Combining Implicit and Explicit Regularization for Efficient Learning in Deep Networks

Dan Zhao
New York University
dz1158@nyu.edu

Abstract

Works studying implicit regularization have focused on gradient trajectories during the optimization process in explaining why deep networks favor certain kinds of solutions over others. For deep linear networks, it has been shown that gradient descent implicitly regularizes toward low-rank solutions in matrix completion/factorization tasks. Akin to an accelerative pre-conditioning, these effects become more pronounced with increased depth. Inspired by this, we propose an explicit penalty that mirrors the rank minimization and generalization performance independently of depth but interestingly only takes effect with certain adaptive gradient optimizers (e.g. Adam and some of its close variants)—performing competitively with or outperforming several approaches in matrix completion over a range of parameter and data regimes. Together with the choice of the optimization algorithm, our findings suggest that explicit regularization can play an important role 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 why and how highly over-parameterized, deep neural networks with more parameters than training examples generalize so effectively despite long-held notions to the contrary. This generalization puzzle has only deepened as deep learning models often generalize well simply by optimizing its training error on an under-determined problem.

To explain this [12, 13], previous works have focused on how gradient-based optimization induces an implicit bias on optimization trajectories, particularly in deep (i.e. over-parameterized) settings, tending towards solutions with certain properties [7, 27] like those that generalize well. In contrast, while explicit regularization has seen wide-spread usage in various settings (e.g. weight decay, dropout [60]), its role in explaining generalization has been less certain given its inability to prevent over-fitting on random labels [70] or its absence in deep models that generalize well on their own.

Some works have focused on a simple test-bed to formalize and isolate the mechanisms through which implicit regularization operates—namely, *matrix completion*. Given some observed subset of an unknown, low-rank matrix $W^\star$, the task is to recover the unseen entries. A key observation [27] has been how gradient descent on a shallow linear neural network, with sufficiently small learning rate and near-zero initialization, pushes towards low-rank solutions on its own. This has led to the conjecture [27] that gradient descent induces implicit regularization that minimizes the nuclear norm.

This conjecture has been put into doubt by work [7] showing that gradient descent not only promotes low-rank solutions in the shallow case on matrix completion tasks, but its implicit regularization is further strengthened with increased depth (i.e. via a deep linear neural network [7]), leading to solutions with lower rank and more accurate completion than those corresponding to the nuclear

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

norm. Others [6] have also shown how increased depth or over-parameterization can provide an accelerative pre-conditioning to regression problems for faster convergence on deep linear networks.

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 improved or more efficient learning? Previous work [7] has conjectured that implicit regularization cannot be characterized by explicit norm-based penalties, but whether these penalties can produce similar effects is unclear.
- Do implicit and explicit forms of regularization interact in any meaningful way? Can we modify the implicit biases of optimizers with explicit regularizers so as to promote better kinds of performance? Some work [11] has begun to draw inspiration from implicit regularization to create explicit regularizers, but their interactions are less clear.
- Previous works [6, 7] have shown depth, or over-parameterization, can act as a pre-conditioner to accelerate convergence or enhance implicit tendencies towards certain solutions. Can this accelerative effect be partially replicated or, at least, imitated without depth? It has been conjectured that the accelerative effect or implicit preconditioning from depth is somewhat orthogonal to, and unlikely to be produced by, explicit regularization methods.

To try and shed more light on these questions, we propose an explicit penalty that takes the ratio between the nuclear norm of a matrix and its Frobenius norm ($\|W\|_\star / \|W\|_F$) and study its effects on the task of matrix completion. This penalty can be interpreted as an adaptation of the Hoyer measure [32] to the spectral domain or as a particular normalization of the nuclear-norm penalty that is commonly used as a proxy for rank in convex relaxations of the problem.

Studying implicit regularization can be difficult as it is not always possible to account for all other sources of implicit regularization. For a more precise study of the implicit biases of optimizers and their interactions with our penalty, we use matrix completion and deep linear networks as tractable, yet expressive and well-understood test-beds that admit a crisp formulation of the mechanism through which implicit regularization operates [7, 28]. In particular, we show the following:

1. A depth 1 linear neural network (i.e. a degenerate network without any depth) trained with this penalty can produce the same rank reduction and deliver comparable, if not better, generalization performance as a deep linear network or deep matrix factorization. In short, depth is no longer necessary for learning here, making learning more efficient.
2. The above result only occurs under Adam and, to some extent, its close variants. This suggests that different optimizers, each with their own inductive biases, can interact differently with explicit regularizers in promoting certain solutions over others.
3. With the penalty, we can achieve better generalization and rank-reduction performance beyond various other 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 networks 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.

These properties can make deep linear networks faster and more efficient for a variety of purposes relating to matrix completion, estimation, or factorization, ranging from applications in efficient reinforcement learning [58] to adversarial robustness [68] and others [1].

Previous conjectures and works [6, 7, 70] 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 [61, 62] 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 a popular real-world benchmark and compare our method with other popular methods. We conclude in Section 5 with a discussion on future work.

2 Related Work

Implicit regularization has been studied extensively, with more recent work focusing on optimization trajectories in deep, over-parameterized settings [6, 7, 10]. One avenue has focused on linear neural networks (LNNs) [26, 29, 37, 57] given their relative tractability and the similarity in learning dynamics between linear and non-linear networks. In settings with multiple optima, [28] has shown how LNNs trained on separable data can converge with an implicit bias to the max margin solution.

Others [7] have demonstrated how gradient flow converges towards low-rank solutions where depth acts as an accelerative pre-conditioning [6]. In [3], for natural and vanilla gradient descent (GD), different kinds of pre-conditioning are shown to impact bias-variance and risk trade-offs in over-parameterized linear regression, but this is less clear in the case of adaptive gradient optimizers.

Building upon the study of acceleration and pre-conditioning dating back to Nesterov [50] and Newton, Adam's [39] effectiveness—and its closely inspired variants [24, 44, 45]—in optimizing deep networks faster makes clear the importance of adaptive pre-conditioning. Though some [61, 62] have doubted their effectiveness due to potential issues that can harm generalization, others [31, 42, 67, 71] have shown advantages from optimizers with adaptive preconditioning; despite speeding up optimization, however, the effect of pre-conditioning on generalization has been less clear as some [2, 23, 65] have argued that the "sharpness" of minima achieved can vary depending on the choice of optimizer. More recent works have characterized pre-conditioning and mechanisms of implicit regularization around "edges of stability" for optimizers where the training regime occurs within a certain sharpness threshold, defined as the maximum eigenvalue of the loss Hessian [8, 21, 22].

Matrix completion and factorization have themselves long been an important focus for areas like signal recovery [17] and recommendation systems [14] from theoretical bounds for recovery and convergence [18, 19, 46, 54] to practical algorithms and implementations [34, 47]. We refer to [20] for a comprehensive survey. These and related have also served as test-beds for understanding implicit regularization; [7, 27] use matrix factorization and sensing to study gradient flow's implicit bias towards low-rank solutions, leading to conjectures that algorithmic regularization may not correspond to minimization of any norm-based penalty. [64] studies the implicit bias of mirror descent on matrix sensing, showing that the solution interpolates between the nuclear and Frobenius norms depending on the underlying matrix. In a related thread, [43] has shown that gradient flow with any commuting parameterization is equivalent to continuous mirror descent with a specific Legendre function, generalizing previous results [9, 27, 63] that have characterized the implicit bias of GD.

Interestingly, [11] illustrates how the discrete steps of GD can regularize optimization trajectories away from large gradients towards flatter minima. They develop an explicit regularizer to embody this implicit gradient regularization in attempts to control and reinforce this implicit effect more directly. To our knowledge, our paper is the first to propose the ratio of the nuclear 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 $\mathcal{P}_\Omega(W^\star)$ 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:

$$\min_W \mathcal{L}(W) \coloneqq \min_W \|\mathcal{P}_\Omega(W^\star) - \mathcal{P}_\Omega(W)\|_F^2 + \lambda R(W) \tag{1}$$

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 [4], 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 $W_i \in \mathbb{R}^{d_i \times d_{i-1}}, i \in \{1, \dots, N\}$ denotes the weight matrix corresponding to depth i or the i-th layer. Here, N 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/factorization). We refer to the matrix W, the product of the N weight matrices in Eq. (2), as the *end-product matrix* as per [7]. As such, the end-product matrix W is the solution produced in estimating $W^\star$ or, conveniently, the DLNN itself.

In our analyses, we focus on rank 5 matrices as the ground truth W^* and parameterize our DLNN W with $d_0 = \ldots = d_N = m = n = 100$ (i.e. 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 work [7] and employ the effective rank [56] of a matrix to track and quantify the rank of W in our experiments, defined as: $\text{e-rank}(W) = \exp\{H(p_1, \ldots, p_n)\}$ where H is the Shannon entropy, $p_i = \sigma_i / \|\sigma\|_1$, $\{\sigma_i\}$ 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 [56], resulting in unreliable and unstable 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 characterizing the inductive biases of un-regularized gradient descent and un-regularized Adam to better understand their dynamics in the presence of our penalty.

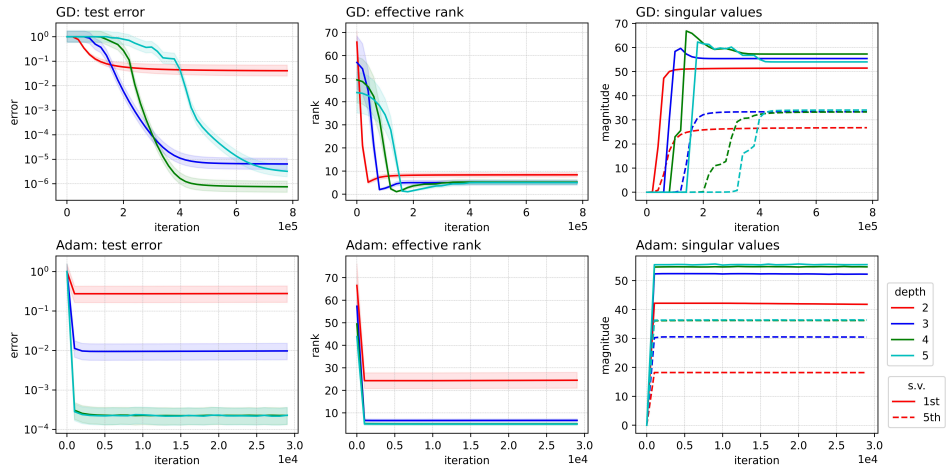

Figure 1: Dynamics of *un-regularized gradient descent (GD) and Adam*. Plots show the performance of GD over LNNs of depths 2 to 5 for rank-5 matrices of size 100×100. Colors correspond to different depth levels and shaded regions correspond to error bands. The left column depicts generalization error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

Gradient Descent Previous work [7] has shown that depth enhances gradient descent's 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 [6, 7]):

$$\dot{\sigma}_i = -N(\sigma_i(t)^2)^{\frac{N-1}{N}} \cdot \mathbf{u}_i^\top \nabla_W \mathcal{L}(W(t))\mathbf{v}_i \tag{3}$$

$$\text{vec}(\dot{W}) = -P_W \text{vec}\left(\nabla_W \mathcal{L}(W)\right) \tag{4}$$

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 $\sigma_i(t)$, 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, $\dot{W} = dW/dt$ is the time evolution of the end-product matrix or (equivalently) the DLNN itself, $P_W = \sum_{j=1}^{N} (W^\top W)^{\frac{N-j}{N}} \otimes (WW^\top)^{\frac{j-1}{N}}$, and $\otimes$ denotes the Kronecker product. Moving forward, we suppress the explicit dependency on t for simplicity. We note that full dynamics in Eq. (3) require non-degeneracy (i.e. depth $N > 1$); otherwise, they reduce to just the product of the loss gradient and singular vectors.

In Eq. (4), P_W can be seen as a pre-conditioning onto the gradient that, with sufficient depth ($N \geq 2$), accelerates movements already taken in the optimization [6]. As depth (i.e. over-parameterization)

increases, this acceleration intensifies while larger singular values and their movements become more pronounced than their smaller counterparts, driving singular value separation and a decrease in rank of the recovered matrix (Fig. 1 top row). The singular values also evolve at uneven paces depending on the depth; 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 and produce better solutions (i.e. low test error).

Adam Analyzing Adam can be difficult given its exponentially moving average of gradients; to simplify our analysis, we borrow some assumptions from [5] to approximately characterize Adam's dynamics via gradient flow 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 [7], the trajectory of the singular values σ_i of the end-product matrix W can be approximately characterized as:*

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

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 $P_{W,G} = \sum_{j=1}^{N} ((WW^\top)^{\frac{j-1}{N}} \otimes (W^\top W)^{\frac{N-j}{N}}) G_j$ is p.s.d. and G_j is a diagonal matrix for layers $j \in \{1, \ldots, N\}$. Specifically, $G_j = \text{diag}(\text{vec}(S_j))$, $[S_j]_{m,n} = [(\nabla_{W_j} \mathcal{L}(W)^2 + s_j^2)^{-1/2}]_{m,n}$, $\nabla_{W_j} \mathcal{L}(W) = \partial \mathcal{L}(W) / \partial W_j$ is layer j's loss gradient, 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, which now normalizes each layer by the square-root 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, its normalization by a function of the variance can now either dampen or further accelerate the trajectory.

Empirically, we see that depth enhances the implicit bias towards low-rank solutions for both Adam and gradient descent albeit differently (Fig. 1, middle column); in deeper regimes ($N > 2$), Adam minimizes rank to exact/near-exact rank recovery more smoothly than gradient descent via faster (10^4 vs. 10^5 iterations) and more uniform convergence (Fig. 1, bottom row). With Adam, singular value dynamics exhibit significantly more uniform evolution regardless of depth in contrast to gradient descent (Fig. 1, right), leading to different types of trajectories and solutions.

3.3 Depth, with penalty

Gradient Descent We now characterize the dynamics of gradient flow with our penalty.

Theorem 2. *Under the assumptions of [6], the evolution of the singular values of the end-product matrix, under gradient descent with the penalty, can be approximated in the following fashion:*

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

where $\lambda > 0$ is the regularization parameter and $\dot{\sigma}_r^{GF}$ denotes the un-regularized singular value trajectory under gradient flow in Eq. (3). Similarly, the evolution of W can be approximated via:

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

where U, V contain the left and right singular vectors of W, P_W is the pre-conditioning in Eq. (4), and $\tilde{\Sigma} = \frac{||W||_*}{||W||_F} \Sigma$ where Σ contains the singular values of W along its diagonal.

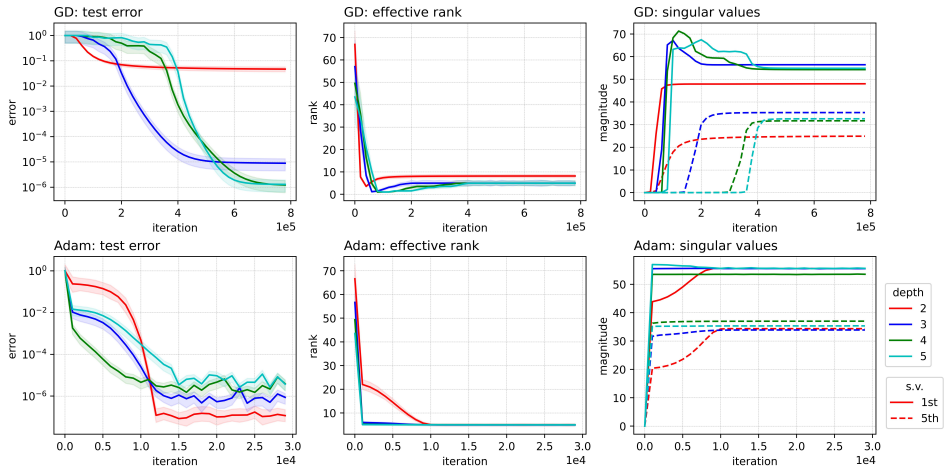

Figure 2: Dynamics of *regularized gradient descent (GD) and regularized Adam* with our penalty. Plots above show the performance of GD with our proposed penalty over DLNNs of depths 2 to 5 for rank-5 matrix completion. Setup is identical to Fig. 3. Here, $\lambda = 10^{-4}$ but results hold for a range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

Proof. See Appendix A.4 for details. □

The penalty can be seen as an additive component to the original trajectory in Eq. (3) that also intensifies with increased depth, making movements in larger singular values more pronounced than those of smaller ones. As a result, it can enable more pronounced singular value separation than before ($\frac{2N-1}{N}$ vs. $\frac{3N-2}{2}$), depending on λ. Increasing depth continues to push down rank but, unlike before (Eq. (3), Eq. (4)), the penalty now allows singular value trajectories to depend on their own magnitudes even without depth (Eq. (7) with $N = 1$), 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.

In Eq. (8), we also see that the depth-dependent accelerative pre-conditioning P_W now acts on a new term: while the first term can be interpreted as the typical influence of reducing the loss via training and gradient optimization on the network's trajectory, the new term can be interpreted as a spectral-based component that can be used by P_W to further enhance the spectral trajectory of W at higher depths, like in Eq. (7). 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: $\frac{\mathbf{u}_i^\top \mathbf{v}_i}{||W||_F^2}(1 - \frac{||W||_*}{||W||_F}\sigma_i)$.

Empirically, comparing the un-regularized case (Fig. 1 top row) to the regularized case (Fig. 2 top row), we see that the penalty helps increase the speed at which rank is reduced, inducing faster rates of rank reduction earlier in training. Unlike in un-regularized gradient descent where deeper networks take longer to exhibit rank reduction, the penalty enables near simultaneous reductions in rank across all depths (Fig. 2 top row, middle), making it less dependent on depth. However, the DLNN and end-product matrix W still achieves similar performance in terms of test error under the penalty as it does under un-regularized gradient descent.

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 evolution of the end-product matrix and its singular values can be approximated via the following:*

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

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 large degree of depth invariance as rank is pushed down almost independently of depth (Fig. 2, bottom row), even at depth 2. Exact rank recovery of the underlying matrix is now possible at all depths, unlike gradient descent. From a shallow network ($N = 2$), increasing the depth does not induce any material changes in the solutions produced as the penalized DLNN under Adam produces low-rank solutions that achieve exact rank recovery and low test 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 does not vary much with depth. As singular values evolve, we see that their paths are relatively robust to depth, producing the near-identical behavior with much less dependence on depth (Fig. 2, right).

3.4 No depth, with penalty

Given the non-trivial interaction effects and depth invariance seen in the previous section, we now consider a depth 1 network (i.e. no depth, $N = 1$). It is known [70] that gradient descent in this setting converges to the minimum Frobenius (i.e. ℓ_2) norm solution, which does not necessarily induce low ranked-ness. With a depth 1 network under gradient descent, the penalty does little to affect the solution (Fig. 3, top row) as learning is heavily constrained without depth.

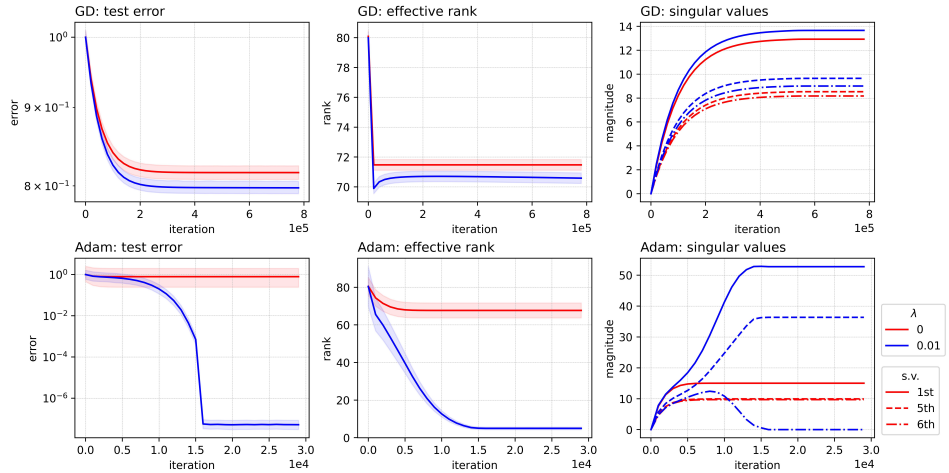

Figure 3: Performance comparison between choice of optimization and regularizer in a depth-1 linear network. Top row corresponds to gradient descent and bottom corresponds to Adam. Note that $\lambda = 0$ corresponds to the un-regularized setting. Here, $\lambda = 10^{-2}$ but results hold for values $\lambda \in [10^{-6}, 10^{-1}]$.

Yet, despite being ineffective under gradient descent, the penalty is effective under Adam (Fig. 3, bottom) even without depth or over-parameterization, generalizing as well as, if not better than, deep network ($N > 2$). We note that replacing our penalty with other proxies of rank or spectral sparsity, like the nuclear norm, does not work (Table 1). As mentioned earlier, under un-regularized gradient flow at depth 1, network dynamics collapse as singular value trajectories reduce to $\mathbf{u}_i^\top \nabla_W \mathcal{L}(W(t))\mathbf{v}_i$ 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 from un-regularized gradient descent at depth 1 generalize poorly and fail at rank recovery.

Without depth, we see that a key component in helping a depth 1 network learn under Adam and the penalty 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 $\nabla_W \mathcal{L}(W) \cdot G$ resembles an inverse signal-to-noise ratio that both have come up in other works as important quantities that are strongly predictive of generalization capabilities [35] or are essential to finding optimal variance adaption factors of stochastic gradients [5].

3.5 Comparison with other penalties and optimizers

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) of the estimated matrix. **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 the nuclear norm penalty, **None** is no penalty, and $a \cdot e\ b$ denotes $a \cdot 10^b$. Best results—in terms of both absolute test error (lower the better) and rank reduction (closer to 5 the better) as well as depth invariance in terms of error and rank—are in bold. For more results, 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 [39]	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 [25]	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 [39]	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

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 [39], 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. Though the nuclear norm shows some extent of depth invariance in rank reduction, it is unable to find well-generalizing solutions and 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.

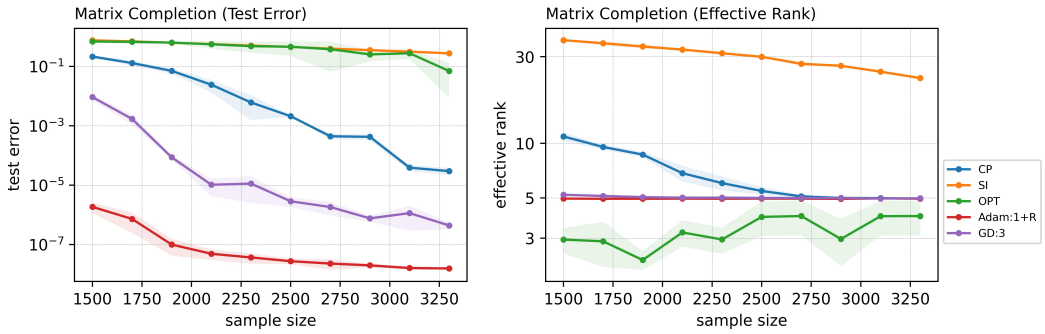

Figure 4: Comparative performance in generalization error and rank minimization for rank-5 matrix completion (100×100). x-axis stands for the number of observed entries (out of 10^5 entries) and shaded regions indicate error bands. **Adam:1+R** refers to a depth 1 network trained with Adam and our penalty,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

Comparative Performance We also note our penalty's comparative performance (Fig. 4) against other methodologies for matrix completion across various sample sizes (i.e. the number of entries out of 100,000 made available for training). A 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. Even at lower sample sizes, a penalized depth 1 network generalizes better than methods such as `SoftImpute` [47], `OptSpace` [38], the minimum nuclear norm solution [19], 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 small sample sizes, but also improves as the sample size grows.

4 Results on real data

Table 2: Performance evaluations on MovieLens100K. Results are categorized by model, whether additional data or features (e.g. external/side information, implicit feedback, graph features, etc.) beyond the explicit ratings in the interaction matrix are used, and test error as measured by root mean squared error (RMSE, lower is better) on different splits in **(a)** and **(b)**. Since various approaches use different train-test proportions, results [16, 53] on two common splits are included. Results from using Adam with the penalty are in bold.

Model	Uses side info, add. features, or other info, etc?	90% RMSE
Depth 1 LNN	No	
w. GD		2.814
w. GD+penalty		2.808
w. Adam		1.844
w. Adam+penalty		**0.915**
User-Item Embedding	No	
w. GD		2.453
w. GD+penalty		2.535
w. Adam		1.282
w. Adam+penalty		**0.906**
NMF [48]	No	0.958
PMF [48]	No	0.952
SVD++ [41]	Yes	0.913
NFM [30]	No	0.910
FM [55]	No	0.909
GraphRec [53]	No	0.898
AutoSVD++ [59]	Yes	0.904
GraphRec+sidefeat.[53]	Yes	0.899
GraphRec+graph/side feat.[53]	Yes	0.883

(a) Performance on 90:10 (90%) train-test split

Model	Uses side info, add. features, or other info, etc?	80% RMSE
Depth 1 LNN	No	
w. GD		2.797
w. GD+penalty		2.821
w. Adam		1.822
w. Adam+penalty		**0.921**
User-Item Embedding	No	
w. GD		2.532
w. GD+penalty		2.519
w. Adam		1.348
w. Adam+penalty		**0.919**
IMC [33, 66]	Yes	1.653
GMC [36]	Yes	0.996
MC [18]	Yes	0.973
GRALS [52]	Yes	0.945
sRGCNN (sRMGCNN) [49]	Yes	0.929
GC-MC [16]	Yes	0.910
GC-MC+side feat. [16]	Yes	0.905

(b) Performance on 80:20 (80%) train-test split

Lastly, a natural question might be: how well do our results extend to real-world data? To answer this, we consider MovieLens100K [15]—a popular benchmark for recommendation systems that has seen widespread usage for evaluating different approaches. It consists of ratings from 944 users on 1,683 movies, forming an interaction matrix $M \in \mathbb{R}^{944 \times 1683}$ where the goal is to predict the rest of M after having observed a subset.

Unlike our earlier experiments, the values here are discrete in the range $\{1, 2, 3, 4, 5\}$ and M is of high, or near full, rank. Given these differences and more conventional empirical approaches in 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 [40]) that combines user-specific and item-specific biases with a dot product between a latent user and latent item embedding; we apply our penalty separately on either the item or the user embedding layer. Though approaches solely utilizing explicit ratings have fallen out of favor in lieu of ones incorporating additional information and complex designs (e.g. graph-based, implicit feedback, deep networks), we nonetheless evaluate the effects of our penalty with this simple framework. We compare the results from these two approaches with a variety of approaches that use specialized architectures, deep non-linear networks, additional side information, etc. beyond M.

From Table 2, we see that Adam and the penalty (**w. Adam+penalty**) can improve performance over the baseline of gradient descent (GD) or Adam alone. Surprisingly, a depth 1 network with Adam and the penalty can outperform or come close to other more specialized approaches despite its simplicity; however, in contrast to the others, it does so without any specialized or additional architectures (e.g. helper models/networks), further/external information beyond M (e.g. implicit feedback, side info), creation of new graphs or features, non-linearites, higher-order interactions (e.g. factorization machines), or—for the depth 1 case—even any factorization at all. More precise

tuning (e.g. better initialization, learning schedules) or usage of other info/modules may yield further improvements on this task and others that involve matrix completion or factorization [1, 58, 68]. We leave these for fuller evaluation and further study in future work.

5 Discussion

The dynamics of optimization trajectories—induced together by the choice of optimizer, parameterization, loss function, and architecture—can play an important role in the solutions produced and their ability to generalize. Depth and gradient descent-like algorithms have been key ingredients to the successes of deep learning. On matrix completion/factorization, the proposed penalty helps produce well-generalizing solutions and perfect rank recovery even with a degenerate depth 1 (i.e. depth-less) 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?

We suspect not. While we concur with the conjecture in [7]—namely, a reductionist view which suggests that implicit regularization can be entirely encapsulated by an explicit norm-based penalty is likely an incorrect over-simplification—we believe that there is merit in studying both implicit and explicit forms of regularization to examine their interplay. Our work suggests that we may be able to partially replicate the successes of *deep* learning by selectively combining optimization methods with explicit penalties via better model design or encoding of inductive biases, but this remains unclear.

Many questions remain open from our limited analyses which, due to space considerations, we leave for future work. For instance, how well do lessons from DLNNs translate to their non-linear counterparts and on other tasks (e.g. classification)? How does this relate to learning regimes with larger learning rates or discrete trajectories (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 networks. Nonetheless, our findings suggest the possibility that it can be harnessed and transformed to desirable effect.

Acknowledgments and Disclosure of Funding

We thank Kellin Pelrine for his feedback and Derek Feng for his help with a previous collaboration on which this work builds. We also thank the anonymous reviewers for their valuable comments and feedback throughout the process.

References

[1] Anish Acharya, Rahul Goel, Angeliki Metallinou, and Inderjit Dhillon. Online embedding compression for text classification using low rank matrix factorization. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

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

[8] Sanjeev Arora, Zhiyuan Li, and Panigrah Abhishek. Understanding gradient descent on edge of stability in deep learning. *arXiv preprint arXiv:2205.09745*, 2022.

[9] Moroshko E. Nacson M. S. Woodworth B. Srebro N. Globerson A. Azulay, S. and D. Soudry. On the implicit bias of initialization shape: Beyond infinitesimal mirror descent. In *Proceedings of the 38th International Conference on Machine Learning (PMLR)*, 2021.

[10] 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.

[11] David G.T. Barrett and Benoit Dherin. Implicit gradient regularization. *International Conference on Learning Representations (ICLR)*, 2021.

[12] 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.

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

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

[15] Rianne Van Den Berg, Thomas Kipf, and Max Welling. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19*, 2015.

[16] Rianne Van Den Berg, Thomas Kipf, and Max Welling. Graph convolutional matrix completion. *KDD'18 Deep Learning Day)*, 2018.

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

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

[19] 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.

[20] 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.

[21] Jeremy Cohen, Simran Kaur, Li Yuanzhi, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on networks typically occurs at the edge of stability. *International Conference on Learning Representations (ICLR)*, 2021.

[22] Jeremy Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapti, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E. Dahl, and Justin Gilmer. Adaptive gradient methods at the edge of stability. *arXiv preprint arXiv:2207.14484*, 2022.

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

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

[25] 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.

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

[27] 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.

[28] 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.

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

[30] 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.

[31] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, and Bernhard Nessler. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

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

[33] Prateek Jain and Inderjit S. Dhillon. Provable inductive matrix completion. *arXiv:1306.0626*, 2013.

[34] 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.

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

[36] Vassilis Kalofolias, Xavier Bresson, Michael Bronstein, and Pierre Vandergheynst. Matrix completion on graphs. *arXiv preprint arXiv:1408.1717*, 2014.

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

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

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

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

[41] 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.

[42] 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.

[43] Zhiyuan Li, Tianhao Wang, Jason D. Lee, and Sanjeev Arora. Implicit bias of gradient descent on reparametrized models: On equivalence to mirror descent. *arXiv preprint arXiv:2207.04036v1*, 2022.

[44] 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.

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

[46] 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.

[47] 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.

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

[49] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, 2017.

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

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

[52] Nikhil Rao, Hsiang-Fu Yu, Pradeep K. Ravikumar, and Inderjit S. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in Neural Information Processing Systems*, 2015.

[53] 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.

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

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

[56] 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.

[57] 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.

[58] Devavrat Shah, Dogyoon Song, Zhi Xu, and Yuzhe Yang. Sample efficient reinforcement learning via low-rank matrix estimation. In *Advances in Neural Information Processing Systems*, 2019.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] Gunasekar S. Lee J. D. Moroshko E. Savarese P. Golan I. Soudry D. Woodworth, B. and N. Srebro. Kernel and rich regimes in overparametrized models. In *Proceedings of the 33rd Conference on Learning Theory (PMLR)*, 2020.

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

[65] 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.

[66] Miao Xu, Rong Jin, and Zhi-Hua Zhou. Speedup matrix completion with side information: Application to multi-label learning. In *Advances in Neural Information Processing Systems*, 2013.

[67] 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.

[68] Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. ME-Net: Towards effective adversarial robustness with matrix estimation. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

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

[70] 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.

[71] 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.

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]