A Implementation, Further Experiments, & Derivations

A.1 Implementation

In this section, we provide the implementation details behind our experiments. We use `PyTorch` for implementing the linear neural network, `CVXPY` for finding the minimum nuclear norm solution, the `R` package `ROptSpace` for running the `OptSpace` algorithm, and the `Python` package `fancyImpute` for running the `SoftImpute` algorithm. Additionally, for experiments on MovieLens100k, we use the `Spotlight` repo [1] for different kinds of recommendation models in `PyTorch` and modify its common building blocks for our purposes. Experiments were run on a NVIDIA Tesla V100 GPU.

Synthetic Data For synthetic data experiments, our implementation details mirror the experimental design of [7], which we briefly detail. When referring to a random rank r matrix with size $m \times n$, we mean a product $UV^\top$, where the entries of $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ are drawn independently from a standard normal distribution. The observed entries are uniformly sampled at random without repetition. The goal is to recover the underlying matrix by observing a portion of it and trying to recover the remaining entries as well as producing a solution matrix whose overall rank is low or is close to the rank of the underlying ground truth matrix. During training, deep linear neural networks are trained by gradient descent, or whichever other optimizer under consideration, under the Frobenius loss (i.e. ℓ_2 loss).

Weights are initialized via independent samples from a Gaussian distribution with zero mean and standard deviation 10^{-3} as in [7]. Learning rates are fixed through training; in line with previous work [7] and the assumptions of gradient flow, we set our initial learning rates (for gradient descent, Adam, and our other optimizers) $\alpha = 10^{-3}$ for the results shown in this paper, but we also conducted the same experiments with $\alpha \in \{5 \cdot 10^{-4}, 10^{-4}\}$ and saw no qualitative differences. During training, stopping criteria consist of either reaching a total of $5 \cdot 10^5$ iterations or training loss reaching values below 10^{-7}. Results relating to the ratio penalty are robust across varying strengths of regularization $\lambda \in [10^{-4}, 10^{-1}]$. For Adam, aside from the initial learning rate, the default hyper-parameters under PyTorch's implementation were used, such as $(\beta_1, \beta_2) = (0.9, 0.999)$, no `amsgrad`, etc. unless otherwise stated. Similarly for experiments with other optimizers, aside from the initial learning rate, we use the default hyper-parameters unless otherwise specified. Test error is measured via mean-squared error, unless stated otherwise. To quantify the rank of solutions produced, we use the effective rank measure [56] due to its favorable properties over numerical rank in simulation settings and its usage in [7]. The test error with respect to a ground truth matrix $W^\star$ is given by the mean-squared error (i.e., $\frac{1}{N}\|W - W^\star\|_F^2$) but results continue to hold under similar measures of test error. For rank 5 experiments where the sample size was fixed, sample size is set at 2000 (again, following [7]) but results are similar for sample sizes at 2500, 3000, etc. The sample size for rank 10 experiments were set at 3000 and 3500.

MovieLens100k For our experiments on MovieLens100k, we use the base explicit factorization model available in `Spotlight` in conjunction with Adam and our ratio penalty. The explicit factorization model resembles a shallow factorization but includes several notable differences: an user-embedding layer, an item-embedding layer, an user-specific bias, and an item-specific bias. With our penalty applied to either the user-embedding layer or the item-embedding layer, we evaluate values of $\lambda \in [10^{-6}, 10^{-1}]$, embedding dimensions in $\{2, 4, 8, 16, 32, 64, 128\}$, and batch-sizes of $\{128, 256, 512, 1024\}$ for initial learning rates $\alpha \in [5 \cdot 10^{-3}, 10^{-5}]$. Each model configuration is trained for 100 epochs and evaluated on the full test set at the end. The best model is check-pointed and training is terminated if training error begins increasing for three epochs. The best performing model with the reported test error was with learning rate $\alpha = 10^{-3}$, batch size 256, effective dimension 64, $\lambda_{\text{user}} = 0.01$ while another was $\alpha = 10^{-3}$, batch size 128, effective dimension 64, $\lambda_{\text{user}} = 10^{-4}$ and $\lambda_{\text{item}} = 10^{-1}$. For the depth 1 LNN, we take an approach near-identical to our synthetic experiments but include a bias term that is added to our single weight matrix. We explore values for regularization strengths $\lambda \in [2 \cdot 10^1, 10^{-6}]$ and initial learning rate $\alpha \in [5 \cdot 10^{-3}, 10^{-5}]$. The best performing model with the reported test error was with learning rate $\alpha = 5 \cdot 10^{-4}$ and $\lambda = 1.5$ though many similar configurations came close to the same performance. A more rigorous sweep of configurations is necessary to exhaustively find similar, or the best, configurations; due to space considerations, we leave more in-depth evaluations for future work.

[1] (https://github.com/maciejkula/spotlight)

A.2 Rank 10 Matrix Completion

The following details the same set of experiments as those in Section 3 but for rank 10 matrices as an example to illustrate that our results generalize beyond rank 5.

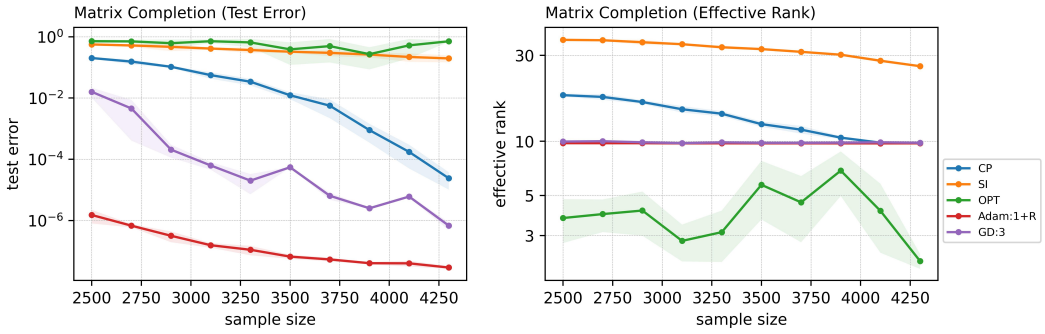

Figure 5: Comparative performance in generalization error and rank minimization for rank 10 matrix completion (of size 100×100). x-axis stands for the number of observed entries (uniformly sampled) and shaded regions indicate standard error bands. **Adam:1+R** refers to Adam at depth 1 with the penalty, **CP** is the minimum nuc. norm solution, **GD:3** is gradient descent with depth 3 (deep matrix factorization), **OPT** is `OptSpace` [38], and **SI** is `SoftImpute` [47]. To reduce clutter, we omit results with similar performance to GD:3 (e.g. GD:4/5 etc.).

Overall, comparative performance results very much resemble those in Fig. 4 for rank 5 completion. As seen in Fig. 5, our penalty's comparative performance relative to other techniques remains unchanged from rank 5. Like before, Adam with penalty, at depth 1, outperforms all other approaches across all data regimes by at least two orders of magnitude, with reconstruction error decreasing even further as sample size is increased. A similar result is seen in terms of its performance on rank: it reduces rank to the point of exact rank recovery independently of sample size.

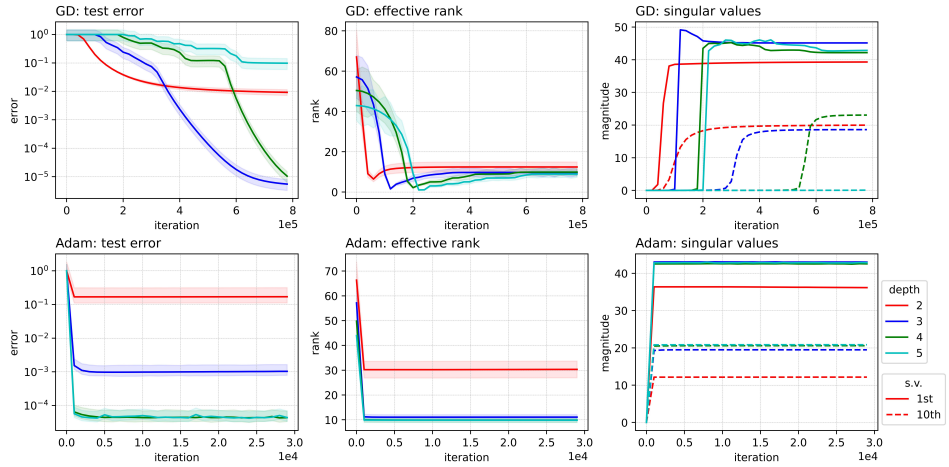

Figure 6: Dynamics of gradient descent and Adam over DLNN. Plots above show the performance of gradient descent over networks of depths 2/3/4/5 for rank 10 matrices of size 100×100. The top row depicts gradient descent and the bottom depicts Adam. The left column depicts generalization error as a function of depth and training iterations. The middle column depicts the change in effective rank across depths and over training iterations. The right column shows the 1st and 10th largest singular values for each depth across training iterations. For singular values, within each depth level (colored lines), a solid line indicates the 1^{st} largest singular value while a dotted line indicates the 10^{th} largest. We omit the other singular values for clarity due to their small magnitude.

Extending our experiments to rank 10 matrix completion, we see extreme similarity with results under rank 5 (Fig. 6). For singular values, we focus on the 1st and 10th largest values in accordance with the changed rank setting (rank 5 to rank 10) to better illustrate the differential behavior of the most prominent values. Our overall results remain the same as in Section 3.

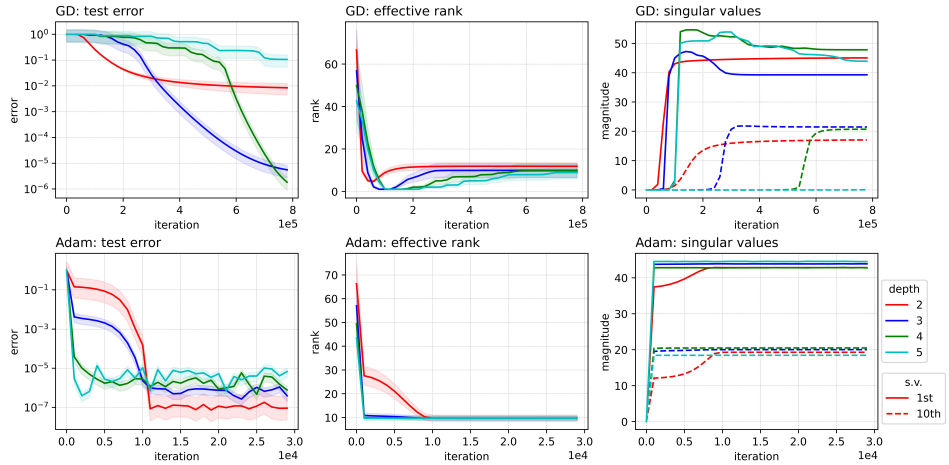

Figure 7: Dynamics of regularized gradient descent and regularized Adam with our penalty. Plots show the performance over networks of depths 2/3/4/5 for rank 10 matrix completion. The penalty's hyper-parameter is set at $\lambda = 10^{-4}$ but results hold for a range of values $\lambda \in [10^{-6}, 10^{-1}]$.

For regularized gradient descent and Adam (Fig. 7), results again resemble the rank 5 case. Under gradient descent, the proposed penalty induces only slight effects on quickening and tightening the convergence behavior of singular values and test error. In contrast, we see nearly identical effects of our penalty under Adam when compared to the rank 5 case. We also see that, like in rank 5, depth 2 becomes effective in generalizing and reducing rank when compared to depth 2 under Adam alone; in depth 2, our proposed penalty is able to push its singular value trajectories of the 1st and 10th largest singular values towards the other respective singular values in higher depth levels as if getting depth 2 to behave similarly as higher depth levels in contrast to un-penalized Adam.

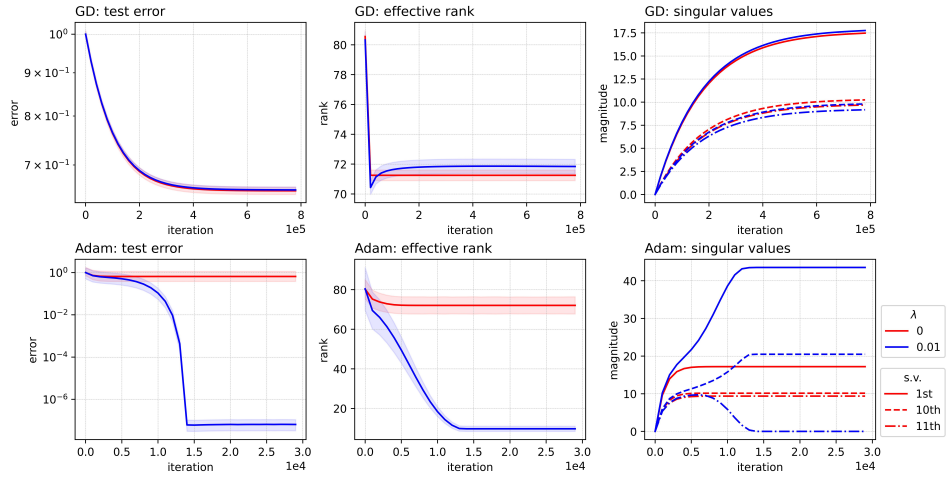

Figure 8: Performance comparison between choice of optimization and regularizer in a depth-1 linear network. Top row corresponds to gradient descent and bottom corresponds to Adam. Note that $\lambda = 0$ corresponds to the un-regularized setting. The penalty's hyper-parameter is set at $\lambda = 0.01$ but results hold for a range of values $\lambda \in [10^{-4}, 10^{-1}]$. For illustrative emphasis, the 11th largest singular value (dashed line) is also included here to depict the trajectory difference under penalized gradient descent versus penalized Adam for depth 1.

A.3 Comparing Different Optimizers & Penalties

In this section, we expand upon our experiments evaluating different combinations of optimizers and explicit penalties beyond those in Table 1. In Table 3 and Table 4, we include other optimizers and evaluate them against several explicit penalties. For all optimizer choices, hyper-parameters, aside

Table 3: Additional results for rank 5 matrix completion across various optimizer/penalty/depth combinations in terms of test error (**Err**) and effective rank (**Rk**, rounded to nearest integer) of resulting solutions at convergence. **Ratio** denotes our ratio penalty ($||\cdot||_*/||\cdot||_F$), **Sch p:q** denotes the ratio of two Schatten (quasi)norms ($||\cdot||_{S_p}/||\cdot||_{S_q}$) as penalty, **Nuc** denotes nuclear norm penalty, and **None** denotes no penalty.

Optimizer	Depth	Ratio		Sch $\frac{1}{2}:\frac{2}{3}$		Sch $\frac{1}{3}:\frac{2}{3}$		Sch $\frac{1}{3}:\frac{1}{2}$		Nuc		None	
		Err	Rk	Err	Rk	Err	Rk	Err	Rk	Err	Rk	Err	Rk
Adam [39] (w. amsgrad)	1	0.83	29	0.99	53	1.00	61	1.00	60	0.34	6	1.00	79
	3	$9e$-3	5	0.05	6	0.04	6	0.10	6	0.32	6	0.06	6
Adadelta [69]	1	0.98	59	1.00	63	0.98	37	1.00	53	0.77	21	1.01	74
	3	$3e$-3	5	$2e$-4	5	$3e$-3	5	$2e$-3	5	0.29	5	$5e$-3	5
GD (w. momentum)	1	0.80	70	0.79	64	0.77	47	0.80	63	0.78	54	0.81	69
	3	$3e$-3	5	$1e$-4	5	0.43	4	0.07	5	0.02	5	0.07	4
AdamW [45]	1	$1e$-3	5	0.98	45	1.00	49	1.00	64	0.34	6	1.00	79
	3	$1e$-3	5	$2e$-4	5	0.01	5	$3e$-2	5	0.31	6	0.02	5
NAdam [24]	1	$1e$-3	5	0.97	43	0.99	54	0.99	61	0.32	6	1.01	79
	3	$2e$-3	5	$1e$-4	5	0.05	5	0.01	5	0.30	5	0.02	5
RAdam [44]	1	$1e$-3	5	0.95	46	0.99	51	1.00	62	0.25	6	1.00	80
	3	$7e$-4	5	$5e$-4	5	0.01	5	0.01	5	0.18	5	0.05	6

from the initial learning rate which is set to 10^{-3}, are set to their default values in `PyTorch` unless otherwise specified (e.g. default momentum values for GD). For all experiments evaluating optimizer-penalty combinations, we try $\lambda \in [10^{-6}, 10^{-1}]$ which all delivered similar results, especially in terms of observed depth invariance (or lack thereof) and generalization/rank reduction performance. The results shown in Table 1 in Appendix A.3 as well as in Table 3 and Table 4 here have $\lambda = 0.05$. Results shown are trained on a sample size of 2000 entries (out of a 100×100 matrix); results show no meaningful difference on other sample sizes (2500, 3000).

Table 4: Additional results for rank 5 matrix completion across various optimizer/penalty/depth combinations in terms of test error (**Err**) and effective rank (**Rk**, rounded to nearest integer) of resulting solutions at convergence. **Sch p** denotes the Schatten (quasi)norm ($||\cdot||_{S_p}$) as penalty.

Optimizer	Depth	Sch $\frac{1}{3}$		Sch $\frac{1}{2}$		Sch $\frac{2}{3}$	
		Err	Rk	Err	Rk	Err	Rk
Adam [39]							
w. amsgrad	1	0.99	69	0.97	11	0.55	7
	3	1.00	5	1.00	1	0.80	2
w.o. amsgrad	1	1.00	50	0.94	5	0.10	5
	3	0.88	1	0.84	1	0.17	4
Adagrad [25]	1	1.00	12	0.97	2	0.70	15
	3	0.81	1	0.85	1	0.08	5
Adamax [39]	1	1.00	54	0.87	1	1.00	41
	3	1.00	23	0.85	1	1.00	4
RMSProp	1	1.04	53	1.00	40	0.10	7
	3	1.00	8	0.74	1	0.08	5
GD							
w. momentum	1	266.7	43	2.14	29	0.65	16
	3	1.00	25	1.00	17	0.41	3
w.o. momentum	1	67.6	45	0.94	3	0.77	19
	3	1.10	34	0.99	1	0.58	2
Adadelta [69]	1	0.99	60	1.05	48	1.00	60
	3	0.99	56	0.94	1	0.99	56
AdamW [45]	1	1.00	48	0.97	9	0.67	6
	3	1.00	10	0.99	1	0.82	2
NAdam [24]	1	1.00	47	0.98	8	0.80	7
	3	1.00	3	0.92	1	0.71	3
RAdam [44]	1	1.00	46	1.01	25	0.57	7
	3	0.99	7	0.91	1	0.68	3

We note that Adam-like variants such as AdamW, NAdam, and RAdam also, unsurprisingly, exhibit varying degrees of depth invariance with respect to both generalization error and rank reduction with

our ratio penalty though their solutions fail to generalize as well as Adam and Adamax under the penalty—this makes sense as Adam and Adamax are much more closely related, and both proposed in the same paper with only slight differences in their update rules, than subsequent variants that have significantly different iterative updates (see [24, 39, 44, 45] for further details).

A.4 Supplemental Derivations

Here, we outline and detail the derivations behind some of our work describing the trajectory of the end-product matrix W and its singular values σ_i under gradient flow, largely building off of [6, 7]. We highlight the main derivations as they relate to the theorems in this paper and defer to the appendices of [6, 7] for additional details and a fuller treatment of gradient flow derivations. We also provide additional commentary and discussion here due to space constraints in the main body of the paper.

A.4.1 Theorem 2 (Gradient flow, with penalty)

Setup & preliminaries. We omit the details of deriving the $\dot{\sigma}$ and $\dot{W}$ under un-regularized gradient flow; instead, we defer to the appendix of [6] for a detailed treatment and build upon their results for our derivations.

Evolution of W and σ under gradient flow without penalty. To begin, we recall the dynamics of the end-product matrix W and its singular values σ under *un-regularized* gradient flow (for details on a full derivation, see the appendices of [6, 7] for a detailed treatment):

$$\dot{\sigma}_i = -N(\sigma_r^2)^{\frac{N-1}{N}} \mathbf{u}_r^\top \nabla_W \mathcal{L}(W) \mathbf{v}_r \tag{10}$$

$$\text{vec}(\dot{W}) = -P_W \text{vec}\left(\nabla_W \mathcal{L}(W)\right) \tag{11}$$

Since the explicit regularization simply adds a regularization term to the original loss function, we can re-calculate the gradient of the new, regularized loss and substitute the gradients back into the dynamics above in order to re-characterize the above dynamics under the penalty. In more formal terms, if we let the new modified loss function be represented by $\tilde{\mathcal{L}}(W) \coloneqq \mathcal{L}(W) + \lambda R(W)$ where $R(W) = ||W||_*/||W||_F$ denotes the explicit penalty, then we can calculate $\nabla_W \tilde{\mathcal{L}}(W) = \nabla_W \mathcal{L}(W) + \lambda \nabla_W R(W)$ to plug back into the un-regularized dynamics in order to characterize the trajectories under the effect of the penalty.

Evolution of W and σ under gradient flow with penalty. To characterize the dynamics of gradient flow in presence of the penalty, we first derive the gradient of the penalty (i.e. $\nabla_W R(W)$) ignoring the regularization strength (hyper)parameter λ. Via the chain rule and the sub-gradient of the nuclear norm, we can express the gradient of the penalty as:

$$\nabla_W R(W) = \frac{1}{||W||_F^2} \left(UV^\top - \frac{||W||_*}{||W||_F} U\Sigma V^\top \right) \tag{12}$$

where U, V, Σ are the singular matrices of the singular value decomposition of W (i.e. $W = U\Sigma V^\top$).

To first characterize the dynamics of the end-product matrix W, we can use the gradient of the penalty and substitute it back into Eq. (11) along with the loss gradient. Taking Eq. (12), plugging it back into Eq. (11) along with the original loss gradient $\nabla_W \mathcal{L}(W)$, and re-arranging terms, we have

$$\begin{aligned}
\text{vec}(\dot{W}) &= -P_W \text{vec}\left(\nabla_W \mathcal{L}(W) + \lambda \nabla_W R(W)\right) \\
&= -P_W \left(\text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top - \frac{||W||_*}{||W||_F} U\Sigma V^\top)}{||W||_F^2} \right) \\
&= -P_W \left(\text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top - U\tilde{\Sigma} V^\top)}{||W||_F^2} \right)
\end{aligned}$$

where $\tilde{\Sigma} = \frac{||W||_*}{||W||_F}\Sigma$, producing the expression in Eq. (8).

Turning now to the singular values, for analytical ease in characterizing the dynamics of the singular values σ_i under gradient flow with the penalty (i.e., $\dot{\sigma}_r^{\text{reg}}$), we can equivalently express Eq. (12) in

terms of its diagonal elements; for instance, if we focus on the r-th diagonal element, this produces

$$\frac{\mathbf{u}_r^\top \mathbf{v}_r}{||W||_F^2}\left(1 - \frac{||W||_*}{||W||_F}\sigma_r\right)$$

where $\{\mathbf{u}_r, \mathbf{v}_r\}$ are the i-th left and right singular vectors of W and σ_r is the r-th diagonal element of Σ. Substituting this term along with the original loss gradient back into Eq. (10) and re-writing/grouping terms, we obtain our desired result in the form of Eq. (7) for the r-th singular value, i.e.,

$$\dot{\sigma}_r^{\text{reg}} = \dot{\sigma}_r^{\text{GF}} - \frac{\lambda N}{||W||_F^2}\left(1 - \frac{||W||_*}{||W||_F}\right)\sigma_r^{\frac{3N-2}{2}}$$

where $\dot{\sigma}_r^{\text{GF}}$ is defined in Eq. (10) (i.e. the trajectory of singular values under un-regularized gradient flow).

A.4.2 Theorem 1 (Adam, no penalty)

Setup & preliminaries. We omit the details of deriving P_W from the beginning and defer to the appendix of [6] for a more comprehensive review. Operations like divisions, squaring, and square-roots on vectors and matrices are to assumed to be performed element-wise unless otherwise stated. In this case, since parameters are matrices, loss gradients will generally be matrices; in this case, when referring to the variance of said objects, the variance is taken to mean the variance of the vectorized matrix (i.e. condensed into a vector) drawn from some distribution.

Per-layer weight updates under Adam in discrete time take the form, assuming no weight-decay:

$$W_j^{(t+1)} = W_j^{(t)} - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

where $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$, $m_t = (1-\beta_1)m_{t-1} + \beta_1 g_{t-1}$, $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$, $v_t = (1-\beta_2)v_{t-1} + \beta_2 g_{t-1}^2$ for each layer $j \in \{1, \dots, N\}$ in a depth N network. α is the initial learning rate, $\hat{m}_t$ is the bias-corrected first moment, $\hat{v}_t$ is the bias-corrected second moment, and ε is a division-by-zero adjustment which we assume to be zero to simplify our later analysis. In turn, m_t is the moving average of stochastic gradients g_t and v_t is the moving average of their element-wise square g_t^2, where $g_t = \nabla_{W_j}\mathcal{L}(W^{(t)})$, the gradient of the loss with respect to the j-th layer weight, g_t^2 is the element-wise squaring of g_t, and $\beta_1, \beta_2 \in (0, 1)$ are discount factors for m_t and v_t respectively. Moving forward, for notational simplicity, we suppress the dependence of the W and W_j on t unless otherwise stated and set $\beta_1 = \beta_2 = \beta = 1 - \epsilon$ for sufficiently small epsilon (e.g. $\epsilon = 0.001$).

Given the loss in Eq. (1), the gradient of the loss with respect to the j-th layer weight matrix is (suppressing notation on time t):

$$g \coloneqq \nabla_{W_j}\mathcal{L}(W) = \prod_{i=j+1}^{N} W_i^\top \left(\nabla_W \mathcal{L}(W)\right) \prod_{i=1}^{j-1} W_i^\top \tag{13}$$

where $\nabla_W \mathcal{L}(W)$ is the gradient of the loss with respect to the end-product matrix W. While gradient flow uses this loss directly in its iterative updates for each j-th weight matrix, Adam makes adjustments to this loss gradient in its calculation of $\hat{m}$ and $\hat{v}$.

Assumptions. As noted earlier in Section 3, to approximately characterize the trajectory under Adam in closed form, albeit imperfectly, we make a few additional assumptions. More formally, in the same spirit as [5], we assume that the underlying distribution of stochastic gradients is stationary and approximately constant over the time horizon of the moving average of gradients so that $\hat{m}_t$ and $\hat{v}_t$ approximate estimate their respective moments of g_t. Namely:

$$\hat{m}_t \to \mathbb{E}(\hat{m}_t) \approx \nabla_{W_j}\mathcal{L}(W^{(t)}), \quad \hat{v}_t \to \mathbb{E}(\hat{v}_t) \approx (\nabla_{W_j}\mathcal{L}(W^{(t)}))^2 + s_{t,j}^2 \tag{14}$$

where $s_{t,j}^2$ is the (element-wise) variance of $\nabla_{W_j}\mathcal{L}(W^{(t)})$ with the assumption that each W_j is drawn i.i.d. from a common stationary distribution. Naturally, this assumption can only hold approximately, at best, and is clearly inaccurate for gradients far in the past—to that end, we assume some sufficient degree of "burn-in" or progression so that past some time t, gradients near initialization or extremely distant past gradients do not contribute meaningfully to the moving average. As noted in [5], this

assumed approximation is more realistic and more likely to hold in certain cases (e.g. high noise or small step size, the latter of which aligns with the spirit of gradient flow).

Evolution of W under Adam without penalty. Under the assumptions above and those in [7], to approximately characterize the trajectory of the end-product matrix, we can rewrite Adam's update of each weight layer W_j using our assumptions above in continuous time (i.e. $\alpha \to 0$), in a fashion similar to gradient flow, as (suppressing notational dependence on time t):

$$\dot{W}_j = -\frac{\nabla_{W_j}\mathcal{L}(W)}{\sqrt{\nabla_{W_j}\mathcal{L}(W)^2 + s_j^2}} \tag{15}$$

where, again, division, etc. here denote element-wise operations. As an aside, we note that the expression in Eq. (15) can also be written as $\dot{W}_j = -(1+\eta_j^2)^{-1/2}$ where $\eta_j^2 \coloneqq s_j^2/\nabla_{W_j}\mathcal{L}(W)^2$, resembling a variance adaptation factor for each layer j that shortens the update in directions of high (relative) variance as an adaptive means to account for varying reliability of the gradient throughout the optimization process or an approximate signal-to-noise ratio of the gradient [5, 39]. Using the definition of $\nabla_{W_j}\mathcal{L}(W)$ from Eq. (13), we can rewrite the right-hand side of Eq. (15) as:

$$-\prod_{i=j+1}^{N} W_i^\top (\nabla_W \mathcal{L}(W)) \prod_{i=1}^{j-1} W_i^\top \odot S_j \tag{16}$$

where $\odot$ denotes the Hadamard product and S_j is a matrix that contains the corresponding entry-wise elements of $(\nabla_{W_j}\mathcal{L}(W)^2 + s_j^2)^{-1}$ to make the element-wise operations more explicit. To derive the evolution of the end-product matrix $\dot{W}$ from $\dot{W}_j$, we follow the approach in [7] but with several modifications due to presence of S_j. For the full details, we defer to the appendix of [7]. We left and right multiply by a product of other weight layers and sum across the depth of the network:

$$\dot{W} = -\sum_{j=1}^{N} \prod_{j+1}^{i=N} W_i \left(\prod_{i=j+1}^{N} W_i^\top ((\nabla_W \mathcal{L}(W))) \prod_{i=1}^{j-1} W_i^\top \odot S_j \right) \prod_{1}^{i=j-1} W_i \tag{17}$$

To simplify notation for the proceeding steps, we define the following:

$$A = \prod_{i=j+1}^{N} W_i^\top,\ B = \prod_{i=1}^{j-1} W_i^\top,\ L = \nabla_W \mathcal{L}(W)$$
$$\alpha = A^\top, \quad \beta = ALB \odot S_j, \quad \gamma = B^\top$$

Re-writing the above expression with these new definitions, we have:

$$-\sum_{j=1}^{N} A^\top (ALB \odot S_j) B^\top = -\sum_{j=1}^{N} \alpha\beta\gamma$$

Ignoring the negative sign to focus on the summation and taking the (column-major order) vectorization of the above expression, we then have:

$$\begin{aligned}
\text{vec}\left(\sum_j \alpha\beta\gamma\right) &= \sum_j \text{vec}(\alpha\beta\gamma) \\
&= \sum_j (\gamma^\top \otimes \alpha)\text{vec}(\beta) \\
&= \sum_j (B \otimes A^\top)\text{vec}(ALB \odot S_j) \\
&= \sum_j (B \otimes A^\top)(\text{vec}(ALB) \odot \text{vec}(S_j)) \\
&= \sum_j (B \otimes A^\top)(B^\top \otimes A)(\text{vec}(L) \odot \text{vec}(S_j)) \\
&= \sum_j (BB^\top \otimes A^\top A)(\text{vec}(L) \odot \text{vec}(S_j)) \\
&= \sum_{j=1}^{N} ((WW^\top)^{\frac{N-j}{N}} \otimes (W^\top W)^{\frac{j-1}{N}})(\text{vec}(L) \odot \text{vec}(S_j))
\end{aligned}$$

where the first two equalities is due to the additivity of the vectorization operator and the association between vectorized products and the Kronecker product ($\text{vec}(ABC) = (A^\top \otimes C)\text{vec}(B)$), the fourth equality is due to the preservation of the Hadamard product over vectorization, the fifth equality is the result of applying the same logic as the first equality to $\text{vec}(ALB)$, the sixth equality is the result of the mixed-product property of Kronecker products, and the simplification in the final equality can be found in the appendix of [6]. Previous work [6] has shown that, in un-regularized gradient flow, $P_W \coloneqq \sum_{j=1}^{N}((WW^\top)^{\frac{N-j}{N}} \otimes (W^\top W)^{\frac{j-1}{N}})$ is a p.s.d. matrix that serves as an accelerative pre-conditioning that acts on the loss gradient. However, now there is an additional component S_j so we can further simplify the previous expression:

$$\begin{aligned}
&= \sum_{j=1}^{N} ((WW^\top)^{\frac{N-j}{N}} \otimes (W^\top W)^{\frac{j-1}{N}})(\text{diag}(\text{vec}(S_j))\text{vec}(L)) \\
&= \sum_{j=1}^{N} ((WW^\top)^{\frac{N-j}{N}} \otimes (W^\top W)^{\frac{j-1}{N}})(G_j\text{vec}(L))
\end{aligned}$$

where $G_j = \text{diag}(\text{vec}(S_j)$ denotes taking the elements of $\text{vec}(S_j))$ and placing them along the diagonal of a zero matrix. Finally, substituting the term back in for L, we have a approximate characterization of the trajectory of the end-product matrix W under Adam:

$$\text{vec}(\dot{W}) = -P_{W,G}\text{vec}(\nabla_W \mathcal{L}(W)) \tag{18}$$

where $P_{W,G} \coloneqq \sum_{j=1}^{N}((WW^\top)^{\frac{N-j}{N}} \otimes (W^\top W)^{\frac{j-1}{N}})G_j$.

Positive semi-definiteness of $P_{W,G}$. Under gradient flow, [6] has shown that $P_W = \sum_{j=1}^{N} P_j$ where $P_j = (W^\top W)^{\frac{N-j}{N}} \otimes (WW^\top)^{\frac{j-1}{N}}$ and P_j is symmetric; to show that P_j is p.s.d., note that, suppressing notation for time dependence t, we can equivalently write for some j,

$$P_j = A((DD^\top)^{\frac{j-1}{N}} \otimes (D^\top D)^{\frac{N-j}{N}})A^\top \tag{19}$$

where $A = U \otimes V$, the Kronecker product of the matrices containing the left and right singular vectors of W respectively, and D is the diagonal matrix containing the singular values of W. Since the term in the middle is the Kronecker product of two diagonal matrices (with non-negative singular values along the diagonal), it's also a diagonal matrix. Therefore, for non-zero x and some fixed j,

$$x^\top A((DD^\top)^{\frac{j-1}{N}} \otimes (D^\top D)^{\frac{N-j}{N}})A^\top x \geq ||A^\top x||^2 c \geq 0$$

where $c = \sigma_1^{2\frac{j-1}{N}} \cdot \sigma_1^{2\frac{N-j}{N}}$ and σ_1 is the smallest singular value of W or, equivalently, the smallest value of the diagonal matrix D which is by definition non-negative. G_j is also symmetric, since it is diagonal, and p.s.d. since its diagonal entries (and therefore eigenvalues) are by definition non-negative. Since $P_j G_j$ is the product of a real symmetric p.s.d. matrix P_j and a real positive diagonal matrix G_j, the product is also p.s.d. [51]. Therefore, $P_{W,G} = \sum_{j=1}^{N} P_j G_j$ is also p.s.d. as a sum of p.s.d. matrices. Similar to P_W under un-regularized gradient flow, $P_{W,G}$ acts as a pre-conditioning on the loss gradient which now also includes the variance of the loss gradient with respect to each layer j.

Implications of $P_{W,G}$. As noted in [6], P_W in the original case of un-regularized gradient flow serves as an accelerative pre-conditioning on the loss gradient (Eq. (4)) that intensifies with depth and can only exist with sufficient depth ($N \geq 2$). More specifically, [6] show that the eigenvalues of P_W are $\sum_{j=1}^{N} \sigma_r^{2\frac{N-j}{N}} \sigma_{r'}^{2\frac{j-1}{N}}$, corresponding to its eigenvectors $\text{vec}(\mathbf{u}_r \mathbf{v}_{r'}^\top)$ where $\mathbf{u}_r$ and $\mathbf{v}_{r'}^\top$ are the left and right singular vectors of the end-product matrix $W \in \mathbb{R}^{m \times n}$. With non-trivial depth ($N \geq 2$), increasing σ_r or $\sigma_{r'}$ results in an increase in the eigenvalue corresponding to the eigen-direction (i.e. rank one matrix) $\mathbf{u}_r \mathbf{v}_{r'}^\top$, which can be interpreted as the pre-conditioning favoring or accelerating in directions that correspond to singular vectors whose presence in the end-product matrix W is stronger. In the case of degenerate depth ($N = 1$), however, the pre-conditioning P_W collapses into the identity, causing all of its non-zero eigenvalues to reduce down to unity—resulting in no accelerative favoring of any eigen-direction(s) during optimization.

In the case of Adam's pre-conditioning at degenerate depth, $P_{W,G}$'s non-zero eigenvalues, unlike P_W, are no longer just unity and static which produce no accelerative favoring in any direction; instead, its eigenvalues become $[(1+\eta^2)^{-1/2}]_{r,r'}$ where now $\eta^2 = s^2 / \nabla_W \mathcal{L}(W)^2$, allowing for continued variation and accelerative effects even at depth 1. For non-degenerate depths ($N \geq 2$), each individual P_j within $P_{W,G}$ is now normalized by G_j so that each layer's individual contribution to the overall pre-conditioning is normalized by a function of that layer's squared gradient and gradient variance, unlike before.

Evolution of σ under Adam without penalty. For the trajectories of the singular values $\{\sigma_i\}$, we largely build atop the approach in [7], whose key steps we highlight here. We start by defining the analytic singular value decomposition of the end-product matrix as $W(t) = U(t) D(t) V(t)^\top$. Differentiating $W(t)$ with respect to time t:

$$\dot{W}(t) = \dot{U}(t) D(t) V(t)^\top + U(t) \dot{D}(t) V(t)^\top + U(t) D(t) \dot{V}(t)$$

If we then left-multiply the above expression by $U(t)^\top$ and right-multiply by $V(t)$, we see that:

$$U(t)^\top \dot{W}(t) V(t) = U(t)^\top \dot{U}(t) D(t) + \dot{D}(t) + D(t) \dot{V}(t) V(t)$$

where the orthonormal columns of $U(t)$ and $V(t)$ have helped simplify the earlier expression. Focusing on the diagonal elements of the above expression and suppressing the notation for time dependence, we see that they are:

$$\mathbf{u}_i^\top \dot{W} \mathbf{v}_i = \mathbf{u}_i^\top \dot{\mathbf{u}}_i^\top \sigma_i + \dot{\sigma}_i + \sigma_i \dot{\mathbf{v}}_i^\top \mathbf{v}_i$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ are the i-th left and right singular vectors associated with the i-th diagonal element or, equivalently, the i-th singular value σ_i. Since the columns of U are orthonormal by definition, and because $\mathbf{u}_i$ and $\mathbf{v}_i$ has constant unit length by definition (i.e. $\mathbf{u}_i^\top \mathbf{u}_i = \frac{1}{2} \frac{d}{dt} ||\mathbf{u}_i(t)||_2^2 = 0$), the above equation can be distilled down into:

$$\dot{\sigma}_i = \mathbf{u}_i^\top \dot{W} \mathbf{v}_i$$

Taking the (column-order first) vectorization of the expression above, then:

$$\text{vec}(\dot{\sigma}_i) = \dot{\sigma}_i = (\mathbf{u}_i^\top \otimes \mathbf{v}_i^\top) \text{vec}(\dot{W}) = \text{vec}(\mathbf{v}_i \mathbf{u}_i^\top)^\top \text{vec}(\dot{W}) \tag{20}$$

We can then substitute our expression for $\text{vec}(\dot{W})$ from Eq. (18) to characterize the singular values under Adam:

$$\dot{\sigma}_i = -\text{vec}(\mathbf{v}_i \mathbf{u}_i^\top)^\top P_{W,G} \text{vec}(\nabla_W \mathcal{L}(W)) \tag{21}$$

A.4.3 Lemma 4 (Adam, with penalty)

We omit the details of deriving $\dot{\sigma}$ under gradient flow from the beginning and defer to the appendix of [6] for a more comprehensive review. We highlight the key parts.

Evolution of W under Adam with penalty. To characterize the evolution of the end-product matrix W under Adam in presence of the penalty, we can simply leverage our earlier results from Eq. (18) and combine them with the gradient of the penalty since the penalty is simply an additive component to the same loss function. Denoting our penalty by $R(W)$, we can characterize $\dot{W}$ as:

$$\text{vec}(\dot{W}) = -P_{W,G}\text{vec}(\nabla_W \mathcal{L}(W) + \lambda \nabla_W R(W))$$

Substituting in the gradient of the penalty and vectorizing accordingly, we have:

$$\text{vec}(\dot{W}) = -P_{W,G}\left(\text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2}\right) \tag{22}$$

where U and $V^\top$ are the matrices containing the left and right singular vectors of the end-product matrix W at time t (time notation suppressed in the expression above), λ is the regularization strength, and $\tilde{\Sigma}$ is a re-weighted version of a rectangular diagonal matrix containing the singular values of W (i.e. $\tilde{\Sigma} = \frac{||W||_*}{||W||_F}\Sigma$). Here, we can see that $P_{W,G}$ in Eq. (22) is accelerating not just the loss gradient in helping with optimization with respect to performance via the loss but also the penalty gradient in its tendency towards low-rankedness.

Interestingly, we also note that the second term in the parenthesis of Eq. (22) can be seen as a re-scaled version of W. Namely, if we ignore the vectorization operator, the term $(UV^\top - U\tilde{\Sigma}V^\top)/||W||_F^2$ can be re-expressed as a new spectrally shifted and re-scaled W that we can define as $\bar{W} \coloneqq U\bar{\Sigma}V^\top$ where $\bar{\Sigma} = (I - \tilde{\Sigma})/||W||_F^2$. In other words, we see that the explicit regularizer affects the network's (i.e., W) trajectory by introducing a new spectrally adjusted version of itself as part of the penalization.

Evolution of σ under Adam with penalty. Similar to the approach taken to characterize $\dot{W}$ under the penalty, we can re-use the expression in Eq. (20) and substitute in the appropriate expression for $\text{vec}(\dot{W})$ in the case of Adam with the penalty:

$$\dot{\sigma}_i = -\text{vec}(\mathbf{u}_i\mathbf{v}_i^\top)^\top P_{W,G}\left(\text{vec}(\nabla_W \mathcal{L}(W)) + \lambda \frac{\text{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2}\right) \tag{23}$$

A.4.4 Depth 1 dynamics

Lastly, we clarify the dynamics of the end-product matrix W and its singular values σ in the case of a depth 1 network.

Un-regularized gradient flow and Adam. As mentioned in Section 3, for un-regularized gradient flow, at $N = 1$, the trajectories of the end-product matrix and its singular values reduce down to:

$$\dot{\sigma}_i = -\mathbf{u}_i^\top \nabla_W \mathcal{L}(W)\mathbf{v}_i \tag{24}$$

$$\text{vec}(\dot{W}) = -\text{vec}\left(\nabla_W \mathcal{L}(W)\right) \tag{25}$$

where $P_W = I_{mn}$ has now reduced down to the identity; as such, the accelerative pre-conditioning that typically strengthens with depth no longer exists, as expected.

For un-regularized Adam, at $N = 1$, we have:

$$\dot{\sigma}_i = -\text{vec}(\mathbf{v}_i\mathbf{u}_i^\top)^\top (G \cdot \text{vec}(\nabla_W \mathcal{L}(W))) \tag{26}$$

$$\text{vec}(\dot{W}) = -G \cdot \text{vec}(\nabla_W \mathcal{L}(W)) \tag{27}$$

where $P_{W,G} = I_{mn}$, $G_j = G$ since now $N = j = 1$, $G = \text{diag}(\text{vec}(S_j))$, and S_j is defined as a matrix whose elements are: $[S_j]_{m,n} = [(\nabla_W \mathcal{L}(W)^2 + s^2)^{-1/2}]_{m,n}$ as defined in Eq. (6). While the depth-dependent accelerative pre-conditioning still no longer exists, we have a new sort of "pre-conditoning" in the form of a p.s.d. matrix G that is a function of the squared loss gradient and its variance.

Regularized gradient flow and Adam. Finally, we describe the dynamics for the end-product matrix and its singular values under gradient flow and Adam at depth 1, but now with the penalty.

For gradient flow with the penalty, we see that:

$$\dot{\sigma}_r = -\mathbf{u}_i^\top \nabla_W \mathcal{L}(W)\mathbf{v}_i - \frac{\lambda}{||W||_F^2}\left(1 - \frac{||W||_*}{||W||_F}\right)\sigma_r^{\frac{1}{2}} \tag{28}$$

$$\text{vec}(\dot{W}) = -\left(\text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2}\right) \tag{29}$$

As noted earlier, in the absence of any pre-conditioning, we do have an additional degree of freedom provided by the penalty both in terms of the evolution of $\dot{W}$ (i.e. via the penalty gradient and not just the loss gradient) and $\dot{\sigma}$ (i.e. to depend on its own relative magnitude unlike un-regularized gradient flow).

For Adam, we have:

$$\dot{\sigma}_i = -\text{vec}(\mathbf{v}_i\mathbf{u}_i^\top)^\top G\left(\text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2}\right) \tag{30}$$

$$\text{vec}(\dot{W}) = -G\left(\text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2}\right) \tag{31}$$

As noted earlier in Section 3, the key is the combination of G, a pre-conditioning that is a function of the variance of the loss gradient and appears under Adam and Adam-like variants, and the gradient penalty from our normalized nuclear norm ratio, that allows a depth 1 network (i.e. no depth) to generalize as well as a deep network, or deep factorization, and perform the same extent of rank reduction, as gradient flow/descent's implicit regularization in the presence of depth, to the point of perfect rank recovery as measured by effective rank like in [7].