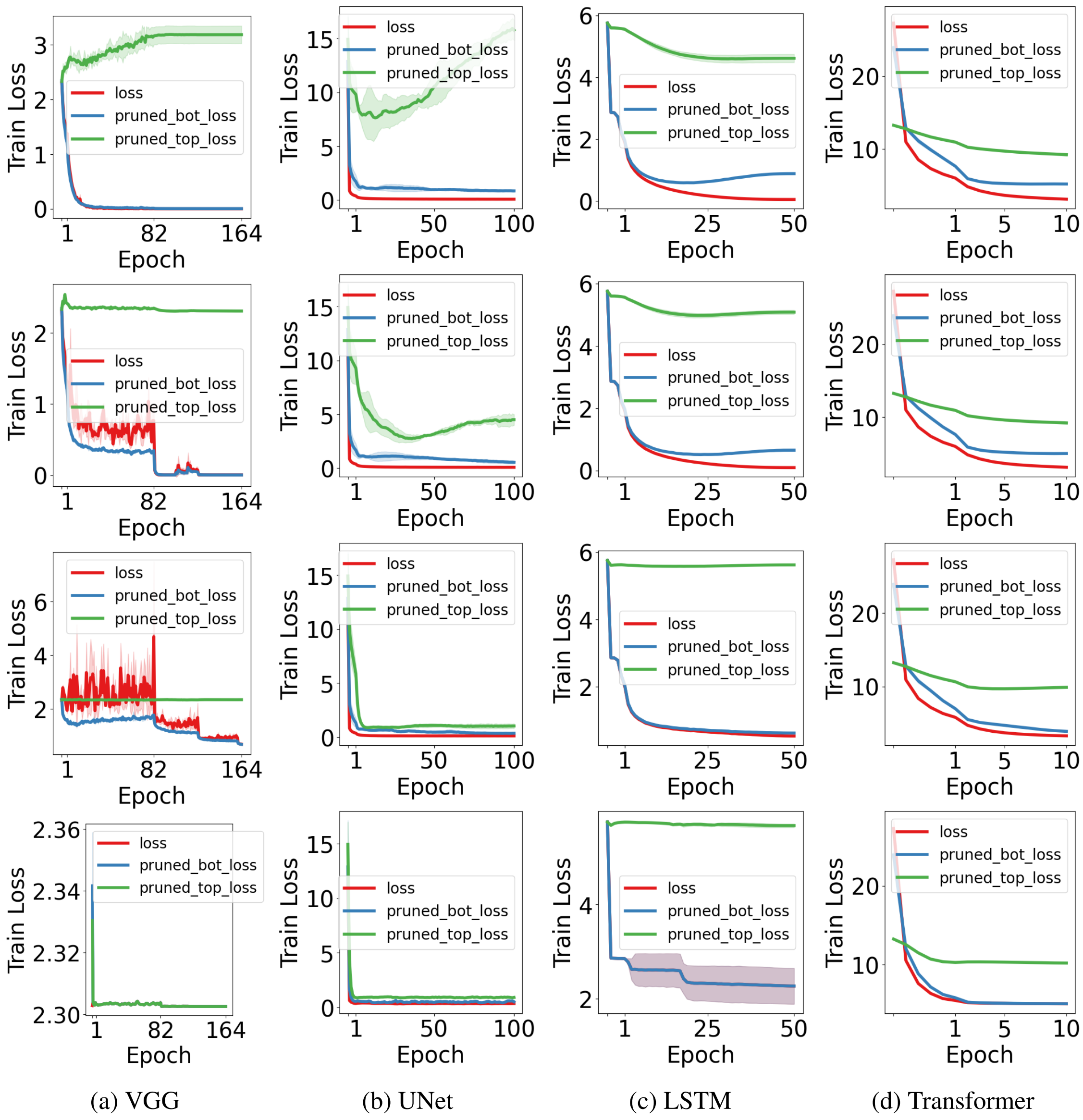


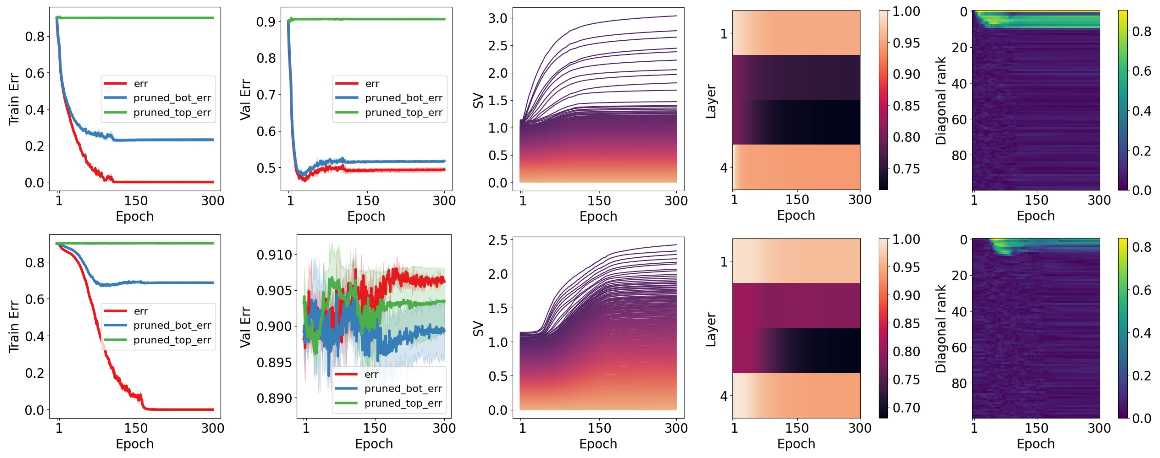
Figure 6: Training loss over time, where the rows use differing amounts of weight decay. From top to bottom, for VGG we use coefficients $\{0, 0.001, 0.01, 0.1\}$, while for other networks we use coefficients $\{0, 0.1, 1, 10\}$. We see that it is still possible to achieve low training loss under high weight decay, and as we increase the amount of weight decay, the gap between pruned and unpruned parameters closes, lending support to the idea that the parameters become lower rank.

might expect from L^2 space, the top half of the SVD is a good approximation to the original model, and with high weight decay the approximation is very tight. In addition, even with very high weight decay the performance does not break down besides the VGG case.

Appendix B. Spectral Dynamics with Random Labels

To investigate the connection between rank and generalization, we revisit the classic random label memorization experiments of Zhang et al. [43] as a testing ground.

We train a 4-layer ReLU MLP to fit random or true labels on CIFAR10 [17] similar to Zhang et al. [43]. We use SGD with momentum of 0.9 and constant learning rate of 0.001, and train for 300 epochs to see the entire trend of training. The major difference to the setting of Zhang et al. [43] is the use of a constant learning rate, as their use of a learning rate schedule might conflate the results.



(a) Train Err. (b) Val. Err. (c) SVs (d) Eff. Rank (e) Alignment

Figure 7: **Top row:** results with true labels. **Bottom row:** results with random labels. We see that the middle layers have a lower effective rank when using true labels and that alignment in the middle layers persists throughout training, unlike in the random label case.

Surprisingly, we see in Figure 7 that even without weight decay, with true labels, inner layers align, while with random labels, this alignment occurs and then disappears with more training. This is particularly intriguing as there are nonlinearities that could theoretically separate the network from the linear case, and yet quite strong alignment occurs despite that. We also see that the middle layer of the network trained on true labels has a lower effective rank, which may make sense as the data with true labels likely shares a common structure among classes. This further suggests that viewing generalization through the lens of rank may be fruitful.

Appendix C. Experimental Details

We use 3 random seeds for all experiments and average all plots over those 3. This is relatively small, but error bars tend to be very tight, and due to the high volume of runs required for this work, we lack the resources to run much more.

To compute alignment, we consider only pairs of layers that directly feed into each other and ignore the influence of residual connections to reduce the number of comparisons. Specifics on individual architectures are given below.

C.1. Image Classification with VGG

We train a VGG-16 [33] on CIFAR-10 [17] for 164 epochs, following hyperparameters and learning rate schedule in [8], but without data augmentation. For the optimizer, we use SGD with a batch size of 128, initial learning rate of 0.1, and momentum of 0.9. We also decay the learning rate 3 times by a factor of 10 at epoch 82, epoch 120, and finally at epoch 160. We also use a minor amount of weight decay with a coefficient of 0.0001.

VGG-16 uses ReLU activations and batch normalization [16] and includes both convolutional and linear layers. For linear layers, we simply compute the SVD of the weight matrix. For convolutional layers, the parameters are typically stored as a 4D tensor of shape $(c_{\text{out}}, c_{\text{in}}, h, w)$ for the output channels, input channels, height and width of the filters respectively. As the filters compute a transformation from each position and input channel to an output channel, we compute the SVD of the flattened tensor $(c_{\text{out}}, c_{\text{in}} \cdot h \cdot w)$, which maps all inputs to outputs, similar to Praggastis et al. [27]. This is not the SVD of the entire transformation of the feature map to the next feature map but rather the transformation from a set of adjacent positions to a particular position in the next layer.

In order to compute the alignment of bases between consecutive convolutional layers, $V_{i+1}^{\top} U_i$, we need to match the dimensionality between U_i and V_{i+1}. For convolutional layers, we are presented with a question as to how to handle the spatial dimensions h and w as naively, the input dimension of the next layer will be a factor of $h \cdot w$ larger dimension. We experimented with multiple cases, including aligning at each spatial position individually or averaging over the alignment at all spatial positions. We eventually settled on aligning the output of one layer to the center spatial input of the next layer. For a 3x3 convolution mapping to a following 3x3 convolution, we compute the alignment only for the position (1,1) of the next layer. This seemed reasonable to us as, on average, the edges of the filters showed poorer alignment overall.

In the single-layer plots we visualize a convolutional layer in the middle of the model and the alignment of two consecutive convolutional layers.

C.2. Image Generation with UNets

We train a UNet [30] diffusion model [12, 34] on MNIST [19] generation. We take model design and hyperparameters from [38]. In particular, we use a 4-layer residual UNet and train with AdamW [21] with batch size 128 and a learning rate 0.0003 for 100 epochs. This model uses swish [28] activations and a combination of linear and convolutional, as well as transposed convolutional layers.

Computing SVDs and alignment is similar to the image classification case described above, except in the case of the transposed convolutions where an extra transpose of dimensions is needed as parameters are stored with the shape $(c_{\text{in}}, c_{\text{out}}, h, w)$.

In single-layer plots we visualize a convolutional layer in the middle of the model and the alignment of two consecutive convolutional layers.

C.3. Speech Recognition with LSTMs

We train a bidirectional LSTM [13] for automatic speech recognition on LibriSpeech [25]. We tune for a simple and well-performing hyperparameter setting. We use AdamW [21] with batch size 32, learning rate 0.0003, and weight decay 0.1 for 50 epochs. We also use a cosine annealing learning rate schedule from 1 to 0 over the entire 50 epochs.

The LSTM only has matrix parameters and biases, so it is straightforward to compute SVDs of the matrices. In the case of alignment, we make a number of connections: first, down depth for the input parameters, then connecting the previous input parameter to the current hidden parameter in both directions, then connecting the previous hidden parameter to the current input parameter.

In the single-layer plots we visualize an input parameter in the middle of the model, and the alignment between two input parameters with depth.

C.4. Language Modeling with Transformers

We train a Transformer [37] language model on Wikitext-103 [22]. We base hyperparameter choices on the Pythia suite [4], specifically the 160 million parameter configuration with sinusoidal position embeddings, 12 layers, model dimension 768, 12 attention heads per layer, and hidden dimension 768. We use AdamW [21] with batch size 256, learning rate 0.0006 and weight decay 0.1. We use a context length 2048 and clip gradients to a maximum norm of 1. We also use a learning rate schedule with a linear warmup and cosine decay to 10% of the learning rate, like Biderman et al. [4].

For SVDs, for simplicity, we take the SVD of the entire $(3d_{\text{model}}, d_{\text{model}})$ parameter that computes queries, keys, and values from the hidden dimension inside the attention layer without splitting into individual heads. This is reasonable as the splitting is done after the fact internally. We also take the SVD of the output parameters and linear layers of the MLPs, which are 2-dimensional matrices.

For alignment, we consider the alignment of W_Q and W_K matrices, W_V and W_O matrices, the alignment between W_O and W_1 of the MLP block, between W_1 and W_2 of the MLP block, and between W_2 and the next attention layer.

In the single-layer plots, we visualize a weight matrix inside the MLP of a middle attention layer, and the alignment between middle weight matrices.

Appendix D. Limitations

There are a few key limitations to our study. As mentioned, we lack the computational resources to run more than 3 random seeds per experiment. However, we do find error bars to be quite tight in general (except for the generalization epoch in the grokking experiments). In addition, as discussed, we ignore 1D parameters in the neural networks, which may be particularly crucial (especially normalization). In addition, due to computational constraints, we do not consider the alignment of layers across residual connections as this quickly becomes combinatorial in-depth. Thus, there may be other interesting interactions that we do not observe. Finally, due to computational constraints we cannot investigate results on larger models than the 12 layer Transformer, which may have different behavior.

Appendix E. Compute Resources

All experiments are performed on an internal cluster with on the order of 100 NVIDIA 2080ti GPUs or newer. All experiments run on a single GPU in less than 8 hours. We estimate that end-to-end, it might take a few days for these resources to rerun all of the experiments in this paper.

Appendix F. Code Sources

We use PyTorch [26], NumPy [11] for all experiments and Weights & Biases [5] for experiment tracking. We make plots with Matplotlib [15] and Seaborn [39]. We also use HuggingFace Datasets [20] for Wikitext-103 [22].