Parsimonious Deep Learning: A Differential Inclusion Approach with Global Convergence

Yanwei Fu\textsuperscript{1}, Chen Liu\textsuperscript{1}, Donghao Li\textsuperscript{1,4}, Xinwei Sun\textsuperscript{2}, Jinshan Zeng\textsuperscript{3,4} and Yuan Yao\textsuperscript{4}

\textsuperscript{1}Fudan University, \textsuperscript{2}Microsoft Research-Asia, \textsuperscript{3}Jiangxi Normal University, \textsuperscript{4}Hong Kong University of Science and Technology

Abstract

Over-parameterization is ubiquitous nowadays in training neural networks to benefit both optimization in seeking global optima and generalization in reducing prediction error. However, compressive networks are desired in many real world applications and direct training of small networks may be trapped in local optima. In this paper, instead of pruning or distilling an over-parameterized model to compressive ones, we propose a parsimonious learning approach based on differential inclusions of inverse scale spaces, that generates a family of models from simple to complex ones with a better efficiency and interpretability than stochastic gradient descent in exploring the model space. It enjoys a simple discretization, the Split Linearized Bregman Iterations, with provable global convergence that from any initializations, algorithmic iterations converge to a critical point of empirical risks. One may exploit the proposed method to boost the complexity of neural networks progressively. Numerical experiments with MNIST, Cifar-10/100, and ImageNet are conducted to show the method is promising in training large scale models with a favorite interpretability.

1 Introduction

The expressive power of Deep Convolutional Neural Networks (DNNs) comes from the millions of parameters, which are optimized by Stochastic Gradient Descent (SGD) algorithms [Bottou, 2010] and variants like Adam [Kingma and Ba, 2015]. Remarkably, model over-parameterization helps both optimization and generalization. For optimization, over-parameterization may simplify the landscape of empirical risks toward locating global optima efficiently by gradient descent method [Mei et al., 2018, 2019, Venturi et al., 2018, Allen-Zhu et al., 2018, Du et al., 2018]. On the other hand, over-parameterization does not necessarily result in a bad generalization or overfitting [Zhang et al., 2017], especially when some weight-size dependent complexities are controlled [Bartlett, 1997, Bartlett et al., 2017, Golowich et al., 2018, Neyshabur et al., 2019].
However, compressive networks are desired in many real world applications, e.g. robotics, self-driving cars, and augmented reality. Despite that $\ell_1$ regularization has been applied to deep learning to enforce the sparsity on the weights toward compact, memory efficient networks, it sacrifices some prediction performance [Collins and Kohli, 2014]. This is because that the weights learned in neural networks are highly correlated, and $\ell_1$ regularization on such weights violates the incoherence or irrepresentable conditions needed for sparse model selection [Donoho and Huo, 2001, Tropp, 2004, Zhao and Yu, 2006], leading to spurious selections with poor generalization. On the other hand, $\ell_2$ regularization is often utilized for correlated weights as some low-pass filtering, sometimes in the form of weight decay [Loshchilov and Hutter, 2019] or early stopping [Yao et al., 2007, Wei et al., 2017]. Furthermore, group sparsity regularization [Yuan and Lin, 2006] has also been applied to neural networks, such as finding optimal number of neuron groups [Alvarez and Salzmann, 2016] and exerting good data locality with structured sparsity [Wen et al., 2016, Yoon and Hwang, 2017].

Yet, without the aid of over-parameterization, directly training a compressive model architecture may meet the obstacle of being trapped in local optima in contemporary experience. Alternatively, researchers in practice typically start from training a big model using common task datasets like ImageNet, and then prune or distill such big models to small ones without sacrificing too much of the performance [Jaderberg et al., 2014, Han et al., 2015, Zhu et al., 2017, Zhou et al., 2017, Zhang et al., 2016, Li et al., 2017, Abbasi-Asl and Yu, 2017, Yang et al., 2018, Arora et al., 2018]. In particular, a recent study [Frankle and Carbin, 2019] created the lottery ticket hypothesis based on empirical observations: “dense, randomly-initialized, feed-forward networks contain subnetworks (winning tickets) that – when trained in isolation – reach test accuracy comparable to the original network in a similar number of iterations”. How to effectively reduce an over-parameterized model thus becomes the key to compressive deep learning. Such a backward pruning approach is in a sharp contrast to traditional parsimonious or sparse learning approach where one starts from simple yet interpretable models, delving into complex models progressively. Examples of such a parsimonious learning include boosting as functional gradient descent or coordinate descent method, regularization paths associated with $\ell_2$ (Tikhonov regularization or Ridge regression) and/or $\ell_1$ (Lasso) penalties, etc. None of these has been systematically studied in deep learning.

In this paper, we aim to fill in this gap by exploiting a dynamic approach to forward parsimonious learning. We are able to establish a family of neural networks, from simple to complex, by following regularization paths as solutions of differential inclusions of inverse scale spaces. Our key idea is to design some dynamics that simultaneously exploit over-parameterized models and structural sparsities. To achieve this goal, the original network parameters are lifted to a coupled pair, with one set of parameters following the standard gradient descend to explore the over-parameterized model space, while the other set of parameters learning structure sparsity in an inverse scale space, that the large-scale important parameters are learned at a fast speed while the small unimportant ones are learned at a slow speed. The two sets of parameters are coupled in an $\ell_2$ regularization. The dynamics enjoys a simple discretization, the Split Linearized Bregman Iteration, with provable global convergence guarantee. Equipped with such a method, one may efficiently boost or grow a network with different sparse models, in a more efficient way than SGD to explore the model space and in favor of better interpretability. For example, Figure 1 exhibits the performance on training ResNet-18 on ImageNet, using SGD, Adam, and our algorithms, where a proper
choice of parameters leads to more efficient training than SGD and Adam in early epochs with a comparable accuracy in the end. Moreover, Figure 3 further demonstrates that the sparse convolutional filters trained on ImageNet, focus more on non-semantic textures than shapes and colors, in support of alternative studies [Abbasi-Asl and Yu, 2017, Geirhos et al., 2019]. How to enhance the semantic shape invariance learning, is arguably a key to improve the robustness of convolutional neural networks.

The organization of the paper is as follows. Section 2 introduces the methodology; Section 3 establishes the global convergence theory; Section 4 discusses stochastic approximation algorithms and boosting; experimental results are presented in Section 5. Proofs are collected in Appendix A.

2 Methodology

The supervised learning task learns a mapping $\Phi_W : \mathcal{X} \rightarrow \mathcal{Y}$, from input space $\mathcal{X}$ to output space $\mathcal{Y}$, with a parameter $W$ such as weights in neural networks, by minimizing certain loss functions on training samples $\check{L}_n(W) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \Phi_W(x_i))$. For example, a neural network of $l$-layer is defined as

$$
\Phi_W(x) = \sigma_l(W_l \sigma_{l-1}(W_{l-1} \cdots \sigma_1(W_1 x)))
$$

where $W = \{W_i\}_{i=1}^{l}$, $\sigma_i$ is the nonlinear activation function of the $i$-th layer and the loss function is usually the cross-entropy.

Differential Inclusion of Inverse Scale Space. Consider the following dynamics,

$$
\frac{\dot{W}_t}{\kappa} = -\nabla_W \check{L}(W_t, \Gamma_t) \\
\dot{V}_t = -\nabla_\Gamma \check{L}(W_t, \Gamma_t) \\
V_t \in \partial\Omega(\Gamma_t)
$$

Figure 1: Comparison on training and test accuracies of training ResNet18 on ImageNet, by SGD, Adam, and SplitLBI with $\kappa = 1, 10$, respectively. For the choice of $\kappa = 1$, SplitLBI enjoys more efficient learning curves than SGD and Adam noticeably in early epochs. Details about the experiments can be found in Section 5.
where \( \bar{\Omega}(\Gamma) = \Omega(\Gamma) + \frac{1}{2\kappa}\|\Gamma\|^2 \) for some sparsity-enforced regularization \( \Omega(\Gamma) \) such as Lasso or grouped Lasso penalties, \( \kappa > 0 \) as a damping parameter such that the solution path is continuous, and the augmented loss function is

\[
\bar{\mathcal{L}}(W, \Gamma) = \hat{\mathcal{L}}_n(W) + \frac{1}{2\nu}\|W - \Gamma\|^2,
\]

with \( \nu > 0 \) controlling the gap admitted between \( W \) and \( \Gamma \). Compared to the original loss function \( \hat{\mathcal{L}}_n(W) \), the \( \bar{\mathcal{L}}(W, \Gamma) \) additionally adopt the variable splitting strategy, by lifting the original neural network parameter \( W \) to \((W, \Gamma)\) with \( \Gamma \) to model the structural sparsity of \( W \). For simplicity, we assumed \( \bar{\mathcal{L}} \) is differentiable with respect to \( W \) here, otherwise the gradient in (1a) is understood as subgradient and the equation becomes an inclusion.

The differential inclusion system (1) is called as the Split Linearized Bregman Inverse Scale Space (SplitISS) for the following reasons. It can be understood as a gradient descent flow of \( W_t \) in the proximity of \( \Gamma_t \) and a mirror descent [Nemirovski and Yudin, 1983] flow of \( \Gamma_t \) associated with a sparsity enforcement penalty \( \bar{\Omega} \). For a large enough \( \nu \), it reduces to the gradient descent method for \( W_t \). Yet the solution path of \( \Gamma_t \) exhibits the following selection property in the separation of scales: starting at the zero, important parameters of large scale will be learned fast, popping up to be nonzeros early, while unimportant parameters of small scale will be learned slowly, appearing to be nonzeros late. In fact, taking \( \Omega(\Gamma) = \|\Gamma\|_1 \) and \( \kappa \to \infty \) for simplicity, \( V_t \) undergoes a gradient descent flow before reaching the \( \ell_\infty \)-unit box, which implies that \( \Gamma_t = 0 \) in this stage and \( W_t \) follows the gradient descent with a standard \( \ell_2 \)-regularization; the earlier a component in \( V_t \) reaching the \( \ell_\infty \)-unit box, the earlier a corresponding component in \( \Gamma_t \) becomes nonzero and rapidly evolves toward a critical point of \( \bar{\mathcal{L}} \) under gradient flow, which drives \( W_t \) closely following dynamics of \( \Gamma_t \) whose important parameters are selected. Such a property is called as the inverse scale space in applied mathematics [Burger et al., 2006] and recently was shown to achieve statistical model selection consistency in high dimensional linear regression [Osher et al., 2016] and general linear models [Huang and Yao, 2018], with a reduction of bias as \( \kappa \) increases. In this paper, we shall see the inverse scale space property still holds for neural networks trained by Eq. (1), e.g. Figure 2 shows a LeNet trained on MNIST by the discretized dynamics, where important sparse filters are selected in early epochs while the popular stochastic gradient descent method returns dense filters.

Compared with directly enforcing a penalty function \( \Omega(W) \) (e.g. \( \ell_1 \) or \( \ell_2 \) regularization)

\[
\min_W \tilde{\mathcal{R}}_n(W) := \hat{\mathcal{L}}_n(W) + \lambda \Omega(W), \quad \lambda \in \mathbb{R}_+.
\]

the SplitISS avoids the parameter correlation problem in over-parameterized models. In fact, a necessary and sufficient condition for Lasso or \( \ell_1 \)-type sparse model selection is the incoherence or irepresentable conditions ([Tropp, 2004, Zhao and Yu, 2006]) that are violated for highly correlated weight parameters, leading to spurious discoveries. In contrast, Huang et al. [Huang et al., 2018] showed that equipped with such a variable splitting where \( \Gamma \) enjoys an orthogonal design, the SplitISS can achieve model selection consistency under weaker conditions than generalized Lasso, relaxing the incoherence or irepresentable conditions when parameters are highly correlated. For weight parameter \( W \), instead of directly being imposed with \( \ell_1 \)-sparsity, it adopts \( \ell_2 \)-regularization in the proximity of the sparse path of \( \Gamma \) that admits simultaneously exploring highly correlated parameters in over-parameterized models and sparsity regularizations.
Figure 2: Visualization of solution path and filter patterns in the third convolutional layer (i.e., conv.c5) of LetNet-5, trained on MNIST. The left figure shows the magnitude changes for each filter of the models trained by SplitLBI and SGD, where x-axis and y-axis indicate the training epochs, and filter magnitudes ($\ell_2$-norm), respectively. The SplitLBI path of filters selected in the support of $\Gamma$ are drawn in blue color, while the red color curves represent the filters that are not important and outside the support of $\Gamma$. We visualize the corresponding learned filters by Erhan et al. [2009] at 20 (blue), 40 (green), and 80 (black) epochs, which are shown in the right figure with the corresponding color bounding boxes, i.e., blue, green, and black, respectively. It shows that our SplitLBI enjoys a sparse selection of filters without sacrificing accuracy (see Table 1).

**Split Linearized Bregman Iterations.** The SplitISS admits an extremely simple discrete approximation, using the Euler forward discretization of the dynamics above:

\[
W_{k+1} = W_k - \kappa \alpha_k \cdot \nabla_W \tilde{\mathcal{L}}(W_k, \Gamma_k), \\
V_{k+1} = V_k - \alpha_k \cdot \nabla_{\Gamma} \tilde{\mathcal{L}}(W_k, \Gamma_k), \\
\Gamma_{k+1} = \kappa \cdot \text{Prox}_\Omega(V_{k+1}),
\]

where $V_0 = \Gamma_0 = 0$, $W_0$ can be small random numbers such as Gaussian distribution in neural networks, the proximal map in Eq (4c) that controls the sparsity of $\Gamma$ is given by

\[
\text{Prox}_\Omega(V) = \arg \min_{\Gamma} \left\{ \frac{1}{2} \|\Gamma - V\|_2^2 + \Omega(\Gamma) \right\},
\]

We shall call such an iterative procedure as **Split Linearized Bregman Iteration** (SplitLBI), that was firstly proposed in [Huang et al., 2016] as an iterative regularization path for sparse modeling in high dimensional statistics with an improved model selection consistency than generalized Lasso, and has been later used in medical image analysis [Sun et al., 2017] and computer vision [Zhao et al., 2018]. In the application to neural networks, the loss becomes highly non-convex, the SplitLBI returns a sequence of sparse models from simple to complex ones with a global convergence guarantee shown below, while solving Eq (3) at various levels of $\lambda$ might not be tractable except for over-parameterized models.

The sparsity-enforcement penalty $\Omega(\cdot)$ used in convolutional neural networks can be chosen as follows. Our choice in this paper aims at regularizing the groups of weight parameters
using the group Lasso penalty [Yuan and Lin, 2006] defined by,

\[ \Omega(\Gamma) := \sum_g \| \Gamma^g \|_2 = \sum_g \sqrt{\sum_{i=1}^{|\Gamma^g|} \left( \Gamma^g_i \right)^2}, \]

where \(|\Gamma^g|\) is the number of weights in \(\Gamma^g\). We treat convolutional layers and fully connected layers in different ways as follows.

(a) For a convolutional layer, \(\Gamma^g = \Gamma^g(c_{in}, c_{out}, \text{size})\) denote the convolutional filters where \(\text{size}\) denotes the kernel size and \(c_{in}\) and \(c_{out}\) denote the numbers of input channels and output channels, respectively. We select \(g = c_{out}\) and regard each group as each convolutional filter.

(b) For a fully connected layer, \(\Gamma = \Gamma(c_{in}, c_{out})\) where \(c_{in}\) and \(c_{out}\) denote the numbers of inputs and outputs of the fully connected layer. Each group \(g\) corresponds to each element \((i, j)\), and the group Lasso penalty degenerates to the Lasso penalty.

With such choices, Eq. (4c) hence has a closed form solution \(\Gamma^g = \kappa \cdot \max (0, 1 - 1/\|Z^g\|_2) Z^g\) for the \(g\)-th filter.

3 Global Convergence of SplitLBI for Neural Networks

We present a theorem that guarantees the \textit{global convergence} of SplitLBI, i.e. from any initialization, the SplitLBI sequence converges to a critical point of \(\bar{\mathcal{L}}\).

First of all, we reformulate Eq. (6) into an equivalent form. Let \(P := (W, \Gamma)\), the SplitLBI algorithm in Eq (4a)-(4c) can be rewritten as the following standard Linearized Bregman Iteration [Huang and Yao, 2018],

\[ P_{k+1} = \arg\min_P \{ \langle P - P_k, \alpha \nabla \bar{\mathcal{L}}(P_k) \rangle + B^\Psi_P (P, P_k) \}, \]

where

\[ \Psi(P) = \Omega(\Gamma) + \frac{1}{2\kappa} \|P\|_2^2 = \Omega(\Gamma) + \frac{1}{2\kappa} \|W\|_2^2 + \frac{1}{2\kappa} \|\Gamma\|_2^2, \]

\(p_k \in \partial \Psi(P_k)\), and \(B^\Psi_P\) is the Bregman divergence associated with convex function \(\Psi\), defined by

\[ B^\Psi_P (P, Q) := \Psi(P) - \Psi(Q) - \langle q, P - Q \rangle, \text{ for some } q \in \partial \Psi(Q). \]

Denote \(R(P) := \Omega(\Gamma)\), then Eq (6) can be rewritten as,

\[ P_{k+1} = \text{Prox}_{\kappa R}(P_k + \kappa (p_k - \alpha \nabla \bar{\mathcal{L}}(P_k))), \]

\[ p_{k+1} = p_k - \kappa^{-1}(P_{k+1} - P_k + \kappa \alpha \nabla \bar{\mathcal{L}}(P_k)), \]

where \(p_k = [0, g_k]^T \in \partial R(P_k)\) and \(g_k \in \partial \Omega(\Gamma_k)\). Thus SplitLBI is equivalent to the following iterations,

\[ W_{k+1} = W_k - \kappa \alpha \nabla W \bar{\mathcal{L}}(W_k, \Gamma_k), \]

\[ \Gamma_{k+1} = \text{Prox}_{\kappa \Omega}(\Gamma_k + \kappa (g_k - \alpha \nabla \Gamma \bar{\mathcal{L}}(W_k, \Gamma_k))), \]

\[ g_{k+1} = g_k - \kappa^{-1}(\Gamma_{k+1} - \Gamma_k + \kappa \alpha \cdot \nabla \Gamma \bar{\mathcal{L}}(W_k, \Gamma_k)). \]
Exploiting the equivalent reformulation \((10a)-(10c)\), one can establish the global convergence of \((W_k, \Gamma_k, g_k)\) under the following assumptions.

**Assumption 1.** Suppose that:

1. \(\hat{L}_n(W)\) is continuous differentiable and \(\nabla \hat{L}_n\) is Lipschitz continuous with a positive constant \(Lip\);
2. \(\hat{L}_n(W)\) has bounded level sets;
3. \(\hat{L}_n(W)\) is lower bounded (without loss of generality, we assume that the lower bound is 0);
4. \(\Omega\) is a proper lower semi-continuous convex function and has locally bounded subgradients, that is, for every compact set \(S \subseteq \mathbb{R}^n\), there exists a constant \(C > 0\) such that for all \(\Gamma \in S\) and all \(g \in \partial \Omega(\Gamma)\), there holds \(\|g\| \leq C\);
5. the Lyapunov function
   \[
   F(P, G) := \alpha \hat{L}(W, \Gamma) + B_{\Omega}^\circ(\Gamma, \tilde{\Gamma}),
   \]
   is a Kurdyka-Lojasiewicz function on any bounded set, where \(B_{\Omega}^\circ(\Gamma, \tilde{\Gamma}) := \Omega(\Gamma) - \Omega(\tilde{\Gamma}) - \langle \tilde{g}, \Gamma - \tilde{\Gamma} \rangle, \Gamma \in \partial \Omega^*(\tilde{g})\), and \(\Omega^*\) is the conjugate of \(\Omega\) defined as
   \[
   \Omega^*(g) := \sup_{U \in \mathbb{R}^n} \{\langle U, g \rangle - \Omega(U)\}.
   \]

**Remark 1.** Assumption 1 (a)-(c) are regular in the analysis of nonconvex algorithm (see, Attouch et al. [2013] for instance), while Assumption 1 (d) is also mild including all Lipschitz continuous convex function over a compact set. Some typical examples satisfying Assumption 1(d) are the \(\ell_1\) norm, group \(\ell_1\) norm, and every continuously differentiable penalties. By Eq (11) and the definition of conjugate, the Lyapunov function \(F\) can be rewritten as follows,

\[
F(W, \Gamma, g) = \alpha \hat{L}(W, \Gamma) + \Omega(\Gamma) + \Omega^*(g) - \langle \Gamma, g \rangle.
\]

Now we are ready to present the main theorem.

**Theorem 1** (Global Convergence of SplitLBI). Suppose that Assumption 1 holds. Let \((W_k, \Gamma_k, g_k)\) be the sequence generated by SplitLBI (Eq (10a)-(10c)) with a finite initialization. If \(0 < \alpha_k = \alpha < \frac{2}{\kappa(Lip + \nu - 1)}\), then \((W_k, \Gamma_k, g_k)\) converges to a critical point of \(F\). Moreover, \((W_k, \Gamma_k)\) converges to a stationary point of \(\hat{L}\) defined in Eq (2), and \(\{W^k\}\) converges to a stationary point of \(\hat{L}_n(W)\).

Applying to the neural network training settings, typical examples satisfying the assumption of this theorem are summarized in the following corollary.

**Corollary 1.** Let \(\{W_k, \Gamma_k, g_k\}\) be a sequence generated by SLBI (10a)-(10c) for the deep neural network training problem where
(a) \( \ell \) is any smooth definable loss function, such as the square loss \( t^2 \), exponential loss \( e^t \), logistic loss \( \log(1 + e^{-t}) \), and cross-entropy loss;

(b) \( \sigma_i \) is any smooth definable activation, such as linear activation, sigmoid \( \frac{1}{1 + e^{-t}} \), hyperbolic tangent \( \frac{e^t - e^{-t}}{e^t + e^{-t}} \), and softplus \( \frac{1}{c} \log(1 + e^{ct}) \) for some \( c > 0 \) as a smooth approximation of ReLU;

(c) \( \Omega \) is the group Lasso penalty (i.e., \( \Omega(\Gamma) = \sum_g \| \Gamma_g \|_2 \)).

Moreover, suppose that the empirical risk \( \hat{\mathcal{L}}_n(W) \) has lower bounded level set and its gradient is Lipschitz continuous with a constant \( \text{Lip} > 0 \). Then the sequence \( \{W_k\} \) converges to a stationary point of \( \hat{\mathcal{L}}_n(W) \) if the step size \( 0 < \alpha_k = \alpha < \frac{1}{\kappa (\text{Lip} + \nu - 1)} \).

Proofs of Theorem 1 and Corollary 1 are given in Appendix A.

4 Stochastic Approximation of SplitLBI and Applications

For neural network training with large datasets, stochastic approximation of the gradients in Split LBI over the mini-batch \( (X, Y)_{\text{batch}} \) is adopted to update the parameter \( W \),

\[
\nabla^t W = \nabla W \hat{\mathcal{L}}_n(W) \mid (X, Y)_{\text{batch}}. \tag{13}
\]

Moreover, inspired by the variants of SGD, the momentum term can be also incorporated to the standard Split LBI that leads to the following updates of \( W \) by replacing Eq (4a) with,

\[
\begin{align*}
v_{t+1} &= \tau v_t + \nabla W \hat{\mathcal{L}}(W_t, \Gamma_t) \tag{14a} \\
W_{t+1} &= W_t - \kappa \alpha v_{t+1} \tag{14b}
\end{align*}
\]

where \( \tau \) is the momentum factor, empirically setting as 0.9 in default. One immediate application of such stochastic algorithms of SplitLBI is to “boost networks”, i.e. growing a network from the null to a complex one by sequentially applying our algorithm on subnets with increasing complexities.

**Boosting Networks.** We propose the task of boosting network by SplitLBI, i.e., learning to expand the convolutional filters and network parameters during training. Remarkably, our boosting network is very different from previous tasks, including AutoML [Wong et al., 2018], or life-long learning [Wang et al., 2017, Thrun and Mitchell, 1995, Pentina and Lampert, 2015, Li and Hoiem, 2016], knowledge distill [Hinton et al., 2014]. Starting from very few filters of each convolutional layer, algorithms of boosting network require not only efficiently optimizing the parameters of filters, but also adding more filters if the existing filters do not have enough capacity to model the distribution of training data. Note that the existing work mentioned above does not allow additional expanding or fine-tuning algorithms which are very computational expensive in practice.

To apply the SplitLBI to boost networks, we define a projection of the convolutional filters \( W \) on the support set of \( \Gamma \),

\[
\tilde{W}_{k+1} = \text{Proj} \text{supp}(\Gamma_{k+1}) (W_{k+1}). \tag{15}
\]
The basic idea is to monitor the gap between $W$ and its projection $\tilde{W}$ along the iterations: when the gap becomes small, we are going to expand the network by adding new filters.

We give the description of the procedure of expanding a network by SplitLBI. Starting from a small number of filters (e.g. 2) of convolutional layer, more and more filters tend to be with non-zero value as the algorithm iterates. At each step, we can compute the ratio of $\tilde{W}_k$ to be non-zero. If this ratio passes a pre-set threshold (empirically setting as 0.8), we add $m$ (e.g. 2) new convolutional filters into $W$; and the newly added parameters are randomly initialized. Then we continue learning the network from training data. This process is repeated until model converged or maximum epochs reached.

5 Experiments

The proposed SplitLBI is evaluated on MNIST, Cifar-10/100 and ImageNet-2012 datasets, which are widely utilized as the benchmarks in deep learning. Particularly, Sec. 5.1 compares SplitLBI against SGD and Adam in image classification. Furthermore, Sec. 5.2 shows that SplitLBI can efficiently boost a network from null to complex ones from data.

**Implementation.** Various algorithms are evaluated over the LeNet (LeCun et al. [2015]), AlexNet and ResNet (He et al. [2016]), respectively. By default, the minibatch-based SplitLBI is employed here with the batch size 256. For MNIST and Cifar-10, the hyper-parameters are $\kappa = 10$, $\nu = 10$ and $\alpha_k$ starts from 0.1 and decays by 0.1 every 40 epochs. In ImageNet-2012, we use $\kappa \in \{1, 10\}$, $\nu = 1000$, and $\alpha_k$ starts from 0.1 and decays by 0.1 every 30 epochs. Adam follows the default PyTorch setting and SGD follows He et al. [2016] with initial learning rate 0.1, momentum $\tau = 0.9$ and weight decay $5 \times 10^{-4}$.

5.1 Image Classification

In the supervised setting, SGD, Adam and SplitLBI are adopted in optimizing DNNs on these datasets. The results are shown in Table 1. To train ResNet-18 on ImageNet-2012 dataset, the model trained by SplitLBI ($\kappa = 1$) can achieve the Top-1 accuracy of 67.76 by 40 epochs, which is 2% lower than the best ResNet-18 results report in He et al. [2016] optimized by 120 epochs. Still this indicates that the SplitLBI may have a more efficient training than SGD. Note that, (1) In Fig. 1, we qualitatively compare our SplitLBI of $\kappa = 1, 10$ against SGD and Adam. Remarkably, at the early stage, our SplitLBI of $\kappa = 1$ has a much better efficiency than Adam and SGD within the same number of training epochs; (2) our SLBI requires the

| Dataset              | MNIST | Cifar-10 | ImageNet-2012 |
|----------------------|-------|----------|---------------|
| Models               | LeNet | ResNet-20| AlexNet       |
| $SGD^*$              | 99.21 | 91.25    | 56.55/79.09   |
| $Adam^*$             | 99.19 | 90.86    | –/–           |
| SplitLBI             | 99.18 | 91.50    | 56.09/78.86   |
| SplitLBI with momentum| 99.19| 91.57    | 56.09/78.86   |

Table 1: Top-1/Top-5 accuracy(%) comparison between SGD and Split LBI on each dataset. *: the results are reproduced from the official PyTorch website. All models are trained with 100 epochs.
ResNet-18@ImageNet (Conv1)

Figure 3: Training curves and visualization of filters in the first convolutional layer of ResNet-18 trained on ImageNet-2012. SplitLBI with $\kappa = 1$ shows a better efficiency than SGD and Adam in early epochs. The important filters selected by SplitLBI focus more on textures than shapes and colors.

extra 7% – 8% more GPU usage than that of SGD, in order to save the parameters of $V_k$ and $\Gamma_k$. Comparing with SGD, the SplitLBI adds little extra memory cost to store $V$ and $\Gamma$. The total running time of each epoch is comparable to that of SGD.

To interpret the learned filters by SplitLBI with structural sparsity, we visualize a LeNet-5 trained on MNIST and a ResNet-18 on ImageNet.

Visualization of LeNet-5 on MNIST. We visualize the solution paths and filter patterns of the third convolutional layer (i.e., conv.c5) of LeNet-5 trained by SplitLBI and SGD trained on MNIST as in Fig. 2. X-axis and Y-axis indicate the training epochs, and magnitudes of each filter respectively. In particular, the left two figures compares the magnitude value changes of each filter trained by SplitLBI and SGD, individually. The filters that are not in the support of $\Gamma$, are drawn by red color. In the right figure, the model parameters learned at 20 (blue), 40 (green), and 80 (black) epochs are visualized as Erhan et al. [2009], with the corresponding color bounding boxes, i.e., blue, green, and black, respectively. The conv.c5 has 120 convolutional filters, which are ordered by the magnitude of filters. We can observe that the filters of high magnitude learned by our SplitLBI are much sparser and have very much clearer patterns that those trained by SGD, showing that SplitLBI encourages structural sparsity of filters.

Visualization of ResNet-18 on ImageNet-2012. To further reveal the insights of learned patterns of SplitLBI, we visualize the first convolutional layer of ResNet-18 on ImageNet-2012 in Fig. 3. The left figure compares the training and validation accuracy of SplitLBI ($\kappa = 1, 10$) and SGD. The right figure compares visualizations of the filters learned by SplitLBI and SGD using Springenberg et al. [2014]. To be specific, denote the weights of an $l$-layer network as $\{W^1, W^2, \cdots, W^l\}$. For the $i$-th layer weights $W^i$, denote the $j$-th channel $W^i_j$. Then we compute the gradient of the sum of the feature map computed from each filter $W^i_j$ with respect to the input image (here a snake image). We further conduct the min-max normalization to
Table 2: Results of boosting networks on MNIST, Cifar-10, and Cifar-100. All models are trained by SplitLBI in 1000 epochs ($\kappa = 1, \nu = 1000$). [·] indicates the corresponding layer boosted.

|       | number@size (Min) | Min  | number@size (Max) | Max  | O.P. | Boost |
|-------|------------------|------|------------------|------|------|-------|
| MNIST | [8@3 × 3]        | 96.11| [8@3 × 3]        | 98.61| 98.67| 98.40 |
|       | [2@3 × 3]        | 46.11| [8@3 × 3]        | 63.45| 67.56| 63.00 |
| Cifar-10 | 8@5 × 5-[2@3 × 3]| 51.95| 8@5 × 5-[172@3 × 3]| 80.90| 84.37| 79.57 |
|       | 8@5 × 5-8@3 × 3-2@3 × 3 | 57.21| 8@5 × 5-8@3 × 3-190@3 × 3 | 80.69| 84.90| 78.78 |
| Cifar-100 | 8@5 × 5-[2@3 × 3]| 22.37| 8@5 × 5-[154@3 × 3]| 52.99| 58.17| 52.22 |
|       | 8@5 × 5-8@3 × 3-2@3 × 3 | 23.00| 8@5 × 5-8@3 × 3-180@3 × 3 | 53.19| 58.57| 49.89 |

the gradient image, and generate the final visualization map. The right figure compares the visualized gradient images of first convolutional layer of 64 filters with $7 \times 7$ receptive fields. We visualize the models parameters at 20 (purple), 40 (green), and 60 (black) epochs, respectively, which corresponds to the bounding boxes in the right figure annotated by the corresponding colors, i.e., purple, green, and black. We order the gradient images produced from 64 filters by the descending order of the magnitude ($\ell_2$-norm) of filters, i.e., images are ordered from the upper left to the bottom right. For comparison, we also provide the visualized gradient from random initialized weights. We find that the filters with high magnitudes mostly focus on the texture and shape information, while color information is kept in the filters with small magnitudes. This phenomenon is in accordance with observation of Abbasi-Asl and Yu [2017] that filters mainly contain color information can be pruned for saving computational cost. Moreover, among the filters of high magnitudes, most of them capture non-semantic textures while few pursue shapes. This shows that the first convolutional layer of ResNet-18 trained on ImageNet learned non-semantic textures rather than shape to do image classification tasks, in accordance with recent studies [Geirhos et al., 2019].

5.2 Boosting a Network

In this experiment, we explore the performance that SplitLBI boosts filters of a network with a fixed number of layers. In each individual layer, we start from a small initial number of filters (Min), and gradually grow to a maximum number of filters (Max), adaptively learned by the SplitLBI-Boosting described in Sec. 4. We compare the results of directly using small number of filters, maximum number of filters (Max) and our boosting method in Table 2. Note that, the maximum number of filters are computed by our boosting algorithm. We do not know it beforehand. In addition, we also include the networks with over-parameterized (O.P.) convolutional filters, which is twice the maximum number of filters of each corresponding layer. By default, all models are trained to 1000 epochs. We employ MNIST, Cifar-10 and Cifar-100 as the evaluation benchmarks here.

As shown in Table 2, our SplitLBI can successfully boost the filters of networks from Min to Max. In Fig. 4, Fig. 5, Fig. 6, and 7, we give the full training paths of Min (Minimum Model), Max (Maximum Model) and Boosting (Grow) as compared in Table 2. Each black line indicates filters are added. In our experiments, two filters ($m = 2$) are added each time. To the best of our knowledge, this is the first algorithm that can simultaneously learn the network structures and parameters from data. It is desired that the Max and Boosting hit the same
performance with enough training epochs; and impressively, after the 1000 epochs, Max and Boosting can indeed achieve almost comparable results on all of the two layer networks. This further indicates the efficacy of proposed SplitLBI. On Cifar-100 with a three convolutional layer network, the results of boosting are inferior to that of Max. A further inspection on the full training paths of Min, Max and Boosting (Fig. 7) shows that the boosting may require more training epochs in such a relative complex structures. Thus empirically, we fix the Boosted network in Table 2, continue training another 500 epochs, and we find the trained model indeed can report almost the same performance as the Max.

6 Conclusion

In this paper, a parsimonious deep learning method is proposed based on differential inclusions of inverse scale spaces. Its simple discretization – SplitLBI has a provable global convergence on DNNs, and thus is employed to train deep networks. Furthermore, the proposed method can be applied to boost a network by simultaneously learning both network structure and parameters. The extensive experiments and visualization show that the proposed method may not only have better efficiency than SGD, but also capture the structural sparsity of convolutional filters for interpretability.
Figure 5: One Layer Network on Cifar-10 dataset. Each dash black line indicates that new filters are added.

Figure 6: Two Layer Network on Cifar-10 and Cifar-100 datasets. Each dash black line indicates that new filters are added.
Figure 7: Three Layer Network on Cifar-10 and Cifar-100 datasets. Each dash black line indicates that new filters are added.

Appendix

A Proof of Theorem 1

A.1 Kurdyka-Łojasiewicz Property

To introduce the definition of the Kurdyka-Łojasiewicz (KL) property, we need some notions and notations from variational analysis, which can be found in Rockafellar and Wets [1998].

The notion of subdifferential plays a central role in the following definitions. For each $x \in \text{dom}(h) := \{x \in \mathbb{R}^p : h(x) < +\infty\}$, the Fréchet subdifferential of $h$ at $x$, written $\hat{\partial}h(x)$, is the set of vectors $v \in \mathbb{R}^p$ which satisfy

$$\liminf_{y \to x, y \neq x} \frac{h(y) - h(x) - \langle v, y - x \rangle}{\|y - x\|} \geq 0.$$ 

When $x \notin \text{dom}(h)$, we set $\hat{\partial}h(x) = \emptyset$. The limiting-subdifferential (or simply subdifferential) of $h$ introduced in Mordukhovich [2006], written $\partial h(x)$ at $x \in \text{dom}(h)$, is defined by

$$\partial h(x) := \{v \in \mathbb{R}^p : \exists x^k \to x, h(x^k) \to h(x), v^k \in \hat{\partial}h(x^k) \to v\}. \quad (16)$$

A necessary (but not sufficient) condition for $x \in \mathbb{R}^p$ to be a minimizer of $h$ is $0 \in \partial h(x)$. A point that satisfies this inclusion is called limiting-critical or simply critical. The distance between a point $x$ to a subset $\mathcal{S}$ of $\mathbb{R}^p$, written $\text{dist}(x, \mathcal{S})$, is defined by $\text{dist}(x, \mathcal{S}) = \inf\{\|x - s\| : s \in \mathcal{S}\}$, where $\| \cdot \|$ represents the Euclidean norm.
Let $h : \mathbb{R}^p \rightarrow \mathbb{R} \cup \{+\infty\}$ be an extended-real-valued function (respectively, $h : \mathbb{R}^p \Rightarrow \mathbb{R}^q$ be a point-to-set mapping), its graph is defined by

\[
\text{Graph}(h) := \{(x, y) \in \mathbb{R}^p \times \mathbb{R} : y = h(x)\},
\]

(resp. \(\text{Graph}(h) := \{(x, y) \in \mathbb{R}^p \times \mathbb{R}^q : y \in h(x)\}\)), and its domain by \(\text{dom}(h) := \{x \in \mathbb{R}^p : h(x) < +\infty\}\) (resp. \(\text{dom}(h) := \{x \in \mathbb{R}^p : h(x) \neq \emptyset\}\)). When $h$ is a proper function, i.e., when \(\text{dom}(h) \neq \emptyset\), the set of its global minimizers (possibly empty) is denoted by

\[
\arg \min h := \{x \in \mathbb{R}^p : h(x) = \inf h\}.
\]

The KL property [Lojasiewicz, 1963, 1993, Kurdyka, 1998, Bolte et al., 2007a,b] plays a central role in the convergence analysis of nonconvex algorithms [Attouch et al., 2013, Wang et al., 2019]. The following definition is adopted from Bolte et al. [2007b].

**Definition 1** (Kurdyka-Lojasiewicz property). A function $h$ is said to have the Kurdyka-Lojasiewicz (KL) property at $\bar{u} \in \text{dom}(\partial h) := \{v \in \mathbb{R}^n | \partial h(v) \neq \emptyset\}$, if there exists a constant $\eta \in (0, \infty)$, a neighborhood $N$ of $\bar{u}$ and a function $\phi : [0, \eta) \rightarrow \mathbb{R}_+$, which is a concave function that is continuous at 0 and satisfies $\phi(0) = 0$, $\phi \in C^1((0, \eta))$, i.e., $\phi$ is continuous differentiable on $(0, \eta)$, and $\phi'(s) > 0$ for all $s \in (0, \eta)$, such that for all $u \in N \cap \{u \in \mathbb{R}^n | h(\bar{u}) < h(u) < h(\bar{u}) + \eta\}$, the following inequality holds

\[
\phi'(h(u) - h(\bar{u})) \cdot \text{dist}(0, \partial h(u)) \geq 1.
\]

(17)

If $h$ satisfies the KL property at each point of $\text{dom}(\partial h)$, $h$ is called a KL function.

KL functions include real analytic functions, semialgebraic functions, tame functions defined in some o-minimal structures [Kurdyka, 1998, Bolte et al., 2007b], continuous subanalytic functions [Bolte et al., 2007a] and locally strongly convex functions. In the following, we provide some important examples that satisfy the Kurdyka-Lojasiewicz property.

**Definition 2** (Real analytic). A function $h$ with domain an open set $U \subset \mathbb{R}$ and range the set of either all real or complex numbers, is said to be real analytic at $u$ if the function $h$ may be represented by a convergent power series on some interval of positive radius centered at $u$: $h(x) = \sum_{j=0}^{\infty} \alpha_j (x-u)^j$, for some $\{\alpha_j\} \subset \mathbb{R}$. The function is said to be real analytic on $V \subset U$ if it is real analytic at each $u \in V$ [Krantz and Parks, 2002, Definition 1.1.5]. The real analytic function $f$ over $\mathbb{R}^p$ for some positive integer $p > 1$ can be defined similarly.

According to Krantz and Parks [2002], typical real analytic functions include polynomials, exponential functions, and the logarithm, trigonometric and power functions on any open set of their domains. One can verify whether a multivariable real function $h(x)$ on $\mathbb{R}^p$ is analytic by checking the analyticity of $g(t) := h(x + ty)$ for any $x, y \in \mathbb{R}^p$.

**Definition 3** (Semialgebraic).

(a) A set $D \subset \mathbb{R}^p$ is called semialgebraic [Bochnak et al., 1998] if it can be represented as

\[
D = \bigcup_{i=1}^{s} \bigcap_{j=1}^{t} \{x \in \mathbb{R}^p : P_{ij}(x) = 0, Q_{ij}(x) > 0\},
\]

where $P_{ij}, Q_{ij}$ are real polynomial functions for $1 \leq i \leq s, 1 \leq j \leq t$. 

15
(b) A function \( h : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\} \) (resp. a point-to-set mapping \( h : \mathbb{R}^p \rightrightarrows \mathbb{R}^q \)) is called semialgebraic if its graph \( \text{Graph}(h) \) is semialgebraic.

According to Lojasiewicz [1965], Bochnak et al. [1998] and Shiota [1997, I.2.9, page 52], the class of semialgebraic sets are stable under the operation of finite union, finite intersection, Cartesian product or complementation. Some typical examples include polynomial functions, the indicator function of a semialgebraic set, and the Euclidean norm [Bochnak et al., 1998, page 26].

A.2 KL Property in Deep Learning and Proof of Corollary 1

In the following, we consider the deep neural network training problem. Consider a \( l \)-layer feedforward neural network including \( l - 1 \) hidden layers of the neural network. Particularly, let \( d_i \) be the number of hidden units in the \( i \)-th hidden layer for \( i = 1, \ldots, l - 1 \). Let \( d_0 \) and \( d_l \) be the number of units of input and output layers, respectively. Let \( W^i \in \mathbb{R}^{d_i \times d_{i-1}} \) be the weight matrix between the \((i - 1)\)-th layer and the \( i \)-th layer for any \( i = 1, \ldots, l - 1 \).

According to Theorem 1, one major condition is to verify the introduced Lyapunov function \( F \) defined in (11) satisfies the Kurdyka-Lojasiewicz property. For this purpose, we need an extension of semialgebraic set, called the o-minimal structure (see, for instance Coste [1999], van den Dries [1986], Kurdyka [1998], Bolte et al. [2007b]). The following definition is from Bolte et al. [2007b].

Definition 4 (o-minimal structure). An o-minimal structure on \((\mathbb{R}, +, \cdot)\) is a sequence of boolean algebras \( \mathcal{O}_n \) of “definable” subsets of \( \mathbb{R}^n \), such that for each \( n \in \mathbb{N} \)

(i) if \( A \) belongs to \( \mathcal{O}_n \), then \( A \times \mathbb{R} \) and \( \mathbb{R} \times A \) belong to \( \mathcal{O}_{n+1} \);

(ii) if \( \Pi : \mathbb{R}^{n+1} \to \mathbb{R}^n \) is the canonical projection onto \( \mathbb{R}^n \), then for any \( A \) in \( \mathcal{O}_{n+1} \), the set \( \Pi(A) \) belongs to \( \mathcal{O}_n \);

(iii) \( \mathcal{O}_n \) contains the family of algebraic subsets of \( \mathbb{R}^n \), that is, every set of the form

\[ \{ x \in \mathbb{R}^n : p(x) = 0 \}, \]

where \( p : \mathbb{R}^n \to \mathbb{R} \) is a polynomial function.

(iv) the elements of \( \mathcal{O}_1 \) are exactly finite unions of intervals and points.

Based on the definition of o-minimal structure, we can show the definition of the definable function.

Definition 5 (Definable function). Given an o-minimal structure \( \mathcal{O} \) (over \((\mathbb{R}, +, \cdot)\)), a function \( f : \mathbb{R}^n \to \mathbb{R} \) is said to be definable in \( \mathcal{O} \) if its graph belongs to \( \mathcal{O}_{n+1} \).

According to van den Dries and Miller [1996], Bolte et al. [2007b], there are some important facts of the o-minimal structure, shown as follows.

\(^{1}\)To simplify notations, we regard the input and output layers as the 0-th and the \( l \)-th layers, respectively, and absorb the bias of each layer into \( W^i \).

16
(i) The collection of semialgebraic sets is an o-minimal structure. Recall the semialgebraic sets are Boolean combinations of sets of the form
\[ \{ x \in \mathbb{R}^n : p(x) = 0, q_1(x) < 0, \ldots, q_m(x) < 0 \}, \]
where \( p \) and \( q_i \)'s are polynomial functions in \( \mathbb{R}^n \).

(ii) There exists an o-minimal structure that contains the sets of the form
\[ \{ (x, t) \in [-1, 1]^n \times \mathbb{R} : f(x) = t \} \]
where \( f \) is real-analytic around \([-1, 1]^n\).

(iii) There exists an o-minimal structure that contains simultaneously the graph of the exponential function \( \mathbb{R} \ni x \mapsto \exp(x) \) and all semialgebraic sets.

(iv) The o-minimal structure is stable under the sum, composition, the inf-convolution and several other classical operations of analysis.

The Kurdyka-Lojasiewicz property for the smooth definable function and non-smooth definable function were established in [Kurdyka, 1998, Theorem 1] and [Bolte et al., 2007b, Theorem 11], respectively. Now we are ready to present the proof of Corollary 1.

**Proof of Corollary 1.** To justify this corollary, we only need to verify the associated Lyapunov function \( F \) satisfies Kurdyka-Lojasiewicz inequality. In this case and by (12), \( F \) can be rewritten as follows
\[ F(W, \Gamma, \mathcal{G}) = \alpha \left( \hat{\mathcal{L}}(W, \Gamma) + \frac{1}{2\nu} \|W - \Gamma\|^2 \right) + \Omega(\Gamma) + \Omega^*(g) - \langle W, g \rangle. \]

Because \( \ell \) and \( \sigma_i \)'s are definable by assumptions, then \( \hat{\mathcal{L}}(W, \Gamma) \) are definable as compositions of definable functions. Moreover, according to Krantz and Parks [2002], \( \|W - \Gamma\|^2 \) and \( \langle W, g \rangle \) are semi-algebraic and thus definable. Since the group Lasso \( \Omega(\Gamma) = \sum_g \|\Gamma_g\|_2 \) is the composition of \( \ell_2 \) and \( \ell_1 \) norms, and the conjugate of group Lasso penalty is the maximum of group \( \ell_2 \)-norm, i.e. \( \Omega^*(\Gamma) = \max_g \|\Gamma_g\|_2 \), where the \( \ell_2 \), \( \ell_1 \), and \( \ell_{\infty} \) norms are definable, hence the group Lasso and its conjugate are definable as compositions of definable functions. Therefore, \( F \) is definable and hence satisfies Kurdyka-Lojasiewicz inequality by [Kurdyka, 1998, Theorem 1].

The verifications of these specific cases listed in assumptions can be found in the proof of [Zeng et al., 2019, Proposition 1]. This finishes the proof of this corollary. \( \square \)

**A.3 Proof of Theorem 1**

Our analysis is mainly motivated by a recent paper [Benning et al., 2017], as well as the influential work [Attouch et al., 2013]. According to [Attouch et al., 2013, Lemma 2.6], there are mainly four ingredients in the analysis, that is, the sufficient descent property, relative error property, continuity property of the generated sequence and the Kurdyka-Lojasiewicz property of the function. More specifically, we first establish the sufficient descent property of the generated sequence via exploiting the Lyapunov function \( F \) (see, (11)) in Lemma A.4.
in Section A.4, and then show the relative error property of the sequence in Lemma A.5 in Section A.5. The continuity property is guaranteed by the continuity of $\mathcal{L}(W, \Gamma)$ and the relation $\lim_{k \to \infty} B_{\Omega}^{\nu}(\Gamma_{k+1}, \Gamma_k) = 0$ established in Lemma 1(i) in Section A.4. Thus, together with the Kurdyka-Lojasiewicz assumption of $F$, we establish the global convergence of SLBI following by [Attouch et al., 2013, Lemma 2.6].

Let $(\bar{W}, \bar{\Gamma}, \bar{g})$ be a critical point of $F$, then the following holds

$$\partial_W F(\bar{W}, \bar{\Gamma}, \bar{g}) = \alpha (\nabla \mathcal{L}_n(\bar{W}) + \nu^{-1}(\bar{W} - \bar{\Gamma})) = 0,$$

$$\partial_T F(\bar{W}, \bar{\Gamma}, \bar{g}) = \alpha \nu^{-1}(\bar{\Gamma} - \bar{W}) + \partial \Omega(\bar{\Gamma}) - \bar{g} \geq 0,$$

$$\partial_g F(\bar{W}, \bar{\Gamma}, \bar{g}) = \bar{\Gamma} - \partial \Omega^*(\bar{g}) \geq 0.$$  \hfill (18)

By the final inclusion and the convexity of $\Omega$, it implies $\bar{g} \in \partial \Omega(\bar{\Gamma})$. Plugging this inclusion into the second inclusion yields $\alpha \nu^{-1}(\bar{\Gamma} - \bar{W}) = 0$. Together with the first equality implies

$$\nabla \mathcal{L}(\bar{W}, \bar{\Gamma}) = 0, \quad \nabla \mathcal{L}_n(\bar{W}) = 0.$$

This finishes the proof of this theorem.

### A.4 Sufficient Descent Property along Lyapunov Function

Let $P_k := (W_k, \Gamma_k)$, and $Q_k := (P_k, g_{k-1}), k \in \mathbb{N}$. In the following, we present the sufficient descent property of $Q_k$ along the Lyapunov function $F$.

**Lemma.** Suppose that $\mathcal{L}_n$ is continuously differentiable and $\nabla \mathcal{L}_n$ is Lipschitz continuous with a constant $\text{Lip} > 0$. Let $\{Q_k\}$ be a sequence generated by SLBI with a finite initialization. If $0 < \alpha < \frac{2}{\text{Lip} + \nu^{-1}}$, then

$$F(Q_{k+1}) - F(Q_k) \leq \rho \|Q_{k+1} - Q_k\|_2^2,$$

where $\rho := \frac{1}{\kappa} - \alpha \frac{(\text{Lip} + \nu^{-1})}{2}$.

**Proof.** By the optimality condition of (9a) and also the inclusion $p_k = [0, g_k]^T \in \partial R(P_k)$, there holds

$$\kappa (\alpha \nabla \mathcal{L}(P_k) + p_{k+1} - p_k) + P_{k+1} - P_k = 0,$$

which implies

$$-\langle \alpha \nabla \mathcal{L}(P_k), P_{k+1} - P_k \rangle = \kappa^{-1} \|P_{k+1} - P_k\|_2^2 + D(\Gamma_{k+1}, \Gamma_k) \quad \hfill (19)$$

where

$$D(\Gamma_{k+1}, \Gamma_k) := \langle g_{k+1} - g_k, \Gamma_{k+1} - \Gamma_k \rangle.$$

Noting that $\mathcal{L}(P) = \mathcal{L}_n(W) + \frac{1}{2\nu} \|W - \Gamma\|^2_2$ and by the Lipschitz continuity of $\nabla \mathcal{L}_n(W)$ with a constant $\text{Lip} > 0$ implies $\nabla \mathcal{L}$ is Lipschitz continuous with a constant $\text{Lip} + \nu^{-1}$. This implies

$$\mathcal{L}(P_{k+1}) \leq \mathcal{L}(P_k) + \langle \nabla \mathcal{L}(P_k), P_{k+1} - P_k \rangle + \frac{\text{Lip} + \nu^{-1}}{2} \|P_{k+1} - P_k\|_2^2.$$
Substituting the above inequality into (19) yields
\[
\alpha \hat{L}(P_{k+1}) + D(\Gamma_{k+1}, \Gamma_k) + \rho\|P_{k+1} - P_k\|^2 \leq \alpha \hat{L}(P_k).
\]  

(20)

Adding some terms in both sides of the above inequality and after some reformulations implies
\[
\alpha \hat{L}(P_{k+1}) + B^{\theta_k}_{\Omega}(\Gamma_{k+1}, \Gamma_k)
\leq \alpha \hat{L}(P_k) + B^{\theta_k-1}_{\Omega}(\Gamma_k, \Gamma_{k-1}) - \rho\|P_{k+1} - P_k\|^2 - (D(\Gamma_{k+1}, \Gamma_k) + B^{\theta_k}_{\Omega}(\Gamma_k, \Gamma_{k-1}) - B^{\theta_k}_{\Omega}(\Gamma_{k+1}, \Gamma_k))
\]

(21)

where the final equality holds for
\[
D(\Gamma_{k+1}, \Gamma_k) - B^{\theta_k}_{\Omega}(\Gamma_{k+1}, \Gamma_k) = B^{\theta_k+1}_{\Omega}(\Gamma_k, \Gamma_{k-1}).
\]

Therefore, both
\[
\alpha \hat{L}(P_{k+1}) + B^{\theta_k}_{\Omega}(\Gamma_{k+1}, \Gamma_k)
\]

and the finiteness of Lemma 1.

Suppose that assumptions of Lemma A.4 hold. Suppose further that Assumption 1 (b)-(d) hold. Then

(i) both \(\alpha\{\hat{L}(P_k)\}\) and \(\{F(Q_k)\}\) converge to the same finite value, and \(\lim_{k \to \infty} B^{\theta_k}_{\Omega}(\Gamma_{k+1}, \Gamma_k) = 0\).

(ii) the sequence \(\{(W_k, \Gamma_k, g_k)\}\) is bounded,

(iii) \(\lim_{k \to \infty} \|P_{k+1} - P_k\|^2 = 0\) and \(\lim_{k \to \infty} D(\Gamma_{k+1}, \Gamma_k) = 0\),

(iv) \(\frac{1}{K} \sum_{k=0}^K \|P_{k+1} - P_k\|^2 \to 0\) at a rate of \(O(1/K)\).

Proof. By (20), \(\hat{L}(P_k)\) is monotonically decreasing due to \(D(\Gamma_{k+1}, \Gamma_k) \geq 0\). Similarly, by (23), \(F(Q^k)\) is also monotonically decreasing. By the lower boundedness assumption of \(\hat{L}_n(W)\), both \(\hat{L}(P)\) and \(F(Q)\) are lower bounded by their definitions, i.e., (2) and (11), respectively. Therefore, both \(\{\hat{L}(P_k)\}\) and \(\{F(Q_k)\}\) converge, and it is obvious that \(\lim_{k \to \infty} F(Q_k) \geq \lim_{k \to \infty} \alpha \hat{L}(P_k)\). By (22),

\[
B^{\theta_k-1}_{\Omega}(\Gamma_k, \Gamma_{k-1}) \leq F(Q_k) - F(Q_{k+1}), \ k = 1, \ldots
\]

By the convergence of \(F(Q_k)\) and the nonnegativeness of \(B^{\theta_k-1}_{\Omega}(\Gamma_k, \Gamma_{k-1})\), there holds

\[
\lim_{k \to \infty} B^{\theta_k-1}_{\Omega}(\Gamma_k, \Gamma_{k-1}) = 0.
\]

By the definition of \(F(Q_k) = \alpha \hat{L}(P_k) + B^{\theta_k-1}_{\Omega}(\Gamma_k, \Gamma_{k-1})\) and the above equality, it yields

\[
\lim_{k \to \infty} F(Q_k) = \lim_{k \to \infty} \alpha \hat{L}(P_k).
\]

Since \(\hat{L}_n(W)\) has bounded level sets, then \(W_k\) is bounded. By the definition of \(\hat{L}(W, \Gamma)\) and the finiteness of \(\hat{L}(W_k, \Gamma_k)\), \(\Gamma_k\) is also bounded due to \(W_k\) is bounded. The boundedness of \(g_k\) is due to \(g_k \in \partial \Omega(\Gamma_k)\), condition (d), and the boundedness of \(\Gamma_k\).
By (23), summing up (23) over \(k = 0, 1, \ldots, K\) yields
\[
\sum_{k=0}^{K} (\rho \|P_{k+1} - P_k\|^2 + D(\Gamma_{k+1}, \Gamma_k)) < \alpha \bar{L}(P_0) < \infty. \tag{24}
\]

Letting \(K \to \infty\) and noting that both \(\|P_{k+1} - P_k\|^2\) and \(D(\Gamma_{k+1}, \Gamma_k)\) are nonnegative, thus
\[
\lim_{k \to \infty} \|P_{k+1} - P_k\|^2 = 0, \quad \lim_{k \to \infty} D(\Gamma_{k+1}, \Gamma_k) = 0.
\]

Again by (24),
\[
\frac{1}{K} \sum_{k=0}^{K} (\rho \|P_{k+1} - P_k\|^2 + D(\Gamma_{k+1}, \Gamma_k)) < K^{-1} \alpha \bar{L}(P_0),
\]
which implies \(\frac{1}{K} \sum_{k=0}^{K} \|P_{k+1} - P_k\|^2 \to 0\) at a rate of \(O(1/K)\).

\[\square\]

**A.5 Relative Error Property**

In this subsection, we provide the bound of subgradient by the discrepancy of two successive iterates. By the definition of \(F\) (11),
\[
H_{k+1} := \begin{pmatrix}
\alpha \nabla W \hat{L}(W_{k+1}, \Gamma_{k+1}) \\
\alpha \nabla \Gamma \hat{L}(W_{k+1}, \Gamma_{k+1}) + g_{k+1} - g_k
\end{pmatrix} \in \partial F(Q_{k+1}), \quad k \in \mathbb{N}. \tag{25}
\]

**Lemma.** Under assumptions of Lemma 1, then
\[
\|H_{k+1}\| \leq \rho_1 \|Q_{k+1} - Q_k\|, \quad \text{for } H_{k+1} \in \partial F(Q_{k+1}), \quad k \in \mathbb{N},
\]
where \(\rho_1 := 2\alpha^{-1} + \alpha (\text{Lip} + 2\nu^{-1})\). Moreover, \(\frac{1}{K} \sum_{k=1}^{K} \|H_k\|^2 \to 0\) at a rate of \(O(1/K)\).

**Proof.** Note that
\[
\nabla W \hat{L}(W_{k+1}, \Gamma_{k+1}) = (\nabla W \hat{L}(W_{k+1}, \Gamma_{k+1}) - \nabla W \hat{L}(W_{k+1}, \Gamma_k)) + (\nabla W \hat{L}(W_{k+1}, \Gamma_k) - \nabla W \hat{L}(W_k, \Gamma_k)) + \nabla W \hat{L}(W_k, \Gamma_k).
\]

By the definition of \(\hat{L}\) (see (2)),
\[
\|\nabla W \hat{L}(W_{k+1}, \Gamma_{k+1}) - \nabla W \hat{L}(W_{k+1}, \Gamma_k)\| = \nu^{-1} \|\Gamma_k - \Gamma_{k+1}\|,
\]
\[
\|\nabla W \hat{L}(W_{k+1}, \Gamma_k) - \nabla W \hat{L}(W_k, \Gamma_k)\| = \|(\nabla \hat{L}_n(W_{k+1}) - \nabla \hat{L}_n(W_k)) + \nu^{-1}(W_{k+1} - W_k)\|
\leq (\text{Lip} + \nu^{-1})\|W_{k+1} - W_k\|,
\]

where the last inequality holds for the Lipschitz continuity of \(\nabla \hat{L}_n\) with a constant \(\text{Lip} > 0\), and by (10a),
\[
\|\nabla W \hat{L}(W_k, \Gamma_k)\| = (\kappa \alpha)^{-1}\|W_{k+1} - W_k\|.
\]

20
Substituting the above (in)equalities into (26) yields
\[
\|\nabla W \tilde{L}(W_{k+1}, \Gamma_{k+1})\| \leq \left[ (\kappa \alpha)^{-1} + \text{Lip} + \nu^{-1} \right] \cdot \|W_{k+1} - W_k\| + \nu^{-1}\|\Gamma_{k+1} - \Gamma_k\|
\]
Thus,
\[
\|\alpha \nabla W \tilde{L}(W_{k+1}, \Gamma_{k+1})\| \leq \left[ \kappa^{-1} + \alpha(\text{Lip} + \nu^{-1}) \right] \cdot \|W_{k+1} - W_k\| + \alpha \nu^{-1}\|\Gamma_{k+1} - \Gamma_k\|. \tag{27}
\]
By (10c), it yields
\[
g_{k+1} - g_k = \kappa^{-1}(\Gamma_k - \Gamma_{k+1}) - \alpha \nabla \Gamma \tilde{\mathcal{L}}(W_k, \Gamma_k).
\]
Noting that \(\nabla \Gamma \tilde{\mathcal{L}}(W_k, \Gamma_k) = \nu^{-1}(\Gamma_k - W_k)\), and after some simplifications yields
\[
\|\alpha \nabla \Gamma \tilde{\mathcal{L}}(W_{k+1}, \Gamma_{k+1}) + g_{k+1} - g_k\| = \left\| (\kappa^{-1} - \alpha \nu^{-1}) \cdot (\Gamma_k - \Gamma_{k+1}) + \alpha \nu^{-1}(W_k - W_{k+1})\right\| \\
\leq \alpha \nu^{-1}\|W_k - W_{k+1}\| + (\kappa^{-1} - \alpha \nu^{-1})\|\Gamma_k - \Gamma_{k+1}\|, \tag{28}
\]
where the last inequality holds for the triangle inequality and \(\kappa^{-1} > \alpha \nu^{-1}\) by the assumption.

By (27), (28), and the definition of \(H_{k+1}\) (25), there holds
\[
\|H_{k+1}\| \leq \left[ \kappa^{-1} + \alpha(\text{Lip} + 2\nu^{-1}) \right] \cdot \|W_{k+1} - W_k\| + \left( \kappa^{-1} + 1 \right)\|\Gamma_{k+1} - \Gamma_k\| \\
\leq \left[ 2\kappa^{-1} + 1 + \alpha(\text{Lip} + 2\nu^{-1}) \right] \cdot \|P_{k+1} - P_k\| \leq \left[ 2\kappa^{-1} + 1 + \alpha(\text{Lip} + 2\nu^{-1}) \right] \cdot \|Q_{k+1} - Q_k\|. \tag{29}
\]

By (29) and Lemma 1(iv), \(\frac{1}{K} \sum_{k=1}^{K} \|H_k\|^2 \to 0\) at a rate of \(O(1/K)\).

This finishes the proof of this lemma.

\[\square\]

References

R. Abbasi-Asl and B. Yu. Structural compression of convolutional neural networks based on greedy filter pruning. \textit{arXiv preprint arXiv:1705.07356}, 2017. 1, 1, 5.1

Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. 2018. arXiv:1811.03962. 1

J. M. Alvarez and M. Salzmann. Learning the number of neurons in deep networks. In \textit{NIPS}, 2016. 1

S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. \textit{arXiv preprint arXiv:1802.05296}, 2018. 1

H. Attouch, J. Bolte, and B. F. Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. \textit{Mathematical Programming}, 137:91–129, 2013. 1, A.1, A.3

P. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. In \textit{The 31st Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA}. 2017. 1

21
P. L. Bartlett. For valid generalization the size of the weights is more important than the size of the network. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 134–140. MIT Press, 1997.

M. Benning, M. M. Betcke, M. J. Ehrhardt, and C.-B. Schönlieb. Choose your path wisely: gradient descent in a bregman distance framework. arXiv preprint arXiv:1712.04045, 2017.

J. Bochnak, M. Coste, and M.-F. Roy. *Real algebraic geometry*, volume 3. Ergeb. Math. Grenzgeb. Springer-Verlag, Berlin, 1998.

J. Bolte, A. Daniilidis, and A. Lewis. The Lojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17:1205–1223, 2007a.

J. Bolte, A. Daniilidis, A. Lewis, and M. Shiota. Clark subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18:556–572, 2007b.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010.

M. Burger, G. Gilboa, S. Osher, and J. Xu. Nonlinear inverse scale space methods. *Communications in Mathematical Sciences*, 4(1):179–212, 2006.

M. Collins and P. Kohli. Memory bounded deep convolutional networks. In arXiv preprint arXiv:1412.1442, 2014.

M. Coste. *An introduction to o-minimal geometry*. RAAG Notes, 81 pages, Institut de Recherche Mathematiques de Rennes, 1999.

D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001.

S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. 2018. arXiv:1811.03804.

D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations (ICLR)*, 2019. arXiv preprint arXiv:1803.03635.

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *International Conference on Learning Representations (ICLR)*, 2019. arXiv preprint arXiv:1811.12231.

N. Golowich, A. Rakhlin, and O. Shamir. Size-independent sample complexity of neural networks. *Conference on Learning Theory (COLT)*, 2018. arXiv preprint arXiv:1712.06541.
S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 1

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5, 5.1

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*, 2014. 4

C. Huang and Y. Yao. A unified dynamic approach to sparse model selection. In *The 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, Lanzarote, Spain, 2018. 2, 3

C. Huang, X. Sun, J. Xiong, and Y. Yao. Split lbi: An iterative regularization path with structural sparsity. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS) 29*, pages 3369–3377. 2016. 2

C. Huang, X. Sun, J. Xiong, and Y. Yao. Boosting with structural sparsity: A differential inclusion approach. *Applied and Computational Harmonic Analysis*, 2018. arXiv preprint arXiv:1704.04833. 2

M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014. 1

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

S. Krantz and H. R. Parks. *A primer of real analytic functions*. Birkhäuser, second edition, 2002. 2, A.2

K. Kurdyka. On gradients of functions definable in o-minimal structures. *Annales de l’institut Fourier*, 48:769–783, 1998. A.1, A.1, A.2, A.2

Y. LeCun et al. Lenet-5, convolutional neural networks. URL: http://yann.lecun.com/exdb/lenet, 20, 2015. 5

H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *ICLR*, 2017. 1

Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2016. 4

S. Lojasiewicz. Une propriétetopologique des sous-ensembles analytiques réels. In: Les Équations aux dérivées partielles. *Éditions du centre National de la Recherche Scientifique, Paris*, pages 87–89, 1963. A.1

S. Lojasiewicz. *Ensembles semi-analytiques*. Institut des Hautes Etudes Scientifiques, 1965. A.1

S. Lojasiewicz. Sur la geometrie semi-et sous-analytique. *Annales de l’institut Fourier*, 43:1575–1595, 1993. A.1
I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*, 2019. arXiv preprint arXiv:1711.05101.

S. Mei, A. Montanari, and P.-M. Nguyen. A mean field view of the landscape of two-layers neural network. *Proceedings of the National Academy of Sciences (PNAS)*, 2018.

S. Mei, T. Misiakiewicz, and A. Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. *Conference on Learning Theory (COLT)*, 2019.

B. S. Mordukhovich. *Variational analysis and generalized differentiation I: Basic Theory*. Springer, 2006.

A. Nemirovski and D. Yudin. *Problem complexity and Method Efficiency in Optimization*. New York: Wiley, 1983. Nauka Publishers, Moscow (in Russian), 1978.

B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations (ICLR)*, New Orleans, Louisiana, USA. 2019.

S. Osher, F. Ruan, J. Xiong, Y. Yao, and W. Yin. Sparse recovery via differential inclusions. *Applied and Computational Harmonic Analysis*, 2016.

A. Pentina and C. H. Lampert. Lifelong learning with non-i.i.d. tasks. In *NIPS*. 2015.

R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*. Grundlehren Math. Wiss. 317, Springer-Verlag, New York, 1998.

M. Shiota. *Geometry of subanalytic and semialgebraic sets*, volume 150 of *Progress in Mathematics*. Birkhäuser, Boston, 1997.

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806, 2014.

X. Sun, L. Hu, Y. Yao, and Y. Wang. Gsplit lbi: Taming the procedural bias in neuroimaging for disease prediction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 107–115. Springer, 2017.

S. Thrun and T. M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 1995.

J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inform. Theory*, 50(10):2231–2242, 2004.

L. van den Dries. A generalization of the tarski-seidenberg theorem and some nondefinability results. *Bull. Amer. Math. Soc. (N.S.)*, 15:189–193, 1986.

L. van den Dries and C. Miller. Geometric categories and o-minimal structures. *Duke Mathematical Journal*, 84:497–540, 1996.

L. Venturi, A. S. Bandeira, and J. Bruna. Spurious valleys in two-layer neural network optimization landscapes. 2018. arXiv:1802.06384.
Y. Wang, D. Ramanan, and M. Hebert. Growing a brain: Fine-tuning by increasing model capacity. In CVPR, 2017. 4

Y. Wang, W. Yin, and J. Zeng. Global convergence of admm in nonconvex nonsmooth optimization. Journal of Scientific Computing, 78(1):29–63, 2019. A.1

Y. Wei, F. Yang, and M. J. Wainwright. Early stopping for kernel boosting algorithms: A general analysis with localized complexities. The 31st Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 2017. 1

W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning the number of neurons in deep networks. In NIPS, 2016. 1

C. Wong, N. Houlsby, Y. Lu, and A. Gesmundo. Transfer learning with neural automl. In NIPS, 2018. 4

H. Yang, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. In IJCAI 2018, 2018. 1

Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. Constructive Approximation, 26(2):289–315, 2007. 1

J. Yoon and S. J. Hwang. Combined group and exclusive sparsity for deep neural networks. In ICML, 2017. 1

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68(1):49–67, 2006. 1, 2

J. Zeng, T. T.-K. Lau, S.-B. Lin, and Y. Yao. Global convergence of block coordinate descent in deep learning. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, 2019. A.2

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. International Conference on Learning Representations (ICLR), 2017. arXiv:1611.03530. 1

X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(10):1943–1955, 2016. 1

B. Zhao, X. Sun, Y. Fu, Y. Yao, and Y. Wang. Mspli lbi: Realizing feature selection and dense estimation simultaneously in few-shot and zero-shot learning. In International Conference on Machine Learning (ICML), 2018. 2

P. Zhao and B. Yu. On model selection consistency of lasso. J. Machine Learning Research, 7: 2541–2567, 2006. 1, 2

A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. ICLR, 2017. 1

C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. ICLR, 2017. 1