SPARSE TRAINING OF NEURAL NETWORKS BASED ON MULTILEVEL MIRROR DESCENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a dynamic sparse training algorithm based on linearized Bregman iterations / mirror descent that exploits the naturally incurred sparsity by alternating between periods of static and dynamic sparsity pattern updates. The key idea is to combine sparsity-inducing Bregman iterations with adaptive freezing of the network structure to enable efficient exploration of the sparse parameter space while maintaining sparsity. We provide convergence guaranties by embedding our method in a multilevel optimization framework. Furthermore, we empirically show that our algorithm can produce highly sparse and accurate models on standard benchmarks. We also show that the theoretical number of FLOPs compared to SGD training can be reduced from 38% for standard Bregman iterations to 6% for our method while maintaining test accuracy.

1 INTRODUCTION

Deep neural networks have produced astonishing results in various areas such as computer vision and natural language processing (Mohd Noor & Ige, 2025) but demand significant memory and specialized hardware, contributing to growing concerns about their environmental impact, particularly the carbon footprint of training and inference (Dhar, 2020).

In response, researchers have explored techniques for developing compact and efficient models, such as sparse neural networks, wherein many neuron connections are absent. Within this context, the Lottery Ticket Hypothesis (Frankle & Carbin, 2019) plays a central role, suggesting that every dense network contains a sparse subnetwork that, when trained independently, can achieve comparable accuracy.

There are two main approaches to obtain sparse neural networks: pruning and sparse training. In pruning, a dense model is first trained and unwanted connections are removed afterward. Because this usually causes a drop in performance, the pruned model is often retrained with the sparsity pattern fixed. By contrast, sparse training incorporates mechanisms that encourage sparsity already during training. Pruning methods themselves vary widely, depending on how weights are selected for removal and at what stage of training pruning is applied. A key distinction is between unstructured pruning, which eliminates individual weights, and structured pruning, which removes entire components such as neurons or filters, see Hoefler et al. (2021) for an extensive overview.

In addition to pruning-based methods, another technique to encourage sparsity is to include explicit regularization terms in the loss function. A common example is Lasso regularization, which uses the ℓ_1-norm as a penalty in the objective function (Tibshirani, 1996). The resulting optimization problem can be solved using algorithms such as Proximal Gradient Descent (Rosasco et al., 2020; Mosci et al., 2010). A conceptually different approach is to enforce sparsity through *implicit regularization* which can be achieved using mirror descent (Nemirovskij & Yudin, 1983). Here we would like to highlight a series of works (Huang et al., 2016; Azizan et al., 2021; Bungert et al., 2021; 2022; Wang & Benning, 2023; Heeringa et al., 2025b) that utilize mirror descent or linearized Bregman iterations to induce sparsity in neural networks without explicit regularization.

Linearized Bregman iterations are equivalent to mirror descent, but they are typically formulated and analyzed with less regularity assumptions on the mirror map (e.g., compared to Azizan et al. (2021)), thereby lending themselves toward non-smooth sparsity-promoting mirror maps. This was exploited in (Bungert et al., 2022) to devise the LinBreg algorithm which essentially is a stochastic mirror descent algorithm applied with a non-smooth mirror map.

Within the context of sparse training, we also highlight the relevance of genetic evolutionary algorithms, particularly Sparse Evolutionary Training (SET) (Mocanu et al., 2018). SET applies a dynamic sparse training approach by performing pruning at the end of each epoch—removing a fraction of the active connections—and regrowing an equal number of new connections at random positions. This iterative process maintains a fixed sparsity level. Beyond sparse training, evolutionary algorithms have also been successfully applied to Neural Architecture Search (Miikkulainen et al., 2024; Elsken et al., 2019); however, they typically lack rigorous convergence guarantees.

In this paper, we propose a training algorithm based on linearized Bregman iterations, designed to promote sparsity in neural networks *during training*. A key benefit of Bregman iterations over regularization methods like the Lasso is that for the former the number of non-zero parameters of the trained networks usually increases monotonically. Hence, algorithms like LinBreg lend themselves to exploiting sparsity early on in the training process. In our method, we periodically freeze the network's structure: that is, we restrict updates to parameters that are non-zero in the current iteration. This approach offers two key benefits. First, stricter sparsity is enforced than through LinBreg alone, as the number of non-zero parameters cannot increase during the frozen phases. Second, during the frozen phases only derivatives corresponding to active parameters are required which provides scope for significant computational savings during training.

We embed the resulting algorithm within a multilevel optimization framework (Nash, 2000), which enables us to leverage existing convergence theory. In particular, we adapt the convergence analysis of Multilevel Bregman Proximal Gradient Descent (Elshiaty & Petra, 2025) to our sparse training setup. We find that our method can outperform the standard LinBreg algorithm by yielding models that are sparser while achieving comparable or even superior performance for image classification.

The remainder of this paper is structured as follows. First, we explain our algorithm and present how it can be interpreted as a multilevel optimization method. We proceed by proving a sublinear convergence result for the algorithm. Finally, we perform numerical experiments comparing our method to other methods that aim at achieving sparse but performative models.

2 Related Work

Bregman Iterations / Mirror descent Bregman iterations were originally introduced by Osher et al. (2005) as iterative reconstruction method for imaging inverse problems to overcome the bias of regularization methods like total variation denoising (Rudin et al., 1992). Later they were applied to compressed sensing (Yin et al., 2008) and nonlinear inverse problems (Bachmayr & Burger, 2009; Benning et al., 2021). In the context of machine learning, they were used for sparsity (Bungert et al., 2021; 2022; Heeringa et al., 2025b;a) of neural network representations as well as for training networks with non-smooth activations of proximal type (Wang & Benning, 2023). While Bregman iterations in their original form generalize the implicit Euler method, so-called linearized Bregman iterations are closely related to mirror descent (Nemirovskij & Yudin, 1983) or more precisely to lazy mirror descent / Nestorov's dual averaging (Nesterov, 2009). The method is also referred to as Bregman proximal gradient descent and a stochastic gradient version of it was coined LinBreg by Bungert et al. (2022). It must be emphasized that, just like different communities use different terminologies, they also developed different mathematical tools to analyze the convergence behavior. In particular, the inverse problems community put a lot of effort into analyzing Bregman iterations with sparsity-promoting regularizers which translates to mirror descent with non-smooth mirror maps. This will also be our approach in this paper.

Multilevel Optimization Multilevel optimization methods originate from multigrid techniques, which were initially developed to solve differential equations efficiently. The MGOPT algorithm (Nash, 2000) was among the first to adapt these ideas to optimization problems. More recently, Elshiaty & Petra (2025) extended this framework by incorporating linearized Bregman iterations. Their work provides convergence guarantees via a Polyak–Łojasiewicz-type inequality for the ML BPGD algorithm and demonstrates its effectiveness in image reconstruction tasks. Hovhannisyan et al. (2016) provide a connection between multilevel optimization and mirror descent, noting that the latter is equivalent to linearized Bregman iterations. They further incorporate acceleration techniques, prove convergence of their algorithm, and illustrate its performance through numerical experiments on face recognition. Multilevel strategies have also been explored in deep learning, particularly for training residual neural networks (ResNets). Kopaničáková & Krause (2023) introduce a hierarchy based on networks of varying depth and width, leveraging the fact that smaller models are faster and cheaper to train. Similar approaches have been made e.g. by Gaedke-Merzhäuser et al. (2021). Other approaches embed the multilevel hierarchy directly into the objective function rather than the model architecture. For example, Braglia et al. (2020) use varying batch sizes to compute the loss, effectively inducing a multiscale structure during training.

Sparse neural networks Within the context of sparse training, several methods have been proposed to obtain sparse yet performant models without the need to train a dense network beforehand. One approach is to keep a static sparsity pattern throughout training which is identified in advance based upon, say, the magnitude of the initialized weights' gradient of the loss (Lee et al., 2019) or second order derivative information (Singh & Alistarh, 2020). Another way is to allow for the sparsity pattern to dynamically change throughout training by including mechanisms that remove and regrow connections according to certain criterion. For example, DEEP-R (Bellec et al., 2018) uses a random regrowth strategy whereas RigL (Evci et al., 2020) activates connections according to gradient magnitude. Other criterion may include: accumulated gradient information (Liu et al., 2021); momentum (Dettmers & Zettlemoyer, 2019); the difference to the weights of a previous training step (Li et al., 2023). DFBST (Pote et al., 2023) applies binary masks during both the forward and backward passes in which the forward pass mask sparsifies the weights, while the backward pass mask restricts gradient updates, enabling sparse training. Top-KAST (Jayakumar et al., 2020) maintains a fixed sparse model by using only the top-D weights for the forward pass, but employs a slightly larger set of parameters during the backward pass to update and explore potential alternative masks. Ji et al. (2024b) decouple the optimization of masks and weights, treating the mask as an explicit optimization variable. Such a decoupling can be viewed in a time-dependent mirror flow framework (Jacobs & Burkholz, 2025) in which convergence and optimality results can be stated. The AC/DC method (Peste et al., 2021) uses alternating phases of full-support and sparse-support optimization to produce accurate sparse–dense model pairs, while providing convergence guarantees of the underlying iterative hard thresholding algorithm for non-convex training under a Polyak–Łojasiewicz-type inequality. Yin et al. (2023) propose an algorithm to obtain channel-level sparse networks by performing channel pruning during training, where channels are removed based on the mean magnitude of their weights. Techniques have also been developed to further enhance existing sparse training algorithms such as applying the soft thresholding operator with learnable thresholds to layer weights (Kusupati et al., 2020) and performing weight updates that minimize over a local neighborhood (Ji et al., 2024a).

3 Method

A typical training problem to find optimal network parameters $\theta \in \mathbb{R}^d$ consists of solving

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta), \tag{1}$$

where $\mathcal{L} \colon \mathbb{R}^d \to \mathbb{R}$ is a differentiable and non-negative loss function, for example, the empirical loss. One way to enforce constraints or encourage sparsity in solutions is to specify a proper, convex, and lower semicontinuous function $J \colon \mathbb{R}^d \to (-\infty, \infty]$, such as the ℓ_1-norm or indicator function of a closed convex set, and consider a minimization of $\mathcal{L} + J$. The approach of Bregman iterations is to directly minimize $\mathcal{L}$, while implicitly minimizing J. This is achieved through the scheme

$$\begin{aligned} \theta^{(k+1)} &= \operatorname*{arg\,min}_{\theta \in \mathbb{R}^d} D_J^{p^{(k)}}(\theta, \theta^{(k)}) + \tau^{(k)} \mathcal{L}(\theta), \\ p^{(k+1)} &= p^{(k)} - \tau^{(k)} \nabla \mathcal{L}(\theta^{(k+1)}) \in \partial J(\theta^{(k+1)}), \end{aligned} \tag{2}$$

where the so-called Bregman divergence (associated to J) is defined as:

$$D_J^p(\tilde{\theta}, \theta) := J(\tilde{\theta}) - J(\theta) - \langle p, \tilde{\theta} - \theta \rangle. \tag{3}$$

Here $\theta \in \operatorname{dom}(\partial J)$, $p \in \partial J(\theta)$ is a subgradient, and $\tau^{(k)} > 0$ is a sequence of step sizes. The Bregman divergence (3) can be interpreted as the difference between J and its linearization around θ and satisfies properties such as $D_J^p(\theta, \theta) = 0$ and, due to convexity of J, $D_J^p(\tilde{\theta}, \theta) \geq 0$.

Since solving the optimization problem (2) is typically almost as hard as solving (1), one typically replaces $\mathcal{L}$ in (2) with its first order approximation $\mathcal{L}(\theta^{(k)}) + \langle \nabla\mathcal{L}(\theta^{(k)}), \theta - \theta^{(k)}\rangle$ and J with the strongly convex elastic-net regularizer

$$J_\delta(\theta) := \frac{1}{2\delta}\|\theta\|^2 + J(\theta), \quad \delta \in (0,\infty),$$

to obtain (see, e.g., Bungert et al. (2022) for a derivation)

$$v^{(k+1)} = v^{(k)} - \tau\nabla\mathcal{L}(\theta^{(k)}), \tag{4a}$$

$$\theta^{(k+1)} = \operatorname{prox}_{\delta J}(\delta v^{(k+1)}), \tag{4b}$$

starting at some $\theta^{(0)}$ and $v^{(0)} \in \partial J_\delta(\theta^{(0)})$. The algorithm involves the proximal operator

$$\operatorname{prox}_{\delta J}(\theta) := \operatorname*{arg\,min}_{\tilde\theta\in\mathbb{R}^d} \frac{1}{2\delta}\|\tilde\theta - \theta\|^2 + J(\tilde\theta).$$

To make (4) feasible for the high-dimensional and non-convex problems arising in machine learning, Bungert et al. (2022) in their LinBreg method replaced the gradient $\nabla\mathcal{L}(\theta^{(k)})$ in (4a) by an unbiased stochastic estimator and provided a convergence analysis.

Note that if $J \equiv 0$, the proximal operator is the identity map and, taking $\delta = 1$, (4) recovers Gradient Descent. More generally, (4) coincides with mirror descent (Beck & Teboulle, 2003) applied to the distance generating function J_δ. This can be seen by noting that $\nabla J_\delta^* = \operatorname{prox}_{\delta J}(\delta\cdot)$, where J_δ^* denotes the convex conjugate (Bauschke & Combettes, 2011) of J_δ.

Although evaluating the proximal operator in (4b) is phrased as a minimization problem, particular choices of J admit closed forms, e.g., $J = \lambda\|\cdot\|_1$ yields the soft shrinkage operator

$$\operatorname{prox}_{\delta J}(\delta v) = \delta\,\operatorname{sign}(v)\max(|v| - \lambda, 0) \tag{5}$$

applied componentwise. This particular choice also demonstrates how Bregman iterations can lead to sparse networks. In (5); only parameters whose corresponding dual variable v exceeds the threshold λ in absolute value will be non-zero. Thus, we can view (4b) as a pruning step inherent to the optimizer, where the pruning criterion employs information associated with the regularizer J, and not just the magnitude of the parameter or of the gradient of the training loss $\mathcal{L}$.

In practice, θ can represent the parameters of several network layers, and as such one may wish to employ a different regularizer for each layer. To represent this, we split the parameter vector θ into groups via $\theta = (\theta_{(1)}, \dots, \theta_{(G)})$, where each group $\theta_{(g)} \in \mathbb{R}^{d_g}$ contains d_g scalar parameters. Furthermore, we assume the regularizer J acts on these groups separately, taking the form

$$J(\theta) = \sum_{g=1}^{G} J_g(\theta_{(g)}), \tag{6}$$

where each $J_g \colon \mathbb{R}^{d_g} \to (-\infty, \infty]$ is proper, convex, and lower-semicontinuous. We see that (6) includes the standard ℓ_1-norm but also the group $\ell_{1,2}$-norm (Scardapane et al., 2017), given by

$$J(\theta) := \sum_{g=1}^{G} \sqrt{n_g}\|\theta_{(g)}\|_2, \tag{7}$$

where n_g denotes the number of parameters in the group. While the ℓ_1-norm encourages individual parameters to become zero, the group $\ell_{1,2}$-norm can be used to enforce entire structures, such as convolutional kernels, to vanish.

The main idea of our algorithm is to use this induced sparsity of the iterations (4) by only performing a full update with this rule every m iterations. In all other iterations, we update only the non-zero parameters and consequently only require gradients with respect to the active parameters. Depending on the induced sparsity pattern, this provides scope for significant computational savings during most training steps. We show that we can interpret this idea as a multilevel optimization scheme, allowing us to adapt convergence proofs from the multilevel optimization literature.

More precisely, we consider a two-level framework consisting of the actual minimization problem (1) and a coarse problem with fewer variables. To map between these levels, the restriction and prolongation operators are used. The restriction operator at iteration k, denoted $R^{(k)} \colon \mathbb{R}^d \to \mathbb{R}^{D_k}$ with

$D_k < d$, is a linear map that decides which of the variables we restrict ourselves to. Consequently, the rows of the matrix $R^{(k)}$ are standard unit vectors and θ_i is selected by $R^{(k)}$ if and only if one of the rows of the matrix is the i-th standard unit vector. We only consider the case where entire groups are selected by the restriction operator, so if one component of a group $\theta_{(g)}$ is selected, then all the other components must be selected as well. Given the number of selected groups $G^{(k)}$, we can define an injective function $r^{(k)} \colon \{1, 2, \ldots, G^{(k)}\} \to \{1, 2, \ldots, G\}$ such that

$$R^{(k)}\theta = (\theta_{(r^{(k)}(1))}, \ldots, \theta_{(r^{(k)}(G^{(k)}))}), \quad \theta \in \mathbb{R}^d.$$

For a given coarse variable $\hat{\theta}$, this function is useful to determine where its parameter groups belong on the fine level.

The corresponding prolongation operator $P^{(k)} \colon \mathbb{R}^{D_k} \to \mathbb{R}^d$ maps from the coarse to the fine level and is defined as the transpose of the restriction $P^{(k)} = (R^{(k)})^T$. This means that the prolongation operator maps the groups that were selected by the restriction back and simply completes the parameter vector θ by setting zero for every parameter group that was not selected by the restriction. Using the previously defined function $r^{(k)}$, we can explicitly write the prolongation of a coarse variable $\hat{\theta} \in \mathbb{R}^{D_k}$ as

$$(P^{(k)}\hat{\theta})_{(g)} = \begin{cases} \hat{\theta}_{(i)}, & \text{if } g = r^{(k)}(i), \\ 0, & \text{otherwise.} \end{cases}$$

While the case where the restriction operator chooses the groups that are non-zero at iteration k is the most interesting for us, our analysis works for arbitrary selections of parameter groups.

In our algorithm, during each iteration k, a predefined criterion is evaluated to decide whether to invoke the coarse-level model. If the coarse model is employed, the restriction operator $R^{(k)}$ maps the current iterate $\theta^{(k)}$ and the corresponding subgradient $v^{(k)}$ to the coarse level. We denote these restrictions by $\hat{\theta}^{0,k} := R^{(k)}\theta^{(k)}$ and $\hat{v}^{0,k} := R^{(k)}v^{(k)}$. The algorithm then performs m LinBreg steps to minimize the coarse loss

$$\hat{\mathcal{L}}^{(k)}(\hat{\theta}) := \mathcal{L}(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})) \tag{8}$$

using $\hat{J}^{(k)}_{\delta}(\hat{\theta}) := \frac{1}{2\delta}\|\hat{\theta}\|^2 + \hat{J}^{(k)}(\hat{\theta})$ as the regularizer, where

$$\hat{J}^{(k)}(\hat{\theta}) := \sum_{i=1}^{G^{(k)}} J_{r^{(k)}(i)}(\hat{\theta}_{(i)}).$$

The result of these coarse iteration steps $\hat{\theta}^{m,k}$ with corresponding subgradient $\hat{v}^{m,k}$ is then mapped back to the fine level via

$$\begin{aligned} \tilde{\theta}^{(k+1)} &:= \theta^{(k)} + P^{(k)}(\hat{\theta}^{m,k} - \hat{\theta}^{0,k}), \\ \tilde{v}^{(k+1)} &:= v^{(k)} + P^{(k)}(\hat{v}^{m,k} - \hat{v}^{0,k}), \end{aligned}$$

before performing a LinBreg step on the fine level to obtain $\theta^{(k+1)}$ and $v^{(k+1)}$. See Algorithm 1.

If the restriction operator at iteration k only selects the parameter groups that are non-zero for $\theta^{(k)}$, the coarse loss (8) and the prolongation to the fine level simplify to

$$\hat{\mathcal{L}}^{(k)}(\hat{\theta}) = \mathcal{L}(P^{(k)}\hat{\theta}), \qquad \tilde{\theta}^{(k+1)} = P^{(k)}\hat{\theta}^{m,k}.$$

For the subgradients, however, such simplification is not available.

Typically in multilevel optimization, a linear correction term is added to the coarse objective function to ensure first-order coherence, i.e., critical points on the fine level are transferred to ones on the coarse level. In our case, due to the special structure of $\hat{\mathcal{L}}^{(k)}$ no correction term is necessary, since

$$\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k}) = R^{(k)}\nabla\mathcal{L}(\theta^{(k)}).$$

Hence, if $\theta^{(k)}$ is a critical point of $\mathcal{L}$, then $\hat{\theta}^{0,k} := R^{(k)}\theta^{(k)}$ is a critical point of $\hat{\mathcal{L}}^{(k)}$.

A criterion to decide when to use the coarse model is (Wen & Goldfarb, 2009; Vanmaele et al., 2025)

$$\|R^{(k)}\nabla\mathcal{L}(\theta^{(k)})\| \geq \kappa\|\nabla\mathcal{L}(\theta^{(k)}\|, \qquad \|R^{(k)}\nabla\mathcal{L}(\theta^{(k)})\| > \varepsilon, \tag{9}$$

where κ and ε are positive hyperparameters. Inequalities (9) prevent coarse iteration updates in the case that $\hat{\theta}^{0,k}$ is a critical point of $\hat{\mathcal{L}}^{(k)}$ (which does not necessarily mean that $\theta^{(k)}$ is a critical point of $\mathcal{L}$). MAGMA (Hovhannisyan et al., 2016) uses a similar criterion to decide when to invoke the coarse model. Specifically, it checks the first part of (9) and, in addition, whether the current iterate is sufficiently far from the point of the last coarse update or the number of consecutive fine updates is below a predefined limit. This ensures coarse updates are triggered only when they are beneficial—either because the iterate has changed significantly, or because not too many consecutive fine steps have yet been taken.

Since we work with subgradients, we carefully investigate the transfer between the different levels. In particular, we show that restricting the subgradients from the fine level yields subgradients of the coarse regularizer and mapping the coarse subgradients back to the fine level as described also leads to subgradients on the fine level due to the structure of J from (6). The proofs of the following propositions can be found in Appendix A.

Proposition 1. *If $v^{(k)} \in \partial J_\delta(\theta^{(k)})$, then defining $\hat{v}^{0,k} := R^{(k)}v^{(k)}$ and $\hat{\theta}^{0,k} := R^{(k)}\theta^{(k)}$ we have*

$$\hat{v}^{0,k} \in \partial\hat{J}_\delta^{(k)}(\hat{\theta}^{0,k}).$$

This implies that in Algorithm 1 we start the coarse iteration with a feasible pair $(\hat{\theta}^{0,k}, \hat{v}^{0,k})$. Consequently, due to the structure of the iteration, we subsequently obtain $\hat{v}^{i,k} \in \partial\hat{J}_\delta^{(k)}(\hat{\theta}^{i,k})$ for $i = 1, \ldots, m$. We also observe that the way of transferring the updated variables back to the fine level preserves the fact that $\tilde{v}^{(k+1)}$ is a subgradient of J_δ at $\tilde{\theta}^{(k+1)}$.

Proposition 2. *If $\hat{\theta}^{m,k} \in \operatorname{dom}(\partial\hat{J}^{(k)})$ and $\hat{v}^{m,k} \in \partial\hat{J}_\delta^{(k)}$, then defining*

$$\begin{aligned}\tilde{\theta}^{(k+1)} &:= \theta^{(k)} + P^{(k)}(\hat{\theta}^{m,k} - \hat{\theta}^{0,k}),\\ \tilde{v}^{(k+1)} &:= v^{(k)} + P^{(k)}(\hat{v}^{m,k} - \hat{v}^{0,k})\end{aligned}$$

we have $\tilde{v}^{(k+1)} \in \partial J_\delta(\tilde{\theta}^{(k+1)})$.

Algorithm 1 Multilevel LinBreg

```
Input: Initial guess $\theta^{(0)} \in \mathbb{R}^d, v^{(0)} \in \partial J_\delta(\theta^{(0)})$
for $k \in \mathbb{N}$ do
  if condition to use coarse model is satisfied at $\theta^{(k)}$ then
    $\hat{\theta}^{0,k} = R^{(k)}\theta^{(k)}$
    $\hat{v}^{0,k} = R^{(k)}v^{(k)}$
    for $i = 1, \ldots, m$ do
      $\hat{g}^{(k)} \leftarrow$ unbiased estimator of $\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i-1,k})$
      $\hat{v}^{i,k} = \hat{v}^{i-1,k} - \hat{\tau}^{i-1,k}\hat{g}^{(k)}$
      $\hat{\theta}^{i,k} = \operatorname{prox}_{\delta\hat{J}^{(k)}}(\delta\hat{v}^{i,k})$
    end for
    $\tilde{v}^{(k+1)} = v^{(k)} + P^{(k)}(\hat{v}^{m,k} - \hat{v}^{0,k})$
    $\tilde{\theta}^{(k+1)} = \theta^{(k)} + P^{(k)}(\hat{\theta}^{m,k} - \hat{\theta}^{0,k})$
    $g^{(k+1)} \leftarrow$ unbiased estimator of $\nabla\mathcal{L}(\tilde{\theta}^{(k+1)})$
    $v^{(k+1)} = \tilde{v}^{(k+1)} - \tau g^{(k+1)}$
    $\theta^{(k+1)} = \operatorname{prox}_{\delta J}(\delta v^{(k+1)})$
  else
    $g^{(k)} \leftarrow$ unbiased estimator of $\nabla\mathcal{L}(\theta^{(k)})$
    $v^{(k+1)} = v^{(k)} - \tau g^{(k)}$
    $\theta^{(k+1)} = \operatorname{prox}_{\delta J}(\delta v^{(k+1)})$
  end if
end for
```

To conclude this section, we would like to highlight the key differences between our algorithm and the methods ML BPGD (Elshiaty & Petra, 2025) and MGOPT (Nash, 2000). Firstly, we use adaptive restriction and prolongation operators, which are necessary to focus on different parameters at different iterations. Secondly, we do not perform a line search when mapping from the coarse level to the fine level. MGOPT and ML BPGD both use arbitrary convex coarse objective functions with a linear correction term to ensure that the direction found with the coarse iterations is a descent direction, as well as a line search to ensure that the fine loss actually decreases. Our specific selection of the coarse objective function ensures a decrease in the fine loss (see Lemma 3), eliminating the need for a line search. Finally, unlike ML BPGD, our regularizer J_δ can be non-differentiable. Therefore, rather than working with gradients, we need to consider subgradients which must be handled with care when mapping between different levels (see Propositions 1 and 2).

4 Convergence Analysis

For the convergence analysis, we need to make further assumptions. We follow Bauschke et al. (2019); Elshiaty & Petra (2025) and require the loss function to be smooth relative to the regularizer and to satisfy a Polyak–Łojasiewicz-like inequality.

Assumption 1. *We assume that $\mathcal{L}$ is L-smooth with respect to J_δ, i.e.*

$$\mathcal{L}(\tilde{\theta}) \leq \mathcal{L}(\theta) + \langle \nabla \mathcal{L}(\theta), \tilde{\theta} - \theta \rangle + L D^{v}_{J_\delta}(\tilde{\theta}, \theta)$$

for $\tilde{\theta} \in \mathbb{R}^d$, $\theta \in \operatorname{dom}(\partial J)$ and $v \in \partial J_\delta(\theta)$.

Since J_δ is strongly convex, its induced Bregman divergence is bounded from below by the squared Euclidean norm. Therefore, from the descent lemma for L-smooth functions (Bauschke & Combettes, 2011; Beck, 2017) it follows that this assumption holds in particular for loss functions $\mathcal{L}$ with a Lipschitz-continuous gradient which, however, is generally a much stronger condition than Assumption 1.

Assumption 2. *For $\theta \in \operatorname{dom}(\partial J), v \in \partial J_\delta(\theta)$ and $\tau > 0$, let $v^+_\tau := v - \tau \nabla \mathcal{L}(\theta)$ and $\theta^+_\tau := \operatorname{prox}_{\delta J}(\delta v^+_\tau)$. We assume the existence of a function $\lambda \colon (0,\infty) \to (0,\infty)$ and some $\eta > 0$ such that*

$$D^{v^+_\tau}_{J_\delta}(\theta, \theta^+_\tau) \geq \lambda(\tau) D^{v^+_1}_{J_\delta}(\theta, \theta^+_1) \tag{10}$$

for $\theta \in \operatorname{dom}(\partial J), v \in \partial J_\delta(\theta), \tau > 0$ and

$$D^{v^+_1}_{J_\delta}(\theta, \theta^+_1) \geq \eta(\mathcal{L}(\theta) - \mathcal{L}^*), \tag{11}$$

where $\mathcal{L}^* := \inf_{\mathbb{R}^d} \mathcal{L}$.

In the gradient descent case where $J_\delta = \frac{1}{2}\|\cdot\|^2$, assumption (10) is satisfied with $\lambda(\tau) = \tau^2$ and (11) becomes the well-known Polyak–Łojasiewicz inequality

$$\|\nabla \mathcal{L}(\theta)\|^2 \geq 2\eta(\mathcal{L}(\theta) - \mathcal{L}^*)$$

which is weaker than convexity but requires every stationary point of $\mathcal{L}$ to be a minimizer.

For our convergence analysis, we use the exact gradients on the fine level and stochastic gradient estimators on the coarse level. We assume that these gradient estimators are unbiased with bounded variance. To be precise, let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $\hat{g}^{(k)} : \mathbb{R}^{D_k} \times \Omega \to \mathbb{R}^{D_k}$. We make the following assumption regarding $\hat{g}^{(k)}$.

Assumption 3. *We assume that for $\hat{\theta} \in \mathbb{R}^{D_k}$,*

$$\mathbb{E}\left[\hat{g}^{(k)}(\hat{\theta}, \omega) | \hat{\theta}\right] = \nabla \hat{\mathcal{L}}^{(k)}(\hat{\theta}).$$

Moreover, there exists a positive constant σ such that

$$\mathbb{E}\left[\|\hat{g}^{(k)}(\hat{\theta}, \omega) - \nabla \hat{\mathcal{L}}^{(k)}(\hat{\theta})\|^2 | \hat{\theta}\right] \leq \sigma^2$$

for any $k \in \mathbb{N}$ and $\hat{\theta} \in \mathbb{R}^{D_k}$.

Using these two assumptions, following Bauschke et al. (2019); Elshiaty & Petra (2025) and extending the coarse descent to the stochastic case, we can prove a decrease property.

Theorem 1. *Let Assumptions 1, 2 and 3 be satisfied and $(\theta^{(k)})$ be the sequence generated by Algorithm 1 with exact gradients on the fine level. If the step size τ is chosen such that $\tau < \frac{1}{L}$ and the coarse step sizes such that $\hat{\tau}^{i,k} \leq \frac{1}{4L}$then*

$$\mathbb{E}\left[\mathcal{L}(\theta^{(k)}) - \mathcal{L}^*\right] \leq (1-r)^k(\mathcal{L}(\theta^{(0)}) - \mathcal{L}^*) + \sum_{i=0}^{k-1}(1-r)^{k-i}\hat{\rho}_i,$$

where $r = \frac{\eta\lambda(\tau)}{\tau}$ and

$$\hat{\rho}_i = \begin{cases} \sum\limits_{j=0}^{m-1} \frac{\delta\sigma^2}{2}\hat{\tau}^{j,i} - \frac{1}{4\hat{\tau}^{j,i}}\mathbb{E}\left[D^{\text{sym}}_{\hat{J}^{(k)}_{\delta}}(\hat{\theta}^{j+1,i}, \hat{\theta}^{j,i})\right] & \textit{if } i \textit{ triggers a coarse step}, \\ 0 & \textit{otherwise}. \end{cases}$$

A direct consequence is the convergence in expectation of the Algorithm 1 if the coarse step-sizes converge to zero.

Corollary 1. *Let the assumptions from Theorem 1 hold. Define $\hat{\tau}^{(k)} := \max_{i=0,\dots,m-1}\hat{\tau}^{i,k}$, the maximum stepsize on the coarse level in iteration k. If $\tau^{(k)}$ converges to zero, then*

$$\lim_{k\to\infty}\mathbb{E}\left[\mathcal{L}(\theta^{(k)})\right] = \mathcal{L}^*.$$

We note that ultimately, we would like to prove the same results when also using stochastic gradients on the fine level. However, this appears to be out of reach unless one is willing to assume (strong) convexity (Bungert et al., 2022), increased smoothness (Zhang & He, 2018), or perform variance reduction (Li et al., 2022). The only results for standard mirror descent in this direction that we are aware of are due to Fatkhullin & He (2024), who show that, in expectation, the best iterate is within one mirror-descent step of an ε-optimal point once sufficiently many steps have been taken and the step-sizes are chosen correctly depending on ε. Their analysis relies on a differentiable mirror map, weak relative convexity of the objective, and a different variant of the Polyak–Łojasiewicz inequality.

In Appendix B, we examine convergence in a slightly different setting. Specifically, when the loss function has a finite-sum structure, we show that using a mini-batch approximation as the coarse loss, together with an added linear correction term, makes the multilevel algorithm closely related to standard variance-reduction methods. This connection enables us to establish convergence of the resulting algorithm.

5 Numerical Experiments

All gradient estimators in Algorithm 1 are computed using mini-batch approximations. It is also important to note that the ordering of coarse and fine updates is flexible. For instance, one may perform a single fine update – using a mini-batch to update the fine model – followed by m coarse updates. Alternatively, switching can occur at the epoch level, where fine updates are carried out for an entire epoch, and then coarse updates are applied for m consecutive epochs.

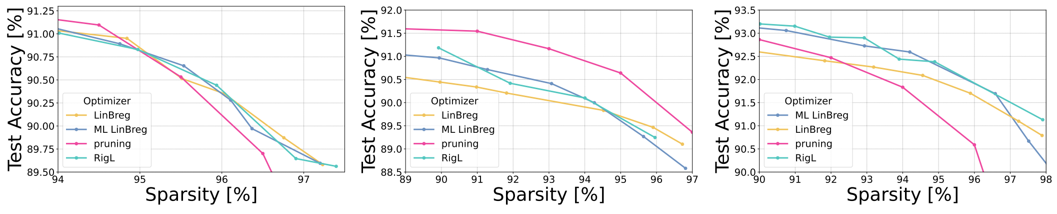
As mentioned above, we choose the linear map that selects all non-zero parameters in the current state as the restriction operator. The sparsity of a model is defined to be the percentage of the models' parameters that are zero. We consider promoting unstructured sparsity with standard ℓ_1-regularization, that is, $J = \lambda\|\cdot\|_1$, and initialize all networks with a sparsity of 99%. Due to our considerations regarding the initialization (see Appendix D), we would need to multiply every sparsified group of weights by $\frac{1}{\sqrt{0.01}} = 10$. However, we instead use a multiplication of 5 as this leads to better results. For training we use the step-wise switching strategy between fine and coarse model rather than switching epoch-wise. To simplify the algorithm, we use the coarse model at every stage, and perform a fine level update every 100 training steps, i.e. take $m = 99$ in Algorithm 1. For all experiments, we initialize the learning rate at 0.1 and apply a cosine annealing scheduler.

For ablation studies investigating the effects of the hyperparameters λ and m for medium and larger scale experiments we refer to Appendix D

CIFAR10 training In order to evaluate our proposed training algorithm, we train different neural network architectures on the CIFAR10 dataset, containing 60,000 32-by-32 color images in 10 different classes. The objective is to construct sparse models that exhibit only marginal reductions in test accuracy.

The hyperparameter λ plays a significant role in determining the characteristics of the trained network. As expected, stronger regularization – i.e., larger values of λ – leads to increased sparsity in the resulting model, potentially at the cost of reduced accuracy (see Figure 4a in the Appendix). Therefore, our aim is to find a trade-off between sparsity and accuracy.

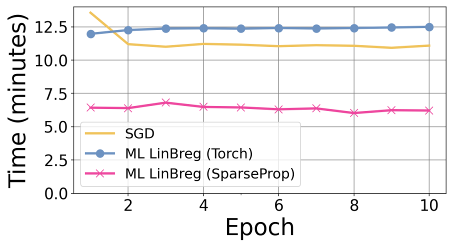
We train models using Algorithm 1 with different choices of the regularization parameter λ across multiple random seeds to produce models with different levels of sparsity. As baselines, we include dense models trained with stochastic gradient descent (SGD), LinBreg, RigL (Evci et al., 2020) and pruned versions of the SGD-trained models. The latter were pruned to match the sparsity levels achieved by LinBreg and then fine-tuned. For a fair comparison, the pruned models were trained for only 180 epochs before undergoing an additional 20 epochs of fine-tuning, whereas the baseline methods were trained for the full 200 epochs. Thus, the overall training budget is aligned at 200 epochs across all methods. For models trained with RigL, we used the proposed Erdős–Rényi-Kernel (ERK) initialization method, as we found it yielded the best results. While our proposed method is robust to different initialization schemes, RigL relies on ERK initialization to achieve competitive results (see Table 2 in Appendix D). We visualize our results in Figure 1 plotting the mean values over multiple random seeds of the test accuracy at the corresponding sparsity levels for different regularizers, for LinBreg, our proposed algorithm, RigL and the pruned and fine-tuned models. We observe that, across all the architectures considered, our method achieves results comparable to those of RigL. Moreover, we emphasize that, while achieving results comparable to LinBreg and pruning on ResNet18, our method requires substantially less gradient information. For the VGG16 architecture, the pruned architectures outperform the other approaches, however, at the cost of requiring training of a full architecture. Moreover, for WideResNet28-10, our algorithm outperforms both LinBreg and the pruning approach by producing models that are sparser and equally accurate. Compared to pruning, our method (and to some extent also LinBreg) has the advantage of not requiring a dense model to be trained.



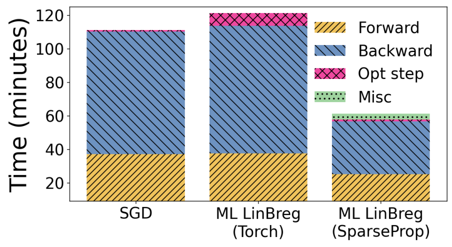
(a) ResNet18 (b) VGG16 (c) WideResNet28-10

Figure 1: Accuracy and sparsity for different regularizers with LinBreg and ML LinBreg as well as RigL and a pruning baseline on CIFAR10 for different architectures.

Exploiting sparsity in practice – especially unstructured sparsity as focused on in these experiments – is notoriously difficult, especially on GPUs. To this end, it is common to report theoretical computational savings – see Appendix C for how our method compares to standard training with SGD and LinBreg in this regard. Furthermore, implementations of sparse training algorithms often calculate dense gradients and only simulate sparse training via masking hooks on forward and backward passes. To illustrate that ML LinBreg can lead to real-time savings, we use SparseProp (Nikdan et al., 2023), a package which has implementations of linear and convolutional layers that exploit unstructured sparsity - with the caveat that it is only applicable on CPUs. We train ResNet18 on an AMD Ryzen 5 4500U CPU for 10 epochs where, after each fine update, we replace Torch linear and convolutional layers with SparseProp analogues if doing so leads to faster forward and backward passes. We report the computational performance of SGD and the Torch and SparseProp implementations of ML LinBreg in Figure 2. Utilizing SparseProp reduces training time by 49% compared to the Torch implementation, with time spent for forward and backward passes reduced by 32% and 58%, respectively. In contrast, the Torch variant of ML LinBreg takes slightly longer than SGD since full gradients are still calculated under the hood and the optimizer step has added complexity.



(a) Per-epoch training time on CPU

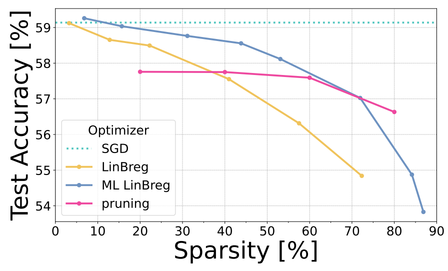


(b) Breakdown of total training time on CPU

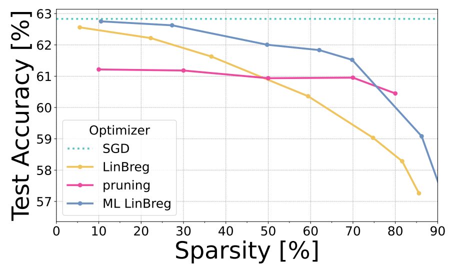
Figure 2: Timing breakdown for training ResNet18 for 10 epochs on an AMD Ryzen 5 4500U CPU when employing SparseProp modules to exploit unstructured sparsity. The sparsity induced by ML LinBreg can lead to real-time computational savings.

Further CIFAR10 experiments are reported in Appendix D, including an ablation study investigating the effect of the duration of the coarse period and the regularization strength (see Figure 4).

TinyImageNet training We further evaluate our approach on the TinyImageNet dataset, containing 100,000 64-by-64 color images in 200 different classes, using two different architectures. As in the CIFAR10 experiments, we train models with the regularizer $J = \lambda\|\cdot\|_1$ for different values of λ across multiple random seeds. The results are summarized in Figure 3, where we present the mean values of the achieved test accuracies located at the sparsity of the models. For comparison, we also include a dense baseline trained with standard SGD. We additionally provide the results that can be achieved by pruning and fine-tuning the dense SGD models to different levels of sparsity. Our findings show that the proposed multilevel method consistently outperforms the standard LinBreg algorithm by producing models that are sparser while maintaining, or even improving, test accuracy. Compared to the pruning approach, we achieve comparable or better test accuracy across all sparsity levels.



(a) ResNet18



(b) WideResNet28-10

Figure 3: Accuracy and sparsity for different regularizers with LinBreg and ML LinBreg as well as SGD and pruning baselines on TinyImageNet.

6 Conclusion and Outlook

We proposed a multilevel framework for linearized Bregman iterations in the context of sparse training. We established convergence of the function values for our method assuming a PL-type condition for the objective, and we demonstrated its effectiveness in producing sparse yet accurate models for image classification tasks. An interesting direction for future work is to analyze the convergence of our algorithm in a fully stochastic setting, using gradient estimators in place of exact gradients for both the lower and the higher level problem. Also, relaxing the PL-condition to a local condition of Kurdyka–Łojasiewicz type is an interesting but challenging follow-up problem. Moreover, sparsity-informed training implementations of our method have the potential to reduce training time and resource requirements.

Reproducibility Statement

The code used to produce the results will be released at a GitHub repository that will be made available after blind review.

References

Navid Azizan, Sahin Lale, and Babak Hassibi. Stochastic mirror descent on overparameterized nonlinear models. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7717–7727, 2021. doi: 10.1109/TNNLS.2021.3087480.

Markus Bachmayr and Martin Burger. Iterative total variation schemes for nonlinear inverse problems. *Inverse Problems*, 25(10):105004, 2009. doi: 10.1088/0266-5611/25/10/105004.

Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Publishing Company, Incorporated, 1st edition, 2011. ISBN 1441994661.

Heinz H. Bauschke, Jérôme Bolte, Jiawei Chen, Marc Teboulle, and Xianfu Wang. On linear convergence of non-Euclidean gradient methods without strong convexity and Lipschitz gradient continuity. *J. Optim. Theory Appl.*, 182(3):1068–1087, 2019. doi: 10.1007/s10957-019-01516-9.

Amir Beck. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017. doi: 10.1137/1.9781611974997.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003. doi: 10.1016/S0167-6377(02)00231-6.

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BJ_wN01C-`.

Martin Benning, Marta M. Betcke, Matthias J. Ehrhardt, and Carola-Bibiane Schönlieb. Choose your path wisely: gradient descent in a Bregman distance framework. *SIAM J. Imaging Sci.*, 14 (2):814–843, 2021. doi: 10.1137/20M1357500.

Vanessa Braglia, Alena Kopanicáková, and Rolf Krause. A multilevel approach to training. *ICML 2020 Workshop: Beyond First Order Methods in Machine Learning*, 2020. doi: 10.48550/arXiv.2006.15602.

Leon Bungert, Tim Roith, Daniel Tenbrinck, and Martin Burger. Neural architecture search via bregman iterations. *arXiv*, 2021. doi: 10.48550/arXiv.2106.02479.

Leon Bungert, Tim Roith, Daniel Tenbrinck, and Martin Burger. A Bregman learning framework for sparse neural networks. *J. Mach. Learn. Res.*, 23:Paper No. [192], 43, 2022. ISSN 1532-4435,1533-7928.

Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv*, 2019. doi: 10.48550/arXiv.1907.04840.

Payal Dhar. The carbon impact of artificial intelligence. *Nature Machine Intelligence*, 2:423–425, 08 2020. doi: 10.1038/s42256-020-0219-9.

Yara Elshiaty and Stefania Petra. Multilevel bregman proximal gradient descent. *arXiv*, 2025. doi: 10.48550/arXiv.2506.03950.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ByME42AqK7`.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2943–2952. PMLR, 2020. URL `https://proceedings.mlr.press/v119/evci20a.html`.

Ilyas Fatkhullin and Niao He. Taming nonconvex stochastic mirror descent with general bregman divergence. In *International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 3493–3501. PMLR, 2024. URL `https://proceedings.mlr.press/v238/fatkhullin24a.html`.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Lisa Gaedke-Merzhäuser, Alena Kopaničáková, and Rolf Krause. Multilevel minimization for deep residual networks. In *FGS'2019—19th French-German-Swiss conference on Optimization*, volume 71 of *ESAIM Proc. Surveys*, pp. 131–144. EDP Sci., Les Ulis, 2021. doi: 10.1051/proc/202171131.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015. doi: 10.1109/ICCV.2015.123.

Tjeerd Jan Heeringa, Christoph Brune, and Mengwu Guo. Sparsifying dimensionality reduction of pde solution data with bregman learning. *SIAM Journal on Scientific Computing*, 47(5):C1033–C1058, 2025a. doi: 10.1137/24M1684566.

Tjeerd Jan Heeringa, Tim Roith, Christoph Brune, and Martin Burger. Learning a sparse representation of barron functions with the inverse scale space flow. *Journal of Machine Learning*, 4(1): 48–88, 2025b. doi: 10.4208/jml.240123.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021. URL `http://jmlr.org/papers/v22/21-0366.html`.

Vahan Hovhannisyan, Panos Parpas, and Stefanos Zafeiriou. MAGMA: Multilevel accelerated gradient mirror descent algorithm for large-scale convex composite minimization. *SIAM J. Imaging Sci.*, 9(4):1829–1857, 2016. ISSN 1936-4954. doi: 10.1137/15M104013X.

Chendi Huang, Xinwei Sun, Jiechao Xiong, and Yuan Yao. Split LBI: An iterative regularization path with structural sparsity. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, volume 29 of *NIPS'16*, pp. 3377–3385, 2016. ISBN 9781510838819.

Tom Jacobs and Rebekka Burkholz. Mask in the Mirror: Implicit Sparsification. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=U47ymTS3ut`.

Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, volume 33 of *NIPS '20*, pp. 20744–20754, 2020. ISBN 9781713829546.

Jie Ji, Gen Li, Jingjing Fu, Fatemeh Afghah, Linke Guo, Xiaoyong Yuan, and Xiaolong Ma. A Single-Step, Sharpness-Aware Minimization is All You Need to Achieve Efficient and Accurate Sparse Training. 37:44269–44290, 2024a. doi: 10.52202/079017-1405.

Jie Ji, Gen Li, Lu Yin, Minghai Qin, Geng Yuan, Linke Guo, Shiwei Liu, and Xiaolong Ma. Advancing dynamic sparse training by exploring optimization opportunities. In *Forty-first International Conference on Machine Learning*, 2024b.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper_files/paper/2013/file/ac1dd209cbcc5e5d1c6e28598e8cbbe8-Paper.pdf`.

Alena Kopaničáková and Rolf Krause. Globally convergent multilevel training of deep residual networks. *SIAM Journal on Scientific Computing*, 45(3):S254–S280, 2023. doi: 10.1137/21M1434076.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International conference on machine learning*, pp. 5544–5555. PMLR, 2020.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Gen Li, Lu Yin, Jie Ji, Wei Niu, Minghai Qin, Bin Ren, Linke Guo, Shiwei Liu, and Xiaolong Ma. NeurRev: Train Better Sparse Neural Network Practically via Neuron Revitalization. 2023. URL `https://openreview.net/forum?id=60lNoatp7u`.

Wenjie Li, Zhanyu Wang, Yichen Zhang, and Guang Cheng. Variance reduction on general adaptive stochastic mirror descent. *Machine Learning*, 111(12):4639–4677, 2022.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. *Advances in Neural Information Processing Systems*, 34:9908–9922, 2021.

Filippo Marini, Margherita Porcelli, and Elisa Riccietti. A multilevel stochastic regularized first-order method with application to training. *arXiv*, 2024. doi: 10.48550/arXiv.2412.11630.

Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. 14 - evolving deep neural networks. In Robert Kozma, Cesare Alippi, Yoonsuck Choe, and Francesco Carlo Morabito (eds.), *Artificial Intelligence in the Age of Neural Networks and Brain Computing (Second Edition)*, pp. 269–287. Academic Press, second edition edition, 2024. ISBN 978-0-323-96104-2. doi: 10.1016/B978-0-323-96104-2.00002-6.

Decebal Mocanu, Elena Mocanu, Peter Stone, Phuong Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9, 06 2018. doi: 10.1038/s41467-018-04316-3.

Mohd Halim Mohd Noor and Ayokunle Olalekan Ige. A survey on state-of-the-art deep learning applications and challenges. *Engineering Applications of Artificial Intelligence*, 159:111225, 2025. ISSN 0952-1976. doi: 10.1016/j.engappai.2025.111225.

Sofia Mosci, Lorenzo Rosasco, Matteo Santoro, Alessandro Verri, and Silvia Villa. Solving structured sparsity regularization with proximal methods. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 418–433, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15883-4.

Stephen G. Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14(1-2):99–116, 2000. doi: 10.1080/10556780008805795.

Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009. doi: 10.1007/s10107-007-0149-x.

Mahdi Nikdan, Tommaso Pegolotti, Eugenia Iofinova, Eldar Kurtic, and Dan Alistarh. SparseProp: Efficient Sparse Backpropagation for Faster Training of Neural Networks at the Edge. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 26215–26227. PMLR, 2023.

Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2): 460–489, 2005.

Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. AC/DC: Alternating compressed/decompressed training of deep neural networks. *Advances in neural information processing systems*, 34:8557–8570, 2021.

Tejas Pote, Muhammad Athar Ganaie, Atif Hassan, and Swanand Khare. Dynamic forward and backward sparse training (DFBST): Accelerated deep learning through completely sparse training schedule. In Emtiyaz Khan and Mehmet Gonen (eds.), *Proceedings of The 14th Asian Conference on Machine Learning*, volume 189 of *Proceedings of Machine Learning Research*, pp. 848–863. PMLR, 12–14 Dec 2023. URL `https://proceedings.mlr.press/v189/pote23a.html`.

Lorenzo Rosasco, Silvia Villa, and Bundefinedng Công Vundefined. Convergence of stochastic proximal gradient algorithm. *Appl. Math. Optim.*, 82(3):891–917, December 2020. ISSN 0095-4616. doi: 10.1007/s00245-019-09617-7.

Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

Sidak Pal Singh and Dan Alistarh. WoodFisher: Efficient Second-Order Approximation for Neural Network Compression. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18098–18109. Curran Associates, Inc., 2020.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 00359246.

Ferdinand Vanmaele, Yara Elshiaty, and Stefania Petra. Multilevel optimization: Geometric coarse models and convergence analysis. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 95–108. Springer, 2025.

Xiaoyu Wang and Martin Benning. Lifted bregman training of neural networks. *Journal of Machine Learning Research*, 24(232):1–51, 2023.

Zaiwen Wen and Donald Goldfarb. A line search multigrid method for large-scale nonlinear optimization. *SIAM J. Optim.*, 20(3):1478–1503, 2009. doi: 10.1137/08071524X.

Lu Yin, Gen Li, Meng Fang, Li Shen, Tianjin Huang, Zhangyang Wang, Vlado Menkovski, Xiaolong Ma, Mykola Pechenizkiy, Shiwei Liu, et al. Dynamic sparsity is channel-level sparsity learner. *Advances in Neural Information Processing Systems*, 36:67993–68012, 2023.

Wotao Yin, Stanley Osher, Donald Goldfarb, and Jerome Darbon. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging sciences*, 1 (1):143–168, 2008.

Siqi Zhang and Niao He. On the convergence rate of stochastic mirror descent for nonsmooth nonconvex optimization. *arXiv*, 2018. doi: 10.48550/arXiv.1806.04781.

A Theoretical proofs

A first very important observation is that $\hat{J}_\delta^{(k)}$ coincides with $\hat{J} \coloneqq J_\delta(\theta^{(k)} + P^{(k)}(\cdot - \hat{\theta}^{0,k}))$ up to a constant depending on $\theta^{(k)}$. To see this, note that

$$(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k}))_{(g)} = \begin{cases} \hat{\theta}_{(i)}, & \text{if } g = r^{(k)}(i), \\ (\theta^{(k)})_{(g)}, & \text{otherwise} \end{cases} \tag{12}$$

due to the definitions of $P^{(k)}$ and $\hat{\theta}^{0,k}$. Using this, it is straight-forward to compute

$$\begin{aligned} \hat{J}(\hat{\theta}) &= \frac{1}{2\delta}\|\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})\|^2 + J(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})) \\ &= \frac{1}{2\delta}\|\hat{\theta}\|^2 + \frac{1}{2\delta}\sum_{g \in X}\|(\theta^{(k)})_{(g)}\|^2 + \sum_{i=1}^{G^{(k)}} J_{r^{(k)}(i)}(\hat{\theta}_{(i)}) + \sum_{g \in X} J_g((\theta^{(k)})_{(g)}) \\ &= \hat{J}_\delta^{(k)}(\hat{\theta}) + \text{const}(\theta^{(k)}), \end{aligned}$$

where we use the abbreviation

$$X \coloneqq \{1, 2, \dots, G\} \setminus r^{(k)}(\{1, 2, \dots, G^{(k)}\})$$

to denote all groups that are not selected by $R^{(k)}$. Therefore, $\hat{J}_\delta^{(k)}$ and $\hat{J}$ share the same subdifferential and also induce the same Bregman divergence. We can make use of this fact to prove Proposition 1.

Proof of Proposition 1. Due to the sum and chain rule for subdifferentials,

$$\partial\hat{J}(\hat{\theta}) = \frac{1}{\delta}\hat{\theta} + (P^{(k)})^T \partial J(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})).$$

for $\hat{\theta} \in \mathbb{R}^{D_k}$ such that $J(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})) < \infty$ and $\partial J(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})) \neq \emptyset$. Consequently, since we have already established that $\hat{J}_\delta^{(k)}$ and $\hat{J}$ share the same subdifferential, for $\hat{\theta} = \hat{\theta}^{0,k}$, we have

$$\begin{aligned} \partial\hat{J}_\delta^{(k)}(\hat{\theta}^{0,k}) &= \partial\hat{J}(\hat{\theta}^{0,k}) \\ &= \frac{1}{\delta}\hat{\theta}^{0,k} + R^{(k)}\partial J(\theta^{(k)}) \\ &= R^{(k)}\partial J_\delta(\theta^{(k)}). \end{aligned}$$

Obviously, $\hat{v}^{0,k} = R^{(k)}v^{(k)}$ belongs to the latter, concluding the proof. $\square$

Proof of Proposition 2. Using (12) with $\hat{\theta} = \hat{\theta}^{m,k}$ together with (6), it is easy to see that

$$\begin{aligned} \partial_{(g)} J_\delta(\tilde{\theta}^{(k+1)}) &= \frac{1}{\delta}\tilde{\theta}^{(k+1)}_{(g)} + \partial J_g((\tilde{\theta}^{(k+1)})_{(g)}) \\ &= \begin{cases} \partial_{(i)}\hat{J}_\delta^{(k)}((\hat{\theta}^{m,k})_{(i)}), & \text{if } g = r^{(k)}(i), \\ \partial_{(g)} J_\delta(\theta^{(k)}), & \text{otherwise.} \end{cases} \end{aligned}$$

Since by definition

$$\tilde{v}^{(k+1)} = \begin{cases} (\hat{v}^{m,k})_{(i)}, & \text{if } g = r^{(k)}(i), \\ (v^{(k)})_{(g)}, & \text{otherwise,} \end{cases}$$

together with $\hat{v}^{m,k} \in \partial\hat{J_\delta}^{(k)}(\hat{\theta}^{m,k})$ and $v^{(k)} \in \partial J_\delta(\theta^{(k)})$, we obtain overall that $\tilde{v}^{(k+1)} \in \partial J_\delta(\tilde{\theta}^{(k+1)})$. $\square$

As a next step, we prove that the coarse objective $\hat{\mathcal{L}}^{(k)}$ is L-smooth relative to the coarse regularizer $\hat{J}_\delta^{(k)}$.

Lemma 1. *Let Assumption 1 be satisfied. Then, the coarse objective $\hat{\mathcal{L}}^{(k)}$ defined in* (8) *is L-smooth relative to $\hat{J}_\delta^{(k)}$.*

Proof. Let $\hat{\theta} \in \operatorname{dom} \partial J$ with $\hat{v} \in \partial \hat{J}(\hat{\theta})$ and $\hat{\theta}' \in \mathbb{R}^{D_k}$. We define

$$\tilde{v} := v^{(k)} + P^{(k)}(\hat{v} - \hat{v}^{0,k}),$$
$$\tilde{\theta} := \theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k}),$$

and note that this leads to $\tilde{v} \in \partial J_\delta(\tilde{\theta})$ following from the same argumentation as in the proof of Proposition 2. In particular, $\tilde{\theta} \in \operatorname{dom} \partial J$. Hence, from the L-smoothness of $\mathcal{L}$ relative to J_δ, we deduce

$$\hat{\mathcal{L}}^{(k)}(\hat{\theta}') - \hat{\mathcal{L}}^{(k)}(\hat{\theta}) - \langle \nabla \mathcal{L}(\tilde{\theta}), P^{(k)}(\hat{\theta}' - \hat{\theta}) \rangle \leq L(\hat{J}(\hat{\theta}') - \hat{J}(\hat{\theta}) - \langle \tilde{v}, P^{(k)}(\hat{\theta}' - \hat{\theta}) \rangle)$$

and using $(P^{(k)})^T \nabla \mathcal{L}(\tilde{\theta}) = \nabla \hat{\mathcal{L}}^{(k)}(\hat{\theta})$, we obtain

$$\hat{\mathcal{L}}^{(k)}(\hat{\theta}') - \hat{\mathcal{L}}^{(k)}(\hat{\theta}) - \langle \nabla \hat{\mathcal{L}}^{(k)}(\hat{\theta}), \hat{\theta}' - \hat{\theta} \rangle \leq L(\hat{J}(\hat{\theta}') - \hat{J}(\hat{\theta}) - \langle (P^{(k)})^T \tilde{v}, \hat{\theta}' - \hat{\theta} \rangle).$$

Due to the definition of $\tilde{v}$, we have $(P^{(k)})^T \tilde{v} = R^{(k)} \tilde{v} = \hat{v}$, since $R^{(k)} P^{(k)} = \mathrm{id}$. Consequently,

$$\hat{\mathcal{L}}^{(k)}(\hat{\theta}') - \hat{\mathcal{L}}^{(k)}(\hat{\theta}) - \langle \nabla \hat{\mathcal{L}}^{(k)}(\hat{\theta}), \hat{\theta}' - \hat{\theta} \rangle \leq L D_{\hat{J}}^{\hat{v}}(\hat{\theta}', \hat{\theta}),$$

meaning that $\hat{\mathcal{L}}^{(k)}$ is L-smooth relative to $\hat{J}$. Since we have already established that $\hat{J}$ and $\hat{J}_\delta^{(k)}$ induce the same Bregman divergence, relative smoothness with respect to these two functions is equivalent. □

For the proof of our main result Theorem 1 we require two more lemmas, the proofs of which work very similar to the corresponding results in Bauschke et al. (2019); Elshiaty & Petra (2025) with the key differences being that we utilize subgradients instead of classical gradients of the regularizer J_δ and that we use stochastic gradients on the coarse level.

The first lemma asserts a linear convergence rate for the linearized Bregman iterations (4) without a multilevel component.

Lemma 2. *Let Assumption 1 and Assumption 2 be satisfied. Moreover, for $\theta \in \operatorname{dom}(\partial J), v \in \partial J_\delta(\theta)$ and $0 < \tau < \frac{1}{L}$, let $v_\tau^+ := v - \tau \nabla \mathcal{L}(\theta)$ and $\theta_\tau^+ := \operatorname{prox}_{\delta J}(\delta v_\tau^+)$. Then,*

$$\mathcal{L}(\theta_\tau^+) - \mathcal{L}^* \leq \left(1 - \eta \frac{\lambda(\tau)}{\tau}\right)(\mathcal{L}(\theta) - \mathcal{L}^*).$$

Proof. Using Assumption 1 the defintion of v_τ^+ and Assumption 2 together with the definition of the symmetrized Bregman divergence

$$D_{J_\delta}^{\mathrm{sym}}(\theta, \tilde{\theta}) := D_{J_\delta}^{v}(\tilde{\theta}, \theta) + D_{J_\delta}^{\tilde{v}}(\theta, \tilde{\theta})$$
$$= \langle v - \tilde{v}, \theta - \tilde{\theta} \rangle,$$

for $\theta, \tilde{\theta} \in \operatorname{dom}\partial J$ and $v \in \partial J_\delta(\theta), \tilde{v} \in \partial J_\delta(\tilde{\theta})$, we obtain

$$\begin{aligned}
\mathcal{L}(\theta_\tau^+) &\leq \mathcal{L}(\theta) + \langle \nabla \mathcal{L}(\theta), \theta_\tau^+ - \theta \rangle + L D_{J_\delta}^{v}(\theta_\tau^+, \theta) \\
&= \mathcal{L}(\theta) - \frac{1}{\tau} D_{J_\delta}^{\mathrm{sym}}(\theta_\tau^+, \theta) + L D_{J_\delta}^{v}(\theta_\tau^+, \theta) \\
&\leq \mathcal{L}(\theta) - \frac{1}{\tau} D_{J_\delta}^{v_\tau^+}(\theta, \theta_\tau^+) \\
&\leq \mathcal{L}(\theta) - \frac{\lambda(\tau)}{\tau} D_{J_\delta}^{v_1^+}(\theta, \theta_1^+) \\
&\leq \mathcal{L}(\theta) - \eta \frac{\lambda(\tau)}{\tau}(\mathcal{L}(\theta) - \mathcal{L}^*).
\end{aligned}$$

Subtracting $\mathcal{L}^*$ from both sides yields the desired estimate. □

The second lemma investigates the decay of the function value if a coarse update is taken. The key observation to prove this result is that our coarse objective $\hat{\mathcal{L}}^{(k)}$ is L-smooth relative to the coarse regularizer $\hat{J}_\delta^{(k)}$. This property follows from Assumption 1.

Lemma 3. *Let Assumption 1 and Assumption 3 be satisfied. If iteration step k triggers a coarse update and the coarse step size is chosen such that $\hat{\tau}^{i,k} \le \frac{1}{4L}$, then*

$$\mathbb{E}\left[\mathcal{L}(\tilde{\theta}^{(k+1)})\right] \le \mathbb{E}\left[\mathcal{L}(\theta^{(k)})\right] - \sum_{j=0}^{m-1} \frac{1}{4\hat{\tau}^{j,k}} \mathbb{E}\left[D^{\mathrm{sym}}_{\hat{J}_\delta^{(k)}}(\hat{\theta}^{j+1,k}, \hat{\theta}^{j,k})\right] + \frac{\delta\sigma^2}{2}\sum_{j=0}^{m-1}\hat{\tau}^{j,k}.$$

Proof. By Lemma 1, $\hat{\mathcal{L}}^{(k)}$ is L-smooth relative to $\hat{J}_\delta^{(k)}$, we obtain

$$\begin{aligned}
\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i+1,k}) &\le \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) + \langle \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}), \hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\rangle + LD^{\hat{v}^{i,k}}_{\hat{J}_\delta^{(k)}}(\hat{\theta}^{i+1,k}, \hat{\theta}^{i,k}) \\
&= \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) + \langle \hat{g}^{(k)}(\hat{\theta}^{i,k}, \omega), \hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\rangle \\
&\quad + \langle \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) - \hat{g}^{(k)}(\hat{\theta}^{i,k}, \omega), \hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\rangle + LD^{\hat{v}^{i,k}}_{\hat{J}_\delta^{(k)}}(\hat{\theta}^{i+1,k}, \hat{\theta}^{i,k}) \\
&\le \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) - \frac{1}{\hat{\tau}^{i,k}} D^{\mathrm{sym}}_{\hat{J}_\delta^{(k)}}(\hat{\theta}^{i+1,k}, \hat{\theta}^{i,k}) + \frac{\varepsilon\hat{\tau}^{i,k}}{2}\|\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) - \hat{g}^{(k)}(\hat{\theta}^{i,k}, \omega)\|^2 \\
&\quad + \frac{1}{2\varepsilon\hat{\tau}^{i,k}}\|\hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\|^2 + LD^{\hat{v}^{i,k}}_{\hat{J}_\delta^{(k)}}(\hat{\theta}^{i+1,k}, \hat{\theta}^{i,k}) \\
&\le \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) - \frac{2\varepsilon - 2L\varepsilon\hat{\tau}^{i,k} - \delta}{2\varepsilon\hat{\tau}^{i,k}} D^{\mathrm{sym}}_{\hat{J}_\delta^{(k)}}(\hat{\theta}^{i+1,k}, \hat{\theta}^{i,k}) \\
&\quad + \frac{\varepsilon\hat{\tau}^{i,k}}{2}\|\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) - \hat{g}^{(k)}(\hat{\theta}^{i,k}, \omega)\|^2
\end{aligned}$$

for any $\varepsilon > 0$ using Young's inequality. Consequently, taking the expectation conditioned on $\hat{\theta}^{i,k}$, that we denote by $\mathbb{E}_{i,k}$ for simplicity, and letting $\varepsilon = \delta$, we arrive at

$$\mathbb{E}_{i,k}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i+1,k})\right] \le \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) - \frac{\delta - 2L\delta\hat{\tau}^{i,k}}{2\delta\hat{\tau}^{i,k}}\mathbb{E}_{i,k}\left[D^{\mathrm{sym}}_{\hat{J}_\delta^{(k)}}(\hat{\theta}^{i+1,k}, \hat{\theta}^{i,k})\right] + \frac{\delta\hat{\tau}^{i,k}}{2}\sigma^2.$$

Thus, for $\hat{\tau}^{i,k} \le \frac{1}{4L}$, taking the full expectation and using the tower property, iterating this inequality leads to

$$\mathbb{E}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i+1,k})\right] \le \mathbb{E}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k})\right] - \sum_{j=0}^{i}\frac{1}{4\hat{\tau}^{j,k}}\mathbb{E}\left[D^{\mathrm{sym}}_{\hat{J}_\delta^{(k)}}(\hat{\theta}^{j+1,k}, \hat{\theta}^{j,k})\right] + \sum_{j=0}^{i}\frac{\delta\sigma^2}{2}\hat{\tau}^{j,k}.$$

In particular,

$$\mathbb{E}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{m,k})\right] \le \mathbb{E}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k})\right] - \sum_{j=0}^{m-1}\frac{1}{4\hat{\tau}^{j,k}}\mathbb{E}\left[D^{\mathrm{sym}}_{\hat{J}_\delta^{(k)}}(\hat{\theta}^{j+1,k}, \hat{\theta}^{j,k})\right] + \sum_{j=0}^{m-1}\frac{\delta\sigma^2}{2}\hat{\tau}^{j,k}.$$

Since $\mathcal{L}(\tilde{\theta}^{(k+1)}) = \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{m,k})$ and $\mathcal{L}(\theta^{(k)}) = \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k})$, this yields the desired estimate. □

Combining the previous results we arrive at the proof of our main result.

Proof of Theorem 1. The result follows by combining Lemmas 2 and 3 in the appendix and iterating. □

Proof of Corollary 1. To prove this result, we consider the worst-case scenario in Theorem 1, that is

$$\hat{\rho}_i = \sum_{j=0}^{m-1}\frac{\delta\sigma^2}{2}\hat{\tau}^{j,i}$$

for any $i \in \mathbb{N}$. This means that we assume that a coarse step is taken at each iteration and that the potential benefit in the loss descent with the symmetrized Bregman divergence vanishes. In this case,

$$\sum_{i=0}^{k-1}(1-r)^{k-i}\hat{\rho}_i = \sum_{i=0}^{k-1}(1-r)^{k-i}\sum_{j=0}^{m-1}\frac{\delta\sigma^2}{2}\hat{\tau}^{j,i} \leq C\sum_{i=0}^{k-1}(1-r)^{k-i}\hat{\tau}^{(i)} = C\sum_{i=0}^{k-1}(1-r)^{i}\hat{\tau}^{(k-i)}$$

for some positive constant C depending on m, δ and σ^2. We prove that the sum on the right-hand side converges to zero as k approaches infinity. Let $\varepsilon > 0$ be arbitrary and define $M := \sup_{i\in\mathbb{N}} \hat{\tau}^{(i)} < \infty$. Due to the convergence of the geometric series, we can choose $J \in \mathbb{N}$ such that

$$\sum_{i=J+1}^{\infty}(1-r)^i < \frac{\varepsilon}{2M}.$$

Consequently, for $k > J$, we have

$$\sum_{i=J+1}^{k}(1-r)^i\hat{\tau}^{(k-1)} \leq M\sum_{i=J+1}^{\infty}(1-r)^i \leq \frac{\varepsilon}{2}.$$

Let additionally k be large enough such that $\sup_{i=k-J,\dots,k}\hat{\tau}^{(i)} < \frac{\varepsilon}{2(J+1)}$. Then, we can immediately conclude

$$\sum_{i=0}^{J}(1-r)^i\hat{\tau}^{(k-i)} \leq \sum_{i=0}^{J}\hat{\tau}^{(k-i)} \leq (J+1)\sup_{i=k-J,\dots,k}\hat{\tau}^{(i)} < \frac{\varepsilon}{2}$$

leading to

$$\sum_{i=0}^{k}(1-r)^i\hat{\tau}^{(k-i)} = \sum_{i=0}^{J}(1-r)^i\hat{\tau}^{(k-i)} + \sum_{i=J+1}^{k}(1-r)^i\hat{\tau}^{(k-i)} < \varepsilon.$$

Therefore, Theorem 1 yields

$$\lim_{k\to\infty}\mathbb{E}\left[\mathcal{L}(\theta^{(k)})\right] = \mathcal{L}^*.$$

□

B Convergence in the case of finite sums

In this section, we take a slightly different perspective and demonstrate that multilevel optimization methods can be interpreted as variance reduction methods. To this end, we consider a slightly modified algorithm compared to Algorithm 1. Specifically, we use a different objective function on the coarse level. Moreover, we analyze the resulting algorithm solely from a theoretical perspective without conducting numerical experiments.

For our analysis in this section, we assume that the loss $\mathcal{L}$ has a finite sum structure

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{j=1}^{n}\mathcal{L}_j(\theta)$$

and each $\mathcal{L}_j$ has a Lipschitz-continuous gradient with Lipschitz-constant $L > 0$. In this setting, we use a mini-batch approximation of the loss as the loss of the coarse level. To be precise, we sample a mini-batch $J^{i,k}$ of size $b < n$ from $[n] := \{1, \dots, n\}$ at the i-th coarse step within the k-th total cycle in Algorithm 1 and define

$$\hat{\mathcal{L}}^{i,k}(\hat{\theta}) := \frac{1}{b}\sum_{j\in J^{i,k}}\mathcal{L}_j(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})).$$

as the coarse loss. As usual in multilevel optimization algorithms (Elshiaty & Petra, 2025; Nash, 2000), we add a linear correction term to this coarse loss to ensure first-order coherence. Precisely, the function, we aim at minimizing in the i-th coarse step of the k-th total cycle is given by

$$\hat{\Psi}^{i,k}(\hat{\theta}) := \hat{\mathcal{L}}^{i,k}(\hat{\theta}) + \langle R^{(k)}\nabla\mathcal{L}(\theta^{(k)}) - \nabla\hat{\mathcal{L}}^{i,k}(\hat{\theta}^{0,k}), \hat{\theta} - \hat{\theta}^{0,k}\rangle. \tag{13}$$

Computing the gradient of this function yields

$$\nabla\hat{\Psi}^{i,k}(\hat{\theta}) = \nabla\hat{\mathcal{L}}^{i,k}(\hat{\theta}) + R^{(k)}\nabla\mathcal{L}(\theta^{(k)}) - \nabla\hat{\mathcal{L}}^{i,k}(\hat{\theta}^{0,k}).$$

In particular, if $\theta^{(k)}$ is a critical point of $\mathcal{L}$, then $\hat{\theta}^{0,k} = R^{(k)}\theta^{(k)}$ is a critical point of $\hat{\Psi}^{i,k}$ for any $i = 1, \ldots, m$. A different perspective on the gradient of the coarse objective is to see it as a variance reduction method. Making use of the special finite sum structure of both $\mathcal{L}$ and $\hat{\mathcal{L}}^{i,k}$, we compute

$$\nabla\hat{\Psi}^{i,k}(\hat{\theta}) = \frac{1}{b}\sum_{j\in J^{i,k}} R^{(k)}\nabla\mathcal{L}_j(\tilde{\theta}) + R^{(k)}\frac{1}{n}\sum_{j=0}^{n}\nabla\mathcal{L}_j(\theta^{(k)}) - \frac{1}{b}\sum_{j\in J^{i,k}} R^{(k)}\nabla\mathcal{L}_j(\theta^{(k)}).$$

where we use the abbreviation $\tilde{\theta} := \theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})$. Thus, $\nabla\hat{\Psi}^{i,k}$ is a combination of current and previous gradients, providing scope for variance reduction. The connection between multilevel optimization and variance reduction has already been investigated, e.g. in Marini et al. (2024); Braglia et al. (2020). We would in particular like to point out the similarity to stochastic variance reduced gradient (SVRG) (Johnson & Zhang, 2013), which uses the same ideas for variance reduction. As before, we denote

$$\hat{\mathcal{L}}^{(k)}(\hat{\theta}) := \mathcal{L}(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})) = \frac{1}{n}\sum_{j=1}^{n}\mathcal{L}_j(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k}))$$

as the actual (non-stochastic) objective, we would like to minimize on the coarse level. We observe that $\nabla\hat{\Psi}^{i,k}(\hat{\theta})$ is an unbiased estimator of $\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta})$ and we can prove a variance bound.

Lemma 4. *$\nabla\hat{\Psi}^{i,k}(\hat{\theta})$ is an unbiased estimator of $\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta})$, i.e.*

$$\mathbb{E}\left[\nabla\hat{\Psi}^{i,k}(\hat{\theta})\right] = \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta})$$

and satisfies the variance bound

$$\mathbb{E}\left[\|\nabla\hat{\Psi}^{i,k}(\hat{\theta}) - \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta})\|^2\right] \le \frac{L^2(n-b)}{b(n-1)}\|\hat{\theta} - \hat{\theta}^{0,k}\|^2,$$

where all expected values are taken with respect to the mini-batch sampling.

Proof. Taking the expected value with respect to drawing the mini-batch, we can easily compute

$$\mathbb{E}\left[\nabla\hat{\Psi}^{i,k}(\hat{\theta})\right] = R^{(k)}\nabla\mathcal{L}(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})) = \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}).$$

Moreover, turning towards the variance, we observe that

$$\begin{aligned}
&\mathbb{E}\left[\|\nabla\hat{\Psi}^{i,k}(\hat{\theta}) - \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta})\|^2\right] \\
&= \mathbb{E}\left[\|\nabla\hat{\mathcal{L}}^{i,k}(\hat{\theta}) - \nabla\hat{\mathcal{L}}^{i,k}(\hat{\theta}^{0,k}) - (\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}) - \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k}))\|^2\right] \\
&= \mathbb{E}\left[\|\frac{1}{b}\sum_{j\in J^{i,k}} R^{(k)}(\nabla\mathcal{L}_j(\tilde{\theta}) - \nabla\mathcal{L}_j(\theta^{(k)})) - (\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}) - \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k}))\|^2\right]
\end{aligned}$$

using the abbreviation $\tilde{\theta} := \theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})$ In the case $b = 1$, we can estimate the variance by the second moment and arrive at

$$\begin{aligned}
&\mathbb{E}_{j\sim\mathrm{Unif}(1,\ldots,n)}\left[\|R^{(k)}(\nabla\mathcal{L}_j(\tilde{\theta}) - \nabla\mathcal{L}_j(\theta^{(k)})) - (\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}) - \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k}))\|^2\right] \\
&\le \mathbb{E}_{j\sim\mathrm{Unif}(1,\ldots,n)}\left[\|R^{(k)}(\nabla\mathcal{L}_j(\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k})) - \nabla\mathcal{L}_j(\theta^{(k)}))\|^2\right] \\
&\le \|R^{(k)}\|^2 L^2\|\theta^{(k)} + P^{(k)}(\hat{\theta} - \hat{\theta}^{0,k}) - \theta^{(k)}\|^2 \\
&\le L^2\|\hat{\theta} - \hat{\theta}^{0,k}\|^2,
\end{aligned}$$

using the Lipschitz continuity of $\nabla\mathcal{L}_j$ and the fact that $\|R^{(k)}\|, \|P^{(k)}\| \leq 1$. In the case $b > 1$, since we draw without replacement, the variance scales with $\frac{n-b}{b(n-1)}$, leading to

$$\mathbb{E}\left[\|\nabla\hat{\Psi}^{i,k}(\hat{\theta}) - \nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta})\|^2\right] \leq \frac{L^2(n-b)}{b(n-1)}\|\hat{\theta} - \hat{\theta}^{0,k}\|^2.$$

□

In this setting, the linearized Bregman iteration scheme on the coarse level reads

$$\begin{aligned}\hat{g}^{i,k} &= \nabla\hat{\Psi}^{i,k},\\ \hat{v}^{i+1,k} &= \hat{v}^{i,k} - \hat{\tau}\hat{g}^{i,k},\\ \hat{\theta}^{i+1,k} &= \operatorname{prox}_{\delta\hat{J}^{(k)}}(\delta\hat{v}^{i+1,k}).\end{aligned}$$

We investigate the descent of $\hat{\mathcal{L}}^{(k)}$ during the coarse level iterations and assume that the number of steps on the coarse level m is at least 2. Note that for our setup to make sense, we require the full gradient of the loss at $\theta^{(k)}$ and therefore $m = 1$ would correspond to no stochasticity at all.

Lemma 5. *If the step size $\hat{\tau}$ is chosen such that*

$$\hat{\tau} \leq \min\left\{\frac{1}{2L\delta}, \sqrt{\frac{b(n-1)}{2\delta^2(n-b)m(m-1)}}\right\}$$

and iteration k triggers a coarse update, then

$$\mathbb{E}\left[\mathcal{L}(\tilde{\theta}^{(k+1)})\right] \leq \mathbb{E}\left[\mathcal{L}(\theta^{(k)})\right] - \frac{1}{\hat{\tau}}\sum_{j=1}^{m}\mathbb{E}\left[D^{\text{sym}}_{\hat{J}^{(k)}}(\hat{\theta}^{j,k}, \hat{\theta}^{j-1,k})\right] - \frac{1}{8\delta\hat{\tau}}\sum_{j=1}^{m}\mathbb{E}\left[\|\hat{\theta}^{j,k} - \hat{\theta}^{j-1,k}\|^2\right].$$

Proof. Using the L-smoothness of $\hat{\mathcal{L}}^{(k)}$ with respect to $\frac{1}{2}\|\cdot\|^2$, which follows from the L-smoothness with respect to $\frac{1}{2}\|\cdot\|^2$ of each $\mathcal{L}_j$ (see the descent lemma for L-smooth functions (Bauschke & Combettes, 2011; Beck, 2017) and Lemma 1), yields

$$\begin{aligned}\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i+1,k}) - \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) &\leq \langle\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}), \hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\rangle + \frac{L}{2}\|\hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\|^2\\ &= \langle\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) - \hat{g}^{i,k}, \hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\rangle + \langle\hat{g}^{i,k}, \hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\rangle + \frac{L}{2}\|\hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\|^2\\ &\leq \frac{\hat{\tau}}{2\varepsilon}\|\nabla\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) - \hat{g}^{i,k}\|^2 - \frac{1}{\hat{\tau}}D^{\text{sym}}_{\hat{J}^{(k)}_\delta}(\hat{\theta}^{i+1,k}, \hat{\theta}^{i,k}) + \frac{L\hat{\tau} + \varepsilon}{2\hat{\tau}}\|\hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\|^2.\end{aligned}$$

for all $\varepsilon > 0$. Taking the expectation conditioned on $\hat{\theta}^{i,k}$, that we denote by $\mathbb{E}_{i,k}$ for abbreviation and using the variance bound established in Lemma 4, we obtain

$$\begin{aligned}\mathbb{E}_{i,k}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i+1,k})\right] &\leq \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) + \frac{\hat{\tau}L^2(n-b)}{2\varepsilon b(n-1)}\|\hat{\theta}^{i,k} - \hat{\theta}^{0,k}\|^2 - \frac{1}{\hat{\tau}}\mathbb{E}_{i,k}\left[D^{\text{sym}}_{\hat{J}^{(k)}_\delta}(\hat{\theta}^{i+1,k}, \hat{\theta}^{i,k})\right]\\ &\quad + \frac{L\hat{\tau} + \varepsilon}{2\hat{\tau}}\mathbb{E}_{i,k}\left[\|\hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\|^2\right]\\ &\leq \hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i,k}) + \frac{\hat{\tau}L^2(n-b)}{2\varepsilon b(n-1)}i\sum_{j=1}^{i}\|\hat{\theta}^{j,k} - \hat{\theta}^{j-1,k}\|^2\\ &\quad - \frac{1}{\hat{\tau}}\mathbb{E}_{i,k}\left[D^{\text{sym}}_{\hat{J}^{(k)}_\delta}(\hat{\theta}^{i+1,k}, \hat{\theta}^{i,k})\right] + \frac{L\hat{\tau} + \varepsilon}{2\hat{\tau}}\mathbb{E}_{i,k}\left[\|\hat{\theta}^{i+1,k} - \hat{\theta}^{i,k}\|^2\right].\end{aligned}$$

We take the full expectation and iterate this inequality to arrive at

$$\begin{aligned}\mathbb{E}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{i+1,k})\right] &\leq \mathbb{E}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k})\right] + \frac{\hat{\tau}L^2(n-b)}{2\varepsilon b(n-1)}\frac{i(i+1)}{2}\sum_{j=1}^{i}\mathbb{E}\left[\|\hat{\theta}^{j,k} - \hat{\theta}^{j-1,k}\|^2\right]\\ &\quad - \frac{1}{\hat{\tau}}\sum_{j=1}^{i+1}\mathbb{E}\left[D^{\text{sym}}_{\hat{J}^{(k)}_\delta}(\hat{\theta}^{j,k}, \hat{\theta}^{j-1,k})\right] + \frac{L\hat{\tau} + \varepsilon}{2\hat{\tau}}\sum_{j=1}^{i+1}\mathbb{E}\left[\|\hat{\theta}^{j,k} - \hat{\theta}^{j-1,k}\|^2\right].\end{aligned}$$

In particular, for $i+1=m$, we obtain

$$\begin{aligned}
&\mathbb{E}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{m,k})\right] \\
&\leq \mathbb{E}\left[\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k})\right] - \frac{1}{\hat{\tau}}\sum_{j=1}^{m}\mathbb{E}\left[D^{\mathrm{sym}}_{\hat{j}^{(k)}}(\hat{\theta}^{j,k},\hat{\theta}^{j-1,k})\right] \\
&\quad - \frac{2b\varepsilon(n-1)(2-\delta\varepsilon-L\delta\hat{\tau}) - \hat{\tau}^2L^2(n-b)m(m-1)\delta}{4\varepsilon b(n-1)\delta\hat{\tau}}\sum_{j=1}^{m}\mathbb{E}\left[\|\hat{\theta}^{j,k}-\hat{\theta}^{j-1,k}\|^2\right]
\end{aligned}$$

Considering this fraction, we observe that for $\varepsilon = \frac{1}{\delta}$ and $\hat{\tau} \leq \frac{1}{2L\delta}$, we have

$$\begin{aligned}
&\frac{2b\varepsilon(n-1)(2-\delta\varepsilon-L\delta\hat{\tau}) - \hat{\tau}^2L^2(n-b)m(m-1)\delta}{4\varepsilon b(n-1)\delta\hat{\tau}} \\
&= \frac{\frac{2b}{\delta}(n-1)(1-L\delta\hat{\tau}) - \hat{\tau}^2L^2(n-b)m(m-1)\delta}{4b(n-1)\hat{\tau}} \\
&\geq \frac{\frac{b}{\delta}(n-1) - \hat{\tau}^2L^2(n-b)m(m-1)\delta}{4b(n-1)\hat{\tau}}
\end{aligned}$$

Moreover, if $\hat{\tau} \leq \sqrt{\frac{b(n-1)}{2\delta^2(n-b)m(m-1)}}$,

$$\frac{\frac{b}{\delta}(n-1) - \hat{\tau}^2L^2(n-b)m(m-1)\delta}{4b(n-1)\hat{\tau}} \geq \frac{1}{8\delta\hat{\tau}}.$$

Thus, with $\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{m,k}) = \mathcal{L}(\tilde{\theta}^{(k+1)})$ and $\hat{\mathcal{L}}^{(k)}(\hat{\theta}^{0,k}) = \mathcal{L}(\theta^{(k)})$ we obtain the desired result. $\square$

Combining Lemma 5 and Lemma 2 yields the following convergence statement with constant step-sizes on the coarse level.

Theorem 2. *Let Assumption 2 be satisfied and the step sizes chosen such that*

$$\tau \leq \frac{1}{L} \quad \text{and} \quad \hat{\tau} \leq \min\left\{\frac{1}{2L\delta}, \sqrt{\frac{b(n-1)}{2\delta^2(n-b)m(m-1)}}\right\}.$$

Let $(\theta^{(k)})$ be the sequence generated by Algorithm 1 with exact gradients on the fine level and the coarse objective defined in (13). *Then,*

$$\mathbb{E}\left[\mathcal{L}(\theta^{(k)}) - \mathcal{L}^*\right] \leq (1-r)^k(\mathcal{L}(\theta^{(0)}) - \mathcal{L}^*) - \sum_{i=0}^{k-1}(1-r)^{k-i}\hat{\rho}_i,$$

where $r = \frac{\eta\lambda(\tau)}{\tau}$ and

$$\hat{\rho}_i = \begin{cases} \frac{1}{\hat{\tau}}\sum\limits_{j=1}^{m}\mathbb{E}\left[D^{\mathrm{sym}}_{\hat{j}^{(k)}}(\hat{\theta}^{j,k},\hat{\theta}^{j-1,k})\right] + \frac{1}{8\delta\hat{\tau}}\sum\limits_{j=1}^{m}\mathbb{E}\left[\|\hat{\theta}^{j,k}-\hat{\theta}^{j-1,k}\|^2\right], & \text{if } i \text{ triggers coarse step},\\ 0, & \text{otherwise}. \end{cases}$$

C Computational savings

Regarding the computational effort, we follow Evci et al. (2020) and estimate the theoretical FLOPs required for training. As in their approach, we approximate the FLOPs for one training step by summing the forward pass FLOPs and twice the forward pass FLOPs for the backward pass. We only account for the FLOPs of convolutional and linear layers. Other operations such as BatchNorm, pooling, and residual additions are ignored, as their contribution to the total FLOPs is negligible compared to the dominant convolutional and linear computations. Denoting the FLOPs of a sparse forward pass by f_S and of a dense forward pass by f_D, the expected FLOPs per training step are

$$\frac{m \cdot 3f_S + (2f_S + f_D)}{m+1},$$

since every $(m+1)$-th step computes a full gradient. This matches the estimation used in RigL, where the network structure is fixed for most iterations and the full gradient is only computed periodically. In our method, however, the sparsity level evolves during training, whereas RigL enforces a fixed sparsity. As a result, particularly in the early stages of training, our method requires substantially fewer FLOPs. By contrast, LinBreg incurs $2f_S + f_D$ FLOPs per training step, since a full gradient is computed at every iteration.

We estimate the theoretical training FLOPs for the WideResNet28-10 models reported in Table 1. Using standard SGD training as the dense baseline, our analysis indicates that LinBreg requires only $0.38\times$ the FLOPs of the baseline, while our method further reduces this to $0.06\times$.

For the VGG16 models from Table 1, LinBreg requires $0.47\times$ the FLOPs of the dense model, whereas our method only needs $0.18\times$. Finally, for ResNet18, we consider the LinBreg approach with $\lambda = 0.2$, which leads to 96.03% sparsity on average, and the ML LinBreg with $\lambda = 0.007$, which reaches 96.12% sparsity. This results in a theoretical FLOPs reduction of $0.43\times$ for the former and $0.12\times$ for the latter.

D Background on numerical experiments

Initialization As previously mentioned, the network is initialized in a highly sparse fashion, allowing the training algorithm to subsequently activate additional parameters by setting them to non-zero values. For this initialization, we follow the approach proposed by Bungert et al. (2022).

He et al. (2015) suggest that for ReLU activation functions, the variance of the weights should satisfy

$$\mathrm{Var}\left[W^l\right] = \frac{2}{n_{l-1}}, \tag{14}$$

where n_l denotes the size of the l-th layer.

To obtain a sparse initialization, we apply binary masks to dense weight matrices, defining the initial weights as

$$W^l = \tilde{W}^l \odot M^l,$$

where $\odot$ denotes the element-wise (Hadamard) product, and each entry of the mask M^l is independently drawn from a Bernoulli distribution with parameter $r \in [0,1]$. This construction implies that the variance of the resulting weights scales as

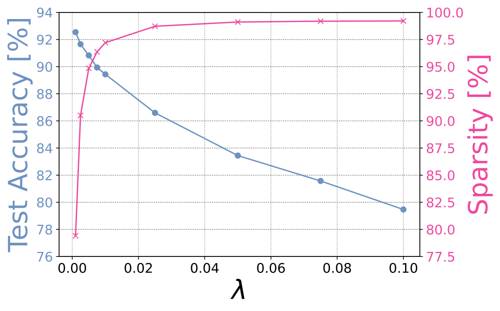
$$\mathrm{Var}\left[W^l\right] = r\,\mathrm{Var}\left[\tilde{W}^l\right].$$

To ensure that the condition in (14) is met, we adjust the distribution of $\tilde{W}^l$ accordingly. For instance, if $\tilde{W}^l$ originally satisfies (14), we scale it accordingly.

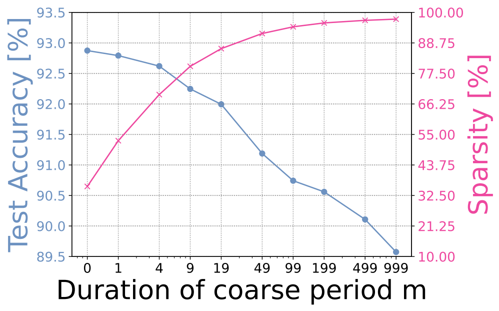
CIFAR10 training We summarize some of the results from our training on the CIFAR10 dataset in Table 1, most of which are also visualized in Figure 1. As previously mentioned, we use $J = \lambda\|\cdot\|_1$ as the regularizer and choose to perform a full update on the fine level every 100 iterations, meaning that $m = 99$ in Algorithm 1.

To demonstrate the effect of the regularization parameter λ in our training procedure, we train models with varying values of λ and evaluate both the test accuracy and sparsity of the resulting models. As expected, smaller values of λ yield highly accurate but less sparse models, whereas increasing λ results in reduced accuracy but greater sparsity. The results of this ablation study are presented in Figure 4a.

In Figure 4b, we perform an ablation study to investigate the effect of the coarse update duration m in Algorithm 1 on test accuracy and sparsity of the trained models. All other parameters are fixed; in particular, the regularizer is chosen as $J = 0.005\|\cdot\|_1$. The network is trained on multiple random seeds, and we report the mean values in the figure. Note that the value $m = 0$ corresponds to the case without coarse updates, i.e., the standard LinBreg algorithm. As expected, increasing m – and thus prolonging the freezing period – quickly yields sparser models, where the reduction in accuracy is minor.



(a) Accuracy and sparsity for ML LinBreg with different values of λ with fixed m



(b) Accuracy and sparsity for ML LinBreg with different values of m with fixed λ

Figure 4: Ablation study comparing accuracy and sparsity across different choices of the hyperparameters λ and m for ResNet18 on CIFAR10.

Table 1: Total sparsity and test accuracy for different network architectures and different optimizers

Architecture	**Optimizer**	**Sparsity** [%]	**Test acc** [%]
ResNet18	SGD	0.39 ± 0.08	92.93 ± 0.23
	Prune+Fine-Tuning	95.00	90.84 ± 0.30
	ML LinBreg ($\lambda = 0.005$)	94.76 ± 0.12	90.89 ± 0.21
	ML LinBreg ($\lambda = 0.007$)	96.12 ± 0.08	90.24 ± 0.33
	ML LinBreg ($\lambda = 0.01$)	97.20 ± 0.06	89.60 ± 0.27
	LinBreg ($\lambda = 0.2$)	96.03 ± 0.04	90.35 ± 0.22
VGG16	SGD	0.11 ± 0.02	91.98 ± 0.13
	Prune+Fine-Tuning	92.00	91.39 ± 0.18
	LinBreg ($\lambda = 0.1$)	91.82 ± 0.05	90.21 ± 0.24
	ML LinBreg ($\lambda = 0.003$)	91.29 ± 0.18	90.71 ± 0.19
WideResNet28-10	SGD	0.17 ± 0.03	93.79 ± 0.08
	Prune+Fine-Tuning	96.00	90.55 ± 0.32
	LinBreg ($\lambda = 0.1$)	95.89 ± 0.16	91.70 ± 0.25
	ML LinBreg ($\lambda = 0.005$)	96.58 ± 0.14	91.69 ± 0.27

To obtain some additional information on the training procedure, we track the mean and standard deviation of the validation accuracy and the sparsity as well as the train loss throughout the training procedure. The resulting plots for ResNet18 are displayed in Figure 5. We observe that freezing the network structure neither affects the final characteristics of the model nor alters the training speed.

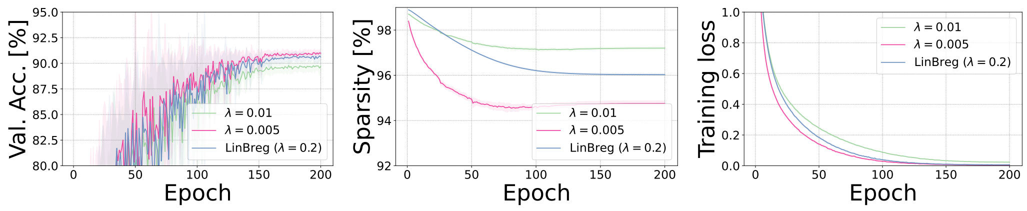
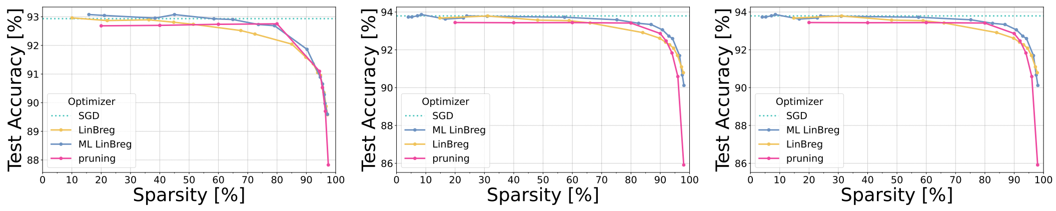


Figure 5: Mean and standard deviation of validation accuracy, sparsity, and train loss of the model parameters over training epochs, shown for different regularization parameters, $\lambda = 0.005$ and $\lambda = 0.01$ and for LinBreg with $\lambda = 0.2$.

Moreover, in addition to comparing the different algorithms for extremely sparse models in Figure 1, we also compare them over the full range of possible levels of sparsity in Figure 6. As before, we include SGD as a dense baseline, as well as pruned versions of the SGD-trained models at various levels of sparsity. For sparsity levels that are not as extreme as those considered in Figure 1, we observe that our method is at least as performant as LinBreg and the pruning approach across different network architectures.



(a) ResNet18 (b) VGG16 (c) WideResNet28-10

Figure 6: Accuracy and sparsity for different regularizers with LinBreg and ML LinBreg as well as a pruning baseline on CIFAR10 for different architectures

Mask initialization We explore the affect of the mask initialization scheme for CIFAR10 ResNet18 models trained by both ML LinBreg and RigL (Evci et al., 2020). In particular, we consider three mask initialization schemes as proposed in Evci et al. (2020).

- *Uniform:* The mask for each layer of the network is uniformly sampled to achieve the target initial sparsity
- *Erdős–Rényi (ER):* Both linear and convolutional layers have sparsities proportional to scores
$$\frac{n_{in} + n_{out}}{n_{in} n_{out}}$$
where n_{in}/n_{out} refers to the dimensions of the input/output of a linear layer and the number of input/output channels for a convolutional layer
- *Erdős–Rényi-Kernel (ERK):* Similiar to ER except the scores for convolutional layers with $k \times k$ kernels are given by
$$\frac{n_{in} + n_{out} + 2k}{n_{in} n_{out} k^2}$$

In the case of ER and ERK, one would in practice calculate the scores and then determine a global rescaling factor of said scores such that the determined masked model achieves the specified target sparsity. The motivation of ER is that layers with many parameters should be very sparse, and ERK makes this more explicit in the case of convolutional layers.

We remark that while the variance-preserving rescaling factor r will be constant across layers when using a uniform mask initialization, for ER and ERK one must adjust the value of r according to the layer's determined sparsity.

We report the achieved sparsity and test accuracy of RigL (target sparsity of 95%) and ML LinBreg ($\lambda = 0.005$) for different mask initialization schemes in Table 2. We see that, when using the variance-preserving rescaling for the weight initialization, the mask initialization can vary the accuracy of models trained via RigL by 2.5% whereas ML LinBreg is far more robust to the initialization and varies by only 0.6%.

Table 2: Performance of RigL and ML LinBreg for various mask initialization schemes across 5 seeds.

Optimizer	**Mask initialization**	**Sparsity** $[\%]$	**Test acc** $[\%]$
RigL	Uniform	94.918	88.280 ± 0.14
	ER	94.918	88.274 ± 0.40
	ERK	94.975	90.830 ± 0.32
ML LinBreg	Uniform	94.687 ± 0.10	90.812 ± 0.33
	ER	94.788 ± 0.06	91.448 ± 0.15
	ERK	94.835 ± 0.05	91.180 ± 0.23

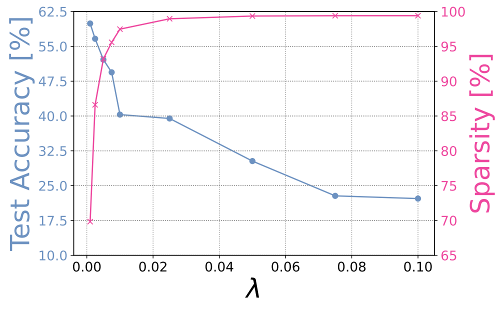
Structured sparsity Even though, we do not induce any structured sparsity through the regularizer J, we observe that for our trained models many of the 2D-kernels are zero. To measure this kind of sparsity, we define the convolutional sparsity as

$$S_{\text{conv}} := 1 - \frac{\sum_{l \in \mathcal{I}_{\text{conv}}} |\{K^l_{ij} \neq 0\}|}{\sum_{l \in \mathcal{I}_{\text{conv}}} c_{l-1} \cdot c_l},$$

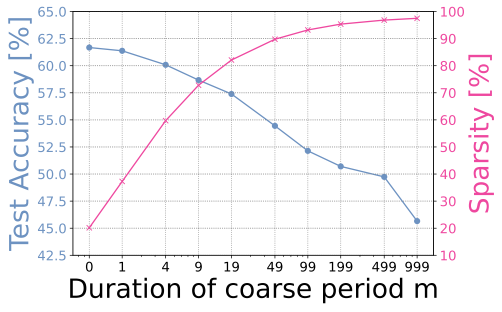
where $\mathcal{I}_{\text{conv}}$ denotes the index set of all convolutional layers, c_{l-1} and c_l are the number of input and output channels of layer $l \in \mathcal{I}_{\text{conv}}$, respectively, and K^l_{ij} is the kernel connecting input channel i to output channel j.

For example, in the WideResNet28-10 case with $\lambda = 0.005$, where we observed a test accuracy of $91.69\% \pm 0.27\%$ we obtain $S_{\text{conv}} = 78.21\% \pm 0.82\%$, meaning that almost 80% of input-output connections are zero. We can further increase this structured sparsity by taking the group $\ell_{1,2}$ norm from (7) as the regularizer. We again scale it by some positive factor λ to control its influence on the optimization procedure. With this approach, we achieve a convolutional sparsity of $S_{\text{conv}} = 84.06\% \pm 0.79\%$ while maintaining a test accuracy of $91.45\% \pm 0.59\%$. By increasing the strength of the regularizer, this can be further improved to $93.09\% \pm 0.46\%$ convolutional sparsity, with a corresponding test accuracy of $90.48\% \pm 0.72\%$.

Scaling up To investigate how increasing the scale of the experiments affects the results when keeping the hyperparameters fixed, we generated the same plots as in Figure 4, this time for the WideResNet28-10 architecture on the TinyImageNet dataset. This setup simultaneously increases both the network capacity and the size and complexity of the dataset. The results of this analysis are shown in Figure 7.



(a) Accuracy and sparsity for ML LinBreg with different values of λ with fixed m



(b) Accuracy and sparsity for ML LinBreg with different values of m with fixed λ

Figure 7: Ablation study comparing accuracy and sparsity across different choices of the hyperparameters λ and m for WideResNet28-10 on TinyImageNet.

Although the sparsity and test accuracy values are, of course, not identical to those obtained on CIFAR-10, the sparsity levels achieved under the exact same hyperparameters remain in a similar range. As expected, the test accuracies are not directly comparable, since the classification task on TinyImageNet with 200 classes is substantially more challenging than CIFAR-10 with only 10 classes. Furthermore, we observe that small values of $\lambda \approx 0.005$ give best result for both the small scale and the large scale experiment, suggesting that this parameter is pretty scale-invariant. On the other hand, it appears like the large scale experiment yields better results for smaller values of m than the small scale one which is potentially also due to the increased difficulty of the learning problem requiring more exact gradient information.