

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 performance error), 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~~ Ziyin & Wang (2023) show that a loss with this parameterization $m \odot w$ with a separate weight decay $\alpha(m^2 + w^2)$. 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 L_1 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 (Evci 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~~) 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 PLoT 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 PLoT. 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 PLoT 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 α_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 α_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, ∇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 spread, 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_∞ 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 ∇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_∞ . In the case of diagonal linear networks, we can even prove optimality with respect to the final Bregman potential R_{a_∞} .

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.

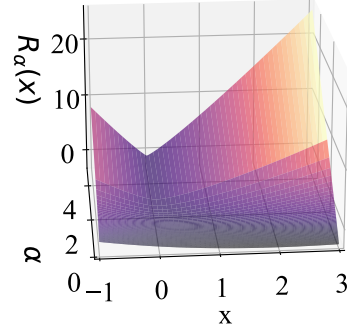


Figure 1: Evolution of the time-dependent Bregman potential. $\alpha = \int_0^t \alpha_s ds$ is the exponent of a_t .

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 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 2α :

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

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 β 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 η is not altered.

Our second and main improvement is induced by the time dependence of α_t . The α_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 spread. 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 spread 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 α_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 α_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 α_0 . Note that the setting $\delta = 1$ and $\beta = 0$ corresponds to spread, therefore. Therefore, PILOT is a strict generalization of spread. Furthermore, of spread. In contrast to spread, 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 α_{init} , initialization x_{init} , scaling constant β
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 spread in a so-called (Kusupati et al., 2020) and spread (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 α_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

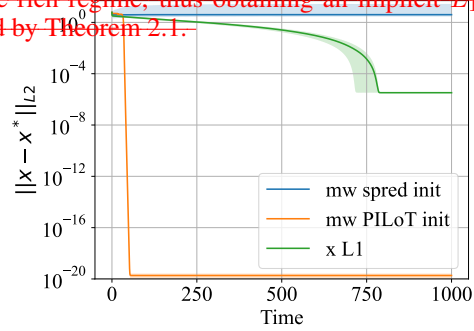


Figure 2: A simulation of gradient flow on a diagonal linear network is given for the different regularizations.

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

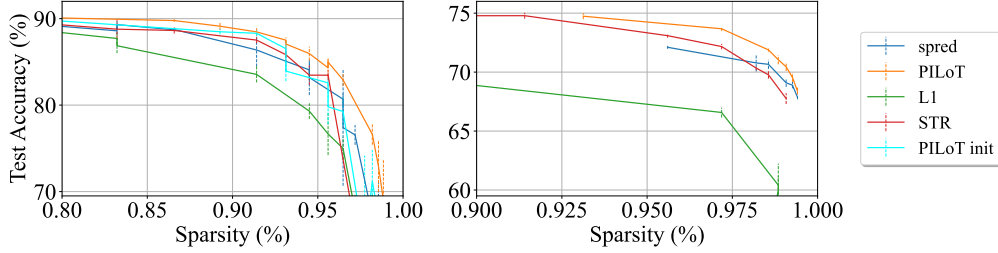


Figure 3: One-shot sparsification. Acc. versus sparsity for CIFAR10 (left) and CIFAR100 (right).

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

Method	Top-1 Acc	Sparsity
ResNet-50	77.01	0
STR	76.19	79.55
STR	76.12	81.27
spread ¹	75.5	80.00
spread	72.64	79.03
PLoT	75.62	80.00
STR	74.73	87.7
STR	74.01	90.55
spread	71.84	89.26
PLoT	74.73	88.00
PLoT	74.04	91.00

Method	Top-1 Acc	Sparsity
ResNet-50	77.01	0
STR	70.4	95.03
spread	69.47	94.50
PLoT	72.67	94.00
PLoT	71.30	95.00
PLoT	71.05	95.60
PLoT	70.49	96.00
STR	67.22	96.53
spread	66.12	97.19
PLoT	68.49	97.19
STR	61.46	98.05
spread	62.71	98.20
PLoT	66.49	97.75
PLoT	64.06	98.20

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

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

¹Starting from a pretrained model with 77% validation accuracy

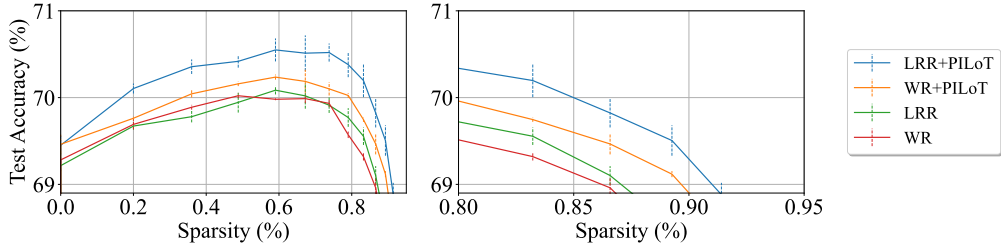


Figure 4: Learning Rate Rewinding (LRR) and Weight Rewinding (WR) with PILoT on ImageNet ResNet-18. The left plot is the complete plot and the right plot is zoomed-in on the higher sparsity regime.

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

- Aws Albarghouthi. Introduction to neural network verification, 2021.
- Felipe Alvarez, Jérôme Bolte, and Olivier Brahic. Hessian riemannian gradient flows in convex programming. *SIAM Journal on Control and Optimization*, 43(2):477–501, January 2004. ISSN 1095-7138. doi: 10.1137/s0363012902419977. URL <http://dx.doi.org/10.1137/S0363012902419977>.
- Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003. ISSN 0167-6377. doi: [https://doi.org/10.1016/S0167-6377\(02\)00231-6](https://doi.org/10.1016/S0167-6377(02)00231-6). URL <https://www.sciencedirect.com/science/article/pii/S0167637702002316>.
- Stephen P. Boyd and Lieven Vandenbergh. Convex optimization. 2009. URL <https://web.stanford.edu/~boyd/cvxbook/>.
- Rebekka Burkholz. Convolutional and residual networks provably contain lottery tickets. In *International Conference on Machine Learning*, 2022a.
- Rebekka Burkholz. Most activation functions can win the lottery without excessive depth. In *Advances in Neural Information Processing Systems*, 2022b.
- Rebekka Burkholz, Nilanjana Laha, Rajarshi Mukherjee, and Alkis Gotovos. On the existence of universal lottery tickets. In *International Conference on Learning Representations*, 2022.
- Tianlong Chen, Zhenyu Zhang, Jun Wu, Randy Huang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Can you win everything with a lottery ticket? *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL <https://openreview.net/forum?id=JL6MU9XFzW>.
- Lénaïc Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In Jacob Abernethy and Shivani Agarwal (eds.), *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 1305–1338. PMLR, 09–12 Jul 2020. URL <https://proceedings.mlr.press/v125/chizat20a.html>.
- Hung-Hsu Chou, Johannes Maly, and Dominik Stöger. How to induce regularization in linear models: A guide to reparametrizing gradient flow, 2024.
- Arthur da Cunha, Emanuele Natale, and Laurent Viennot. Proving the lottery ticket hypothesis for convolutional neural networks. In *International Conference on Learning Representations*, 2022.

- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- Steffen Dereich and Sebastian Kassing. Convergence of stochastic gradient descent schemes for lojasiewicz-landscapes, 2024.
- Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. 2019.
- Simon S. Du, Chi Jin, Jason D. Lee, Michael I. Jordan, Barnabas Poczos, and Aarti Singh. Gradient descent can take exponential time to escape saddle points, 2017.
- Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.
- Mathieu Even, Scott Pesme, Suriya Gunasekar, and Nicolas Flammarion. (s)gd over diagonal linear networks: Implicit regularisation, large stepsizes and edge of stability. *ArXiv*, abs/2302.08982, 2023. URL <https://api.semanticscholar.org/CorpusID:268042036>.
- Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.
- Benjamin Fehrman, Benjamin Gess, and Arnulf Jentzen. Convergence rates for the stochastic gradient descent method for non-convex objective functions, 2019.
- Damien Ferbach, Christos Tsirigotis, Gauthier Gidel, and Bose Avishek. A general framework for proving the equivariant strong lottery ticket hypothesis, 2022.
- Jonas Fischer and Rebekka Burkholz. Plant ‘n’ seek: Can you find the winning ticket?, 2021.
- Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv: Learning*, 2018. URL <https://api.semanticscholar.org/CorpusID:53388625>.
- Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

- Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. *CoRR*, abs/1912.05671, 2019. URL <http://arxiv.org/abs/1912.05671>.
- Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *International Conference on Learning Representations*, 2021.
- Advait Gadhikar and Rebekka Burkholz. Masks, signs, and learning rate rewinding. In *Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=qODvxQ8TXW>.
- Advait Harshal Gadhikar, Sohom Mukherjee, and Rebekka Burkholz. Why random pruning is all we need to start sparse. In *International Conference on Machine Learning*, 2023.
- Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit regularization in matrix factorization, 2017.
- Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry, 2020.
- Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.
- Intekhab Hossain, Jonas Fischer, Rebekka Burkholz, and John Quackenbush. Not all tickets are equal and we know it: Guiding pruning with domain-specific knowledge, 2024.
- Tian Jin, Michael Carbin, Daniel M. Roy, Jonathan Frankle, and Gintare Karolina Dziugaite. Pruning’s effect on generalization through the lens of training and regularization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=OrcLKV9sKWp>.
- John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Židek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.
- Lynn H Kaack, Priya L Donti, Emma Strubell, George Kamiya, Felix Creutzig, and David Rolnick. Aligning artificial intelligence with climate change mitigation. *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 Jiangyuan 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 Jiangyuan 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.

- Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In *International Conference on Learning Representations*, 2021.
- Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization, 2018.
- Alexandra Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? *arXiv preprint arXiv:2311.16863*, 2023.
- Jaron Maene, Mingxiao Li, and Marie-Francine Moens. Towards understanding iterative magnitude pruning: Why lottery tickets win, 2021.
- Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, 2020.
- Nina Narodytska, Hongce Zhang, Aarti Gupta, and Toby Walsh. In search for a sat-friendly binarized neural network architecture. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=SJx-j64FDr>.
- A.S. Nemirovski and D.B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. A Wiley-Interscience publication. Wiley, 1983. ISBN 9780471103455. URL <https://books.google.de/books?id=6ULvAAAAMAAJ>.
- Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. *Advances in Neural Information Processing Systems*, 33, 2020.
- Shreyas Malakarjun Patil and Constantine Dovrolis. Phew: Constructing sparse networks that learn fast and generalize well without training data. In *International Conference on Machine Learning*, pp. 8432–8442. PMLR, 2021.
- Mansheej Paul, Feng Chen, Brett W. Larsen, Jonathan Frankle, Surya Ganguli, and Gintare Karolina Dziugaite. Unmasking the lottery ticket hypothesis: What’s encoded in a winning ticket’s mask? In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=xSsW2Am-ukZ>.
- Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic overparameterization is sufficient. In *Advances in Neural 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 Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and
922 Dan Alistarh. Ac/dc: Alternating compressed/de-
923 compressed training of deep neural networks, 2021.
- 924 Hoang Pham, Shiwei Liu, Lichuan Xiang, Dung D Le,
925 Hongkai Wen, Long Tran-Thanh, et al. Towards
926 data-agnostic pruning at initialization: What makes
927 a good sparse mask? In *Thirty-seventh Conference
928 on Neural Information Processing Systems*, 2023.
- 929 Adityanarayanan Radhakrishnan, Mikhail Belkin, and
930 Caroline Uhler. Linear convergence of generalized
931 mirror descent with time-dependent mirrors, 2021.
- 932 Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey
933 Chu, and Mark Chen. Hierarchical text-conditional
934 image generation with clip latents. *arXiv preprint
935 arXiv:2204.06125*, 2022.
- 936 Alex Renda, Jonathan Frankle, and Michael Carbin.
937 Comparing rewinding and fine-tuning in neural net-
938 work pruning. In *International Conference on
939 Learning Representations*, 2020.
- 940 Tyrrel R Rockafellar and Werner Fenchel. *Con-
941 vex Analysis*. 1970. URL [https://api.
942 semanticscholar.org/CorpusID:
943 198120397](https://api.semanticscholar.org/CorpusID:198120397).
- 944 Pedro Savarese, Hugo Silva, and Michael Maire. Win-
945 ning the lottery with continuous sparsification, 2021.
- 946 Jonathan Schwarz, Siddhant M. Jayakumar, Razvan
947 Pascanu, Peter E. Latham, and Yee Whye Teh. Pow-
948 erpropagation: A sparsity inducing weight reparam-
949 eterisation, 2021.
- 950 Heejune Sheen, Siyu Chen, Tianhao Wang, and Harri-
951 son H. Zhou. Implicit regularization of gradient flow
952 on one-layer softmax attention, 2024.
- 953 Kartik Sreenivasan, Jy-yong Sohn, Liu Yang, Matthew
954 Grinde, Alliot Nagle, Hongyi Wang, Eric Xing,
955 Kangwook Lee, and Dimitris Papailiopoulos. Rare
956 gems: Finding lottery tickets at initialization. *Ad-
957 vances in Neural Information Processing Systems*,
958 35:14529–14540, 2022.
- 959 Natalie Stephenson, Emily Shane, Jessica Chase, Ja-
960 son Rowland, David Ries, Nicola Justice, Jie Zhang,
961 Leong Chan, and Renzhi Cao. Survey of machine
962 learning techniques in drug discovery. *Current drug
963 metabolism*, 20(3):185–193, 2019.
- 964 Hidenori Tanaka, Daniel Kunin, Daniel L. Yamins, and
965 Surya Ganguli. Pruning neural networks without any
966 data by iteratively conserving synaptic flow. In *Ad-
967 vances in Neural Information Processing Systems*,
968 2020.

- 972 Tomas Vaškevičius, Varun Kanade, and Patrick Rebes-
 973 chini. Implicit regularization for optimal sparse
 974 recovery, 2019. URL [https://arxiv.org/](https://arxiv.org/abs/1909.05122)
 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.](https://openreview.net/forum?id=Dvs-a3aymPe)
 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 Wojtowytsch. 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/](http://dx.doi.org/10.1093/biomet/asac010)
 1018 [10.1093/biomet/asac010](http://dx.doi.org/10.1093/biomet/asac010).
- 1019
 1020 Xiao Zhou, Weizhong Zhang, Zonghao Chen, Shizhe
 1021 Diao, and Tong Zhang. Efficient neural network
 1022 training via forward and backward propagation spar-
 1023 sification. *Advances in Neural Information Process-*
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*

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

Liu Ziyin. Symmetry induces structure and constraint of learning. 2023. URL <https://api.semanticscholar.org/CorpusID:263310669>.

Liu Ziyin and Zihao Wang. spread: Solving l_1 penalty with sgd, 2023. URL <https://arxiv.org/abs/2210.01212>.

A MIRROR FLOW FRAMEWORK

In this section we present some known results from the mirror flow framework for completeness. We derive the Bregman potential associated with $m \odot w$ in Theorem A.1. Next, we will also show the convergence of the loss and provide optimality guarantees in Theorem A.2. In addition, we extend Theorem A.2 with Theorem A.3. Finally, we give the optimality result for diagonal linear networks in Theorem A.4

Theorem A.1 *Let the initialization of m and w satisfy $m_{0,i} > |w_{0,i}|$ for all $i \in [n]$. Then the corresponding mirror function is:*

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

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

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

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

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

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

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

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

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

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

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

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

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.

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

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

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

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

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

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

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

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

Applying Gronwall’s Lemma concludes the proof.
 \square

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

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

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

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

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

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

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

In this section, we have shown the viability of using the implicit bias framework to induce an implicit regularization. Furthermore, we have given two known benefits of using the implicit bias framework over explicit regularization. The benefits are convergence to the optimal solution of the original problem and optimality in the case of underdetermined regression. To add to this, we have extended the convergence theorem using the PL-inequality in 9. Moreover, we again highlight the importance of the initialization of m_0 and w_0 with the influence on the smoothness of the Bregman potential and convergence of the loss. The initialization insight as in the main text is used to improve upon spread (Ziyin, 2023) as their initialization has scaling $2u_0v_0 = 0$, it follows already from the mirror flow framework that $2u_0v_0 = 1$ is a better initialization. Furthermore, we also do not initialize at zero though, and scaling 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 ∇f needs to be locally Lipschitz continuous. This is a sufficient condition given that α_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*

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

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

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

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

where the second equality follows from hyperbolic identities. \square

It follows from Lemma B.1 and the local Lipschitz condition on ∇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

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

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 ∇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_∞ 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 $x \in \mathbb{R}^n$,

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

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 μ -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.*

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

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

The divergence is bounded by:

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

The evolution of the bound is

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

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

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

where the first inequality follows from convexity of the loss and the third inequality from the fact 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 iterates converge, concluding the convex case.

In case when f satisfies the PL-inequality, we proceed in the same way as Theorem A.3. The evolution of f is given by

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

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

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

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

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

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

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

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

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

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

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

B.1 DISCUSSION OF THE PROOF

Most of the proof follows the same arguments as in (Alvarez et al., 2004; Li et al., 2022; Pesme et al., 2021). The notable differences are showing that the loss becomes decreasing over time and the observations made in (13).

C DETAILS EXPERIMENTS

In this section, we provide the details of the experiments. In addition, there are additional figures given.

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

C.1 DIAGONAL LINEAR NETWORK

For each setting, different regularization schemes are tried. In total, 7 options are tried. 4 of the schedules are constant i.e. the regularization stays the same during training. The remaining 3 are decaying schedules. These schedules we name harmonic, quadratic, and geometric. The schedules are described by the following recurrent relations:

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

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

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

that our regularization performs the best with the decaying schedules as predicted by the theory. The other methods need constant regularization to perform well as already mentioned in Remark 2.1. Note that, PILOt also can perform well with constant regularization (see Figure 5).

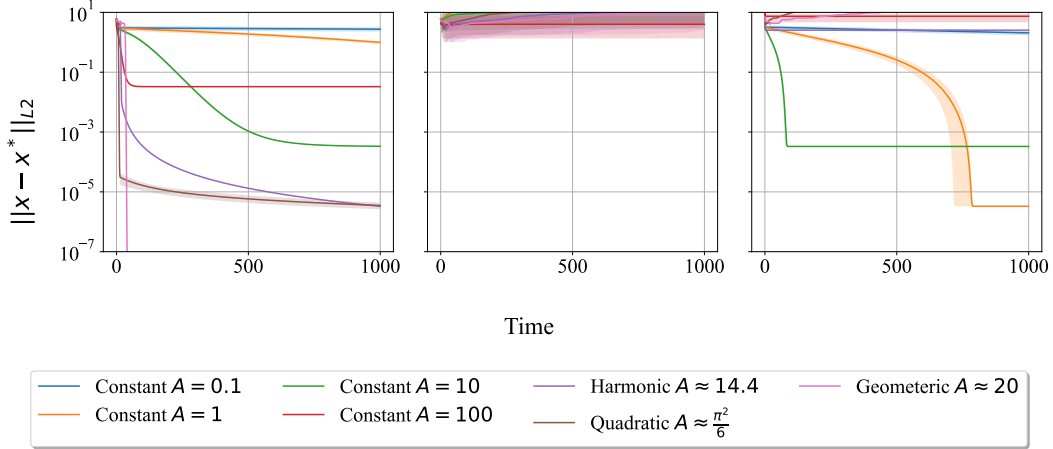


Figure 5: All runs for the diagonal linear network. From left to right $m \odot w$ with PILOt initialization, $m \odot w$ with spread initialization, and x with L_1 regularization

C.2 ONE-SHOT

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

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

Label smoothing Although PILOt is competitive in the 80% – 90% sparsity range it is not SOTA. Nevertheless, if we turn off label smoothing in the experiment PILOt outperforms STR in this range as well. The only change for STR is turning of labelsmoothing. For PILOt we use two different configurations.

²Applied to the other parameters

³Starting 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 ²	10^{-4}	
Base learning rate	$\{0.1, 0.2\}$	
Epochs	150	
Warmup period	0	
Initialization	Kaiming normal	
Scaling	1	Only for $m \odot w$
δ	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	α_{init}	K	sparsity
spred ³	$2e-5$	-	80.00
spred	$3e-6$	-	79.03
PILoT	$7e-6$	60000	80.00
spred	$5e-6$	-	89.26
PILoT	$1e-5$	60000	88.00
PILoT	$1.4e-5$	60000	91.00
spred	$2e-5$	-	94.50
PILoT	$2e-5$	60000	94.00
PILoT	$3e-5$	60000	95.00
PILoT	$3e-5$	60000	96.00
spred	$3e-5$	-	97.19
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. $(m \odot w)^2$ 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	79.03
PILoT	74.72	79.03
STR	71.6	89.26
PILoT	73.21	91.41

C.3 ITERATIVE PRUNING

In Table 5 the details of the ImageNet the experiment are given. Note the base learning rate 0.1 is for the baseline and 0.2 is used for our parameterization combined with scaling 1. In addition, the L_2 regularization denotes the reparameterization of the original weight decay. Thus for PILOT, in this case, we use that instead. Moreover, all runs have been done for 3 different seeds. Furthermore, we provide additional experiments on CIFAR 10 and 100 with ResNet-20 and ResNet-18 respectively in Figure 6. The details are given in Table 6

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 $m \odot w$
δ	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 $m \odot w$
δ	1	Only for $m \odot w$
K	—	Only for $m \odot w$

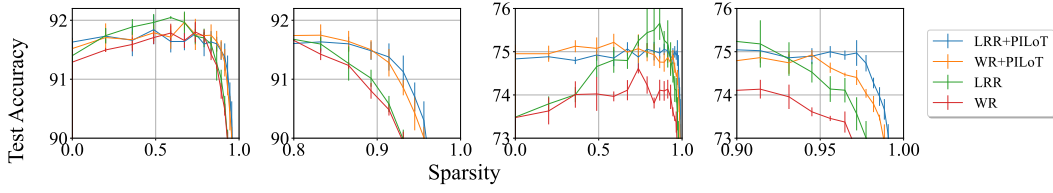


Figure 6: Learning Rate Rewinding (LRR) and Weight Rewinding (WR) with PiLoT shows improvement over the baseline iterative pruning methods for CIFAR 10 and 100.

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 ∂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:

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

Computing $Hg_i \nabla g_j$

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

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

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

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

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

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 cases this implies that $g(m, w) \in \{0\}$. This implies that a mirror flow is only well-defined if the parameterization is zero. Hence, neuron-wise continuous sparsification does not induce a mirror flow. \square