# 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 some of 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 $\ell_p$ norm weight constraints or dropout [43], its effectiveness and role in generalization in deep networks is not certain given its inability to prevent over-fitting to random labels [50] 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.

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

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 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? Similarly, 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.
2. The above result only occurs under Adam and 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 in related work [5, 6, 50] 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 [44, 45] 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, 41] 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 [36] 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, 31, 32]—in optimizing deep networks faster makes the importance of adaptive pre-conditioning more explicit. Though some [44, 45] have doubted the effectiveness of adaptive gradient methods and illustrated potential issues in certain cases, such as harming generalization, others [30, 48, 51] 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, 47] 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, 33, 38] have led to practical algorithms [24, 34]. 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, [46] 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, [52] showed how SGD's 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 $\Omega$. We define the projection $P_\Omega(W^\star)$ to be a $m \times n$ matrix such that the entries with indices in $\Omega$ remain while the rest are masked with zeros. We are interested in the following optimization:

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

where $\|\cdot\|_F$ is the Frobenius norm, $R(\cdot)$ is an explicit penalty, and $\lambda \geq 0$ is the tuning parameter. Following [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 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 similarly to other ranks (e.g. see Appendix A.2). 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.

**Gradient Descent** Previous work [6] has shown that, via gradient flow, depth enhances the implicit regularization towards low rank, characterized by the following singular value trajectory:

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

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, and $\nabla_W \mathcal{L}(W(t))$ is the

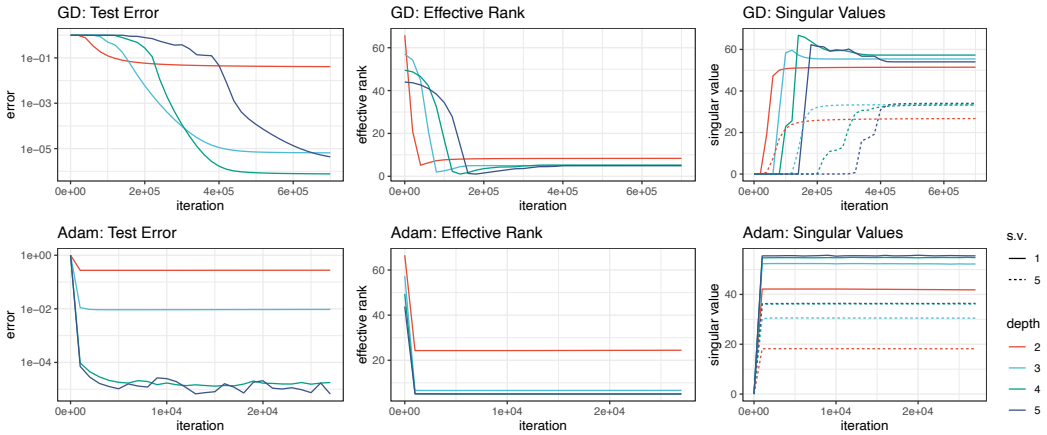


**Figure 1:** Dynamics of *un-regularized gradient descent (GD) and Adam*. Plots show the performance of GD over LNNs of depths 2 to 5 for rank-5 matrices of size $100 \times 100$. Colors correspond to different depth levels. The left column depicts generalization error as a function of depth and training 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 $1^{\text{st}}$ largest singular value while a dotted line indicates the $5^{\text{th}}$ largest. We omit the other singular values for clarity due to their small magnitude.

loss gradient with respect to the end-product matrix $W$ at time $t$. Moving forward, we suppress the dependency on $t$ for simplicity. Note that full dynamics 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 $1^{\text{st}}$ and $5^{\text{th}}$ 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. Additionally, [5, 6] have also described the trajectory dynamics of the end-product matrix $W$ itself (i.e. the produced solution to Eq. (1)) under the same assumptions:

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

where $\text{vec}(\cdot)$ 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. $P_W$ 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 the assumptions from [4] to, though imperfect, characterize Adam's dynamics via gradient flow (see Appendix A.4 for more details). We set the discount factors equal $\beta_1 = \beta_2 = \beta$ and assume that the estimate of discounted squared gradients $\hat{v}$ can be well-approximated by its expectation.

**Lemma 1.** *Under the assumptions above and of [6], the trajectory of the end-product matrix $W$ and its singular values $\sigma_i$, respectively, can be approximately characterized as:*

$$\text{vec}(\dot{W}) = -P_{W,G} \text{vec}(\nabla_W \mathcal{L}(W)) \tag{5}$$

*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 our 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 in the regularized case, 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). The trajectory and convergence of singular values, the effective rank of $W$, and test error appear *accelerated*. Given these differences, Adam appears to arrive at different kinds of solutions than gradient descent.

### 3.3 Depth, with penalty

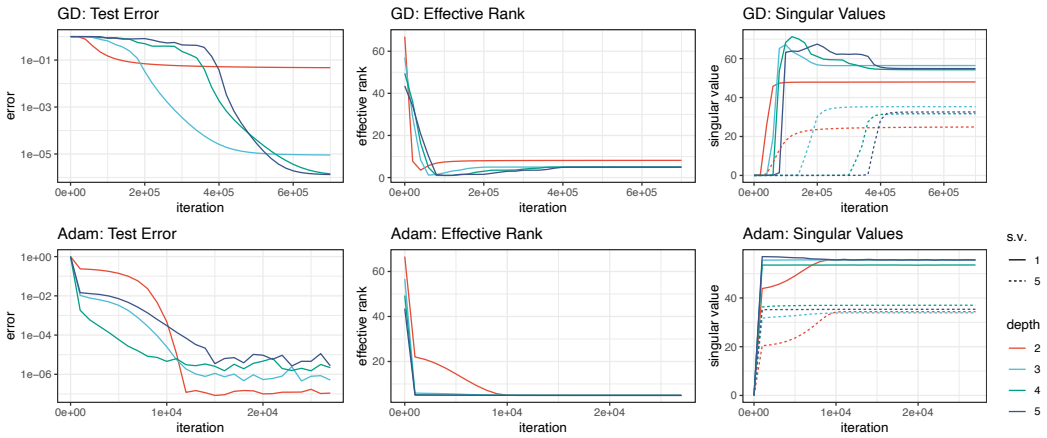


**Figure 2:** Dynamics of *regularized gradient descent (GD) and regularized Adam* with our penalty. Plots above show the performance of GD with our proposed penalty over DLNNs of depths 2 to 5 for rank-5 matrix completion. Setup is identical to Fig. 3. here, $\lambda = 10^{-4}$ but results hold for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

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

**Lemma 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:*

$$\dot{\sigma}_r^{reg} = \dot{\sigma}_r^{GD} - \frac{\lambda N}{||W||_F^2}\left(1 - \frac{||W||_*}{||W||_F}\right)\sigma_r^{\frac{3N-2}{2}} \tag{6}$$

*where $\lambda > 0$ is the regularization parameter and $\dot{\sigma}_r^{GF}$ denotes the singular value trajectory under un-regularized gradient flow (Eq. (3)).*

*Proof.* Follows from the gradient of our penalty, applied to Eq. (3). □

The penalty can be seen as an additive component to the original trajectories in Eq. (3). 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 $\lambda$.

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. (3), Eq. (4)), 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 can similarly characterize the trajectory of the end-product matrix $W$.

**Lemma 3.** *Under the same assumptions, with the proposed penalty, the end-product matrix $W$ evolves in the following fashion:*

$$\text{vec}(\dot{W}) = -P_W \left( \text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top + U\tilde{\Sigma}V^\top)}{||W||_F^2} \right) \tag{7}$$

*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 $\Sigma$ contains the singular values of $W$ along its diagonal.*

*Proof.* Follows from the gradient of our penalty, applied to Eq. (4). □

We see that the depth-dependent accelerative pre-conditioning $P_W$ now acts on a new term, similar to Eq. (6), 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: $\frac{\mathbf{u}_i^\top \mathbf{v}_i}{||W||_F^2}\left(1 - \frac{||W||_*}{||W||_F}\sigma_i\right)$. Empirically, we see (Fig. 2 top row, left plot) that the end-product matrix $W$ produced 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 found.

**Lemma 4.** *Under the same assumptions, with the proposed penalty, the end-product matrix $W$ and its singular values evolves in the following fashion:*

$$\begin{aligned} \text{vec}(\dot{W}) &= -P_{W,G} \left( \text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top + U\tilde{\Sigma}V^\top)}{||W||_F^2} \right) \\ \dot{\sigma} &= -(\mathbf{u}_r^\top \mathbf{v}_r) P_{W,G} \left( \text{vec}\left(\nabla_W \mathcal{L}(W)\right) + \lambda \frac{\text{vec}(UV^\top + U\tilde{\Sigma}V^\top)}{||W||_F^2} \right) \end{aligned} \tag{8}$$

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

Empirically, we note a new development: namely, the network displays a large degree of depth invariance in that it can now push rank 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 also 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, if at all. As singular values evolve near identically, we see that their paths are relatively robust to depth—all producing the near-identical behavior without much 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 [50] that gradient descent in this setting converges to the minimum Frobenius (i.e. $\ell_2$) 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). This similarly holds for un-regularized Adam—without depth, learning is heavily constrained.

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

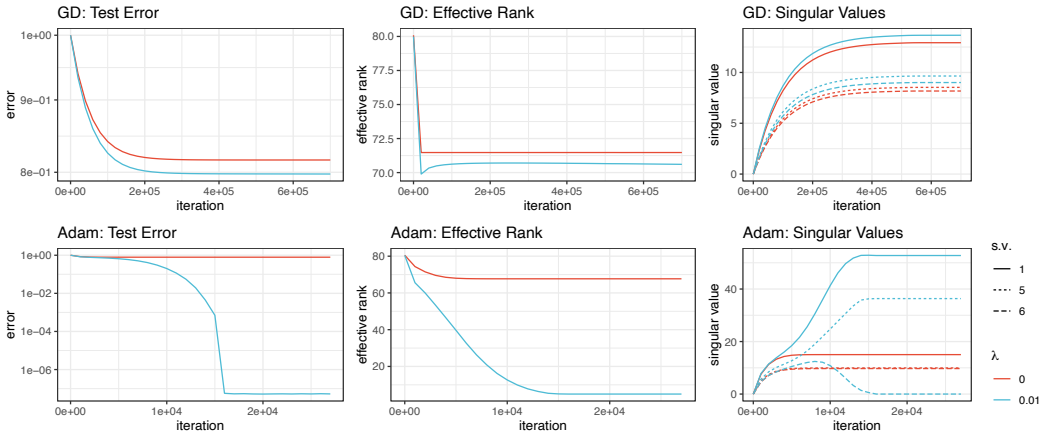


**Figure 3:** Performance comparison between choice of optimization and regularizer in a depth-1 linear network. Top row corresponds to gradient descent and bottom corresponds to Adam. Note that $\lambda = 0$ corresponds to the un-regularized setting. Here, $\lambda = 10^{-4}$ but results hold for values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

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. (5), along with the term $P_{W,G}\left(\frac{\lambda \text{vec}(UV^\top + U\tilde{\Sigma}V^\top)}{||W||_F^2}\right)$, which reduces to $G\left(\frac{\lambda \text{vec}(UV^\top + U\tilde{\Sigma}V^\top)}{||W||_F^2}\right)$ without depth. Interestingly, the variance of the loss gradient and the ratio $\eta^2 = \text{var}(\nabla_W \mathcal{L}(W))/\nabla_W \mathcal{L}(W)^2$ 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 Comparative Performance

**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** is nuclear norm penalty, **None** is no penalty, and $a\cdot$ e $b$ denotes $a \cdot 10^b$. 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 |

**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

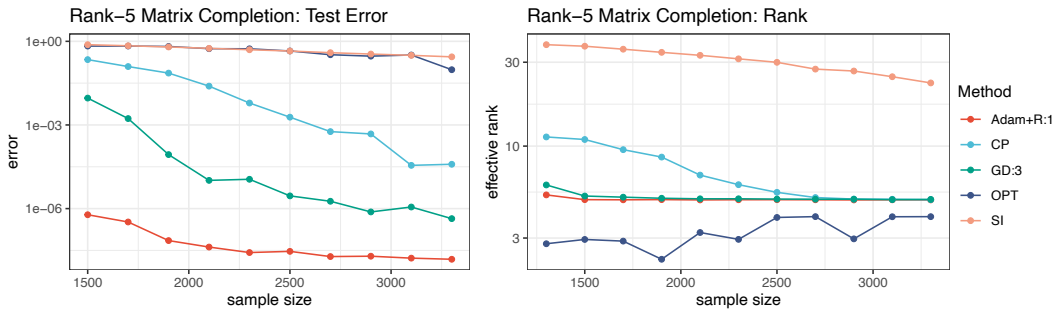


**Figure 4:** Comparative performance in generalization error and rank minimization for rank-5 matrix completion ($100 \times 100$). $x$-axis stands for the number of observed entries (uniformly sampeled). **Adam+R:1** refers to Adam with our penalty at depth 1, **CP** is the minimum nuclear norm solution, **GD:3** is a depth 3 network trained with gradient descent, **OPT** is `OptSpace` [27], and **SI** is `SoftImpute` [34]. To reduce clutter, we omit results with similar performance (e.g. GD:4, GD:5 etc.).

[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 depth 1, the penalized network generalizes better than methods such as `SoftImpute` [34], `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 but it continues to improve with larger sample sizes in addition to scarcer data regimes.

## 4 Empirical Results

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 $M \in \mathbb{R}^{944 \times 1683}$ with the goal of predicting the rest of $M$ after having observed some subset of $M$.

Unlike our earlier experiments, however, the values are discrete in the range $\{1, 2, 3, 4, 5\}$ and $M$ is full-rank. Given these differences and conventional approaches in tackling more empirical recommendation system problems, we apply our penalty in two ways. The first way is as before, training a simple 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) 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. Although this relatively simple approach utilizing explicit ratings has fallen out of favor relative to more recent ones (e.g. graph-based approaches, implicit feedback, deep networks, etc.) utilizing additional information and more 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 for ratings prediction, reflecting a wide variety of approaches, architectures, and methods that use additional side information/features 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

**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** | **0.955** |
| User-Item Embedding | No | | |
| w. Adam | | 0.956 | 0.978 |
| **w. Adam+penalty** | | **0.906** | **0.950** |
| PMF [35] | No | 0.952 | 0.977 |
| SVD++ [29] | No | 0.913 | 0.938 |
| NFM [22] | No | 0.910 | 0.935 |
| FM [39] | No | 0.909 | 0.935 |
| GraphRec [37] | No | 0.898 | 0.932 |
| AutoSVD++ [42] | Yes | 0.904 | 0.926 |
| GraphRec+graph feat. [37] | Yes | 0.887 | 0.918 |
| GraphRec+graph feat.+side info. [37] | Yes | 0.883 | 0.911 |

to the other methodologies, 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 (AutoSVD++) that leverages additional information/features, despite not needing to do so itself. We leave details of our experiment 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 the key ingredients to some of its successes. In our work focusing on matrix completion, our penalty is able to achieve well-generalizing solutions and achieve 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, that 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, and examining 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.

Naturally, many questions remain open from our limited analyses which, due to space considerations, we leave as possible 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 and under different settings 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] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

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

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

[37] 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.

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

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

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

[47] 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.

[48] 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.

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

[50] 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.

[51] 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.

[52] 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 ($\lambda$) and several initial learning rates $\alpha$ 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]