Combining Implicit and Explicit Regularization for Efficient Learning in Deep Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent work on implicit regularization has focused on gradient trajectories during the optimization process in attempting to explain why deep networks favor certain kinds of solutions over others. For deep linear neural networks, it has been shown that gradient descent/flow implicitly regularizes toward low-rank solutions in matrix completion/factorization tasks, akin to an accelerative pre-conditioning, whose effects become more pronounced with increased depth. In light of this, we propose an explicit penalty that mirrors the rank minimization behavior and generalization performance independently of depth, but interestingly only takes effect with Adam and its close variants—it outperforms many approaches in matrix completion and is robust to a wide range of parameter and data regimes. Our findings suggest that explicit regularization can play a key role together with the choice of optimization algorithm in designing different, desirable forms of regularization, and that a more nuanced understanding of this interplay may be necessary.

1 Introduction

Much work has poured into understanding and explaining how and why highly over-parameterized neural networks generalize so effectively with more parameters than training examples despite long-held notions to the contrary. This generalization puzzle has deepened as over-parameterized deep learning models often generalize well even without any explicit penalization, simply by optimizing its training error on an under-determined problem.

To explain this and related phenomena [8, 9], previous works have focused on how gradient-based optimization induces an implicit bias on optimization trajectories, tending towards solutions with certain properties [6, 19] like those that generalize well or are simpler in some respect. In contrast, although explicit regularization has seen wide-spread usage for improving performance in a variety of model frameworks such as ℓ_p norm weight constraints or dropout [45], its effectiveness and role in generalization in deep networks is not certain given its inability to prevent over-fitting to random labels [52] or its absence in deep models which generalize well on their own.

Some more recent work has focused on a simple test-bed setting that admits a crisp formulation for understanding the mechanisms through which implicit regularization operates, namely, *matrix completion*: given some subset of an unknown, low-rank matrix $W^\star$, the task is to recover the unseen entries. A key observation has focused on how gradient descent on a shallow linear neural network, with sufficiently small learning rate and near-zero initialization, pushes towards low-rank solutions without explicit regularization or constraints [19]. This phenomenon led [19] to conjecture that gradient descent is inducing an implicit regularization that minimizes the nuclear norm.

This conjecture has been put into doubt by work showing that implicit regularization influences the trajectory of gradient descent in such a way so as to (a) promote solutions with few large singular

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

values and many small ones [6], with the gap becoming more pronounced as depth increases (i.e. *deep matrix factorization* by [6]), and (b) resemble a combination of *momentum* and *adaptive regularization* [5], itself akin to explicit accelerated gradient methods [28] popular in deep learning.

Contributions Despite these findings, questions still remain:

- Are there explicit norm-based penalties that can mirror the effect of implicit regularization so as to better harness this phenomenon for better or more efficient learning? Previous work [6] has conjectured that implicit regularization cannot be characterized by explicit norm-based penalties, but whether norm-based penalties can produce similar effects is unclear.
- What role does acceleration play, if at all? Can this accelerative effect be replicated or, at least, imitated? Previous work [5] conjectured that the implicit preconditioning or accelerative effect from depth is somewhat orthogonal to explicit acceleration methods.

To try and shed more light on these questions, we propose and consider the following penalty that takes the ratio between the nuclear norm of a matrix and its Frobenius norm: $\|W\|_\star / \|W\|_F$. This penalty can be viewed as an adaptation of the Hoyer measure [23] but for matrices in the spectral domain, measuring the relative spectral sparsity of a non-zero matrix W. We show the following:

1. A depth 1 linear neural network (i.e. a degenerate network without any depth) equipped with our penalty can produce the same rank reduction with comparable, or better, generalization performance as deep matrix factorization. In short, depth is no longer necessary for learning. Given that shallow networks are faster and more lightweight, these attributes also make them more efficient than their deeper counterparts.
2. The above result only occurs under Adam and, to some extent, its close variants. This suggests that different optimizers may have different inductive biases that interact differently with explicit regularizers in preferring certain solutions over others.
3. With the penalty, we achieve better generalization and rank-reduction performance beyond many techniques (Fig. 4), even in low data regimes (i.e. fewer observed entries during training) where the nuclear norm minimizer has no recovery guarantees.
4. In settings with non-trivial depth, this penalty induces the same overall effect on rank and generalization largely independent of depth, showing a degree of depth invariance.

Previous conjectures and works [5, 6, 52] have largely dismissed the importance of explicit regularization in understanding generalization in deep learning, leaving its overall role unclear. Similarly, despite its relative popularity, Adam and other adaptive optimizers have received their share of doubt [46, 47] regarding their effectiveness in producing desirable solutions. Our results suggest a subtle but perhaps important interplay between the choice of the optimizer, its inductive bias, and the explicit penalty in producing different trajectories, and, hence, different kinds of solutions. If so, then in these interactions, explicit regularization may yet have a role to play.

Paper Overview In Section 2, we review previous work. Section 3 details our experiments and findings. In Section 4, we extend our experiments to an empirical benchmark and we conclude in Section 5 with a discussion on future work.

2 Related Work

Implicit regularization and its role in explaining optimization and generalization have been studied extensively, with more recent work focusing on optimization trajectories in deep, over-parameterized settings [5–7]. One avenue has focused on linear neural networks (LNNs) [18, 21, 26, 43] given their relative tractability and the similarity in learning dynamics between linear and non-linear networks. For instance, in settings with multiple optima, [20] has shown how LNNs trained on separable data can converge with an implicit bias towards the maximum margin solution.

Others [6] have demonstrated how gradient flow converges towards low-rank solutions where increased depth acts as an accelerative pre-conditioning [5]. [2] has shown more directly, in the case of natural and vanilla gradient descent, how different choices of pre-conditioning impact bias-variance trade-offs and population risk in over-parameterized linear regression, but leaves the question open for pre-conditioning in adaptive gradient optimizers.

Acceleration in optimization dates back to Nesterov [37] and even Newton's method, using second-order information or a limited memory of past quantities (e.g. gradients) to inform optimization as a form of pre-conditioning. Adam's [28] effectiveness—and its closely inspired variants [16, 32, 33]—in optimizing deep networks faster makes the importance of adaptive pre-conditioning more explicit. Though some [46, 47] have doubted the effectiveness of adaptive gradient methods and illustrated potential issues in certain cases, such as harming generalization, others [31, 50, 53] have shown advantages from preconditioning and adaptive optimizers; despite speeding up optimization, however, the effect of pre-conditioning on generalization has been less clear. Some [1, 15, 49] have argued that the "sharpness" of minima can vary depending on the choice of optimizer.

Matrix completion and factorization have themselves long been an important focus for areas like signal recovery [11] and recommendation systems [10] while theoretical bounds for exact recovery and convergence [12, 13, 34, 40] have led to practical algorithms [24, 35]. We refer to [14] for a more comprehensive survey. These and related tasks have also served as test-beds to better understand implicit regularization. [6, 19] use matrix factorization and sensing to study gradient flow's implicit bias towards low-rank solutions and conjecture that solutions produced by algorithmic regularization may not correspond to any minimization of norm-based penalties at all. More recently, [48] studied the implicit bias of mirror descent on matrix sensing, showing its correspondence to a solution that either interpolates between the nuclear and Frobenius norms depending on the underlying matrix.

Some have started to explicitly compare implicit and explicit regularization; in certain least squares problems, [54] showed how stochastic gradient descent's (SGD) implicit biases can generalize no worse or even outperform ridge regression, requiring quadratically more samples to compete with SGD. To the best of our knowledge, our paper is the first to propose the ratio of the nuclear norm to the Frobenius norm as a penalty in matrix completion to study the interplay between explicit and implicit regularization.

3 Experiments and Findings

3.1 Setup

Formally, we have a ground-truth matrix $W^\star \in \mathbb{R}^{m\times n}$ whose observed entries are indexed by the set Ω. We define the projection $P_\Omega(W^\star)$ to be a $m \times n$ matrix such that the entries with indices in Ω remain while the rest are masked with zeros. We are interested in the following optimization:

$$\min_W \mathcal{L}(W) := \min_W \|P_\Omega(W^\star) - P_\Omega(W)\|_F^2 + \lambda R(W) \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm, $R(\cdot)$ is an explicit penalty, and $\lambda \geq 0$ is the tuning parameter. While we consider various penalties, our main focus is demonstrating the effects of our proposed penalty $R(W) = ||W||_*/||W||_F$. Following earlier works such as [3], we define a *deep linear neural network* (DLNN) through the following over-parameterization, or deep factorization, of W:

$$W = W_N W_{N-1} \dots W_1 \tag{2}$$

under the loss function in (1) where $W_i \in \mathbb{R}^{d_i \times d_{i-1}}, i \in \{1, \dots, N\}$ is the i-th layer/weight matrix. Here, N also denotes the depth of the network/factorization where $N = 2$ corresponds to *matrix factorization*, $N \geq 3$ corresponds to *deep matrix factorization*, and $N = 1$ is the degenerate case (i.e. no depth). We refer to the matrix W, the product of the N weight matrices in Eq. (2), as the *end-product matrix* as per [6], which is the network's produced solution in estimating $W^\star$. In the following analyses, we focus on rank 5 matrices as the ground truth W^* and parameterize our DLNN W with $d_0 = \dots = d_N = n = 100$ (i.e. layers/weight matrices $W_i \in \mathbb{R}^{100\times 100}, \forall i$) for illustrative purposes, but our results extend to other ranks (e.g. see Appendix A.2). We follow previous works such as [6] and employ the effective rank of a matrix [42] to track and quantify the rank of W in our experiments, which is defined as: $\text{e-rank}(W) = \exp\{H(p_1, \dots, p_n)\}$ where H is the Shannon entropy, $p_i = \sigma_i/\|\sigma\|_1$, $\sigma = \{\sigma_i\}$ are the unsigned singular values of W, and $\|\cdot\|_1$ is the ℓ_1 norm. The numerical instability of the numeric rank measure is a known issue [42], resulting in unreliable, high-variance rank estimates. We leave a detailed discussion of experiment settings to Appendix A.1.

3.2 Depth, without penalty

We first establish a baseline by reviewing the dynamics under both un-regularized gradient descent and un-regularized Adam to better understand their dynamics in the presence of our penalty later on.

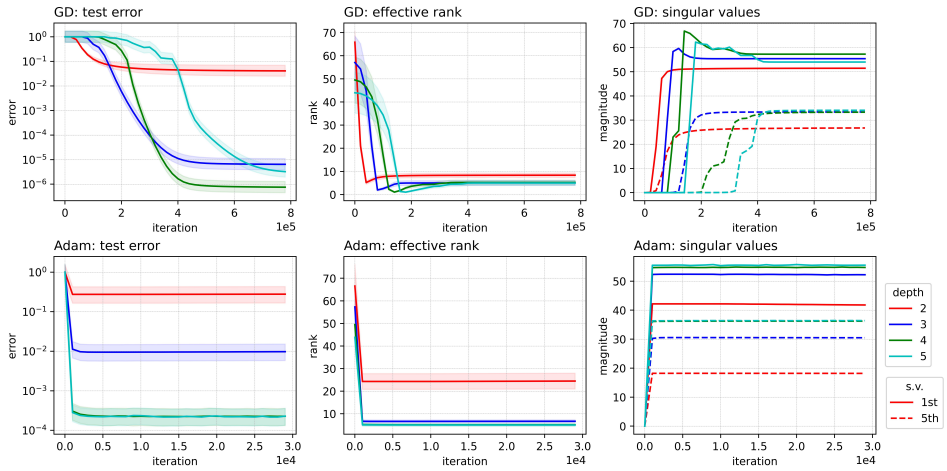

Figure 1: Dynamics of *un-regularized gradient descent (GD) and Adam*. Plots show the performance of GD over LNNs of depths 2 to 5 for rank-5 matrices of size 100×100. Colors correspond to different depth levels and shaded regions correspond to error bands. The left column depicts generalization error 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 5th largest singular values for each depth across training iterations. For singular values, within each depth level (colored lines), a solid line indicates the 1st largest singular value while a dotted line indicates the 5th largest. We omit the remaining singular values to avoid clutter.

Gradient Descent Previous work [6] has shown that depth enhances the implicit regularization towards low rank, characterized by the following trajectories on the end-product matrix W and its singular values $\{\sigma_i\}$ (for details see [5, 6]):

$$\text{vec}(\dot{W}) = -P_W \text{vec}\left(\nabla_W \mathcal{L}(W)\right) \tag{3}$$

$$\dot{\sigma}_i = -N(\sigma_i(t)^2)^{\frac{N-1}{N}} \cdot \mathbf{u}_i^\top \nabla_W \mathcal{L}(W(t)) \mathbf{v}_i \tag{4}$$

where $\dot{\sigma}_i$ is the time derivative of $\sigma_i(t)$, the i-th singular value of $W(t)$, $\{\mathbf{u}_i, \mathbf{v}_i\}$ are the left and right singular vectors of $W(t)$ corresponding to $\sigma(t)_i$, N is the network's depth, $\nabla_W \mathcal{L}(W(t))$ is the loss gradient with respect to the end-product matrix W at time t, $\text{vec}(\cdot)$ denotes (column-first order) vectorization, $\dot{W} = dW/dt$, $P_W = \sum_{j=1}^{N} (W^\top W)^{\frac{N-j}{N}} \otimes (WW^\top)^{\frac{j-1}{N}}$, and $\otimes$ denotes the Kronecker product. Moving forward, we suppress the explicit dependency on t for simplicity.

We note that full dynamics in Eq. (4) require non-degeneracy (i.e. depth $N > 1$); otherwise, they reduce down to the product of the loss gradient and singular vectors. We see these dynamics at play (Fig. 1, top row) where higher depths produce solutions with lower test error and rank (i.e. better singular value separation).

For non-degenerate cases ($N \geq 2$), as depth increases, larger singular values and their changes become more pronounced and enhanced than their smaller counterparts, driving singular value separation and rank decrease—this is seen empirically (Fig. 1 top row), resulting in a decrease in the solution's effective rank. The singular values also evolve at uneven paces across depth levels; increasing depth increases the gap in the time it takes between the 1st and 5th largest singular values to develop while also taking longer to stabilize—these effects are even more pronounced when comparing the five largest singular values to the remaining ones. This differential pace by which the singular values evolve is largely shaped, and constrained, by depth. Only with sufficient depth ($N \geq 3$) do solutions produced by un-penalized gradient descent minimize rank so as to recover the original rank of the matrix in question and generalize better than the nuclear norm solution.

For Eq. (3), the trajectory dynamics of the end-product matrix W itself (i.e. the produced solution to Eq. (1)), P_W can seen as a pre-conditoning onto the gradient that, with sufficient depth ($N \geq 2$), accelerates movements already taken in the optimization [5]. As depth (i.e. over-parameterization) increases, this acceleration intensifies.

Adam Analyzing Adam can be difficult given its exponentially moving average of gradients; to simplify our analysis, we borrow some assumptions from [4] to characterize Adam's dynamics

via gradient flow, though imperfect, by specifically assuming that the discounted gradients can be well-approximated by their expectation. (see Appendix A.4 for more details).

Theorem 1. *Under the assumptions above and of [6], the trajectory of the singular values σ_i of the end-product matrix W can be approximately characterized as:*

$$\dot{\sigma}_i = -\text{vec}(\mathbf{v}_i \mathbf{u}_i^\top)^\top P_{W,G} vec(\nabla_W \mathcal{L}(W)) \tag{5}$$

Similarly, the trajectory of the end-product matrix W itself can be approximately characterized as:

$$\text{vec}(\dot{W}) = -P_{W,G}\text{vec}(\nabla_W \mathcal{L}(W)) \tag{6}$$

where $P_{W,G} = \sum_{j=1}^{N}((W^\top W)^{\frac{N-j}{N}} \otimes (WW^\top)^{\frac{j-1}{N}})G_j$ is p.s.d. and G_j is a diagonal positive matrix. Specifically, $G_j = \text{diag}(\text{vec}(S_j))$, $[S_j]_{m,n} = [(\nabla_{W_j}\mathcal{L}(W)^2 + s_j^2)^{-1/2}]_{m,n}$, $\nabla_{W_j}\mathcal{L}(W) = \partial\mathcal{L}(W)/\partial W_j$, and $s_j^2 = \text{var}(\nabla_{W_j}\mathcal{L}(W))$.

Proof. See Appendix A.4. □

Via this approximation, the pre-conditioning induced by Adam can be characterized as a modification of gradient descent's P_W, normalizing each layer-component by the square-root of its sum of its squared layer-wise loss gradient $(\partial\mathcal{L}(W)/\partial W_j)^2$ and the gradient variance, s_j^2, before summing across all depth levels. Unlike before, the variance of the loss gradient comes into play—whereas before the pre-conditioning serves as a purely accelerative effect that intensifies with depth, normalization by a function of the variance can either dampen or further accelerate the trajectory. Empirically, in the deep regime ($N > 2$), we see that Adam minimizes rank to exact/near-exact rank recovery more efficiently and uniformly than gradient descent via faster (10^4 vs. $10^5 \sim 10^6$), more uniform convergence (Fig. 1, bottom row). Singular value dynamics exhibit significantly more uniform evolution regardless of depth in contrast to gradient descent (Fig. 1, right). Given these differences, Adam appears to arrive at different kinds of solutions than gradient descent.

3.3 Depth, with penalty

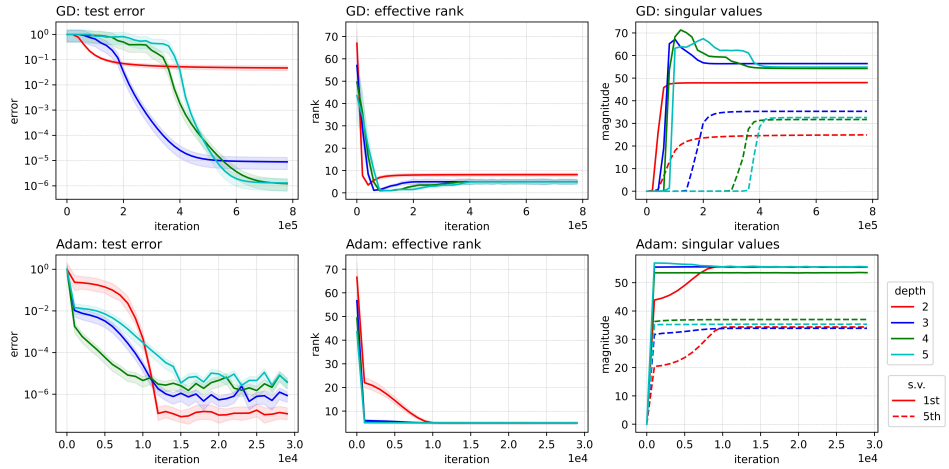

Figure 2: Dynamics of *regularized gradient descent (GD) and regularized Adam* with our penalty. Plots above show the performance of GD with our proposed penalty over DLNNs of depths 2 to 5 for rank-5 matrix completion. Setup is identical to Fig. 3. here, $\lambda = 10^{-4}$ but results hold for a range of values $\lambda \in [10^{-4}, 10^{-1}]$.

Gradient Descent We first characterize dynamics under gradient flow with our proposed penalty.

Theorem 2. *Under the assumptions of [5], with the proposed penalty, the singular values of the end-product matrix W under gradient descent with penalty (i.e. $\dot{\sigma}_r^{reg}$) evolve in the following fashion:*

$$\dot{\sigma}_r^{reg} = \dot{\sigma}_r^{GF} - \frac{\lambda N}{||W||_F^2}\left(1 - \frac{||W||_*}{||W||_F}\right)\sigma_r^{\frac{3N-2}{2}} \tag{7}$$

where $\lambda > 0$ is the regularization parameter and $\dot{\sigma}_r^{GF}$ denotes the singular value trajectory under gradient flow (Eq. (4)). Similarly, the end-product matrix W evolves in the following fashion:

$$\mathrm{vec}(\dot{W}) = -P_W \left(\mathrm{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\mathrm{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2} \right) \tag{8}$$

where U, V contain the left and right singular vectors of W, P_W is the pre-conditioning in Eq. (3), and $\tilde{\Sigma} = \frac{||W||_*}{||W||_F}\Sigma$ where Σ contains the singular values of W along its diagonal.

Proof. See Appendix A.4 for details. □

The penalty can be seen as an additive component to the original trajectories in Eq. (4). This new component also intensifies with increased depth, making movements in larger singular values more pronounced than those of smaller ones, which together can drive faster, more pronounced singular value separation than before ($\frac{2N-1}{N}$ vs. $\frac{3N-2}{2}$), depending on λ.

Empirically, comparing the un-regularized case (Fig. 1 top row) to the now regularized case (Fig. 2 top row), we indeed see that the regularized case features faster convergence in rank and faster development of the top singular values as a result of this accentuation. Here, increased depth continues to push down rank but, unlike before (Eq. (4), Eq. (3)), the penalty now allows singular value trajectories to depend on their own magnitudes even without depth ($N = 1$), providing an additional degree of freedom. The penalty also allows each singular value to depend on its relative weight within the distribution of singular values through $(1 - ||W||_*/||W||_F)$ rather than just its own absolute magnitude. We see that the depth-dependent accelerative pre-conditioning P_W now acts on a new term, similar to Eq. (7), which serves as an additional, spectral-based component that can accelerate the trajectory of the produced solution W, especially at higher depths. Looking at the diagonals, the new term can be seen as a spectrally re-scaled version of W that influences its trajectory in a way that accounts for each singular value's weight relative to its entire spectrum: $\frac{\mathbf{u}_i^\top \mathbf{v}_i}{||W||_F^2}\left(1 - \frac{||W||_*}{||W||_F}\sigma_i\right)$. The end-product matrix W produced now (Fig. 2 top row, left plot) achieves better test error and converges faster at higher depths ($N \geq 4$) compared to analogous depths in the un-regularized case (Fig. 1 top row, left plot).

Adam With Adam, the penalty's effect differs considerably in terms of the solutions produced.

Theorem 3. *Under the same assumptions, with the proposed penalty, the end-product matrix and its singular values evolve in the following fashion:*

$$\begin{aligned} \dot{\sigma} &= -\mathrm{vec}(\mathbf{v}_i \mathbf{u}_i^\top)^\top P_{W,G} \left(\mathrm{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\mathrm{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2} \right) \\ \mathrm{vec}(\dot{W}) &= -P_{W,G} \left(\mathrm{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\mathrm{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2} \right) \end{aligned} \tag{9}$$

Proof. Follows from Eq. (5) and Eq. (6) with our penalty. See Appendix A.4 for details. □

Empirically, we note a new development: there is a larger degree of depth invariance in that rank is pushed down almost independently of depth (Fig. 2, bottom row), even at depth 2. All depths recover the rank of the underlying matrix (i.e. 5) unlike gradient descent. Starting with a shallow network ($N = 2$), increasing the depth does not induce any material changes in the produced solutions as the penalized DLNN under Adam produces low-rank solutions that achieve exact rank recovery and low generalization error on par with, if not better than, before (Fig. 2). It retains some of the accelerative effect of Adam with respect to the speed of convergence and behavior of effective rank and how the singular values develop. Compared to before, we see a beneficial accelerative effect that endows more favorable generalization properties and smoother development of singular values whose convergence speed is at least an order of magnitude faster than under gradient descent—which do not change much with depth. As singular values evolve, we see that their paths are relatively robust to depth, all producing the near-identical behavior with much less dependence on depth (Fig. 2, right).

3.4 Degenerate case, with penalty

Given the non-trivial interaction effects seen thus far, we now consider the degenerate depth 1 network. It is known [52] that gradient descent in this setting converges to the minimum Frobenius (i.e. ℓ_2) norm solution, which does not necessarily induce spectral sparsity or low ranked-ness. Applying our

penalty to this network under gradient descent does little to affect the solution meaningfully (Fig. 3). In this case, without depth, learning is heavily constrained.

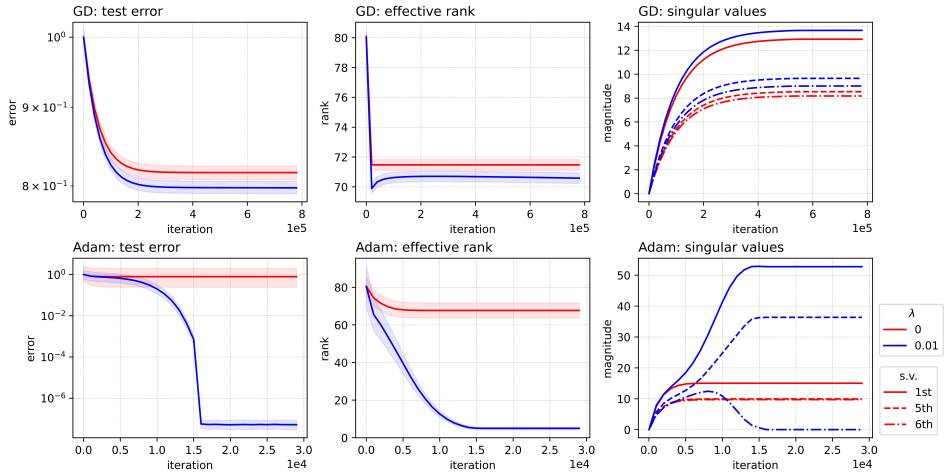

Figure 3: Performance comparison between choice of optimization and regularizer in a depth-1 linear network. Top row corresponds to gradient descent and bottom corresponds to Adam. Note that $\lambda = 0$ corresponds to the un-regularized setting. Here, $\lambda = 10^{-4}$ but results hold for values $\lambda \in [10^{-6}, 10^{-1}]$.

Yet, surprisingly, despite being ineffective with gradient descent, our penalty is effective under Adam (Fig. 3, bottom) even without depth or over-parameterization, achieving generalization on par with, if not better than, deep matrix factorization ($N > 2$) under gradient descent. We note that replacing our penalty with other proxies of rank/spectral sparsity such as the nuclear norm does not work (Table 1). Under un-regularized gradient flow, at depth 1, singular value and end-product dynamics collapse: singular value trajectories reduce to $\mathbf{u}_i^\top \nabla_W \mathcal{L}(W(t))\mathbf{v}_i$, as noted earlier, and the accelerative pre-conditioning reduces to just the identity $P_W = I_{mn}$. We see this empirically (e.g. Fig. 3 and Table 1) where solutions produced un-regularized gradient descent at depth 1 generalize poorly and fail to reduce rank. By removing depth, we see that a key component in helping a depth 1 network learn is its variance-weighted gradient term $\nabla_W \mathcal{L}(W) \cdot G$, as defined in Eq. (6), along with the term $P_{W,G}\left(\lambda \frac{\mathrm{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2}\right)$, which reduces to $G\left(\lambda \frac{\mathrm{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2}\right)$ without depth. Interestingly, the variance of the loss gradient and the ratio $\eta^2 = \mathrm{var}(\nabla_W \mathcal{L}(W))/\nabla_W \mathcal{L}(W)^2$ formed from $\nabla_W \mathcal{L}(W) \cdot G$, akin to a signal-to-noise ratio, have come up in other works as important quantities that is strongly predictive of generalization capabilities [25] or is essential to finding optimal variance adaption factors of stochastic gradients [4].

3.5 Comparison with other penalties and optimizers

Other Combinations Our results do not preclude the possibility that other optimizers and penalties can produce similar or better effects. For completeness, inspired by the properties of our proposed penalty (e.g. ratio-like, spectral, non-convex), we experiment with various optimizers and penalties in Table 1 to compare optimizer-penalty interactions across shallow ($N = 1$) and deep ($N = 3$) settings. We note that our ratio penalty under Adam and its close variant Adamax, both initially proposed in [28], are the only combinations that largely show depth invariance in test error and rank recovery whereas others require depth to either push down rank or generalize better. Although the nuclear norm shows a degree of depth invariance in rank reduction, it is unable to find well-generalizing solutions and it fails in rank reduction under gradient descent where depth is still necessary. Given the space considerations of this paper, we leave a fuller treatment and characterization of each optimizer's inductive biases and their interactions with different regularizers for future work.

Comparative Performance Finally, we note our penalty's comparative performance (Fig. 4) against other methodologies for synthetic matrix completion across various sample sizes (i.e. the number of entries out of 100,000 made available for training). Our depth 1 network with Adam and the penalty outperforms other methods including the un-regularized DLNN in test error, degree of rank compression, and convergence speed across several data regimes. At lower sample sizes, even at

Table 1: 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). **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** the nuclear norm penalty, **None** is no penalty, and $a\cdot$ e b denotes $a \cdot 10^b$. Best results (i.e. test error, rank reduction, and depth invariance) in bold. For more results and details, see Appendix A.3.

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 [28]	1	**4e−7**	**5**	0.72	33	0.80	45	0.81	53	0.36	6	1.00	79
	3	**4e−7**	**5**	3e−6	5	1e−5	5	6e−6	5	0.30	5	0.04	6
Adagrad [17]	1	0.58	31	0.81	60	0.97	32	0.79	60	0.12	8	0.80	70
	3	3e−7	5	9e−7	5	1e−5	5	2e−7	5	0.05	6	4e−3	6
Adamax [28]	1	**4e−7**	**5**	0.76	44	0.85	22	0.80	58	0.05	6	0.81	72
	3	**7e−7**	**5**	3e−6	5	7e+5	1	6e−6	5	0.07	7	0.01	7
RMSProp	1	2e−4	6	0.08	4	1.6e+3	5	1.8e+3	8	0.05	8	0.80	70
	3	0.03	5	8e−4	5	2e−3	5	1.9	5	0.05	6	0.11	14
GD	1	0.81	67	0.81	62	0.80	47	0.81	60	0.82	59	0.83	72
	3	0.51	3	0.25	5	0.56	3	0.39	5	0.24	4	1e−5	5

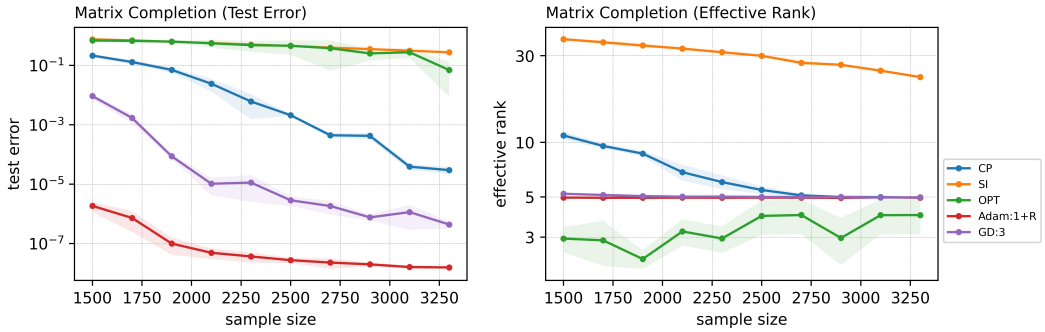

Figure 4: Comparative performance in generalization error and rank minimization for rank-5 matrix completion (100×100). x-axis stands for the number of observed entries (uniformly sampled) and shaded regions indicate standard error bands. **Adam:1+R** refers to Adam at depth 1 plus our penalty, **CP** is the minimum nuclear norm solution, **GD:3** is a depth 3 network trained with gradient descent, **OPT** is `OptSpace` [27], and **SI** is `SoftImpute` [35]. To reduce clutter, we omit results with similar performance (e.g. GD:4, GD:5 etc.).

depth 1, the penalized network generalizes better than methods such as `SoftImpute` [35], `OptSpace` [27], the minimum nuclear norm solution [13], and a DLNN with gradient descent ($N \geq 3$) by at least an order of magnitude. Its performance over other methods not only persists across various data regimes including under scarce data but also improves with larger sample sizes.

4 Results on real world data

Lastly, a natural question might be: how well do the effects of our penalty extend to real-world data? To answer this, we consider MovieLens100k, a popular benchmark for recommendation systems, consisting of 944 users and ratings on 1,683 movies that form an interaction matrix $M \in \mathbb{R}^{944\times1683}$ with the goal of predicting the rest of M after having observed some subset of M.

Unlike our earlier experiments, the values here are discrete in the range $\{1, 2, 3, 4, 5\}$ and M is full-rank. Given these differences and conventional approaches taken in empirical recommendation systems, we apply our penalty in two ways. The first way is as before, training a depth 1 network with Adam and the penalty (Depth 1 LNN, Table 2). The second way is to impose our penalty on a classic user-item embedding model (User-Item Embedding, Table 2 [29]) that combines user-specific and item-specific biases with a dot product between a latent user embedding and a latent item embedding, similar to a shallow explicit factorization model; we apply our penalty separately on either the item embedding layer or the user embedding layer. Though this approach utilizing explicit ratings has fallen out of favor in lieu of more recent ones (e.g. graph-based, implicit feedback, deep networks)

Table 2: Performance evaluations on MovieLens100k. Results are categorized by model type, whether or not additional data/features were used, and test error (RMSE) on 90:10 and 50:50 training-test splits.

Model	Uses side info. or other features?	90% RMSE	50% RMSE
Depth 1 LNN	No		
w. Adam		1.844	1.877
w. Adam+penalty		**0.918**	**0.948**
User-Item Embedding	No		
w. Adam		0.956	0.978
w. Adam+penalty		**0.906**	**0.950**
PMF [36]	No	0.952	0.977
SVD++ [30]	No	0.913	0.938
NFM [22]	No	0.910	0.935
FM [41]	No	0.909	0.935
GraphRec [39]	No	0.898	0.932
AutoSVD++ [44]	Yes	0.904	0.926
GraphRec+graph feat. [39]	Yes	0.887	0.918
GraphRec+graph feat.+side info. [39]	Yes	0.883	0.911

utilizing additional information and complex designs, we nonetheless evaluate the effects of our penalty with this simple framework. With these two approaches, we compare their results with those from both older and more recent state-of-the-art techniques, reflecting a wide variety of approaches and architectures that use additional side information beyond M to improve performance (Table 2).

Surprisingly, we see that the test performance of a simple depth 1 network with Adam and the penalty (**w. Adam+penalty**) can come close to other more specialized approaches—but, in contrast to the others, it does so without any specialized architecture or parameterization, additional side information beyond M, creation of new features, or even any factorization at all. Moreover, we see that the performance of the simple user-embedding model also comes close to the other approaches, even one (e.g. AutoSVD++) that leverages additional information/features, despite not doing so itself. We leave details of our experimental settings to Appendix A.1.

5 Discussion

The dynamics of optimization trajectories—influenced by the choice of optimizer, parameterization, explicit penalties, and even architecture—can play an important role in generalization. With neural networks, depth and gradient descent-like algorithms have been key ingredients to some of its successes. In our work focusing on matrix completion, our penalty achieves well-generalizing solutions and perfect rank recovery even with a depth 1 (i.e. no depth) network. Does that mean our penalty, together with Adam's own inductive bias, is producing an effect similar to implicit regularization under gradient descent with depth, but better? Is it a coincidence that our penalty only obtains such performance with Adam?

We suspect not. While we concur with the overall spirit of the conjecture in [6]—namely, a *reductionist* view of implicit regularization which suggests that it can be entirely encapsulated by an explicit norm-based penalty is likely an incorrect over-simplification—we believe that there is merit in studying and relating implicit regularization to its similar explicit counterparts, as our exposition highlights, to examine their interplay. Our work suggests that we may be able to partially replicate the empirical successes of *deep* learning by selectively combining optimization methods with explicit penalties via better model design or encoding of inductive biases for specific tasks.

Many questions remain open from our limited analyses which, due to space considerations, we leave as directions for future work. For instance, how well do lessons from DLNNs translate to their non-linear counterparts on other tasks (e.g. classification)? How does this relate to learning regimes with larger learning rates (i.e. beyond gradient flow)? A more rigorous analysis of the properties (e.g. convergence) of Adam, adaptive gradient methods, and other optimizers in the presence of explicit regularizers may better our understanding. It remains unclear whether implicit regularization is a bug or a feature in deep, over-parameterized settings. Nonetheless, our findings suggest that it can be harnessed and transformed to desirable effect.

References

[1] Shun-ichi Amari, Jimmy Ba, Roger Grosse, Xuechen Li, Atsushi Nitanda, Taiji Suzuki, Denny Wu, and Ji Xu. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[2] Shun-ichi Amari, Jimmy Ba, Roger Grosse, Xuechen Li, Atsushi Nitanda, Taiji Suzuki, Denny Wu, and Ji Xu. When does preconditioning help or hurt generalization? *International Conference on Learning Representations (ICLR)*, 2021.

[3] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. Why are deep nets reversible: A simple theory, with implications for training. *CoRR*, abs/1511.05653, 2015.

[4] Sanjeev Arora, Nadav Cohen, and Elad Hazan. Dissecting Adam: the sign, magnitude and variance of stochastic gradients. In *Proceedings of the 35th International Conference on Machine Learning (PMLR) 2018*, 2018.

[5] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *35th International Conference on Machine Learning (ICML)*, 2018.

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

[7] Bubacarr Bah, Holger Rauhut, Ulrich Terstiege, and Michael Westdickenberg. Learning deep linear neural networks: Riemannian gradient flows and convergence to global minimizers. *arXiv preprint arXiv:1910.05505*, 2019.

[8] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences of the United States of America*, vol. 80:201907378, April 2020.

[9] Mikhail Belkin, Alexander Rakhlin, and Alexandre B Tsybakov. Does data interpolation contradict statistical optimality? *arXiv.org*, 2018.

[10] James Bennett, Stan Lanning, et al. The netflix prize. Citeseer, 2007.

[11] Emmanuel J. Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2009.

[12] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[13] Emmanuel J Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[14] Yuejie Chi, Yue M Lu, and Yuxin Chen. Non-convex optimization meets low-rank matrix factorization: an overview. *IEEE Transactions on Signal Processing*, 67(20):5239–5269, 2019.

[15] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

[16] Timothy Dozat. Incorporating nesterov momentum into adam. *ICLR Workshop Track*.

[17] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[19] Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pages 6152–6160, 2017.

[20] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pages 9461–9471, 2018.

[21] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

[22] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364, 2017.

[23] Niall Hurley and Scott Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741, 2009.

[24] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pages 665–674, 2013.

[25] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measurs and where to find them. *International Conference on Learning Representations (ICLR)*, 2020.

[26] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

[27] Raghunandan H Keshavan and Sewoong Oh. A gradient descent algorithm on the grassman manifold for matrix completion. *arXiv preprint arXiv:0910.5260*, 2009.

[28] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

[29] Robert Bell Koren, Yehuda and Chris Volinsky. Matrix factorization techniques for recommender systems.

[30] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *14th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 426–434, 2008.

[31] Daniel Levy Levy and John C Duchi. Necessary and sufficient geometries for gradient methods. In *Advances in Neural Information Processing Systems*, pages 11491–11501, 2019.

[32] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. abs/1908.03265, 2019.

[33] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. abs/1711.05101, 2017.

[34] Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion, and blind deconvolution. *Foundations of Computational Mathematics*, pages 1–182, 2019.

[35] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research (JMLR)*, 11: 2287–2322, 2010.

[36] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2008.

[37] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[38] journal=HAL Open Science, hal-03434195 year=2022 Ramuzat, Noelie. On the semi-positive definition of the product of a positive diagonal matrix and a symmetric semi-positive definite matrix.

[39] Ahmed Rashed, Josif Grabocka, and Lars Schmidt-Thieme. Attribute-aware non-linear co-embeddings of graph features. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 314–321, 2019.

[40] Benjamin Recht. A Simpler Approach to Matrix Completion. 12(104):3413–3430, 2011.

[41] Steffen Rendle. Factorization machines. *IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[42] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *15th IEEE European Signal Processing Conference*, pages 606–610. IEEE, 2007.

[43] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[44] Lina Yao Shuai Zhang and Xiwei Xu. Autosvd++: an efficient hybrid collaborative filtering model via contractive auto-encoders. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, RecSys '19, page 957–960, 2017.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014.

[46] Neha S Wadia, Daniel Duckworth, Samuel S Schoenholz, Ethan Dyer, and Jascha Sohl-Dickstein. Whitening and second order optimization both destroy information about the dataset, and can make generalization impossible. *arXiv preprint arXiv:2008.07545*, 2020.

[47] Ashia Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Rech. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.

[48] Fan Wu and Patrick Rebeschini. Implicit regularization in matrix sensing via mirror descent. In *Advances in Neural Information Processing Systems*, pages 20558–20570, 2021.

[49] Lei Wu, Chao Ma, and E Weinan. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. In *Advances in Neural Information Processing Systems*, pages 8279–8288, 2018.

[50] Peng Xu, Fred Roosta, and Michael W Mahoney. Second-order optimization for non-convex machine learning: An empirical study. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 199–207, 2020.

[51] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. abs/1212.5701, 2012.

[52] Chiyuan Zhang, Benjamin Recht, Samy Bengio, Moritz Hardt, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, (ICLR)*, 2017.

[53] Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Gross. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, pages 8194–8205, 2019.

[54] Difan Zou, Jingfeng Wu, Vladimir Braverman, Quanquan Gu, Dean P Foster, and Sham Kakade. The benefits of implicit regularization from sgd in least squares problems. In *Advances in Neural Information Processing Systems*, pages 5456–5468, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 3, Section 5, and Appendix.

(c) Did you discuss any potential negative societal impacts of your work? [N/A] Our work does not present any foreseeable societal impacts or consequences.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3 and Appendix.

(b) Did you include complete proofs of all theoretical results? [Yes] See Section 3 and Appendix.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary material.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] Results under our proposed penalty are valid even across varying degrees of regularization strength (λ) and several initial learning rates α and across several initializations. Other results that were run with random seeds had negligible standard errors (see implementation details under the Appendix).

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [N/A]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Implementation Details, Further Experiments, & Derivations

A.1 Implementation Details

In this section, we provide implementation details of 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 modify and build upon the `Spotlight` repo [1] which uses PyTorch to provide common building blocks for different kinds of recommendation models. Experiments were run on a NVIDIA Tesla V100 GPU.

Synthetic Data For synthetic data experiments, our implementation details mirror the experimental design of [6], 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 were trained by gradient descent, or whichever other optimizer under consideration, via full-batch training applied to the Frobenius loss (i.e. the ℓ_2 loss) over the observed entries.

Weights are initialized via independent samples from a Gaussian distribution with zero mean and standard deviation 10^{-3}. Learning rates are fixed through training; specifically, aligning with 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 $5 \cdot 10^6$ iterations or training loss reaching values below 10^{-7}, whichever applies sooner. Moreover, results relating to the ratio penalty are robust across varying strengths of regularization $\lambda \in [10^{-6}, 10^{-1}]$ and training is full-batch, leaving no room for noise like that in mini-batch training. For Adam, aside from the initial learning rate, the default hyper-parameters under PyTorch's implementation were used, which include $(\beta_1, \beta_2) = (0.9, 0.999)$, no `amsgrad`, etc. unless otherwise stated. Similarly for experiments involving other optimizers, aside from the initial learning rate, we use the default hyper-parameter settings 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 [42] due to its favorable properties over numerical rank in simulation settings and its usage in [6]. The test error with respect to a ground truth matrix $W^\star$ is given by the mean-squared error *i.e.* for a solution W it is $\frac{1}{N}\|W - W^\star\|_F^2$ but results continue to hold across other measures of test error. For rank 5 experiments where the sample size was fixed, sample size is set at 2000 (again, following [6]) 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 base available in `Spotlight`, as mentioned earlier, in conjunction with Adam and our ratio penalty. The explicit factorization model (i.e. baseline) consists of several components akin to a shallow, depth 2 LNN model (with the exception of bias terms and separate inputs into separate embedding layers): a user-embedding layer, an item-embedding layer, a 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 $\alpha \in [5 \cdot 10^{-3}, 10^{-5}]$. For hyper-parameters combination, the model is trained for 100 epochs. 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, and $\lambda = 0.01$.

For our depth 1 LNN, we take an approach near-identical to our synthetic experiments (e.g. same initialization and training/stopping criteria, full-batch training, etc.). For our hyper-parameter space, 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.

[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.

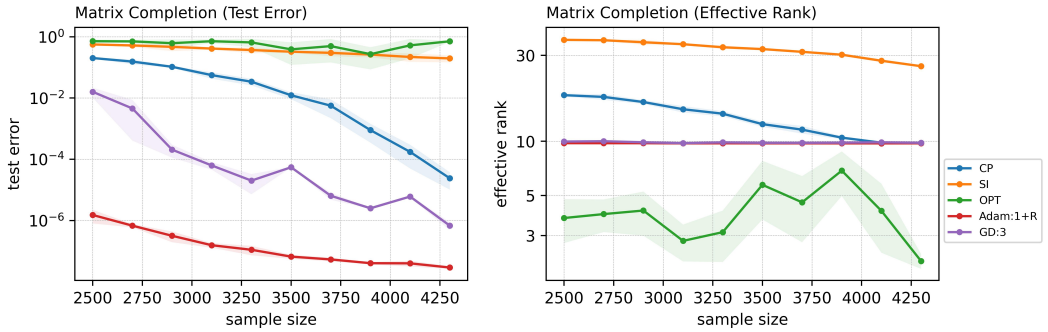

Figure 5: Comparative performance in generalization error and rank minimization for rank 10 matrix completion (of size 100×100). x-axis stands for the number of observed entries (uniformly sampled) and shaded regions indicate standard error bands. **Adam:1+R** refers to Adam at depth 1 plus our penalty, **CP** is the minimum nuclear norm solution, **GD:3** is gradient descent with depth 3 (deep matrix factorization), **OPT** is `OptSpace` [27], and **SI** is `SoftImpute` [35]. To reduce clutter, we omit results with similar performance (e.g. GD:3 vs. 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.

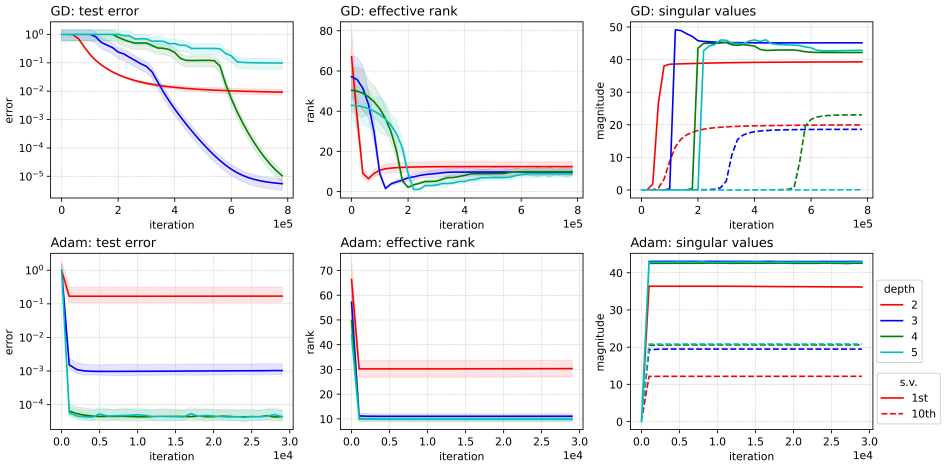

Figure 6: Dynamics of gradient descent and Adam over DLNN. Plots above show the performance of gradient descent over DLNNs of depths 2 to 5 for rank 10 matrices of size 100×100. The top row depicts gradient descent and the bottom depicts Adam. The left column depicts generalization error as a function 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 1st largest singular value while a dotted line indicates the 10th 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.

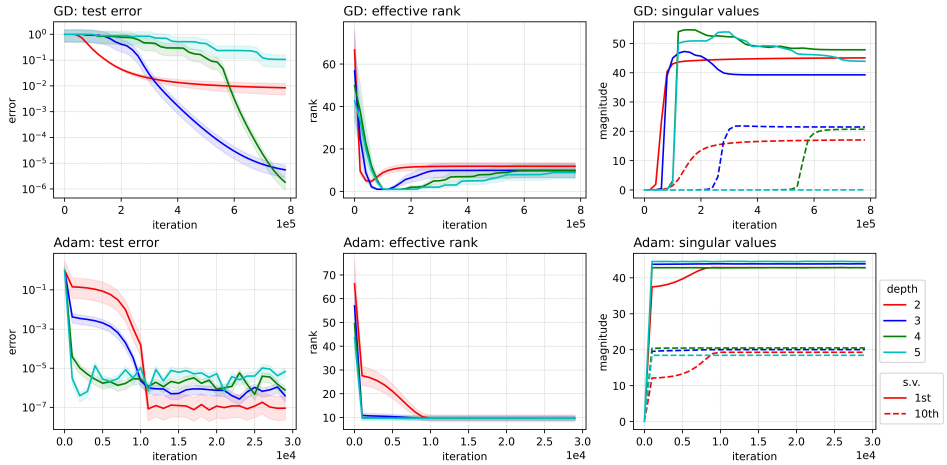

Figure 7: Dynamics of regularized gradient descent and regularized Adam with our penalty. Plots show the performance over DLNNs of depths 2 to 5 for rank 10 matrix completion. The penalty's hyper-parameter is set at $\lambda = 10^{-4}$ but results hold for a range of values $\lambda \in [10^{-6}, 10^{-1}]$.

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.

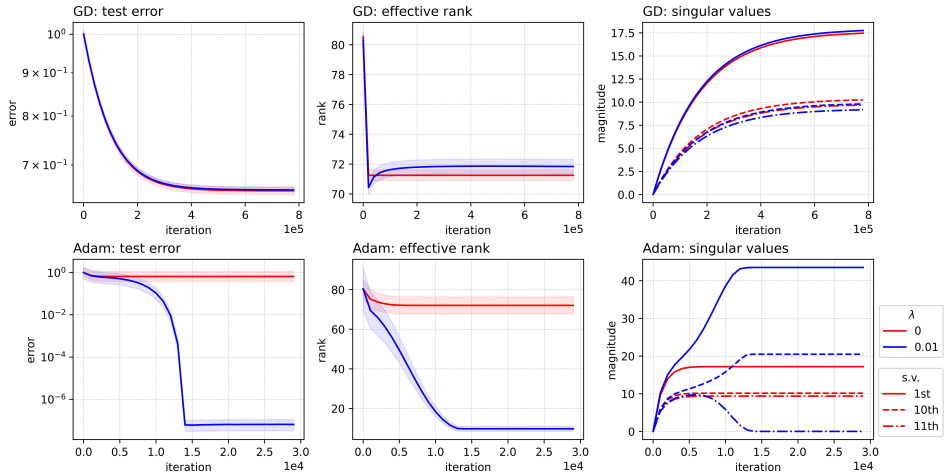

Figure 8: Performance comparison between choice of optimization and regularizer in a depth-1 linear network. Top row corresponds to gradient descent and bottom corresponds to Adam. Note that $\lambda = 0$ corresponds to the un-regularized setting. The penalty's hyper-parameter is set at $\lambda = 0.01$ 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 [28] (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 [51]	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 [33]	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 [16]	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 [32]	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).

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 [16, 28, 32, 33] 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 [5, 6]. We highlight the main derivations as they relate to the theorems in this paper and defer to the appendices of [5, 6] 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 [5] 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 [5, 6] 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 $\tilde{\mathcal{L}}(W) := \mathcal{L}(W) + R(W)$

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 [28]							
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 [17]	1	1.00	12	0.97	2	0.70	15
	3	0.81	1	0.85	1	0.08	5
Adamax [28]	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 [51]	1	0.99	60	1.05	48	1.00	60
	3	0.99	56	0.94	1	0.99	56
AdamW [33]	1	1.00	48	0.97	9	0.67	6
	3	1.00	10	0.99	1	0.82	2
NAdam [16]	1	1.00	47	0.98	8	0.80	7
	3	1.00	3	0.92	1	0.71	3
RAdam [32]	1	1.00	46	1.01	25	0.57	7
	3	0.99	7	0.91	1	0.68	3

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 [5] 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:

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

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}; \qquad m_t = (1-\beta_1)m_{t-1} + \beta_1 g_{t-1}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}; \qquad v_t = (1-\beta_2)v_{t-1} + \beta_2 g_{t-1}^2$$

for each layer $j \in \{1, \ldots, N\}$ in a depth N network where 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 [4], 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:

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

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 [4], 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 [6], 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):

$$\dot{W}_j = -\frac{\nabla_{W_j}\mathcal{L}(W)}{\sqrt{\nabla_{W_j}\mathcal{L}(W)^2 + s_j^2}} \tag{14}$$

where, again, division, etc. here denote element-wise operations. As an aside, we note that the expression in Eq. (14) can also be written as $\dot{W}_j = -(1+\eta_j^2)^{-1/2}$ where $\eta_j^2 := \frac{s_j^2}{\nabla_{W_j}\mathcal{L}(W)^2}$, 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 [4, 28]. Using the definition of $\nabla_{W_j}\mathcal{L}(W)$ from Eq. (13), we can rewrite the right-hand side of Eq. (14) as:

$$-\prod_{i=j+1}^{N} W_i^\top (\nabla_W \mathcal{L}(W)) \prod_{i=1}^{j-1} W_i^\top \odot S_j \tag{15}$$

where $\odot$ denotes the Hadamard product and S_j is a matrix that contains the corresponding entry-wise elements of $1/\sqrt{\nabla_{W_j}\mathcal{L}(W)^2 + s_j^2}$ 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 [6] but with several modifications due to presence of S_j. For the full details, we defer to the appendix of [6]. We left and right multiply by a product of other weight layers and sum across the depth of the network:

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

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

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

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 $\text{vec}(ALB)$, and the sixth equality is the result of the mixed-product property of Kronecker products. Previous work [5] 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:

$$\text{vec}(\dot{W}) = -P_{W,G}\text{vec}(\nabla_W \mathcal{L}(W)) \tag{17}$$

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, [5] 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,

$$P_j = A((D^\top D)^{\frac{N-j}{N}} \otimes (DD^\top)^{\frac{j-1}{N}})A^\top \tag{18}$$

where $A = V \otimes U$, 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,

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

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. [38]. 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 [5], P_W in the original case of un-regularized gradient flow serves as an accelerative pre-conditioning that intensifies with depth and can only exist with sufficient depth ($N \geq 2$). More specifically, they 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 degenerate case $N = 1$, however, the pre-conditioning P_W collapses into the identity, causing all of its non-zero eigenvalues to reduce down to just unity—resulting in no accelerative or favoring of any eigen-direction(s) during the optimization process. In contrast, the eigenvalues of the pre-conditioning $P_{W,G}$ induced by Adam results in a re-scaling of the original eigenvalues of P_W. At non-trivial depth $N \geq 2$, the eigenvalues of $P_{W,G}$ corresponding to its eigenvectors $\text{vec}(\mathbf{u}_r \mathbf{v}_{r'}^\top)$ take the form: $\sum_{j=1}^{N} \sigma_r^{2\frac{N-j}{N}} \sigma_{r'}^{2\frac{j-1}{N}} [(1+\eta_j^2)^{-1/2}]_{r,r'}$ for $r \in \{1, \ldots, m\}$ and $r' \in \{1, \ldots, n\}$, where $\eta_j^2 = s_j^2 / \nabla_{W_j} \mathcal{L}(W)^2$ (again, element-wise division) and $[\cdot]_{r,r'}$ denotes the (r, r')-th diagonal entry of the matrix. As such, the eigen-values are now scaled or normalized by the square root of $1 + \eta_j^2$, a quantity akin to a measure of relative variance, noise-to-signal ratio, or a variance adaptation factor that can help further promote acceleration in directions of low relative variance, or more reliable updates, and vice versa in directions of high relative variance. At degenerate depth, $P_{W,G}$'s non-zero eigenvalues, unlike P_W, are no longer just unity and static, producing 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.

Evolution of σ under Adam without penalty. For the trajectories of the singular values $\{\sigma_i\}$, we largely build atop the approach in [6], 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:

$$\text{vec}(\dot{\sigma}_i) = \dot{\sigma}_i = (\mathbf{u}_i^\top \otimes \mathbf{v}_i^\top)\text{vec}(\dot{W}) = \text{vec}(\mathbf{v}_i \mathbf{u}_i^\top)^\top \text{vec}(\dot{W}) \tag{19}$$

We can then substitute our expression for $\text{vec}(\dot{W})$ from Eq. (17) to characterize the singular values under Adam:

$$\dot{\sigma}_i = -\text{vec}(\mathbf{v}_i \mathbf{u}_i^\top)^\top P_{W,G} \text{vec}(\nabla_W \mathcal{L}(W)) \tag{20}$$

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 [5] 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. (17) 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:

$$\text{vec}(\dot{W}) = -P_{W,G}\left(\text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2}\right) \tag{21}$$

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. (21) 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.

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. (19) and substitute in the appropriate expression for $\text{vec}(\dot{W})$ in the case of Adam with the penalty:

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

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:

$$\dot{\sigma}_i = -\mathbf{u}_i^\top \nabla_W \mathcal{L}(W)\mathbf{v}_i \tag{23}$$

$$\text{vec}(\dot{W}) = -\text{vec}\left(\nabla_W \mathcal{L}(W)\right) \tag{24}$$

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:

$$\dot{\sigma}_i = -\text{vec}(\mathbf{v}_i\mathbf{u}_i^\top)^\top (G \cdot \text{vec}(\nabla_W \mathcal{L}(W))) \tag{25}$$

$$\text{vec}(\dot{W}) = -G \cdot \text{vec}(\nabla_W \mathcal{L}(W)) \tag{26}$$

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:

$$\dot{\sigma}_r = -\mathbf{u}_i^\top \nabla_W \mathcal{L}(W)\mathbf{v}_i - \frac{\lambda}{||W||_F^2}\left(1 - \frac{||W||_*}{||W||_F}\right)\sigma_r^{\frac{1}{2}} \tag{27}$$

$$\text{vec}(\dot{W}) = -\left(\text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top - U\tilde{\Sigma}V^\top)}{||W||_F^2}\right) \tag{28}$$

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:

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

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

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 [6].