

# MASK IN THE MIRROR: IMPLICIT SPARSIFICATION

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

Continuous sparsification strategies are among the most effective methods for reducing the inference costs and memory demands of large-scale neural networks. A key factor in their success is the implicit  $L_1$  regularization induced by jointly learning both mask and weight variables, which has been shown experimentally to outperform explicit  $L_1$  regularization. We provide a theoretical explanation for this observation by analyzing the learning dynamics, revealing that early continuous sparsification is governed by an implicit  $L_2$  regularization that gradually transitions to an  $L_1$  penalty over time. Leveraging this insight, we propose a method to dynamically control the strength of this implicit bias. Through an extension of the mirror flow framework, we establish convergence and optimality guarantees in the context of underdetermined linear regression. Our theoretical findings may be of independent interest, as we demonstrate how to enter the rich regime and show that the implicit bias can be controlled via a time-dependent Bregman potential. To validate these insights, we introduce PILoT, a continuous sparsification approach with novel initialization and dynamic regularization, which consistently outperforms baselines in standard experiments.

## 1 INTRODUCTION

Deep learning continues to impress across disciplines ranging from language and vision (Ramesh et al., 2022) to drug design (Stephenson et al., 2019; Jumper et al., 2021) and even fast matrix multiplication (Fawzi et al., 2022). These accomplishments come at immense costs, as they rely on increasingly large neural network models. Moreover, training such massive models with first-order methods like variants of Stochastic Gradient Descent (SGD) is a considerable challenge and often requires large-scale compute infrastructure (Kaack et al., 2022). Even higher costs are incurred at inference time, if the trained models are frequently evaluated (Wu et al., 2022; Luccioni et al., 2023).

Sparsifying such neural network models is thus a pressing objective. It not only holds the promise to save computational resources, it can also improve generalization (Frankle & Carbin, 2019; Paul et al., 2023), interpretability (Chen et al., 2022; Hossain et al., 2024), denoising (Jin et al., 2022; Wang et al., 2023), and verifiability (Narodytska et al., 2020; Albarghouthi, 2021). However, at its core is a hard large-scale nested optimization problem combining multiple objectives. In addition to minimizing a typical neural network loss  $\min_{w \in \mathbb{R}^n} f(w)$  (and its generalization **performanceerror**), we wish to rely on the smallest possible number of weights, effectively minimizing the  $L_0$  norm  $\min_{w \in \mathbb{R}^n} \|w\|_{L_0}$ . This is an NP-hard problem that is also practically hard to solve due to its mixed discrete and continuous nature. This becomes more apparent when we reformulate it **in-a-way-as like** the best performing sparsification methods **approach it**.

Among such approaches that achieve high sparsity while maintaining high generalization performance **, albeit being computationally expensive, are are continuous sparsification methods and iterative pruning strategies and continuous sparsification methods, which. These methods** explicitly identify for each weight parameter  $w$  of a neural network a binary mask  $m \in \{0, 1\}$  that signifies whether a parameter is pruned **and thus. Thus, a parameter is** set to zero ( $m = 0$ ) or not ( $m = 1$ ), effectively parameterizing the network with parameters  $x = m \odot w$ , **where  $\odot$  is the pointwise multiplication (Hadamard product)**. The introduction of the additional mask parameters  $m$  turns the sparsity objective into a discrete  $L_1$  penalty of  $m$  as  $\|w \odot m\|_{L_0} = \sum_i m_i$  subject to  $m_i \in \{0, 1\}$ , where  $\odot$  denotes elementwise multiplication and we assume that  $w \neq 0$  if  $m = 1$ . The  $L_1$  objective is already more amenable to continuous optimization than the original  $L_0$  objective (Louizos et al., 2018). Nevertheless, the big challenge arises from the fact that  $m$  is binary.

Continuous sparsification addresses this issue by relaxing the optimization problem to continuous or even differentiable variables  $m$ , often with  $m \in [0, 1]$  by learning a parameterization  $m = g(s)$  with  $g: \mathbb{R} \rightarrow [0, 1]$ , e.g., like a sigmoid. This way, the problem becomes solvable with standard first-order optimization methods. Yet, moving from the continuous space back to the discrete space is error-prone. Regularizing and projecting  $m$  towards binary values  $\{0, 1\}$  is generally problematic, requires careful tuning, and often entails robustness issues.

However, this projection step is not necessarily required in a parameterization  $m \odot w$ , where  $m$  can freely attain values in  $\mathbb{R}$ , as it implicitly optimizes for an L1-penalty (Ziyin & Wang, 2023; Ziyin, 2023), which appears to be equivalent to LASSO. To utilize this insight for continuous sparsification, Ziyin & Wang (2023) have proposed a method named *spred*, which combines  $\alpha(\|m\|_{L_2}^2 + \|w\|_{L_2}^2)$ . Interestingly, this approach combined with weight decay  $\alpha(\|m\|_{L_2}^2 + \|w\|_{L_2}^2)$  is equivalent to LASSO and thus an explicit  $L_1$ -regularization (Ziyin, 2023). Yet, their proposed method *spred* significantly outperforms LASSO, which suggests that the posed equivalence of optimized objectives does not suffice to explain its success. cannot explain this success.

The answer to this open question lies in the analysis of the learning dynamics, which fundamentally differs from LASSO, as redundant features are sparsified exponentially fast. We show this by extending the mirror flow framework, based on which we gain additional insights and derive tools to improve over *spred*. By the unification of the explicit regularization in the implicit bias framework, the problem gets indirectly reformulated as a As we show, an important, but overlooked, distinguishing factor is the fact that continuous sparsification with  $m \odot w$  parameterization is driven by an implicit rather than an explicit regularization. This implies that in a sufficiently overparameterized setting, the following hierarchical optimization problem, instead of solving LASSO is solved:

$$\min_{x \in \mathbb{R}^n: f(x)=0} \|x\|_{L_1}. \quad (1)$$

The main idea follows the same philosophy as the lottery ticket hypothesis (Frankle & Carbin, 2019). From a range of models which all attain optimal training loss, we choose it prefers the sparsest model. In other words, we aim to find a subnetwork with a similar accuracy as a dense network. Instead of having two competing objectives, we enforce cooperation between the two objectives by subjugating the sparsification. While this formulation can have advantages, if we can attain zero training loss ( $f(x) = 0$ ), it might still be equivalent to an explicit regularization.

The optimization problem in Eq. (1) is solvable using the implicit bias framework. It is well known that standard gradient flow has an implicit bias which acts as an  $L_2$  regularization (Nemirovski & Yudin, 1983; Beck & Teboulle, 2003). Different parameterizations though induce a different implicit bias (Pesme et al., 2021; Gunasekar et al., 2020; Woodworth et al., 2020; Li et al., 2022) in the The real potential of continuous sparsification and its induced implicit bias (in particular in non-convex settings) becomes apparent when we study the corresponding learning dynamics. Our theoretical analysis of (stochastic) gradient flow framework. We utilize this framework to understand the implicit bias induced by the parameterization applied to  $m \odot w$  with weight decay. The key insight is that the reveals that the dynamics differ fundamentally from the ones of LASSO, where redundant features are sparsified exponentially fast. Instead, we show by extending the mirror flow framework that a dynamic explicit weight decay regularization moves can move the implicit bias from  $L_2$  to  $L_1$  during training. This way, the dynamics resemble In consequence, the sparsification becomes effective only relatively late during training, allowing the overparameterized model to first attain high generalization performance. The dynamics resemble thus a successful strategy that is applied across continuous and iterative pruning methods, as they all acknowledge which all acknowledge and realize the premise that training overparameterized models before they are sparsified usually leads to significant performance benefits (Frankle & Carbin, 2018; Gadhikar & Burkholz, 2024; Paul et al., 2023). Remarkably, we also learn that the strength of the weight decay controls the amount of implicit L1 regularization. (Frankle & Carbin, 2018; Paul et al., 2023; Gadhikar & Burkholz, 2024).

We theoretically extend the framework by proposing a dynamic regularization-

Our analysis extends the implicit bias framework, which covers different parameterizations (Pesme et al., 2021; Gunasekar et al., 2020; Woodworth et al., 2020; Li et al., 2022) and the well-known fact that standard gradient flow has an implicit  $L_2$  bias (Nemirovski & Yudin, 1983; Beck & Teboulle, 2003). Another common use of the framework has been to study when training dynamics enter the so-called rich regime, which is responsible for improved feature learning. In this context, we study two main innovations. a) As we show, the explicit regularization (i.e. weight decay on  $m$  and  $w$ . ~~This regularization leads to even sparser lottery tickets, while still prioritizing the main optimization goal: accuracy. In this sense, it can be interpreted as tuneable implicit regularization.~~) guides the strength of the implicit bias. The fact that this makes the implicit bias tuneable makes it practically relevant for sparsification, as we have to be able to reach a target sparsity. b) By proposing a dynamic regularization (rather than a common static one), we obtain control over the transition speed from  $L_2$  to  $L_1$  regularization, which is crucial for performance gains in the high sparsity regime and enables us to enter the rich regime. ~~We thus show that~~ From a conceptual point of view, we unite explicit and implicit bias ~~can be united with~~ within a time-dependent Bregman potential, which is potentially of independent theoretical interest.

~~Utilizing these insights, we~~ While our general derivations provide insights into various continuous sparsification approaches, including STR (Kusupati et al., 2020), spread (Ziyin & Wang (2023), or (Savarese et al., 2021), we also utilize them to propose a new improved algorithm, PILoT (Parametric Implicit Lottery Ticket), ~~which~~. PILoT combines the  $m \odot w$  parameterization with a dynamic regularization and ~~a better initialization~~. ~~This initialization~~ an initialization that enables sign flips, ~~which are key for~~. Such sign flips are key to effective sparse training (Gadhikar & Burkholz, 2024), but are not feasible with the spread initialization. ~~Furthermore, the dynamic regularization~~ The dynamic regularization and thus implicit bias leads us to ~~improve over spread and other competing methods in particular on~~ outperform state-of-the-art baselines in particular in the high-sparsity regime, as we demonstrate in extensive experiments.

In summary, we make the following **contributions**:

- We gain novel insights into continuous sparsification by highlighting its implicit bias towards sparsity induced by doubling the number of trainable parameters. In particular, we explain the effectiveness of spread (Ziyin & Wang, 2023), which is based on  $m \odot w$ .
- To the best of our knowledge, we are the first to introduce the implicit bias with an explicit regularization resulting in a mirror flow with a time-dependent Bregman potential.
- We provide convergence results for (quasi)-convex loss functions (Theorem 2.2) and optimality for underdetermined linear regression (Theorem 2.3) with time-dependent Bregman potential.
- Improving results by (Alvarez et al., 2004; Li et al., 2022), we replace convexity with the Polyak-Łojasiewicz inequality, quasi-convexity and a growth condition on the Bregman potential (see Theorem A.3).
- Using our extensions of the mirror flow framework, we propose a new continuous sparsification method, PILoT, which controls the implicit regularization dynamically moving from  $L_2$  to  $L_1$ . Its initialization enables sign flips in contrast to spread.
- In experiments for diagonal linear networks and vision benchmarks (including ImageNet), PILoT consistently outperforms baseline sparsification methods such as STR and spread, which demonstrates the utility of our theoretical insights.

## 1.1 RELATED WORK

**Neural network sparsification.** A multitude of neural network sparsification methods have been proposed with different objectives (Liu & Wang, 2023). A popular objective is, for instance, to save computational and memory costs primarily at inference, or also during training, which is linked to the time of pruning, i.e., initially (Frankle et al., 2021; Lee et al., 2019; Tanaka et al., 2020; Wang et al., 2020; Pham et al., 2023; Patil & Dovrolis, 2021; Tanaka et al., 2020; Liu et al., 2021; Gadhikar et al., 2023; Fischer & Burkholz, 2021), early during training (Evcı et al., 2020; Dettmers & Zettlemoyer, 2019), during training like continuous sparsification (Sreenivasan et al., 2022; Kusupati et al., 2020; Savarese et al., 2021; Peste et al., 2021) or within multiple pruning-training iterations (Han

et al., 2015; Frankle & Carbin, 2018; You et al., 2020; Renda et al., 2020; Gadhikar & Burkholz, 2024). Other distinguishing factors are which type of sparsity the methods seek, if they focus on saving computational resources and memory in specific resource-constrained environments or, which methodological approach they follow.

**Unstructured sparsity.** In this work, we focus on unstructured sparsity, i.e., the fraction of pruned weights, and thus seek to remove as many weight entries as possible, which can achieve generally the highest sparsity ratios while maintaining high generalization performance. Structured sparsity, which usually obtains higher computational gains on modern GPUs (Kuzmin et al., 2019; Wen et al., 2016; Lasby et al., 2023), could also be realized in the continuous sparsification setting, for instance, by learning neuron-, group, or even layer-wise masks. Yet, this would not enjoy the same theoretical benefits as we derive here by showing that the unstructured continuous  $m \odot w$  parameterization induces a mirror flow, whereas for example the neuron-wise mask does not (see Section D).

**Iterative pruning.** Iterative pruning is often motivated by ~~to~~ the Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2018), which conjectures the existence of sparse subnetworks of larger dense source networks that can achieve the same accuracy as the dense network when ~~both are~~ trained (Frankle et al., 2021; Liu et al., 2024; Malach et al., 2020; Orseau et al., 2020; Pensia et al., 2020; Burkholz et al., 2022; Fischer & Burkholz, 2021; Burkholz, 2022a;b; da Cunha et al., 2022; Ferbach et al., 2022). In addition to the sparse structure, iterative pruning ~~often~~ tries to identify a trainable parameter initialization, indirectly also implementing an approximate  $L_0$ -regularization. In repeated prune-train iterations, trained weights are thresholded according to an importance score like magnitude. Afterward, the remaining parameters are free to adapt to data in a new training run and not ~~additionally be~~ regularized by a sparsity penalty (like  $L_1$ ). Our proposal PILoT can be combined with such iterative schemes. ~~Our~~ ~~The~~ experiments show that ~~this can boost the it boosts~~ performance of state-of-the-art schemes like ~~Iterative Magnitude Pruning (IMP) (Frankle & Carbin, 2018)~~, Weight Rewinding (WR) (Frankle et al., 2019), and Learning Rate Rewinding (LRR) (~~Maene et al., 2021; Gadhikar & Burkholz, 2024~~) (~~Maene et al., 2021~~).

**Continuous sparsification.** Continuous sparsification characterizes a collection of methods that can compete with iterative pruning techniques, while often requiring fewer training epochs (Sreenivasan et al., 2022; Kusupati et al., 2020). ~~The method by (Savarese et al., 2021) lends its name to the general approach, in which~~ ~~In one of the first proposals by (Savarese et al., 2021)~~, the mask is relaxed to a continuous variable. In general, continuous sparsification can be combined with a probabilistic approach where  $m$  is interpreted as a probability (Louizos et al., 2018; Zhou et al., 2021a;b). Other parameterizations of  $m$  that are not restricted to ~~the range~~  $[0, 1]$  (e.g. Powerpropagation) can also be utilized to regularize towards higher sparsity (Schwarz et al., 2021). Yet, they ~~usually need to be~~ ~~are usually~~ combined with projection ~~approaches like in iterative pruning~~. ~~Furthermore, the spread algorithm proposed by (Ziyin & Wang, 2023) to map  $m$  to a binary mask. The spread algorithm (Ziyin & Wang, 2023) removes any projections and shows that  $m \odot w$  with weight decay induces an implicit  $L_1$  regularization. Yet, this does not solve a LASSO objective. To explain its performance gain over LASSO. ~~By extending, we extend~~ the mirror flow framework, ~~we and~~ find an explanation in the training dynamics ~~that implies a dynamic~~. ~~Our extension, PILoT, dynamically adjusts the weight decay and induces a transition from an implicit  $L_2$  to  $L_1$  regularization. Our extension, PILoT controls this transition dynamically, which leads it to even  $L_2$  to  $L_1$  regularization. This enables it to~~ outperform the state-of-the-art method STR (Kusupati et al., 2020) in the high-sparsity regime. For a survey of other methods see (Kuznedelev et al., 2023).~~

**Implicit bias.** The implicit bias of (S)GD is a well-studied phenomenon (~~Chizat & Bach, 2020; Li et al., 2022; Woodworth et al., 2020; Gunasekar et al., 2020; 2017; Chou et al., 2024~~) (~~Chizat & Bach, 2020; Li et al., 2022; Woodworth et al., 2020; Gunasekar et al., 2020; 2017; Chou et al., 2024; Vaškevičius et al.~~) and can in certain cases be described by a mirror flow or mirror descent (in the discrete case with finite learning rate) (Li et al., 2022). Originally, mirror descent was proposed to generalize gradient descent and other first-order methods in convex optimization (Alvarez et al., 2004; Rockafellar & Fenchel, 1970; Boyd & Vandenberghe, 2009; Nemirovski & Yudin, 1983; Beck & Teboulle, 2003). Moreover, it has been used to study the implicit regularization of SGD in diagonal linear networks (Pesme et al., 2021; Even et al., 2023). More recently, it also has been applied to analyze the implicit bias of attention (Sheen et al., 2024). While (Li et al., 2022) has shown that different parameterizations have a corresponding mirror flow, we find that  $m \odot w$  with our proposed explicit regularization, PILoT, gives rise to a corresponding time-dependent mirror flow. Its time dependence gives us means to control the implicit bias, while still achieving

convergence. Time-dependent mirror descent has so far only been studied in the discrete case as a general possibility (Radhakrishnan et al., 2021). The time dependence also naturally arises in SDE modelling, yet, without control of the implicit bias (Pesme et al., 2021; Even et al., 2023). Here, we not only highlight a practical use case for time-dependent Bregman potentials, we also derive a way to control and exploit it.

**Optimization and convergence proofs.** Loss landscapes and the convergence of first-order methods is a large field of study (Karimi et al., 2016; Fehrman et al., 2019) in its own right. We draw on literature that shows convergence by using the Polyak-Łojasiewicz inequality (Wojtowysch, 2021; Dereich & Kassing, 2024), which is a more realistic assumption in the deep learning context than, for example convexity, because it can hold locally true for non-convex loss functions that are common in machine learning.

## 2 CONTROLLING THE IMPLICIT BIAS WITH EXPLICIT REGULARIZATION

**Structure of theoretical exposition.** Our first goal is to advance the mirror flow framework from (Li et al., 2022) to incorporate time-dependent regularization. ~~This is~~ a key innovation that forms the foundation of our proposed PILoT algorithm (Algorithm 1). The dynamical description also covers constant regularization, as implemented in the spread algorithm. We begin by integrating time-dependent regularization in the mirror flow framework in the case of the parameterization  $m \odot w$  combined with time-varying weight decay. This corresponds to a time-dependent Bregman potential, enabling a more dynamic and powerful form of implicit regularization. The implicit regularization becomes controllable and moves from an  $L_2$  to an  $L_1$  regularization. Building on this, Theorem 2.1 rigorously characterizes this process within this extended framework, offering new insights into the sparsification process. ~~Moreover, using the characterization, Then~~ Theorem 2.2 ~~establishes~~ convergence to a solution of the original optimization problem. Sparsity is still attained according to Theorem 2.1, which can also be observed in the gradient flow in Eq. (6) of PILoT. For diagonal linear networks, which is an analytically tractable setting, we also prove optimality, as stated by Theorem 2.3. This highlights a mechanism how our method PILoT improves over spread, since spread cannot reach optimality.

**Optimization problem.** Consider the following time-dependent optimization problem for a loss function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ :

$$\min_{m, w \in \mathbb{R}^n} f(m \odot w) + \alpha_t (||m||_{L_2}^2 + ||w||_{L_2}^2). \quad (2)$$

where  $\alpha_t \geq 0$  can change during training. ~~(Ziyin & Wang, 2023) for constant~~ In contrast, (Ziyin & Wang, 2023) set  $\alpha_t = \alpha$  that constant and show that Eq. (2) is equivalent to the LASSO objective. Why does spread tend to outperform LASSO then?

**Seeking answers in the training dynamics.** The gradient flow associated with minimizing the continuously differentiable loss function  $f$  is:  $dx_t = -\nabla f(x_t)dt$ ,  $x_0 = x_{\text{init}}$ . Using this gradient flow framework, (Li et al., 2022) show that a reparameterization or overparameterization of the parameters  $x$  leads to a mirror flow. A mirror flow informally minimizes a potential in the background, for example, the  $L_1$  or  $L_2$ -norm. In contrast, explicit regularization forces a direct trade-off. The no need for a trade-off becomes clear in the convergence and optimality theorem.

**Mirror flow.** Concretely, to define a mirror flow, let  $R : \mathbb{R}^n \rightarrow \mathbb{R}$  be a differentiable function. It is described by

$$d\nabla_x R(x_t) = -\nabla_x f(x_t)dt, \quad x_0 = x_{\text{init}}.$$

(Li et al., 2022) provide sufficient conditions for a parameterization  $g : M \rightarrow \mathbb{R}^n$  to induce a mirror flow, where  $M$  is a smooth sub-manifold in  $\mathbb{R}^D$  for  $D \geq n$ . The parameterization  $m \odot w$  falls into this category. The corresponding potential ~~depending on the initialization of  $m_0$  and  $w_0$~~   $R$  is either close to the  $L_1$  or  $L_2$  norm depending on the initialization of  $m_0$  and  $w_0$ . If we could steer it towards  $L_1$ , we could therefore induce an implicit regularization towards sparsity. ~~However, even in this case, we face multiple caveats. Firstly, the, yet, not without issues.~~

**Two caveats and their solution.** a) The potential  $R$  attains its global minimum at the initialization initial  $x_0$  (and not 0), a fact that. This also holds for other reparameterizations (Li et al., 2022). In consequence, we would not promote actual sparsity. ~~Secondly, to induce the for  $x_0 \neq 0$ .~~ b) To

induce  $L_1$  regularization and enter the rich regime, the initialization of both  $m_0$  and  $w_0$  would need to be exponentially small (Woodworth et al., 2020). ~~The explicit~~

~~The explicit dynamic~~ regularization of PILoT in Eq. (2) solves both of these problems. ~~For completeness, the~~, as we show next. The corresponding mirror function  $R$  is stated in Theorem A.1 and the corresponding convergence and optimality theorems in Theorem A.2 and A.4. ~~These results motivate the use and extension of the mirror flow framework.~~

**Dynamic regularization.** ~~We now~~ Next we present the main result, the dynamical description with time-dependent regularization. The exact dynamics are described by the time-dependent mirror flow and is derived in Theorem 2.1.

**Theorem 2.1** *Let  $|w_{0,i}| < m_{0,i}$  for all  $i \in [n]$ , ~~then~~ the time-dependent Bregman potential is given by*

$$R_{a_t}(x) = \frac{1}{2} \sum_{i=1}^n x_i \operatorname{arcsinh} \left( \frac{x_i}{a_{t,i}} \right) - \sqrt{x_i^2 + a_{t,i}^2} - x_i \log \left( \frac{u_{0,i}}{v_{0,i}} \right), \quad (3)$$

with  $a_{t,i} = 2u_{0,i}v_{0,i} \exp \left( -2 \int_0^t \alpha_s ds \right)$  and  $u_{0,i} = \frac{m_{0,i} + w_{0,i}}{\sqrt{2}}$  and  $v_{0,i} = \frac{m_{0,i} - w_{0,i}}{\sqrt{2}}$ . ~~The time-dependent Bregman potential gradient flow of  $x_t = m_t \odot w_t$  induced by Eq. (2) then satisfies~~

$$d\nabla R_{a_t}(x_t) = -\nabla f(x_t) dt, \quad x_0 = m_0 \odot w_0.$$

**Proof.** The proof is given in the appendix. The main steps are: a) Deriving the evolution of the gradient flow (Lemma B.1). b) Showing that it satisfies the time-dependent mirror flow (Lemma B.2). Note that step a) also derives Eq. (6). ~~The~~

~~Observe that the~~ potential in Eq. (3) now depends on  $a_t$ . This ~~has the following effect on the position of the global minimum :~~ ~~changes the global minimum to~~

$$\nabla R_{a_t}(x) = 0 \Leftrightarrow x = \exp \left( -2 \int_0^t \alpha_s ds \right) \odot m_0 \odot w_0.$$

Thus, we gain control over the positional implicit bias, solving our problem with the nonzero global minimum. Next, we characterize the asymptotic behavior, which we control in practice with  $\alpha_t$ , ~~which determines also  $a_t$~~ . The asymptotics follows from Theorem 2 in (Woodworth et al., 2020). For  $a \rightarrow 0$  and  $|\frac{x}{a}| \rightarrow \infty$ , we receive

$$R_a(x) \sim \log \left( \frac{1}{a} \right) \|x\|_{L_1}.$$

Interestingly, the term  $x_i \log \left( \frac{u_{0,i}}{v_{0,i}} \right)$  does not play a role in the asymptotics. The reason is that  $\log \left( \frac{1}{a} \right)$  in front of the other term dominates. Figure 1 illustrates the asymptotics ~~in the case of  $n=1$ . Indeed, we observe that increasing for the one dimensional case. We observe that our two previously identified mirror flow caveats can be resolved: a) Increasing  $a$  moves the minimum to the origin, leading to an  $L_1$  regularization. This implies initializing at zero with an exponentially small scaling (i.e.  $a \rightarrow 0$ ) is not necessary. b) Moreover, the regularization  $\alpha_t$  has an exponential and time-dependent effect on  $a_t$ , thus solving the other issue of. It thus enables steering the dynamics towards an  $L_1$  regularization at the desired speed. In conclusion, our extension and novel analysis have discovered a promising initialization and revealed how explicit regularization solves the problems of the standard mirror flow framework.~~

**Remark 2.1** *At the end of LASSO training, the regularization ~~could~~ ~~can~~ be turned off, ~~to allow a to enable the~~ search for a better solution. However, this ~~approach~~ risks losing the benefits of the regularization. ~~For~~, for example, if the basin of attraction contains non-sparse critical points. In contrast, for the  $m \odot w$  reparameterization, the time-dependent Bregman potential steers (but does not force) the bias towards sparsity. ~~Therefore, dynamically updating the regularization strength makes sense with the reparameterization.~~*

**Convergence and optimality.** It remains to be shown that convergence and optimality results transfer from the mirror flow framework to the time-dependent mirror flow framework. For quasi-convex loss functions, we prove convergence to a critical point. For convex or quasi-convex functions that satisfy the PL-inequality, we derive convergence to a minimizer.

**Theorem 2.2** Assume  $f$  is quasi-convex,  $\nabla f$  is locally Lipschitz and  $\operatorname{argmin}\{f(x)|x \in \mathbb{R}^n\}$  is non-empty. Assume  $\alpha_t \geq 0$  for all  $t \geq 0$  and that  $\int_0^t \alpha_s ds < \infty$  for  $t \in [0, \infty) \cup \{\infty\}$ . Then as  $t \rightarrow \infty$ ,  $x_t$  converges to some critical point  $x^*$ . Furthermore, if  $f$  is either convex or both quasi-convex and satisfies the PL-inequality in Eq. (9). Then  $x_t$  converges to an interpolator  $x^*$  that is a minimizer of  $f$ . Furthermore, in the PL-inequality case, the loss converges linearly such that there is a constant  $C > 0$  such that

$$f(x_t) - f(x^*) \leq B \exp(-\lambda a_\infty t), \quad (4)$$

where  $B = (f(x_0) - f(x^*)) \exp(C \|x^*\|_{L_2} \int_0^\infty \alpha_s ds)$  with  $C$  depending on the smoothness of the loss function.

**Proof.** The main steps of the proof are to show that a) the iterates are bounded and converge to a critical point (Lemma B.3); b) the loss converges (Theorem B.1). A noteworthy tool is a time-dependent Bregman divergence, which we use to bound the iterates. Furthermore, we utilize that  $\alpha_t \geq 0$  converges to zero, resulting in a non-increasing evolution of the loss.

**Potential drawbacks.** Theorem 2.2 guarantees convergence in the case of implicit regularization. Explicit  $L_1$  regularization or spred, on the other hand, cannot achieve the same result due to constant regularization, which we will also highlight in experiments. Regardless, note that the constant  $B$  in Eq. (4) could be large. Furthermore, to reach the implicit  $L_1$ -regularization,  $a_\infty$  needs to be exponentially small similarly as in Theorem 2 in (Woodworth et al., 2020). These two potential drawbacks also reveal where the method will work, namely, in overparameterized settings where the solution  $x^*$  should have less active parameters. Then  $B$  is potentially relatively small.

**Remark 2.2** If  $\nabla f$  is one-sided inversely Lipschitz, a speed-up is possible. The quantity that needs to be bounded for convergence is  $-\nabla f(x_t)^\top x_t$ . In this case, we get

$$-\nabla f(x_t)^\top x_t \leq -\nabla f(x_t)^\top x^* - \|x_t - x^*\|_{L_2}^2 \leq C \|x^*\|_{L_2} - \|x_t - x^*\|_{L_2}^2,$$

where  $C$  is the bound on the smoothness of the loss function  $f$ . This implies that when the interpolator  $x^* \approx 0$  is small, the right-hand side is negative, leading to a speed-up. This condition is also known as coercive.

**Optimality.** Note, the main assumption in Theorem 2.2 is  $\alpha_t \rightarrow 0$ , this ensures convergence to a minimizer of the original problem, while retaining sparsity depending on  $a_\infty$ . In the case of diagonal linear networks, we can even prove optimality with respect to the final Bregman potential  $R_{a_\infty}$ .

**Theorem 2.3** In case of under-determined regression consider the loss function  $f(x) = \tilde{f}(Zx - Y)$ . Assume  $f$  satisfies the conditions with at least one of the convergence criteria of Theorem 2.2. Then  $x_t$  converges to  $x^*$  such that

$$x^* = \operatorname{argmin}_{Zx=Y} R_{a_\infty}(x). \quad (5)$$

**Proof.** We show that the KKT conditions of Problem (5) are satisfied (Theorem B.2).

**Take away.** While we have shown that our extended mirror flow framework can attain convergence and optimality in an analytically tractable scenario, furthermore, from a practical side, it also gives us new tools to derive more promising continuous sparsification techniques. For instance, indirectly defining a time-dependent Bregman potential, we propose next with implicit regularization. In the following section, we propose a way to dynamically control the transition from implicit  $L_2$  regularization to  $L_1$  during training with the help of the derived time-dependent Bregman potential.

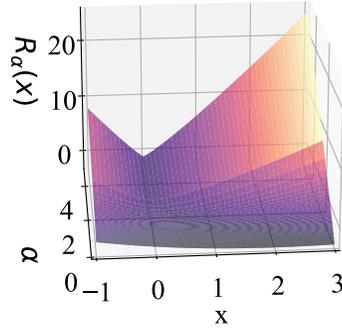


Figure 1: Evolution of the time-dependent Bregman potential.  $\alpha = \int_0^t \alpha_s ds$  is the exponent of  $a_t$ .

### 3 THE ALGORITHM: PILOT

Like spread, our new algorithm PILOT (Algorithm 1) utilizes the parameterization  $m \odot w$ , but proposes a novel initialization and dynamic regularization schedule to control the transition from implicit  $L_2$  to  $L_1$  regularization. To attain the desired results in the original parameterization  $x$ , we first derive its gradient flow.

**Gradient flow.** Essentially, the gradient flow follows from the analysis in Section 2. ~~Inspired by According to~~ Theorem 2.2, we ~~design our algorithm to achieve better performance while guaranteeing convergence. The main consequence is that guarantee convergence by ensuring~~  $\alpha_t \rightarrow 0$ . Concretely, the gradient flow for  ~~$x = m \odot w$~~   $x_t = m_t \odot w_t$  induced by Eq. (2) is given by:

$$dx_t = -\sqrt{x_t^2 + a_t^2} \odot \left( \nabla f(x_t) + 2\alpha_t \frac{x_t}{\sqrt{x_t^2 + a_t^2}} \right) dt, \quad x_0 = x_{init}, \quad (6)$$

where  ~~$a_t = (m_0^2 - w_0^2) \exp(-2 \int_0^t \alpha_s ds)$~~   $a_t = (m_0^2 - w_0^2) \exp(-2 \int_0^t \alpha_s ds)$ .  $m_0$  and  $w_0$  have to be initialized such that  $m_0 \odot w_0 = x_{init}$ . Note that all operations are point-wise. The derivation is based on the time-dependent mirror flow in Section 2.

**Remark 3.1** *The gradient flow ~~PILOT in in~~ Eq. (6) allows us to make a direct comparison to the continuous sparsification method STR (Kusupati et al., 2020). Instead of the soft thresholding operator, we have  $\sqrt{x_t^2 + a_t^2}$ . The main difference is that STR does not change the magnitude of the gradient update outside of the (learnable) threshold, while both PILOT and spread actively change the magnitude depending on the magnitude of the weight. This active sparsification explains why spread and also PILOT can perform better in the high-sparsity regime.*

**Spread.** The gradient flow in Eq. (6) explains why spread ~~usually~~ performs better than LASSO and highlights where spread can further be improved. Note that the balanced initialization of spread is defined such that  $m_0^2 - w_0^2 = 0$  and the regularization is constant  $\alpha_t = \alpha$ . Plugging this into Eq. (6) gives

$$dx_t = -\sqrt{x_t^2} \odot (\nabla f(x_t) + 2\alpha \text{sign}(x_t)) dt, \quad x_0 = x_{init}. \quad (7)$$

Compare this with the gradient flow of LASSO with regularization strength  $2\alpha$ :

$$dx_t = -(\nabla f(x_t) + 2\alpha \text{sign}(x_t)) dt, \quad x_0 = x_{init}.$$

We observe that the main difference to the gradient flow of LASSO is the factor  $\sqrt{x_t^2}$ . This implies the considerable drawback that spread gradient flows cannot sign flip. Therefore, it cannot reach the optimal solution or specific minimizers potentially. Another way to see this is studying the evolution  $x_t = x_0 \exp\left(-4\text{sign}(x_0) \int_0^t \nabla f(x_s) ds - 2\alpha t\right)$  satisfying Eq. (7). In practice, the absence of sign flips might be remedied by using a large learning rate and noise. The evolution also explains why it can perform better than LASSO, as it decays redundant parameters exponentially faster instead of linearly. In other words, the gradient update is proportional to the magnitude of the parameter. Therefore, the evolution of ~~PILOT spread~~ (Eq. (7)) can converge faster and come closer to zero than LASSO.

~~The gradient flow in Eq. (6) allows us to make a direct comparison to the continuous sparsification method STR (Kusupati et al., 2020). Instead of the soft thresholding operator, we have  $\sqrt{x_t^2 + a_t^2}$ . The main difference is that STR does not change the magnitude of the gradient update outside of the (learnable) threshold, while both PILOT and spread actively change the magnitude depending on the magnitude of the weight. This active sparsification explains why spread and also PILOT can perform better in the high-sparsity regime.~~

**PILOT.** Given our insights into the spread algorithm, we want to remedy these by using Our main goal is to remedy the caveats of the spread by inducing the more general gradient flow in Eq. (6). The first improvement is ensuring sign flips are possible to enable sign flips by changing the initialization to  $m_0^2 - w_0^2 = \beta > 0$ , which we will refer to as where  $\beta$  denotes the scaling constant. Our experiments set  $\beta = 1$ , which is motivated by the discretization of the gradient flow. After discretizing Eq. (6),

the effective learning rate at initialization  $x_0 = 0$  is  $\eta|\beta|$ , where  $\eta > 0$  is the learning rate. Therefore, we use  $\beta = 1$  in the experiments so that the learning rate  $\eta$  is not altered.

Our second and main improvement is induced by the time dependence of  $\alpha_t$ . The  $\alpha_t$  together with  $a_t$  control controls the strength of both the implicit and explicit regularization. We observe if via  $a_t$ . If  $a_t \gg x_t$ , then the regularization term in Eq. (6) resembles an  $L_2$  instead of an  $L_1$  norm. Therefore decreasing  $a_t$  moves the implicit regularization from  $L_2$  to  $L_1$ . Accordingly, we sparsify gradually instead of abruptly at initialization, only mildly in early training epochs in contrast to `spred`. We have shown this formally in Section 2. Furthermore, convergence is covered by Theorem 2.2 when  $\alpha_t \rightarrow 0$ . The effect of the regularization remains during training, which is captured by the term  $\sqrt{x_t^2 + a_t^2}$  in Eq. (6). In consequence, PILoT leads to better accuracy while still having a lasting sparsifying effect on the dynamics. Even when PILoT attains a similar sparsity as `spred` at the end of the training dynamics, it can usually still achieve a higher accuracy due to its improved training dynamics.

**Details on PILoT.** These insights lead to The described design choices define Algorithm 1. Our update of the regularization strength depends on  $\alpha_k$ , depends on three quantities: a) the sparsity threshold  $K$  for the weights (a hyperparameter), b) the training accuracy, and c)  $\delta \geq 1$ , the multiplicative factor to gradually increase or decrease the regularization strength. It is proportional to the current order of  $\alpha_k$ . This makes the algorithm The regularization strength (and thus sparsity) grows if the sparsity threshold has not been reached yet and the training accuracy has increased in the previous gradient update step. As the strength is adaptive, the algorithm is less sensitive to the initialization of the regularization strength, as it is adaptive initial strength  $\alpha_0$ . Note that the setting  $\delta = 1$  and  $\beta = 0$  corresponds to `spred`, therefore. Therefore, PILoT is a strict generalization. Furthermore, of `spred`. In contrast to `spred`, however, after half of the training epochs, we decay the regularization strength regardless of whether the sparsity threshold  $K$  is reached. This guarantees convergence of the corresponding gradient flow, as captured by in accordance with Theorem 2.2.

---

#### Algorithm 1 PILoT

---

**Require:** epochs  $T$ , schedule  $\alpha_{init}$ , initialization  $x_{init}$ , scaling constant  $\beta$   
Initialize  $m_0, w_0$  such that  $m_0 \odot w_0 = x_{init}$ ,  $m_0^2 - w_0^2 = 2u_0 \odot v_0 = \beta m_0^2 - w_0^2 = \beta$ ,  $\delta \geq 1$  and,  $K$   
 $\alpha_0 \leftarrow \alpha_{init}$   
 $Current\_training\_acc \leftarrow 0$   
Set  $\tilde{f}(m, w, \alpha_0) := f(m \odot w) + \alpha_0 (\|m\|_{L_2}^2 + \|w\|_{L_2}^2)$   
**for**  $k$  in  $1 \dots T$  **do**  
     $(m_k, w_k) = \text{OptimizerStep}(\tilde{f}(m_{k-1}, w_{k-1}, \alpha_{k-1}))$   
    **if**  $Training\_acc \geq Current\_training\_acc$  and  $\|m_k \odot w_k\|_{L_1} \geq K$  and  $k \leq \frac{T}{2}$  **then**  
         $\alpha_k \leftarrow \alpha_{k-1} \delta$   
    **else**  
         $\alpha_k \leftarrow \alpha_{k-1} / \delta$   
    **end if**  
     $Current\_training\_acc \leftarrow Training\_acc$   
**end for**  
**return** Model  $f(x_T)$  with  $x_T = m_T \odot w_T$

---

## 4 EXPERIMENTS

We demonstrate the effectiveness of PILoT in extensive experiments covering three different scenarios. Firstly, we confirm our theoretical results on the gradient flow in Theorem 2.3. Secondly, we compare PILoT with other state-of-the-art continuous sparsification methods such as STR and `spred` in a so-called (Kusupati et al., 2020) and `spred` (Ziyin & Wang, 2023) in a one-shot setting. In this context, we also isolate the individual contribution of our initialization. Finally, we combine PILoT with iterative pruning methods such as LRR and WR. WR (Frankle & Carbin, 2019) and LRR (Maene et al., 2021).

**Memory requirements.** As most other continuous sparsification approaches, note that PLoT doubles the number of parameters during training. Yet, according to Ziyin & Wang (2023), the training time of a ResNet50 with  $m \odot w$  parameterization on ImageNet increases roughly by 5% only and the memory cost is negligible if the batch size is larger than 50. At inference, we would return to the original representation  $x$  and therefore benefit from the improved sparsification.

**Diagonal Linear Network.** A simulation of gradient flow on a diagonal linear network is given for the different regularizations. The We have proven optimality for the analytically tractable setting of diagonal linear networks, for which we have proven optimality, illustrates. Now we illustrate the benefit of our initialization and dynamic explicit regularization and highlights the importance of choosing the right schedule for. Furthermore, we highlight the impact of a good dynamic schedule of the regularization strength  $\alpha_t$ . The hyperparameters are set to We use  $d = 40$ , amount of data points with feature dimension  $n = 100$  and sample  $z_j \sim N(0, \mathbb{I}_n)$  for  $j \in [d]$ . The ground truth  $x^*$  is set such that  $\|x^*\|_{L_0} = 5$ . Furthermore, the network parameters are initialized with  $x \sim N(0, \mathbb{I}_n \frac{1}{\sqrt{n}})$ ,  $x_0 \sim N(0, \mathbb{I}_n \frac{1}{\sqrt{n}})$ . The step-size is  $\eta = 10^{-4}$  and the trajectories are averaged over 5 initializations. 0.95 confidence regions are indicated by shades. As standard, the The mean squared error is used as loss function for this under-determined linear regression problem. We report the distance of the evolution to the ground truth  $\|x_t - x^*\|_{L_2}$  over training time in Figure 2, which confirms our theoretical insights. While we compare different initializations, Two different initializations, i.e., the one of spread and PLoT, different (fixed) regularization schedules for both, and LASSO, the figure reports the best regularization schedule for each initialization and LASSO. The initialization of  $m$  and  $w$  is critical for performance, as the of PLoT, and different regularization schedules are considered. The schedules are described in Appendix C.1. Results for the best ones are shown in the figure and confirm our theoretical insights. The inability to sign flip prevents spread from reaching the ground truth for all considered schedules. Therefore, in the case of gradient flow, the balanced initialization should be avoided. Furthermore, with a dynamic regularization, our PLoT initialization outperforms both spread and LASSO, reaching the ground truth. The best-performing schedule for the PLoT initialization is a geometrically decaying schedule. This serves as additional motivation for the regularization update step, as also implemented in Algorithm 1. In contrast, for the other two methods, a constant regularization works best. This experimentally confirms Theorem 2.3 and Remark 2.2. Note that LASSO with gradient descent performs as expected. A constant schedule leads to the best performance, with a too small or large constant leading to worse performance. In addition, decaying schedules perform also worse. Decaying schedules perform worse than constant ones supporting Remark 2.1.

The advantage of PLoT is that it has access to the full parameter space (i.e. can sign flip) and the explicit regularization enables us to enter the rich regime, thus obtaining an implicit  $L_1$  regularization, as illustrated by Figure 1 and suggested by Theorem 2.1.

**One-shot sparsification.** Firstly, we compare our method PLoT with STR, spread, and LASSO on CIFAR10 and CIFAR100 training a ResNet-20 or ResNet-18, respectively. Furthermore, in the case of CIFAR10, we also implement the novel initialization ( $m_0^2 - w_0^2 = 1$ ) without dynamic regularization to isolate its benefits. We consider two learning rates  $\{0.1, 0.2\}$  and the weight decay range  $\{1e-5, \dots, 1e-2\}$  for CIFAR10 and range  $\{1e-4, \dots, 1e-3\}$  for CIFAR100 and always show the best result. The same range for the regularization strength is explored for LASSO. The other hyperparameters are reported in the appendix. Secondly, we train ResNet-50 on ImageNet with the setup of STR (Kusupati et al., 2020) and compare directly with their results. Furthermore, we implement both PLoT and spread in this setting.

Figure 3 presents our results for the results for CIFAR10 and CIFAR100. PLoT outperforms all other

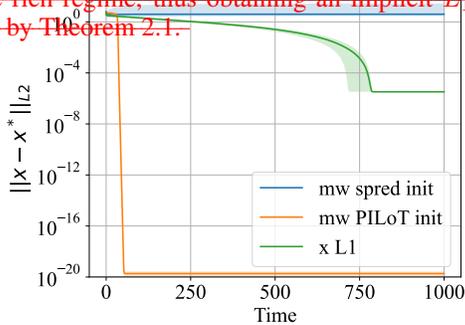
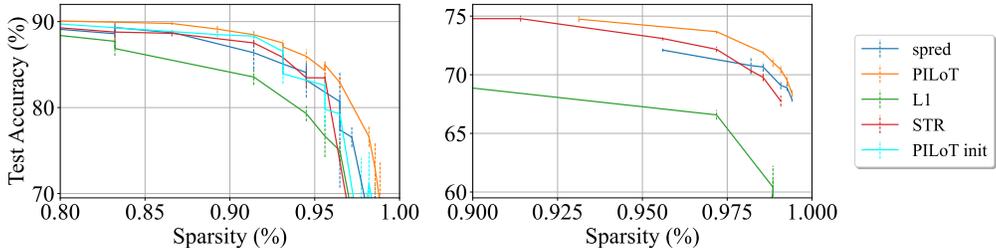


Figure 2: A simulation of gradient flow on a diagonal linear network is given for the different regularizations.

540 methods and is particularly effective in the high-  
 541 sparsity regime. ~~Already, our~~ Our PLoT initial-  
 542 ization leads to improvements over spread and ~~even~~  
 543 STR for medium levels of sparsity. This supports  
 544 our theoretical insight into the role of initializations  
 545 and how they influence the implicit bias. In addition,  
 546 STR is outperformed by spread in the high-sparsity  
 547 regime, confirming the findings of (Ziyin & Wang,  
 548 2023).



558 Figure 3: One-shot sparsification. Acc. versus sparsity for CIFAR10 (left) and CIFAR100 (right).

560 Table 1: ResNet-50 on ImageNet sparsity (%) versus accuracy (%) results.

Method	Top-1 Acc	Sparsity	Method	Top-1 Acc	Sparsity
ResNet-50	77.01	0	ResNet-50	77.01	0
STR	70.4	95.03	STR	67.22	96.53
spread	69.47	94.50	spread	66.12	97.19
PLoT	<b>72.67</b>	94.00	PLoT	<b>68.49</b>	<b>97.19</b>
PLoT	<b>71.30</b>	95.00	STR	61.46	98.05
PLoT	<b>71.05</b>	<b>95.60</b>	spread	62.71	98.20
PLoT	<b>70.49</b>	<b>96.00</b>	PLoT	<b>66.49</b>	97.75
STR	74.73	87.7	PLoT	<b>64.06</b>	<b>98.20</b>
spread <sup>1</sup>	75.5	80.00			
spread	72.64	79.03			
PLoT	75.62	80.00			
STR	<b>74.73</b>	87.7			
STR	74.01	90.55			
spread	71.84	89.26			
PLoT	<b>74.73</b>	88.00			
PLoT	74.04	<b>91.00</b>			

577 In Table 1, we compare PLoT to both STR and  
 578 spread on ImageNet (Deng et al., 2009). See  
 579 Appendix C.2 Table 3 for details on experimental  
 580 configurations. Our method competes with or out-  
 581 performs all baselines at medium and high sparsity  
 582 levels. In addition, it improves over spread ~~for the at~~  
 583 80% sparsity, even when spread is initialized with a  
 584 77% pretrained ResNet-50. This highlights the ef-  
 585 fectiveness of PLoT and confirms the insights from  
 586 the developed theory.

587 **Iterative Pruning.** To demonstrate the versatility  
 588 of PLoT, we also combine it with the state-of-the-art  
 589 iterative pruning methods Learning Rate Rewinding  
 590 (LRR) (Maene et al., 2021) and Weight Rewinding  
 591 (WR) (Frankle et al., 2019) on ImageNet ~~with using~~  
 592 a ResNet-18. For simplicity, we use  $\beta = 1$  and no  
 593

<sup>1</sup>Starting from a pretrained model with 77% validation accuracy

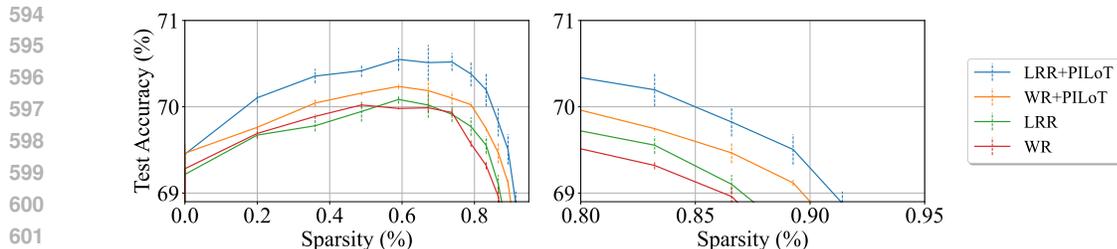


Figure 4: Learning Rate Rewinding (LRR) and Weight Rewinding (WR) with PILoT on ImageNet ResNet-18. The left plot is the complete plot and the right plot is zoomed in on the higher sparsity regime.

regularization. We see in Figure 4 that the parameterization  $m \odot w$  further boosts the performance of iterative methods. Remarkably, WR becomes competitive with LRR. Additional experiments on CIFAR10 and CIFAR100 with regularization are in Appendix C.3. The regularization further helps to reach higher accuracy at high sparsities.

## 5 DISCUSSION

We have shed light on the inner workings of continuous sparsification, ~~a state-of-the-art approach to prune neural networks that tries to solve an intractable optimization problem of mixed discrete and continuous nature.~~ Its basic relaxed formulation utilizes the parameterization  $m \odot w$ , which induces an implicit bias towards sparsity. In contrast to an explicit  $L_1$ -regularization, it enjoys all the benefits of an implicit regularization that caters first to the loss and not a sparsity penalty. Exploiting this insight for neural network sparsification, we have proposed PILoT, which relies on a controllable regularization that acts like an implicit regularization in the original neural network parameter space and, remarkably, corresponds to a time-dependent Bregman potential. ~~It therefore enjoys all the benefits of an implicit regularization that caters first to the loss and not a sparsity penalty. Furthermore~~ As we have shown, the time-dependent control enables the associated mirror flow to enter the rich regime and to so-called rich regime, and thus effectively change the implicit regularization from  $L_2$  to  $L_1$ . This property is central to our proofs that show showing convergence of our approach for (quasi)-convex loss functions and optimality for underdetermined linear regression. ~~Moreover, our analysis of the training dynamics explains why the parameterization  $m \odot w$  performs better than LASSO (and thus an  $L_1$  penalty) and why previous work (Ziyin & Wang, 2023) is limited by its initialization and constant regularization approach.~~ Experiments on standard vision benchmarks further corroborate the utility of our theoretical insights, as our proposal PILoT achieves significant improvements over state-of-the-art baselines.

## REFERENCES

- 648  
649  
650 Aws Albarghouthi. Introduction to neural network ver-  
651 ification, 2021.
- 652 Felipe Alvarez, Jérôme Bolte, and Olivier Brahic.  
653 Hessian riemannian gradient flows in convex  
654 programming. *SIAM Journal on Control and*  
655 *Optimization*, 43(2):477–501, January 2004. ISSN  
656 1095-7138. doi: 10.1137/s0363012902419977.  
657 URL [http://dx.doi.org/10.1137/  
658 S0363012902419977](http://dx.doi.org/10.1137/S0363012902419977).
- 659 Amir Beck and Marc Teboulle. Mirror de-  
660 scent and nonlinear projected subgradient  
661 methods for convex optimization. *Opera-*  
662 *tions Research Letters*, 31(3):167–175, 2003.  
663 ISSN 0167-6377. doi: [https://doi.org/10.](https://doi.org/10.1016/S0167-6377(02)00231-6)  
664 [1016/S0167-6377\(02\)00231-6](https://doi.org/10.1016/S0167-6377(02)00231-6). URL [https:](https://www.sciencedirect.com/science/article/pii/S0167637702002316)  
665 [//www.sciencedirect.com/science/  
666 article/pii/S0167637702002316](https://www.sciencedirect.com/science/article/pii/S0167637702002316).
- 667 Stephen P. Boyd and Lieven Vandenbergh. Con-  
668 vex optimization. 2009. URL [https://web.](https://web.stanford.edu/~boyd/cvxbook/)  
669 [stanford.edu/~boyd/cvxbook/](https://web.stanford.edu/~boyd/cvxbook/).
- 670 Rebekka Burkholz. Convolutional and residual net-  
671 works provably contain lottery tickets. In *Interna-*  
672 *tional Conference on Machine Learning*, 2022a.
- 673 Rebekka Burkholz. Most activation functions can win  
674 the lottery without excessive depth. In *Advances in*  
675 *Neural Information Processing Systems*, 2022b.
- 676  
677 Rebekka Burkholz, Nilanjana Laha, Rajarshi Mukher-  
678 jee, and Alkis Gotovos. On the existence of univer-  
679 sal lottery tickets. In *International Conference on*  
680 *Learning Representations*, 2022.
- 681 Tianlong Chen, Zhenyu Zhang, Jun Wu, Randy Huang,  
682 Sijia Liu, Shiyu Chang, and Zhangyang Wang. Can  
683 you win everything with a lottery ticket? *Trans-*  
684 *actions on Machine Learning Research*, 2022. ISSN  
685 2835-8856. URL [https://openreview.net/  
686 forum?id=JL6MU9XFzW](https://openreview.net/forum?id=JL6MU9XFzW).
- 687 Lénaïc Chizat and Francis Bach. Implicit bias of  
688 gradient descent for wide two-layer neural net-  
689 works trained with the logistic loss. In Jacob  
690 Abernethy and Shivani Agarwal (eds.), *Proceeed-*  
691 *ings of Thirty Third Conference on Learning The-*  
692 *ory*, volume 125 of *Proceedings of Machine Learn-*  
693 *ing Research*, pp. 1305–1338. PMLR, 09–12 Jul  
694 2020. URL [https://proceedings.mlr.](https://proceedings.mlr.press/v125/chizat20a.html)  
695 [press/v125/chizat20a.html](https://proceedings.mlr.press/v125/chizat20a.html).
- 696 Hung-Hsu Chou, Johannes Maly, and Dominik Stöger.  
697 How to induce regularization in linear models: A  
698 guide to reparametrizing gradient flow, 2024.
- 699 Arthur da Cunha, Emanuele Natale, and Laurent Vien-  
700 not. Proving the lottery ticket hypothesis for convo-  
701 lutional neural networks. In *International Confer-*  
*ence on Learning Representations*, 2022.

- 702 Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li,  
703 and Li Fei-Fei. Imagenet: A large-scale hierarchical  
704 image database. In *2009 IEEE Conference on Com-  
705 puter Vision and Pattern Recognition*, pp. 248–255,  
706 2009. doi: 10.1109/CVPR.2009.5206848.
- 707 Steffen Dereich and Sebastian Kassing. Conver-  
708 gence of stochastic gradient descent schemes for  
709 lojasiewicz-landscapes, 2024.
- 710 Tim Dettmers and Luke Zettlemoyer. Sparse networks  
711 from scratch: Faster training without losing perfor-  
712 mance. 2019.
- 713 Simon S. Du, Chi Jin, Jason D. Lee, Michael I. Jordan,  
714 Barnabas Poczos, and Aarti Singh. Gradient descent  
715 can take exponential time to escape saddle points,  
716 2017.
- 717 Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel  
718 Castro, and Erich Elsen. Rigging the lottery: Making  
719 all tickets winners. In *International Conference on  
720 Machine Learning*, pp. 2943–2952. PMLR, 2020.
- 721 Mathieu Even, Scott Pesme, Suriya Gunasekar, and  
722 Nicolas Flammarion. (s)gd over diagonal linear  
723 networks: Implicit regularisation, large stepsizes  
724 and edge of stability. *ArXiv*, abs/2302.08982, 2023.  
725 URL [https://api.semanticscholar.  
726 org/CorpusID:268042036](https://api.semanticscholar.org/CorpusID:268042036).
- 727 Alhussein Fawzi, Matej Balog, Aja Huang, Thomas  
728 Hubert, Bernardino Romera-Paredes, Moham-  
729 madamin Barekatin, Alexander Novikov, Fran-  
730 cisco J. R. Ruiz, Julian Schrittwieser, Grzegorz  
731 Swirszcz, David Silver, Demis Hassabis, and Push-  
732 meet Kohli. Discovering faster matrix multiplication  
733 algorithms with reinforcement learning. *Nature*,  
734 610(7930):47–53, 2022.
- 735 Benjamin Fehrman, Benjamin Gess, and Arnulf  
736 Jentzen. Convergence rates for the stochastic gradi-  
737 ent descent method for non-convex objective func-  
738 tions, 2019.
- 739 Damien Ferbach, Christos Tsirigotis, Gauthier Gidel,  
740 and Bose Avishek. A general framework for prov-  
741 ing the equivariant strong lottery ticket hypothesis,  
742 2022.
- 743 Jonas Fischer and Rebekka Burkholz. Plant ‘n’ seek:  
744 Can you find the winning ticket?, 2021.
- 745 Jonathan Frankle and Michael Carbin. The lot-  
746 tery ticket hypothesis: Finding sparse, trainable  
747 neural networks. *arXiv: Learning*, 2018. URL  
748 [https://api.semanticscholar.org/  
749 CorpusID:53388625](https://api.semanticscholar.org/CorpusID:53388625).
- 750 Jonathan Frankle and Michael Carbin. The lottery  
751 ticket hypothesis: Finding sparse, trainable neural  
752 networks. In *International Conference on Learning  
753 Representations*, 2019.

- 756 Jonathan Frankle, Gintare Karolina Dziugaite,  
757 Daniel M. Roy, and Michael Carbin. Linear  
758 mode connectivity and the lottery ticket hy-  
759 pothesis. *CoRR*, abs/1912.05671, 2019. URL  
760 <http://arxiv.org/abs/1912.05671>.
- 761 Jonathan Frankle, Gintare Karolina Dziugaite, Daniel  
762 Roy, and Michael Carbin. Pruning neural networks  
763 at initialization: Why are we missing the mark? In  
764 *International Conference on Learning Representa-*  
765 *tions*, 2021.
- 767 Advait Gadhikar and Rebekka Burkholz. Masks,  
768 signs, and learning rate rewinding. In *Twelfth*  
769 *International Conference on Learning Representa-*  
770 *tions*, 2024. URL [https://openreview.](https://openreview.net/forum?id=qODvxQ8TXW)  
771 [net/forum?id=qODvxQ8TXW](https://openreview.net/forum?id=qODvxQ8TXW).
- 772 Advait Harshal Gadhikar, Sohom Mukherjee, and Re-  
773 bekka Burkholz. Why random pruning is all we need  
774 to start sparse. In *International Conference on Ma-*  
775 *chine Learning*, 2023.
- 777 Suriya Gunasekar, Blake Woodworth, Srinadh Bho-  
778 janapalli, Behnam Neyshabur, and Nathan Srebro.  
779 Implicit regularization in matrix factorization, 2017.
- 780 Suriya Gunasekar, Jason Lee, Daniel Soudry, and  
781 Nathan Srebro. Characterizing implicit bias in terms  
782 of optimization geometry, 2020.
- 783 Song Han, Jeff Pool, John Tran, and William Dally.  
784 Learning both weights and connections for efficient  
785 neural network. *Advances in neural information pro-*  
786 *cessing systems*, 28, 2015.
- 788 Intekhab Hossain, Jonas Fischer, Rebekka Burkholz,  
789 and John Quackenbush. Not all tickets are equal and  
790 we know it: Guiding pruning with domain-specific  
791 knowledge, 2024.
- 792 Tian Jin, Michael Carbin, Daniel M. Roy, Jonathan  
793 Frankle, and Gintare Karolina Dziugaite. Prun-  
794 ing’s effect on generalization through the lens of  
795 training and regularization. In Alice H. Oh, Alekh  
796 Agarwal, Danielle Belgrave, and Kyunghyun Cho  
797 (eds.), *Advances in Neural Information Processing*  
798 *Systems*, 2022. URL [https://openreview.](https://openreview.net/forum?id=OrcLKV9sKWp)  
799 [net/forum?id=OrcLKV9sKWp](https://openreview.net/forum?id=OrcLKV9sKWp).
- 800 John Jumper, Richard Evans, Alexander Pritzel,  
801 Tim Green, Michael Figurnov, Olaf Ronneberger,  
802 Kathryn Tunyasuvunakool, Russ Bates, Augustin  
803 Žídek, Anna Potapenko, et al. Highly accurate pro-  
804 tein structure prediction with AlphaFold. *Nature*,  
805 596(7873):583–589, 2021.
- 807 Lynn H Kaack, Priya L Donti, Emma Strubell, George  
808 Kamiya, Felix Creutzig, and David Rolnick. Align-  
809 ing artificial intelligence with climate change mitiga-  
tion. *Nature Climate Change*, 12(6):518–527, 2022.

- 810 Hamed Karimi, Julie Nutini, and Mark Schmidt. Lin-  
811 ear convergence of gradient and proximal-gradient  
812 methods under the polyak-łojasiewicz condition. In  
813 Paolo Frasconi, Niels Landwehr, Giuseppe Manco,  
814 and Jilles Vreeken (eds.), *Machine Learning and*  
815 *Knowledge Discovery in Databases*, pp. 795–811,  
816 Cham, 2016. Springer International Publishing.  
817 ISBN 978-3-319-46128-1.
- 818 Aditya Kusupati, Vivek Ramanujan, Raghav Somani,  
819 Mitchell Wortsman, Prateek Jain, Sham Kakade, and  
820 Ali Farhadi. Soft threshold weight reparameteriza-  
821 tion for learnable sparsity. In *Proceedings of the In-*  
822 *ternational Conference on Machine Learning*, July  
823 2020.
- 824 Andrey Kuzmin, Markus Nagel, Saurabh Pitre,  
825 Sandeep Pendyam, Tijmen Blankevoort, and Max  
826 Welling. Taxonomy and evaluation of struc-  
827 tured compression of convolutional neural networks,  
828 2019.
- 829 Denis Kuznedelev, Eldar Kurtic, Eugenia Iofinova,  
830 Elias Frantar, Alexandra Peste, and Dan Alistarh.  
831 Accurate neural network pruning requires rethinking  
832 sparse optimization, 2023.
- 833 Mike Lasby, Anna Golubeva, Utku Evci, Mihai  
834 Nica, and Yani Ioannou. Dynamic sparse train-  
835 ing with structured sparsity. *arXiv preprint*  
836 *arXiv:2305.02299*, 2023.
- 837 Namhoon Lee, Thalaiyasingam Ajanthan, and Philip  
838 H. S. Torr. Snip: single-shot network pruning based  
839 on connection sensitivity. In *International Confer-*  
840 *ence on Learning Representations*, 2019.
- 842 Jianguyan Li, Thanh V. Nguyen, Chinmay Hegde,  
843 and Raymond K. W. Wong. Implicit sparse reg-  
844 ularization: The impact of depth and early stop-  
845 ping, 2021. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2108.05574)  
846 [2108.05574](https://arxiv.org/abs/2108.05574).
- 847 Jianguyan Li, Thanh V. Nguyen, Chinmay Hegde, and  
848 Raymond K. W. Wong. Implicit regularization for  
849 group sparsity, 2023. URL [https://arxiv.](https://arxiv.org/abs/2301.12540)  
850 [org/abs/2301.12540](https://arxiv.org/abs/2301.12540).
- 851 Zhiyuan Li, Tianhao Wang, Jason D. Lee, and San-  
852 jeev Arora. Implicit bias of gradient descent on  
853 reparametrized models: On equivalence to mirror  
854 descent. *ArXiv*, abs/2207.04036, 2022. URL  
855 [https://api.semanticscholar.org/](https://api.semanticscholar.org/CorpusID:250407876)  
856 [CorpusID:250407876](https://api.semanticscholar.org/CorpusID:250407876).
- 857 Bohan Liu, Zijie Zhang, Peixiong He, Zhensen Wang,  
858 Yang Xiao, Ruimeng Ye, Yang Zhou, Wei-Shinn Ku,  
859 and Bo Hui. A survey of lottery ticket hypothesis,  
860 2024.
- 861 Shiwei Liu and Zhangyang Wang. Ten lessons we have  
862 learned in the new ”sparseland”: A short handbook  
863 for sparse neural network researchers, 2023.

- 864 Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen,  
865 Decebal Constantin Mocanu, Zhangyang Wang, and  
866 Mykola Pechenizkiy. The unreasonable effective-  
867 ness of random pruning: Return of the most naive  
868 baseline for sparse training. In *International Con-  
869 ference on Learning Representations*, 2021.
- 870 Christos Louizos, Max Welling, and Diederik P.  
871 Kingma. Learning sparse neural networks through  
872  $l_0$  regularization, 2018.
- 873 Alexandra Sasha Luccioni, Yacine Jernite, and Emma  
874 Strubell. Power hungry processing: Watts driv-  
875 ing the cost of ai deployment? *arXiv preprint  
876 arXiv:2311.16863*, 2023.
- 877 Jaron Maene, Mingxiao Li, and Marie-Francine  
878 Moens. Towards understanding iterative magnitude  
879 pruning: Why lottery tickets win, 2021.
- 881 Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz,  
882 and Ohad Shamir. Proving the lottery ticket hypoth-  
883 esis: Pruning is all you need. In *International Con-  
884 ference on Machine Learning*, 2020.
- 885 Nina Narodytska, Hongce Zhang, Aarti Gupta, and  
886 Toby Walsh. In search for a sat-friendly bi-  
887 narized neural network architecture. In *Inter-  
888 national Conference on Learning Representations*,  
889 2020. URL [https://openreview.net/  
890 forum?id=SJx-j64FDr](https://openreview.net/forum?id=SJx-j64FDr).
- 891 A.S. Nemirovski and D.B. Yudin. *Problem Com-  
892 plexity and Method Efficiency in Optimization*.  
893 A Wiley-Interscience publication. Wiley, 1983.  
894 ISBN 9780471103455. URL [https://books.  
895 google.de/books?id=6ULvAAAAMAAJ](https://books.google.de/books?id=6ULvAAAAMAAJ).
- 896 Laurent Orseau, Marcus Hutter, and Omar Rivasplata.  
897 Logarithmic pruning is all you need. *Advances in  
898 Neural Information Processing Systems*, 33, 2020.
- 900 Shreyas Malakarjun Patil and Constantine Dovrolis.  
901 Phew: Constructing sparse networks that learn fast  
902 and generalize well without training data. In *Inter-  
903 national Conference on Machine Learning*, pp. 8432–  
904 8442. PMLR, 2021.
- 905 Mansheej Paul, Feng Chen, Brett W. Larsen, Jonathan  
906 Frankle, Surya Ganguli, and Gintare Karolina  
907 Dziugaite. Unmasking the lottery ticket hy-  
908 pothesis: What’s encoded in a winning ticket’s  
909 mask? In *The Eleventh International Confer-  
910 ence on Learning Representations*, 2023. URL  
911 [https://openreview.net/forum?id=  
912 xSsW2Am-ukZ](https://openreview.net/forum?id=xSsW2Am-ukZ).
- 913 Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit  
914 Vishwakarma, and Dimitris Papailiopoulos. Opti-  
915 mal lottery tickets via subset sum: Logarithmic over-  
916 parameterization is sufficient. In *Advances in Neu-  
917 ral Information Processing Systems*, volume 33, pp.  
2599–2610, 2020.

- 918 Scott Pesme, Loucas Pillaud-Vivien, and Nicolas  
919 Flammarion. Implicit bias of sgd for diagonal linear  
920 networks: a provable benefit of stochasticity, 2021.  
921
- 922 Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and  
923 Dan Alistarh. Ac/dc: Alternating compressed/de-  
924 compressed training of deep neural networks, 2021.
- 925 Hoang Pham, Shiwei Liu, Lichuan Xiang, Dung D Le,  
926 Hongkai Wen, Long Tran-Thanh, et al. Towards  
927 data-agnostic pruning at initialization: What makes  
928 a good sparse mask? In *Thirty-seventh Conference  
929 on Neural Information Processing Systems*, 2023.
- 930 Adityanarayanan Radhakrishnan, Mikhail Belkin, and  
931 Caroline Uhler. Linear convergence of generalized  
932 mirror descent with time-dependent mirrors, 2021.  
933
- 934 Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey  
935 Chu, and Mark Chen. Hierarchical text-conditional  
936 image generation with clip latents. *arXiv preprint  
937 arXiv:2204.06125*, 2022.
- 938 Alex Renda, Jonathan Frankle, and Michael Carbin.  
939 Comparing rewinding and fine-tuning in neural net-  
940 work pruning. In *International Conference on  
941 Learning Representations*, 2020.
- 942 Tyrrel R Rockafellar and Werner Fenchel. *Con-*  
943 *convex Analysis*. 1970. URL [https://api.](https://api.semanticscholar.org/CorpusID:198120397)  
944 [semanticscholar.org/CorpusID:](https://api.semanticscholar.org/CorpusID:198120397)  
945 [198120397](https://api.semanticscholar.org/CorpusID:198120397).
- 946 Pedro Savarese, Hugo Silva, and Michael Maire. Win-  
947 ning the lottery with continuous sparsification, 2021.  
948
- 949 Jonathan Schwarz, Siddhant M. Jayakumar, Razvan  
950 Pascanu, Peter E. Latham, and Yee Whye Teh. Pow-  
951 erpropagation: A sparsity inducing weight reparam-  
952 eterisation, 2021.
- 953 Heejune Sheen, Siyu Chen, Tianhao Wang, and Harri-  
954 son H. Zhou. Implicit regularization of gradient flow  
955 on one-layer softmax attention, 2024.  
956
- 957 Kartik Sreenivasan, Jy-yong Sohn, Liu Yang, Matthew  
958 Grinde, Alliot Nagle, Hongyi Wang, Eric Xing,  
959 Kangwook Lee, and Dimitris Papailiopoulos. Rare  
960 gems: Finding lottery tickets at initialization. *Ad-*  
961 *vances in Neural Information Processing Systems*,  
962 35:14529–14540, 2022.
- 963 Natalie Stephenson, Emily Shane, Jessica Chase, Ja-  
964 son Rowland, David Ries, Nicola Justice, Jie Zhang,  
965 Leong Chan, and Renzhi Cao. Survey of machine  
966 learning techniques in drug discovery. *Current drug  
967 metabolism*, 20(3):185–193, 2019.
- 968 Hidenori Tanaka, Daniel Kunin, Daniel L. Yamins, and  
969 Surya Ganguli. Pruning neural networks without any  
970 data by iteratively conserving synaptic flow. In *Ad-*  
971 *vances in Neural Information Processing Systems*,  
2020.

- 972 Tomas Vaškevičius, Varun Kanade, and Patrick Rebes-  
973 chini. Implicit regularization for optimal sparse  
974 recovery, 2019. URL [https://arxiv.org/  
975 abs/1909.05122](https://arxiv.org/abs/1909.05122).
- 976  
977 Chaoqi Wang, Guodong Zhang, and Roger B. Grosse.  
978 Picking winning tickets before training by preserv-  
979 ing gradient flow. In *International Conference on  
980 Learning Representations*, 2020.
- 981 Kun Wang, Yuxuan Liang, Pengkun Wang, Xu Wang,  
982 Pengfei Gu, Junfeng Fang, and Yang Wang.  
983 Searching lottery tickets in graph neural net-  
984 works: A dual perspective. In *The Eleventh  
985 International Conference on Learning Represen-  
986 tations*, 2023. URL [https://openreview.  
987 net/forum?id=Dvs-a3aymPe](https://openreview.net/forum?id=Dvs-a3aymPe).
- 988 Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen,  
989 and Hai Li. Learning structured sparsity in deep neu-  
990 ral networks. In *Advances in Neural Information  
991 Processing Systems*, volume 29, 2016.
- 992  
993 Stephan Wojtowysch. Stochastic gradient descent with  
994 noise of machine learning type. part i: Discrete time  
995 analysis, 2021.
- 996  
997 Blake Woodworth, Suriya Gunasekar, Jason D. Lee,  
998 Edward Moroshko, Pedro Savarese, Itay Golan,  
999 Daniel Soudry, and Nathan Srebro. Kernel and rich  
regimes in overparametrized models, 2020.
- 1000  
1001 Carole-Jean Wu, Ramya Raghavendra, Udit Gupta,  
1002 Bilge Acun, Newsha Ardalani, Kiwan Maeng, Glo-  
1003 ria Chang, Fiona Aga, Jinshi Huang, Charles Bai,  
1004 et al. Sustainable ai: Environmental implications,  
1005 challenges and opportunities. *Proceedings of Ma-  
chine Learning and Systems*, 4:795–813, 2022.
- 1006  
1007 Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu,  
1008 Yue Wang, Xiaohan Chen, Richard G. Baraniuk,  
1009 Zhangyang Wang, and Yingyan Lin. Drawing early-  
1010 bird tickets: Toward more efficient training of deep  
1011 networks. In *International Conference on Learning  
Representations*, 2020.
- 1012  
1013 Peng Zhao, Yun Yang, and Qiao-Chu He. High-  
1014 dimensional linear regression via implicit regular-  
1015 ization. *Biometrika*, 109(4):1033–1046, Febru-  
1016 ary 2022. ISSN 1464-3510. doi: 10.1093/  
1017 biomet/asac010. URL [http://dx.doi.org/  
10.1093/biomet/asac010](http://dx.doi.org/10.1093/biomet/asac010).
- 1018  
1019 Xiao Zhou, Weizhong Zhang, Zonghao Chen, Shizhe  
1020 Diao, and Tong Zhang. Efficient neural network  
1021 training via forward and backward propagation spar-  
1022 sification. *Advances in Neural Information Process-  
1023 ing Systems*, 34:15216–15229, 2021a.
- 1024  
1025 Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong  
Zhang. Effective sparsification of neural networks  
with global sparsity constraint. In *Proceedings of*

1026 *the IEEE/CVF Conference on Computer Vision and*  
 1027 *Pattern Recognition*, pp. 3599–3608, 2021b.

1028  
 1029 Liu Ziyin. Symmetry induces structure  
 1030 and constraint of learning. 2023. URL  
 1031 [https://api.semanticscholar.org/](https://api.semanticscholar.org/CorpusID:263310669)  
 1032 [CorpusID:263310669](https://api.semanticscholar.org/CorpusID:263310669).

1033 Liu Ziyin and Zihao Wang. spred: Solving  $l_1$  penalty  
 1034 with sgd, 2023. URL [https://arxiv.org/](https://arxiv.org/abs/2210.01212)  
 1035 [abs/2210.01212](https://arxiv.org/abs/2210.01212).

## 1036 1037 1038 A MIRROR FLOW FRAMEWORK

1039  
 1040 In this section we present some known results from  
 1041 the mirror flow framework for completeness. We de-  
 1042 rive the Bregman potential associated with  $m \odot w$  in  
 1043 Theorem A.1. Next, we will also show the conver-  
 1044 gence of the loss and provide optimality guarantees  
 1045 in Theorem A.2. In addition, we extend Theorem  
 1046 A.2 with Theorem A.3. Finally, we give the opti-  
 1047 mality result for diagonal linear networks in Theo-  
 1048 rem A.4

1049 **Theorem A.1** *Let the initialization of  $m$  and  $w$  sat-*  
 1050 *isfy  $m_{0,i} > |w_{0,i}|$  for all  $i \in [n]$ . Then the corre-*  
 1051 *sponding mirror function is:*

$$1052  
1053 R(x) := \frac{1}{4} \sum_{i=1}^n x_i \operatorname{arcsinh} \left( \frac{x_i}{2u_{0,i}v_{0,i}} \right) - \sqrt{x_i^2 + 4u_{0,i}^2v_{0,i}^2} - x_i \log \left( \frac{u_{0,i}}{v_{0,i}} \right) \quad (8)$$

1054 where  $u_{0,i} = \frac{m_{0,i} + w_{0,i}}{\sqrt{2}}$  and  $v_{0,i} = \frac{m_{0,i} - w_{0,i}}{\sqrt{2}}$ . Fur-  
 1055 thermore,  $R$  is a Bregman function.

1056  
 1057 Proof. The result follows directly from applying  
 1058 Theorem 4.16 in (Li et al., 2022).

1059 Theorem A.1 implies the following: a) The global  
 1060 minima of  $R$  is at the initialization  $x_0 = m_0 \odot w_0$ .  
 1061 b) The Lipschitz coefficient of  $R$  depends on the ini-  
 1062 tialization. The Lipschitz coefficient  $L_R$  of (8) is  
 1063  $L_R = \frac{1}{\min_i 2u_{0,i}v_{0,i}}$ , determining the smoothness of  
 1064 the potential. Following these two observations we  
 1065 make the following remark about Theorem A.1.

1066  
 1067 **Remark A.1** *Note that when the initialization is*  
 1068 *zero, i.e.,  $w_0 = 0, m_0 = \sqrt{a}$  with  $a \geq 0$  then (8)*  
 1069 *is the hyperbolic entropy. The hyperbolic entropy is*

$$1070  
1071 \sum_{i=1}^n x_i \operatorname{arcsinh} \left( \frac{x_i}{a} \right) - \sqrt{x_i^2 + a^2}$$

1072  
 1073 Theorem 2 of (Woodworth et al., 2020) character-  
 1074 izes the behavior in the limit for this case. For the  
 1075 hyperbolic entropy in case  $a \rightarrow 0$  and  $|\frac{x}{a}| \rightarrow \infty$ ,

$$1076  
1077 R(x) \sim \log \left( \frac{1}{a} \right) \|x\|_{L_1}.$$

1080 This means an  $L_1$  bias is induced when  $a$  is small.  
 1081 Nevertheless, we need an exponentially small  $a$  compared to  $x$  to get there as shown in (Woodworth  
 1082 et al., 2020), which can lead to numerical problems.  
 1083 Furthermore,  $m_0 = w_0 = 0$  is a saddle point which  
 1084 can slow down training (exponentially) (Du et al.,  
 1085 2017). Additionally, the asymptotic result only holds for initializing at zero. Note  $L_R = a$  in this  
 1086 case.  
 1087

1088 Remark A.1 shows the potential of using the implicit  
 1089 bias to induce sparsity. To actualize this, we need to  
 1090 solve the two challenges posed in the remark. Both  
 1091 are remedied in Section 2.

1092 In addition to this promising formulation of implicitly  
 1093 minimizing an  $L_1$  norm with the use of the  
 1094 mirror framework, we can get convergence results.  
 1095 These results make it clear why implicit regularization  
 1096 is preferable over explicit regularization. The  
 1097 convergence result from (Li et al., 2022) is stated for  
 1098 our setting. Furthermore, the theorem is extended  
 1099 for a specific class of Bregman functions.

1100 **Theorem A.2** (Theorem 4.14 (Li et al., 2022)) Assume that  $f$  is quasi-convex,  $\nabla f$  is locally Lipschitz  
 1101 and  $\operatorname{argmin}\{f(x)|x \in \mathbb{R}^n\}$  is non-empty. Then as  
 1102  $t \rightarrow \infty$ ,  $x_t$  converges to some critical point  $x^*$ .  
 1103 Moreover, if  $f$  is convex  $x_t$  converges to a minimizer  
 1104 of  $f$ .  
 1105  
 1106

1107 In Theorem A.2 it is shown that with implicit regularization an optimal solution to the original optimization problem can be reached. In contrast, explicit regularization makes this not possible, by definition. Because the optimization problem has fundamentally changed. Showing the benefit of implicit over explicit.

1114 For the extension, the convexity constraint is replaced by the Polyak-Łojasiewicz (PL) inequality in the theorem. The PL-inequality is a more realistic constraint in a machine learning context as loss functions are not locally convex but can satisfy the PL inequality locally (Wojtowysch, 2021; Dereich & Kassing, 2024). The PL-inequality for a continuously differentiable function  $f$  is

$$1121 \|\nabla f(x)\|_{L_2}^2 \geq \lambda(f(x) - f(x^*)) \quad \forall x \in \mathbb{R}^n \quad (9)$$

1123 for some  $\lambda > 0$  and global minima  $x^*$  of  $f$ . This allows us to state the modified theorem.

1125 **Theorem A.3** Consider the same setting as Theorem A.2. Assume  $R$  satisfies for all  $x \in \mathbb{R}^n$ ,

$$1128 z^T (\nabla^2 R(x))^{-1} z \geq \sigma \|z\|_{L_2}^2 \quad \forall z \in \mathbb{R}^n. \quad (10)$$

1129 Furthermore, assume  $f$  satisfies the PL-inequality (9). Then  $x_t$  converges to a minimizer of  $f$ . Furthermore, the loss converges linearly with rate  $\sigma \lambda$ .

1132 Proof. The evolution of  $f(x_t) - f(x^*)$  is described  
 1133 by  $df(x_t) = -\nabla f(x_t)^\top (\nabla^2 R(x_t))^{-1} \nabla f(x_t) dt$ .

1134 From (10) and (9) the evolution is bounded by  
 1135  $df(x_t) \leq -\sigma \|\nabla f(x_t)\|_{L_2}^2 dt \leq -\sigma \lambda (f(x_t) - f(x^*)) dt.$   
 1136

1137 Applying Gronwall’s Lemma concludes the proof.  
 1138  $\square$

1139 Note that Theorem A.3 holds in the same (general)  
 1140 setting as Theorem A.2. Also, note that the PL-  
 1141 inequality together with quasi-convexity does not  
 1142 imply convexity. Theorem A.3 holds for our setting.  
 1143 In this case, it follows from a direct computation that

$$1144 (\nabla^2 R(x))^{-1} = \text{diag} \left( \sqrt{x^2 + 4u_{0,1}^2 v_{0,1}^2}, \dots, \sqrt{x^2 + 4u_{0,n}^2 v_{0,n}^2} \right).  
 1145$$

1146 This implies that  $\eta = 2 \min_i u_{0,i} v_{0,i}$  in Theorem  
 1147 A.3, which again highlights the importance of the  
 1148 initialization.

1149 Finally, in the case of under-determined linear re-  
 1150 gression, we can derive optimality conditions in the  
 1151 form of KKT conditions of  $R$ . Consider a data  
 1152 set  $(z_j, y_j)_{j=1}^d$  with  $z_j \in \mathbb{R}^n$  and  $y_j \in \mathbb{R}$ . Let  
 1153  $Z = (z_1, \dots, z_d)$  and  $Y = (y_1, \dots, y_d)$ . For the re-  
 1154 gression to be called underdetermined  $n > d$ .

1155 **Theorem A.4** (Theorem 4.17 (Li et al., 2022)) *In*  
 1156 *case of under-determined regression consider the*  
 1157 *loss function  $f(x) = \tilde{f}(Zx - Y)$ . Assume  $f$  satisfies*  
 1158 *the conditions of Theorem A.2. Then  $x_t$  converges to*  
 1159  *$x^*$  such that*

$$1160 x^* = \text{argmin}_{Zx=Y} R(x)$$

1162 Note that Theorem 4.17 of (Li et al., 2022) only uses  
 1163 quasi-convexity of the loss. Theorem A.4 guarantees  
 1164 that the optimization problem is solved while implicit-  
 1165 ly minimizing the potential  $R$ . Thus choosing the  
 1166 sparsest model out of the models that predict the data  
 1167 perfectly. This highlights another benefit of implicit  
 1168 regularization over explicit regularization.

1169 In this section, we have shown the viability of us-  
 1170 ing the implicit bias framework to induce an im-  
 1171 plicit regularization. Furthermore, we have given  
 1172 two known benefits of using the implicit bias frame-  
 1173 work over explicit regularization. The benefits are  
 1174 convergence to the optimal solution of the original  
 1175 problem and optimality in the case of underdeter-  
 1176 mined regression. To add to this, we have extended  
 1177 the convergence theorem using the PL-inequality in  
 1178 9. Moreover, we again highlight the importance of  
 1179 the initialization of  $m_0$  and  $w_0$  with the influence on  
 1180 the smoothness of the Bregman potential and con-  
 1181 vergence of the loss. The initialization insight as in  
 1182 the main text is used to improve upon spred (Ziyin,  
 1183 2023) as their initialization has scaling  $2u_0 v_0 = 0$ , it  
 1184 follows already from the mirror flow framework that  
 1185  $2u_0 v_0 = 1$  is a better initialization. Furthermore,  
 1186 we also do not initialize at zero though, and scaling  
 1187 needs to be exponentially small to get a good approximation of the  $L_1$  norm potentially making  
 it hard to escape the saddle point. Therefore, the explicit regularization analyzed in Section 2 is  
 necessary to exploit the implicit bias framework.

## B PROOF MAIN RESULT

We show the main result here. The proof consists of four parts

- $R_{a_t}$  satisfies a mirror flow (Lemmas B.1 and B.2)
- Boundedness of the iterates and convergence to a critical point (Lemma B.3)
- Convergence of the loss (Theorem B.1)
- Optimality in case of underdetermined linear regression (Theorem B.2)

Consider the following gradient flow

$$\begin{cases} dm_t = -\nabla f(m_t \odot w_t) \odot w_t - 2\alpha_t m_t dt \\ dw_t = -\nabla f(m_t \odot w_t) \odot m_t - 2\alpha_t w_t dt \end{cases} \quad (11)$$

For the flow in (11) to be well-posed  $\nabla f$  needs to be locally Lipschitz continuous. This is a sufficient condition given that  $\alpha_t$  is "nice", which will be made more rigorous later. The evolution of  $x_t = m_t \odot w_t$  is derived in Lemma B.1.

**Lemma B.1** *The evolution of  $x_t = m_t \odot w_t$  with 11 is described by*

$$x_t = u_0^2 \odot \exp\left(-2 \int_0^t \nabla f(x_s) ds - 4 \int_0^t \alpha_s ds\right) - v_0^2 \odot \exp\left(2 \int_0^t \nabla f(x_s) ds - 4 \int_0^t \alpha_s ds\right),$$

where  $u_0 = \frac{m_0 + w_0}{\sqrt{2}}$  and  $v_0 = \frac{m_0 - w_0}{\sqrt{2}}$ .

*Proof.* This follows from deriving the flow of  $m_t$  and  $w_t$  and then combining the two. The evolution of both are given by

$$\begin{cases} m_t = \left(m_0 \odot \cosh\left(-\int_0^t \nabla f(x_s) ds\right) + w_0 \odot \sinh\left(-\int_0^t \nabla f(x_s) ds\right)\right) \exp\left(-2 \int_0^t \alpha_s ds\right) \\ w_t = \left(w_0 \odot \cosh\left(-\int_0^t \nabla f(x_s) ds\right) + m_0 \odot \sinh\left(-\int_0^t \nabla f(x_s) ds\right)\right) \exp\left(-2 \int_0^t \alpha_s ds\right). \end{cases}$$

For ease of notation set  $L_t = \int_0^t \nabla f(x_s) ds$  and  $A_t = \int_0^t \alpha_s ds$ . Combining gives us

$$\begin{aligned} x_t &= m_t \odot w_t \\ &= (m_0^2 + w_0^2) \odot \cosh(-L_t) \odot \sinh(-L_t) \exp(-4A_t) \\ &\quad + w_0 \odot m_0 \odot \left(\cosh(-L_t)^2 + \sinh(-L_t)^2\right) \exp(-4A_t) \\ &= \left(\frac{m_0^2 + w_0^2}{2} \odot \sinh(-2L_t)\right) \exp(-4A_t) \\ &\quad + w_0 \odot m_0 \odot (\cosh(-2L_t)) \exp(-4A_t) \\ &= u_0^2 \odot \exp(-2L_t - 4A_t) - v_0^2 \odot \exp(2L_t - 4A_t) \end{aligned}$$

where the second equality follows from hyperbolic identities.  $\square$

It follows from Lemma B.1 and the local Lipschitz condition on  $\nabla f$  and  $\int_0^t \alpha_s ds < \infty$  for all  $t \geq 0$ , that the flow is well-posed. We now define the

(corrected)-hyperbolic entropy function. The corrected hyperbolic entropy is given by

$$R_a(x) = \frac{1}{2} \sum_{i=1}^n x_i \operatorname{arcsinh} \left( \frac{x_i}{a} \right) - \sqrt{x_i^2 + a^2} - x_i \log \frac{u_{0i}}{v_{0,i}},$$

where the last term is the correction. The correction stems from not initializing at zero.

**Lemma B.2** *Let  $|w_{i0}| \leq m_{0i}$  for all  $i \in [n]$ , then  $R_{a_t}(x_t)$  with  $a_t = 2u_0 \odot v_0 \exp \left( -2 \int_0^t \alpha_s ds \right)$  satisfies*

$$d\nabla R_{a_t}(x_t) = -\nabla f(x_t) dt \quad x_0 = m_0 \odot w_0. \quad (12)$$

*Proof.* This follows from Lemma B.1,

$$\begin{aligned} x_t \exp \left( 4 \int_0^t \alpha_s ds \right) &= u_0^2 \exp \left( -2 \int_0^t \nabla f(x_s) ds \right) - v_0^2 \exp \left( 2 \int_0^t \nabla f(x_s) ds \right) \Leftrightarrow \\ &\frac{1}{2} \left( \operatorname{arcsinh} \left( \frac{x_t}{a_t} \right) - \log \left( \frac{u_0}{v_0} \right) \right) = - \int_0^t \nabla f(x_s) ds. \end{aligned}$$

This equivalence follows from setting  $z = \exp \left( -2 \int_0^t \nabla f(x_s) ds \right)$  and solving the resulting quadratic equation. Notice that the left hand side in (12) is  $\nabla R_{a_t}(x_t)$ .  $\square$

**Lemma B.3** *Let  $f$  be a quasi-convex function and  $\alpha_t \geq 0$  for all  $t \geq 0$ . Furthermore, assume the integral  $\int_0^t \alpha_s ds < \infty$ . Then the iterates are bounded and converge to a critical point.*

Consider the time-dependent Bregman divergence

$$D_{a_t}(x^*, x_t) := R_{a_t}(x^*) - R_{a_t}(x_t) - \nabla_x R_{a_t}^T(x^* - x_t) \geq 0$$

The divergence is bounded by:

$$D_{a_t}(x^*, x_t) \leq R_{a_\infty}(x^*) - R_{a_t}(x_t) - \nabla_x R_{a_t}^T(x^* - x_t) =: W_t,$$

this follows from the fact that the map  $a \rightarrow R_a$  is decreasing. We make the following two observations:

$$\frac{d}{da} R_a(x) = -\frac{1}{2} \sum_{i=1}^n \frac{x_i^2 |a| + a^3}{a^2 \sqrt{a^2 + x_i^2}} \leq 0 \quad \text{and} \quad \frac{da_t}{dt} \leq 0 \quad \forall t \geq 0. \quad (13)$$

This allows us to bound the evolution

$$\begin{aligned} \frac{d}{dt} W_t &= \frac{d}{dt} \left( -R_{a_t}(x_t) - \nabla_x R_{a_t}^T(x^* - x_t) \right) \\ &= -\frac{d}{da} R_{a_t}(x_t) \frac{d}{dt} a_t - \frac{d}{dt} \left( \nabla_x R_{a_t}^T \right) (x^* - x_t) \\ &\leq \nabla_x f(x_t)^T (x^* - x_t) \\ &\leq 0, \end{aligned}$$

where the observations in (13) are used in the first inequality.

From Theorem 4.16 in (Li et al., 2022) it follows that for all  $a > 0$ ,  $R_a$  is a Bregman potential. Implying that for all  $a > 0$  the level set for  $\gamma \in \mathbb{R}$ ,

$$\{x \in \mathbb{R}^n : D_a(x^*, x) \leq \gamma\}$$

is bounded. Combining this with the fact that the evolution is bounded, implies the iterates are bounded.

In the next part, it is shown that  $x_t$  converges to a critical point. We first show that the loss becomes eventually non-increasing. There is a  $T$  such that for all  $t \geq T$  the loss  $f$  is non-increasing.,

$$\begin{aligned} df(x_t) &= - \left( \nabla f(x_t)^T \text{diag} \left( \sqrt{x_t^2 + a_t^2} \right) \nabla f(x_t) + 2\alpha_t \nabla f(x_t)^T x_t \right) dt \\ &\leq \left( -\nabla f(x_t)^T \text{diag} \left( \sqrt{x_t^2 + a_t^2} \right) \nabla f(x_t) + 2\alpha_t C \right) dt, \end{aligned}$$

where it is used that the iterates are bounded and  $\nabla f$  is locally Lipschitz. As  $t \rightarrow \infty$  we have that  $\alpha_t \rightarrow 0$  and  $a_t \rightarrow a_\infty > 0$  by assumption. Hence there exists a  $T$  such that for all  $t \geq T$  we have

$$df(x_t) \leq 0.$$

Note if this is not the case then there is a  $T > 0$  such that  $\nabla f(x_T) = 0$  implying convergence (to a critical point).

Now let  $x_\infty$  be an accumulation point of the bounded flow  $x_t$ . We use this to show convergence to a critical point. We have that for all  $x \in \mathbb{R}^n$ ,

$$\nabla f(x_\infty)^T x = \lim_{t \rightarrow \infty} \frac{1}{t} \left( \int_T^{T+t} \nabla f(x_s) ds \right)^T x = \lim_{t \rightarrow \infty} \frac{1}{t} (R_{a_T}(x_T) - R_{a_{T+t}}(x_{T+t}))^T x = 0,$$

where the first equality follows from that the loss is non-increasing and the second one from the time-dependent mirror flow description. Finally, because  $x_t$  converges to an accumulation point we also have  $\lim_{t \rightarrow \infty} R_{a_t}(x_t) = R_{a_\infty}(x_\infty)$  by continuity, giving the last equality.

We use that the accumulation point is a critical point and set  $x^* = x_\infty$  in  $W_t$  such that  $W_t \rightarrow 0$ . This implies  $D_{a_t}(x_\infty, x_t) \rightarrow 0$  by the upperbound. It follows from the fact that the iterates are bounded that  $R_{a_t}$  is  $\mu$ -strongly convex on this bounded convex set where the iterates stay. This gives

$$\|x_\infty - x_t\|_{L_2} \leq \frac{\mu}{2} D_{a_t}(x_\infty, x_t) \rightarrow 0,$$

showing  $x_t$  converges to a critical point.  $\square$

Lemma B.3 gives a condition such that the iterates are bounded and converge to a critical point. It remains to be shown that the loss converges. This is done in Theorem B.1.

**Theorem B.1** *Consider the same setting as Lemma B.3, if  $f$  is convex or satisfies the PL-inequality we have convergence to an interpolator  $x^*$  such that it is a minimizer of  $f$ . Furthermore, in the PL-inequality case, the loss converges linearly.*

1350 Proof. Assume  $f$  is convex, notice first that there is  
 1351 a  $T$  such that for all  $t \geq T$  the loss is non-increasing.  
 1352 Combining this with a bound on the time-dependent  
 1353 Bregman potential gives us convergence of the loss.  
 1354 The time-dependent Bregman divergence is again  
 1355 defined by

$$1356 D_{a_t}(x^*, x_t) = R_{a_t}(x^*) - R_{a_t}(x_t) - \nabla_x R_{a_t}^\top(x^* - x_t) \geq 0.$$

1357 The divergence is bounded by:

$$1358 D_{a_t}(x^*, x_t) \leq W_t.$$

1359 The evolution of the bound is

$$\begin{aligned} 1360 \frac{d}{dt} W_t &= \frac{d}{dt} (-R_{a_t}(x_t) - \nabla_x R_{a_t}^\top(x^* - x_t)) \\ 1361 &= -\frac{d}{da} R_{a_t}(x_t) \frac{d}{dt} a_t - \frac{d}{dt} (\nabla_x R_{a_t}^\top)(x^* - x_t) \\ 1362 &\leq \nabla_x f(x_t)^\top (x^* - x_t) \\ 1363 &\leq f(x^*) - f(x_t), \end{aligned}$$

1364 where again the observations in (13) are used in the  
 1365 first inequality. Therefore the loss converges:

$$\begin{aligned} 1366 f(x_{T+t}) - f(x^*) &\leq \frac{1}{t} \int_T^{T+t} f(x_s) - f(x^*) ds \\ 1367 &\leq \frac{W_T - W_{T+t}}{t} \\ 1368 &\leq \frac{W_T}{t} \rightarrow 0 \end{aligned}$$

1369 where the first inequality follows from convexity of the loss and the third inequality from the fact  
 1370 that  $W_t \geq D_{a_t}(x^*, x_t) \geq 0$ . So the loss converges. We already know from Lemma B.3 that the  
 1371 iterates converge, concluding the convex case.

1372 In case when  $f$  satisfies the PL-inequality, we pro-  
 1373 ceed in the same way as Theorem A.3. The evolution  
 1374 of  $f$  is given by

$$\begin{aligned} 1375 df(x_t) &= - \left( \nabla f(x_t)^\top \text{diag} \left( \sqrt{x_t^2 + a_t^2} \right) \nabla f(x_t) + 2\alpha_t \nabla f(x_t)^\top x_t \right) dt \\ 1376 &\leq (-a_\infty \lambda (f(x_t) - f(x^*)) + \alpha_t C \|x^*\|_{L_2}) dt \end{aligned}$$

1377 where  $C$  is constant depending on the smoothness of  
 1378  $f$ . Then it follows from Gronwall's Lemma that

$$1379 f(x_t) - f(x^*) \leq (f(x_0) - f(x^*)) \exp \left( -a_\infty \lambda t + \int_0^t \alpha_s C \|x^*\|_{L_2} ds \right).$$

1380 It follows from the fact that  $\int_0^t \alpha_s ds < \infty$  for all  $t \geq$   
 1381 0 that the loss  $f$  convergence. Convergence of the  
 1382 iterates now follows in a similar way as the convex  
 1383 case.  $\square$

1384 We now show optimality in the case of under-  
 1385 determined linear regression Consider a data set  
 1386  $(z_j, y_j)_{j=1}^d$  with  $z_j \in \mathbb{R}^n$  and  $y_j \in \mathbb{R}$ . Let  $Z =$   
 1387  $(z_1, \dots, z_d)$  and  $Y = (y_1, \dots, y_d)$ . For the regression  
 1388 to be called underdetermined  $n > d$ .

1389 **Theorem B.2** *In case of under-determined regres-*  
 1390 *sion consider the loss function  $f(x) = \hat{f}(Zx - Y)$ .*

1404 Assume  $f$  satisfies the conditions with at least one  
 1405 of the convergence criteria of Theorem B.1. Then  $x_t$   
 1406 converges to  $x^*$  such that

$$1407 \quad x^* = \operatorname{argmin}_{Zx=Y} R_{a_\infty}(x) \quad (14)$$

1408  
 1409 Proof. Convergence follows from Theorem B.1. It  
 1410 remains to be shown that the optimality conditions of  
 1411 (14) are satisfied. The gradient flow of  $R_{a_t}$  satisfies

$$1412 \quad \nabla R_{a_t}(x_t) = Z^\top \int_0^t \nabla \tilde{f}(x_s) ds \in \operatorname{span}\{Z^\top\}.$$

1413 This quantity is well defined for all  $t \geq 0$  because  
 1414  $\nabla \tilde{f}$  is locally Lipschitz (because  $f$  has to be locally  
 1415 Lipschitz). Therefore taking the  $t \rightarrow \infty$  yields the  
 1416 KKT conditions of the optimization problem in (14).  
 1417  $\square$   
 1418  
 1419

## 1420 B.1 DISCUSSION OF THE PROOF

1421 Most of the proof follows the same arguments as in  
 1422 (Alvarez et al., 2004; Li et al., 2022; Pesme et al.,  
 1423 2021). The notable differences are showing that the  
 1424 loss becomes decreasing over time and the observa-  
 1425 tions made in (13).  
 1426

## 1427 C DETAILS EXPERIMENTS

1428 In this section, we provide the details of the experi-  
 1429 ments. In addition, there are additional figures given.

1430 **Compute** The codebase for the experiments is  
 1431 written in PyTorch and torchvision and their relevant  
 1432 primitives for model construction and data-related  
 1433 operations. The experiments in the paper are trained  
 1434 on an NVIDIA A6000. In addition, the diagonal linear  
 1435 network is trained on a CPU 13th Gen INTEL(R)  
 1436 Core(TM) i9-13900H.  
 1437  
 1438  
 1439

### 1440 C.1 DIAGONAL LINEAR NETWORK

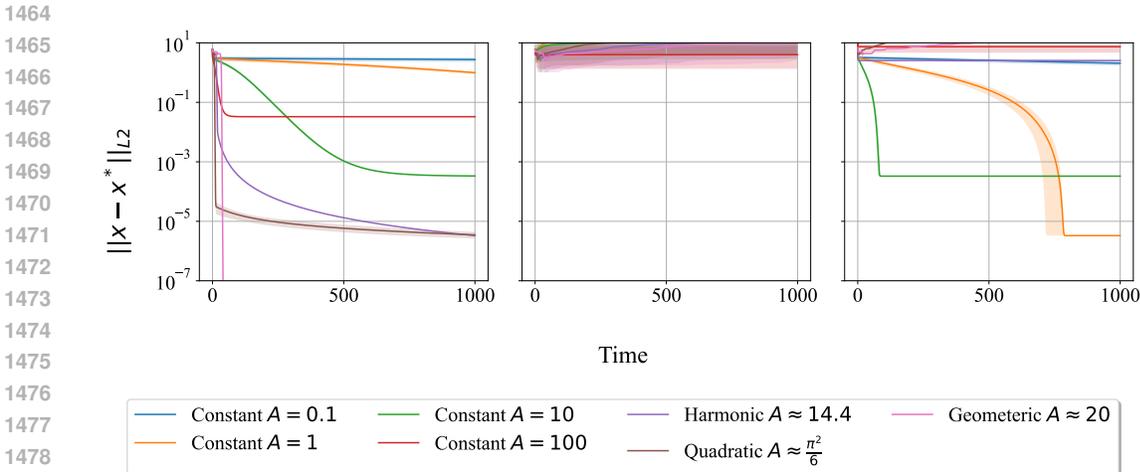
1441 For each setting, different regularization schemes are  
 1442 tried. In total, 7 options are tried. 4 of the sched-  
 1443 ules are constant i.e. the regularization stays the  
 1444 same during training. The remaining 3 are decay-  
 1445 ing schedules. These schedules we name harmonic,  
 1446 quadratic, and geometric. The schedules are de-  
 1447 scribed by the following recurrent relations:  
 1448

$$1449 \quad h_k = \frac{1}{k}, \quad q_k = \frac{1}{k^2} \quad \text{and} \quad g_k = p^k$$

1450 where we have set  $p = 0.95$ . These schedules lead to  
 1451 a total strength of regularization applied. We denote  
 1452  $A := \int_0^t \alpha_s ds$  the total strength of the regulariza-  
 1453 tion. So in practice, it is the weighted sum of the  
 1454 regularization strength.  
 1455

1456 In Figure 5 we present the trajectories of all the  
 1457 schedules for the 3 considered settings. We observe

1458 that our regularization performs the best with the de-  
 1459 caying schedules as predicted by the theory. The  
 1460 other methods need constant regularization to per-  
 1461 form well as already mentioned in Remark 2.1. Note  
 1462 that, PILoT also can perform well with constant reg-  
 1463 ularization (see Figure 5).



1474  
1475  
1476  
1477  
1478  
1479  
1480 Figure 5: All runs for the diagonal linear network. From left to right  $m \odot w$  with PILoT initialization,  
 1481  $m \odot w$  with spread initialization, and  $x$  with  $L_1$  regularization

1482  
1483  
1484 C.2 ONE-SHOT

1485 In this section we give the additional details for the  
 1486 one-shot experiments. In Table 2 the hyperparam-  
 1487 eters for the CIFAR 10 and 100 experiments are  
 1488 given. To determine which configuration is best for  
 1489 which sparsity level we compute the validation accu-  
 1490 racy at multiple levels and choose the level just  
 1491 before the accuracy drops 1% or in the high-sparsity  
 1492 regime 2%. Moreover, we use  $s = -200$  for STR.

1493 For the ImageNet experiment we use the setup of  
 1494 (Kusupati et al., 2020). For PILoT and spread we  
 1495 in addition use  $L_2$  regularization to compensate for  
 1496 the weight decay i.e. we add a term  $(m \odot w)^2$   
 1497 with strength  $0.000030517578125/2$ , which is based on  
 1498 the weight decay strength in (Kusupati et al., 2020)  
 1499 for the baseline. Furthermore, weight decay is  
 1500 turned off for the other parameters. In Table 3 we  
 1501 present the configurations that correspond to the val-  
 1502 ues in Table 1. Note for all PILoT configs  $\delta = 1.01$   
 1503 is used.

1504 **Label smoothing** Although PILoT is competitive in  
 1505 the 80% – 90% sparsity range it is not SOTA. Nev-  
 1506 ertheless, if we turn of label smoothing in the exper-  
 1507 iment PILoT outperforms STR in this range as well.  
 1508 The only change for STR is turning of labelsmooth-  
 1509 ing. For PILoT we use two different configurations.

1510 <sup>2</sup>Applied to the other parameters  
 1511 <sup>3</sup>Starting from a pretrained model with 77% validation accuracy

Table 2: One-shot experiment

Parameter	Setting	Comments
Optimizer	SGD	
Momentum	0.9	
Batch size	256	
Activation function	ReLU	
Weight decay <sup>2</sup>	$10^{-4}$	
Base learning rate	{0.1, 0.2}	
Epochs	150	
Warmup period	0	
Initialization	Kaiming normal	
Scaling	1	Only for $m \odot w$
$\delta$	1.01	
$K$	8000	
CIFAR 10		
Learning rate schedule	cosine warmup	
CIFAR 100		
Learning rate schedule	step warmup	

Table 3: ResNet-50 on ImageNet configurations for each sparsity (%).

Method	$\alpha_{init}$	$K$	sparsity	Method	$\alpha_{init}$	$K$	sparsity
spred <sup>3</sup>	$2e-5$	-	80.00	spred	$2e-5$	-	94.50
spred	$3e-6$	-	79.03	PILoT	$2e-5$	60000	94.00
PILoT	$7e-6$	60000	80.00	PILoT	$3e-5$	60000	95.00
spred	$5e-6$	-	89.26	PILoT	$3e-5$	60000	96.00
PILoT	$1e-5$	60000	88.00	spred	$3e-5$	-	97.19
PILoT	$1.4e-5$	60000	91.00	PILoT	$4e-5$	40000	97.19
				spred	$5e-5$	-	98.20
				PILoT	$5e-5$	20000	97.75
				PILoT	$7e-5$	20000	98.20

We use  $L_2$  regularization i.e.  $(m \odot w)^2 - \|m \odot w\|_{L_2}^2$  regularization set to  $5 \cdot 10^{-5}$  instead of the value from STR experiment. We use  $K = 600000$  and  $\delta = 1.01$ . Furthermore, the strength of the PILoT regularization is initialized at  $\{1 \cdot 10^{-5}, 2 \cdot 10^{-5}\}$  and no weight decay is used on the rest of the parameters. The results are given in Table C.2.

Table 4: Extra experiment ResNet-50 on ImageNet sparsity (%) versus accuracy (%) without label smoothing.

Method	Top-1 Acc	Sparsity
ResNet-50	75.80	0
STR	73.03	<b>79.03</b>
PILoT	<b>74.72</b>	<b>79.03</b>
STR	71.6	89.26
PILoT	<b>73.21</b>	<b>91.41</b>

1566 C.3 ITERATIVE PRUNING  
1567

1568 In Table 5 the details of the ImageNet the experi-  
1569 ment are given. Note the base learning rate 0.1 is  
1570 for the baseline and 0.2 is used for our parameteri-  
1571 zation combined with scaling 1. In addition, the  $L_2$   
1572 regularization denotes the reparameterization of the  
1573 original weight decay. Thus for PILOT, in this case,  
1574 we use that instead. Moreover, all runs have been  
1575 done for 3 different seeds. Furthermore, we provide  
1576 additional experiments on CIFAR 10 and 100 with  
1577 ResNet-20 and ResNet-18 respectively in Figure 6.  
1578 The details are given in Table 6  
1579

Table 5: WR and LRR experiment on ImageNet

Parameter	Setting	Comments
Optimizer	SGD	
Momentum	0.9	
Batch size	512	
Activation function	ReLU	
Weight decay	$\{0, 10^{-4}\}$	
Learning rate schedule	step warmup	
Base learning rate	$\{0.1, 0.2\}$	
Cycles	25	
Pruning rate	0.8	
Epochs per cycle	90	
Warmup period	10	
Initialization	Kaiming normal	
$L_2$ regularization	$5 \cdot 10^{-5}$	Only for $m \odot w$
PILOT regularization	$\{0\}$	Only for $m \odot w$
Scaling	1	Only for $m \odot w$
$\delta$	1	Only for $m \odot w$
$K$	—	Only for $m \odot w$

Table 6: WR and LRR experiment on CIFAR 10 and 100

Parameter	Setting	Comments
Optimizer	SGD	
Momentum	0.9	
Batch size	256	
Activation function	ReLU	
Weight decay	$10^{-4}$	
Learning rate schedule	step warmup	
Base learning rate	$\{0.1, 0.2\}$	
Cycles	25	
Pruning rate	0.8	
Epochs per cycle	150	
Warmup period	50	
Initialization	Kaiming normal	
$L_2$ regularization	0	Only for $m \odot w$
PILOT regularization	$\{10^{-4}\}$	Only for $m \odot w$
Scaling	1	Only for $m \odot w$
$\delta$	1	Only for $m \odot w$
$K$	—	Only for $m \odot w$

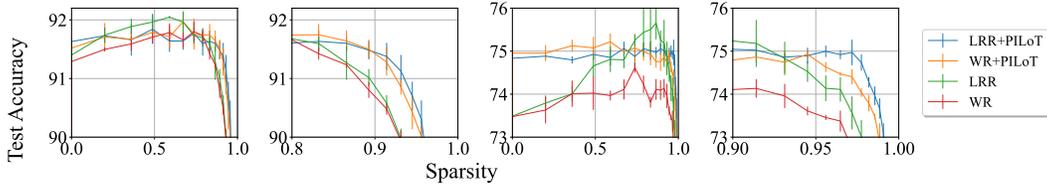


Figure 6: Learning Rate Rewinding (LRR) and Weight Rewinding (WR) with PILoT shows improvement over the baseline iterative pruning methods for CIFAR 10 and 100.

## D REMARK ON NEURONWISE PRUNING

In the main text, we have used mirror flow to describe the implicit bias of parameterwise pruning. In this section, we show that neuronwise pruning can not be analyzed in the same way. We first define neuronwise pruning. Next, we paraphrase the necessary condition such that the implicit bias can be described by a mirror flow from (Li et al., 2022). Finally, we show neuronwise pruning violates this condition pointing out a limitation of the framework.

Consider the parameterization for a function with  $p$  neurons,  $g : \mathbb{R}^p \times \mathbb{R}^{n_1} \times \dots \times \mathbb{R}^{n_p}$ ,

$$g(m, w_1, \dots, w_p) = (m_1 w_1, \dots, m_p w_p)$$

where  $m_i \in \mathbb{R}$  is the mask and  $w_i \in \mathbb{R}^{n_i}$  are the neurons.

We state the necessary condition for a parameterization to induce a mirror flow.

**Theorem D.1** (Theorem 4.10 (Li et al., 2022)) *The Lie bracket span of  $\{\nabla_i g\}_{i=1}^n$  is in the kernel of Jacobian  $\partial g$ .*

Now, we use this theorem to show that neuronwise pruning does not induce a mirror flow.

**Lemma D.1** *Neuronwise pruning violates Theorem D.1.*

*Proof.* We show that for  $p = 1$  the condition is already violated. This implies that in the general case, the condition is also violated as the neurons themselves are commuting with each other as they are parameterized separate.

To see this we can explicitly check the commuting condition for the following parameterization  $g : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$

$$g(m, w) = mw$$

Then the gradients (Jacobians) and Hessian's are given by:

$$\nabla g_i = \begin{pmatrix} w_i \\ m \mathbb{I}_{i=1} \\ \vdots \\ m \mathbb{I}_{i=n} \end{pmatrix} \quad \text{and} \quad H g_i = \begin{pmatrix} 0 & \mathbb{I}_{i=1} & \dots & \mathbb{I}_{i=n} \\ \mathbb{I}_{i=1} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{I}_{i=n} & 0 & \dots & 0 \end{pmatrix} \quad \text{for } i = 1, 2$$

1674 Computing  $Hg_i \nabla g_j$   
 1675

$$1676 \quad Hg_i \nabla g_j = w_j \begin{pmatrix} 0 \\ \mathbb{I}_{i=1} \\ \vdots \\ \mathbb{I}_{i=n} \end{pmatrix}$$

1680 we compute the Lie brackets which span a subspace  
 1681 of the Lie Algebra  $LIE^{\geq 2}(\partial g)$ . The subspace is  
 1682 spanned by  
 1683

$$1684 \quad \text{span} \left( w_j \begin{pmatrix} 0 \\ \mathbb{I}_{i=1} \\ \vdots \\ \mathbb{I}_{i=n} \end{pmatrix} - w_i \begin{pmatrix} 0 \\ \mathbb{I}_{j=1} \\ \vdots \\ \mathbb{I}_{j=n} \end{pmatrix} \text{ for } i, j \in [n] \right) \subset LIE^{\geq 2}(\partial g).$$

1689 Clearly, this span is not in  $Ker(\partial g)$  as can be shown by a direct computation:

$$1691 \quad (\partial g) \left( w_j \begin{pmatrix} 0 \\ \mathbb{I}_{i=1} \\ \vdots \\ \mathbb{I}_{i=n} \end{pmatrix} - w_i \begin{pmatrix} 0 \\ \mathbb{I}_{j=1} \\ \vdots \\ \mathbb{I}_{j=n} \end{pmatrix} \right) = (\mathbb{I}_{i=1}mw_j - \mathbb{I}_{j=1}mw_i, \dots, \mathbb{I}_{i=n}mw_i - \mathbb{I}_{j=1}mw_i)$$

1695 For the span to be in the kernel we need either that all  $w_i = 0$  for all  $i \in [n]$  or  $m = 0$ . In both  
 1696 cases this implies that  $g(m, w) \in \{0\}$ . This implies that a mirror flow is only well-defined if the  
 1697 parameterization is zero. Hence, neuron-wise continuous sparsification does not induce a mirror  
 1698 flow.  $\square$   
 1699

1700  
 1701  
 1702  
 1703  
 1704  
 1705  
 1706  
 1707  
 1708  
 1709  
 1710  
 1711  
 1712  
 1713  
 1714  
 1715  
 1716  
 1717  
 1718  
 1719  
 1720  
 1721  
 1722  
 1723  
 1724  
 1725  
 1726  
 1727