Constrained Implicit Learning Framework for Neural Network Sparsification

Alicia Y. Tsai ALICIATSAI@BERKELEY.EDU
University of California, Berkeley

Wenzhi Gao GWZ@STANFORD.EDU
Stanford University

Laurent El Ghaoui LAURENT.EG@VINUNI.EDU.VN
VinUniveristy; University of California, Berkeley

Editors: Vu Nguyen and Hsuan-Tien Lin

Abstract

This paper presents a novel approach to sparsify neural networks by transforming them into implicit models characterized by an equilibrium equation rather than the conventional hierarchical layer structure. Unlike traditional sparsification techniques reliant on network structure or specific loss functions, our method simplifies the process to a simple constrained least-squared problem with sparsity-inducing constraints or penalties. Additionally, we introduce a scalable algorithm that can be parallelized, addressing the computational complexities associated with this transformation while maintaining efficiency. Experimental results on CIFAR-100 and 20NewsGroup datasets demonstrate the high effectiveness of our method, particularly in scenarios with high pruning rates. This approach offers a versatile and efficient solution for neural network parameter reduction. Furthermore, we observe that a moderate subset of the training data suffices to achieve competitive performance, highlighting the robustness and information-capturing capability of our approach.

Keywords: implicit models; neural network sparsification; constrained LASSO

1. Introduction

Compared to traditional neural networks, implicit neural network (Bai et al., 2019a; Chen et al., 2018; Gu et al., 2020) offer the advantage that their model parameters are simple data matrices. Consequently, after training an implicit model, we can perform model sparsification by running lease-squares-based feature selection tasks. In contrast, traditional neural networks require retraining from scratch with different regularization parameters for sparsification. The architecture of implicit neural networks, as proposed by El Ghaoui et al. (2021); Bai et al. (2019b), has gained increasing popularity due to its simplicity and ability to integrate various traditional neural network architectures. Theories and applications originally developed for traditional neural networks are progressively being extended to implicit networks Peng et al. (2022); Gao and Gao (2022); Geng et al. (2021); Gu et al. (2020).

Deep neural networks are often found to be highly redundant, with a small subset of network coefficients retaining the majority of inference power Ashouri et al. (2018); Denil et al. (2013). This observation has spurred a branch of research dedicated to sparsifying neural network coefficients without compromising inference efficiency. Common methods for network sparsification include: 1) incorporating sparsity-inducing regularization terms into the training objective Louizos et al. (2017); Scardapane et al. (2017), 2) modifying the neural network architecture Wan et al. (2013), and 3) applying post-training sparsification procedures Sun et al. (2016); Louizos et al. (2017); Garg and Candan (2020); Ashouri et al. (2018) that strike a balance between accuracy and sparsity.

While existing sparsification methods yield satisfactory results on established networks, they come with several limitations. Firstly, these techniques tend to be architecture-dependent, necessitating distinct sparsification strategies for different network architectures. Additionally, these methods must navigate the complex interplay between the training objective and sparsity, often resulting in challenging optimization problems. Consequently, devising a universal sparsification scheme for traditional neural networks remains a formidable challenge. In light of this, our paper explores the emerging branch of implicit deep learning networks. As elaborated in the following section, sparsifying an implicit neural network can be conceptualized as a straightforward least-squares problem with sparsity penalties or constraints, irrespective of the underlying network architecture.

2. Constrained Implicit Learning Framework

Notations. Throughout the paper, we use n, m to denote the number of internal states and input samples; p, q denote the dimension of input and output vectors, respectively. Given a matrix V, $|V|$ denotes taking its absolute value element-wise; $\|V\|_0$ denotes the number of non-zero entries; $\|V\|_\infty$ denotes matrix infinity norm; $\|V\|_F = (\sum_{i,j} V_{ij}^2)^{1/2}$ denotes its Frobenious norm.

2.1. Implicit Model

We are given a dataset with input $U \in \mathbb{R}^{p\times m}$ and output $Y \in \mathbb{R}^{q\times m}$, where each column of U and Y represents an input or output vector. An implicit model consists of an equilibrium equation (E) in a "state matrix" $X \in \mathbb{R}^{n\times m}$ and a prediction equation (P):

$$X = \phi(AX + BU) \tag{E}$$

$$\hat{Y}(U) = CX + DU \tag{P}$$

where $\phi : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ is a strictly increasing nonlinear activation function such as ReLU, tanh, or sigmoid. Matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{q\times n}$ and $D \in \mathbb{R}^{q\times p}$ are model parameters. In equation (E), the input feature matrix $U \in \mathbb{R}^{p\times m}$ is passed through a linear transformation B and the internal state matrix X is obtained as the fixed-point solution to equation (E). The output prediction $\hat{Y}$ is then obtained by feeding the state X through the prediction equation (P). The structure is illustrated in Figure 1, where the

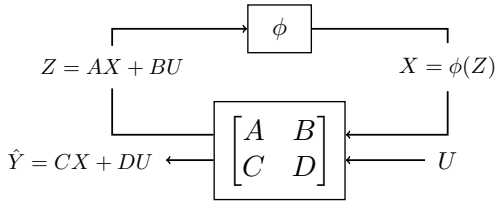

Figure 1: A diagram view of an implicit model, where Z is the pre-activation state "before" passing through the activation function ϕ and X is the post-activation state "after" passing through ϕ.

"pre-activation" and "post-activation" state matrices Z, X are shown; in those matrices, each column corresponds to a single data point.

We provide a simple example of constructing X and Z from a 3-layer fully-connected network of the form:

$$\hat{y}(u) = W_2 x_2,\ x_2 = \phi(W_1 x_1)\ x_1 = \phi(W_0 x_0),\ x_0 = u,$$

where u is a single vector input. For notational simplicity, we exclude the bias terms, which can be easily accounted for by considering the vector $(u, 1)$ instead of u. Each column of Z and X corresponds to the state from a single input. The column z is formed by stacking all the intermediate layers before passing through ϕ and the column x is formed by stacking all the intermediate layers after passing through ϕ:

$$z = \begin{pmatrix} W_1 x_1 \\ W_0 x_0 \end{pmatrix}, \quad x = \phi(z) = \begin{pmatrix} x_2 \\ x_1 \end{pmatrix}.$$

In this example, we can easily verify that its equivalent implicit form is as follows:

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{cc|c} 0 & W_1 & 0 \\ 0 & 0 & W_0 \\ \hline W_2 & 0 & 0 \end{array}\right).$$

For a more complicated network, determining an equivalent implicit form can be a non-trivial task.

2.2. Constrained Implicit Model

The constrained implicit learning framework trains an implicit model with a constraint: it should match both the state X and output $\hat{Y}$ of another "baseline" (implicit or layered) model when the same inputs U are applied. The framework allows us to consider any baseline deep neural networks without ever having to address this challenge: we simply need to extract the pre- and post-activation state matrices. For a given baseline model, the state matrix X can be obtained by running a set of fixed-point iterations (if the baseline is implicit), or a simple forward pass (if the baseline is a standard layered network). In both

cases, we can extract the pre-activation state matrix Z, such that the post-activation state matrix satisfies $X = \phi(Z)$. Each column of matrices Z and X corresponds to a single data point; when the baseline is a layered network, these matrices are constructed by stacking all the intermediate layers into a long column vector, with the first intermediate layer at the bottom and the last intermediate layer at the top.

The conditions $Z = AX + BU$ and $\hat{Y} = CX + DU$ *characterize* the implicit models that match both the state and outputs of the baseline model. We then solve a *convex problem* to find another well-posed model that satisfies Definition 1, with a desired task in mind, under the matching conditions:

$$\begin{aligned} \min_{A,B,C,D} \quad & \ell(A,B,C,D) && \text{(2a)} \\ \text{subject to} \quad & Z = AX + BU, && \text{(2b)} \\ & \hat{Y} = CX + DU, && \text{(2c)} \\ & \|A\|_\infty \leq \kappa. && \text{(2d)} \end{aligned}$$

where ℓ is a user-defined loss function and $\kappa \leq 1$ is a hyper-parameter for model well-posedness define in Definition (1). The reader can refer to Appendix B for proof of the sufficient condition for well-posedness.

Definition 1 (well-posedness) *The $n \times n$ matrix A is said to be well-posed for ϕ if, for any $b \in \mathbb{R}^n$, the solution $x \in \mathbb{R}^n$ of the following equation $x = \phi(Ax + b)$ exists and is unique.*

2.3. Equality Constraints Relaxation

For the equality constraints (2b) and (2c) that match the internal state and exact output, it is not necessary to insist on an exact match. This allows us to relax (2b) and (2c) by introducing penalty terms into the objective function:

$$\begin{aligned} \min_{(A,B,C,D)\in\mathcal{C}, \|A\|_\infty \leq \kappa} \ell(A,B,C,D) + \lambda_1 \|Z - (AX + BU)\|_F^2 \\ + \lambda_2 \|\hat{Y} - (CX + DU)\|_F^2 \end{aligned} \tag{3}$$

where ℓ and $\mathcal{C}$ are user-defined objective function and constraint set for model parameters. λ_1 and λ_2 are hyper-parameters that control the degree of state- and output-matching.

The training problem (3) can be decomposed into a series of parallel, smaller problems, each involving a single row, or a block of rows, if ℓ is decomposable. This is usually the case for most objective functions used in neural network training. For a single row $(a^\top, b^\top)$ of (A, B), and with $z^\top$ the corresponding row in Z, the problem reduces to

$$\begin{aligned} \min_{a,b} \quad & \ell(a,b) + \lambda_1 \left\| z - \left(X^\top, U^\top\right) \begin{pmatrix} a \\ b \end{pmatrix} \right\|^2 && \text{(4)} \\ \text{subject to} \quad & \|a\|_1 \leq \kappa, \end{aligned}$$

where $\|a\|_1 \leq \kappa$ is the well-posedness condition since $\|A\|_\infty$ is separable in terms of rows. The problem of finding C, D is independent of that relative to A, B and takes the same form as problem (4) without having to satisfy the well-posedness condition:

$$\min_{c,d} \quad \ell(c,d) + \lambda_2 \left\| \hat{y} - \left(X^\top, U^\top\right) \begin{pmatrix} c \\ d \end{pmatrix} \right\|^2. \tag{5}$$

When $\ell = 0$, (4) reduces to the basis pursuit problem introduced in Chen and Donoho (1994), for which efficient optimization algorithms have been established in the past years Toh and Yun (2010); Polson et al. (2015). We note that $\ell(c,d)$ can be set to a common sparsity-inducing penalty such as ℓ_1-norm. In the next section, we provide tailored algorithms for implicit model sparsification for solving problem (4).

3. Algorithm for Model Sparsification

In this section, we discuss algorithms for model sparsification on implicit models. On the one hand, note that (5) is a least squares problem and any SGD with its variants can solve it efficiently (Garrigos and Gower, 2023). On the other hand, (4) is more challenging due to the ℓ_1 constraint, and is also known to be LASSO in the constrained form (Tibshirani, 1996). Although variants of projected gradient descent might also work when directly applied to (4), each projection involve a projection onto ℓ_1 norm ball (Laurent, 2016), which is less practical when dimension of (a, b) is large. Moreover, state-of-the-art preprocessing techniques cannot be directly applied to this constrained formulation (Wang, 2015; Ghaoui et al., 2010).

In view of the aforementioned challenge, we propose an alternative algorithm that solves (4) by reducing it to a sequence of least-squares problems with ℓ_1 penalty, to which state-of-the-art LASSO techniques apply. Our algorithm performs bisection on the LASSO regularization path, till an approximate regularization parameter is found to solve the original constrained problem.

3.1. Algorithm Design

For brevity of exposition, we overload notation and let

$$M := (X^\top, U^\top), \quad a := (a, b),$$

thereby getting the following constrained LASSO problem

$$P(\kappa) := \min_a \ \frac{1}{2}\|Ma - z\|^2 \quad \text{subject to} \quad \|a\|_1 \leq \kappa,$$

We assume that the constraint is not trivially satisfied.

A1: (Non-trivial solution) $0 \leq \kappa < \|(M^\top M)^{-1} M^\top z\|_1$.

Note that now $\|a\|_1 \leq \kappa$ simultaneously imposes both the sparsity and the well-posedness constraint.

We define $h(a) := \frac{1}{2}\|Ma - z\|^2$ and use $h(a)$ and $\frac{1}{2}\|Ma - z\|^2$ interchangeably. Before we proceed, we first introduce the unconstrained LASSO problem, which will serve as an important subroutine of our analysis:

$$Q(\lambda) := \min_a \; h(a) + \lambda\|a\|_1 \quad \text{(Unconstrained LASSO)}.$$

3.2. $P(\kappa)$ and Root Finding

Our method is closely related to the following function

$$g(\lambda) := \|a^*_{Q(\lambda)}\|_1, \tag{6}$$

$$a^*_{Q(\lambda)} = \arg\min_a \{h(a) + \lambda\|a\|_1\} \tag{7}$$

and we apply a bisection method to identify some λ such that $g(\lambda) = \kappa$. Before we establish our algorithm, we first show the intuition of our method by analyzing basic properties of $g(\lambda)$.

Lemma 2 *$g(\lambda)$ satisfies the following properties:*

- *$g(\lambda)$ is continuous and piecewise linear. (Osborne et al., 2000)*
- *$g(\lambda_1) - g(\lambda_2) \leq 0$ if $\lambda_1 > \lambda_2$. That is, $g(\lambda)$ is monotonically decreasing.*
- *$g(0) = \|(M^\top M)^{-1}M^\top z\| \leq \frac{1}{m}\|M^\top z\|$.*
- *$g(\lambda) = 0$ if $\lambda \geq \|M^\top z\|_\infty$.*

Lemma 3 *Given $0 < \kappa < \|(M^\top M)^{-1}M^\top z\|_1$, there exists $\lambda_\kappa > 0$ such that $a^*_{P(\kappa)} = a^*_{Q(\lambda_\kappa)}$*

Lemma 2 shows that by identifying a correct λ, the solution to $Q(\lambda)$ and $P(\kappa)$ coincide. Lemma 3 implies that $g(\lambda)$ is monotonic and that finding λ_κ such that $g(\lambda_\kappa) = \kappa$ suffices to give us $a^*_{P(\kappa)}$. Therefore it is natural to employ a bisection routine to identify λ_κ.

Now we are ready to propose the bisection method that searches for the matching λ_κ over the LASSO regularization path.

3.3. Analysis of the Bisection Method

Now that we have established the intuition of the bisection method, we turn to the analysis of method. Now that we know that $\lambda_\kappa \in (0, \|M^\top z\|_\infty)$. Bisection method ends in at most

$$T = \mathcal{O}(\log(1/\varepsilon))$$

iterations. Combined with the standard result on proximal gradient method, we immediately get a complexity for the number of internal iterations.

Algorithm 1 Bisection algorithm on LASSO regularization path

input $\kappa, M, z, \varepsilon$
initialize $u = \|M^\top z\|, l = 0$
while $u - l \geq \varepsilon$
  set $\lambda = \frac{1}{2}(u + l)$
  solve $a^*_{Q(\lambda)} = \arg\min_a \{h(a) + \lambda\|a\|_1\}$
  if $\|a^*_{Q(\lambda)}\|_1 \geq \kappa$
    $l \leftarrow \lambda$
  else
    $u \leftarrow \lambda$
  end
output $\lambda, a^*_{Q(\lambda)}$

Lemma 4 (Karimi et al. (2016)) *If we apply proximal gradient method to $Q(\lambda)$, then*

$$h(a^k) - h(a^*) \leq \exp(-(\mu/L)K)\{h(a^0) - h(a^*)\}$$

where L is the largest eigenvalue of $M^\top M$ and μ is the Polyak-Lojasiewicz constant of the problem; a^k is the output of the k-th iteration; a^* is the unique optimal solution to $Q(\lambda)$.

Intuitively we have linear convergence in both internal and external loops, giving

$$K = \mathcal{O}(T \cdot \log(1/\varepsilon)) = \mathcal{O}(\log^2(1/\varepsilon))$$

worst-case iteration complexity. However, in the next subsection, we show that we can further improve the convergence rate by warm-starting consecutive bisection problems.

3.4. Effect of Warm-Starting

Lemma 5 (Effect of warm-starting) *Given λ_1, λ_2, we have*

$$h(a^*_{Q(\lambda_1)}) + \lambda_1\|a^*_{Q(\lambda_1)}\|_1 - [h(a^*_{Q(\lambda_2)}) + \lambda_2\|a^*_{Q(\lambda_2)}\|_1] \leq \sqrt{n}|\lambda_1 - \lambda_2|A,$$

where $A = \sup_\lambda \|a^*_{Q(\lambda)}\|_\infty$.

Given Lemma 4, we are guaranteed that if λ_1, λ_2 are sufficiently close, warming-starting $Q(\lambda_1)$ with the optimal solution to $Q(\lambda_2)$ would result in a small initial optimality gap bounded by $\sqrt{n}|\lambda_1 - \lambda_2|A$. In other words, the closer we are to λ_κ, the faster each subproblem can be solved.

Theorem 6 *If we warm-start each $Q(\lambda)$, then $K = \mathcal{O}(\log^2(1/\varepsilon))$.*

Theorem 6 establishes that employing a warm-start strategy for initializing each $Q(\lambda)$ restricts the computational complexity K to grow at a rate slower than the square of the logarithm of the inverse of the desired precision ε. This finding facilitates the initialization of the problem with a warm start, leading to notable performance enhancements across successive solver iterations. The complete algorithm is presented in Algorithm 2.

Algorithm 2 Constrained Implicit Model Sparsification

input Data matrix U; A trained standard (layered) neural network $\mathcal{N} : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \times \mathbb{R}$; Hyper-parameter $\kappa \leq 1$ and ε.
initialize Run a single forward pass on $\mathcal{N}$ with U to obtain outputs $\hat{Y}$, *i.e.,* $\hat{Y} = \mathcal{N}(U)$.
Collect all intermediate layers before and after passing through the activation ϕ.
Construct Z and X by stacking all intermediate layers.
set $M := (X^T \; U^T), W := (A, B) = \mathbf{0}$
for each row z_i of Z
  warm start $Q(\lambda_i)$ with the optimal solution to $Q(\lambda_{i-1})$
  solve $Q(\lambda_i)$ with Algorithm 1
  update row i of $W \leftarrow a^*_{i\,Q(\lambda)}$
  end
output W

4. Numerical Experiments

We demonstrate the effectiveness of the implicit model sparsification on CIFAR-100 (Krizhevsky, 2009) and 20NewsGroup[1], leveraging the existing ResNet-20 network (Devries and Taylor, 2017) and DistilBERT model[2] (Sanh et al., 2019) as baseline architectures. The test accuracy is 92.1% for CIFAR-100 and 92.8% for 20NewsGroup, respectively. Throughout the remainder of our paper, we maintain a fixed $\varepsilon = 0.01$. These experiments were solved using Mosek (ApS, 2022) optimization solvers. We also measure the efficiency of the method in terms of the total number of iterations required on the traversal of all rows within the weight matrix.

Table 1: Sparsity levels and accuracy on the CIFAR-100 and 20NewsGroup data.

	CIFAR-100		20NewsGroup	
κ	**Accuracy (%)**	**Sparsity (%)**	**Accuracy (%)**	**Sparsity (%)**
0.005	76.7	98.1	60.9	97.8
0.01	79.1	96.4	74.2	95.9
0.05	84.2	95.2	80.2	95.3
0.1	87.1	93.8	87.2	93.3
0.5	89.0	93.3	88.8	90.7
0.99	91.0	90.1	90.6	87.4
Dense	92.1	0	92.8	0

Table 1 illustrates the trade-off between sparsity and test performance for the CIFAR-100 and 20NewsGroup datasets, employing warm-starting as outlined in Algorithm 2. The hyper-parameter κ serves as a direct control for the sparsity level of the model, where higher κ values correspond to lower sparsity levels. Our results demonstrate the algorithm's capability in computing highly sparse networks, with a mere 2% reduction in test accuracy

1. http://qwone.com/~jason/20Newsgroups/
2. https://huggingface.co/docs/transformers/model_doc/distilbert

observed on both datasets compared to the original dense networks. As the number of non-zero elements increases (i.e., sparsity decreases), a gradual decline in test accuracy is evident, which is the common trade-off between model complexity and performance. However, even with approximately 10% of the model weights retained, the models maintain competitive performance levels, exhibiting only a marginal decrease. This resilience highlights the robustness of our method in scenarios where a significant portion of the model undergoes pruning.

Table 2: Sparsity levels and accuracy on the CIFAR-100 and 20NewsGroup data.

	CIFAR-100		20NewsGroup	
Method	**Accuracy (%)**	**Sparsity (%)**	**Accuracy (%)**	**Sparsity (%)**
SSS	88.4	44.4	89.5	48.7
SPR	89.8	45.9	90.6	50.5
MLA	89.1	50.0	90.3	51.8
Ours	91.0	90.1	90.6	87.4
Dense	92.1	0	92.8	0

In our evaluation, we compare our method with several optimization-based parameter reduction techniques: SSS (Huang and Wang, 2018), SPR (Cacciola et al., 2022), and MLA (Hu et al., 2019). SSS and SPR approach the task as a sparse regularized optimization problem, using ℓ_1-relaxation and perspective relaxation, respectively. MLA focuses on aligning semantic information between intermediate outputs and overall model performance by incorporating feature and semantic correlation losses along with a classification loss, similar to our state- and outputs-matching conditions. We use ResNet-20 for SSS and SPR and ResNet-18 for MLA when experimenting on CIFAR-100 to ensure comparability with the original papers. We replicate the same hyper-parameter settings as reported in the original paper. Table 2 demonstrates that our method surpasses all the baselines while achieving a significantly larger reduction in parameters.

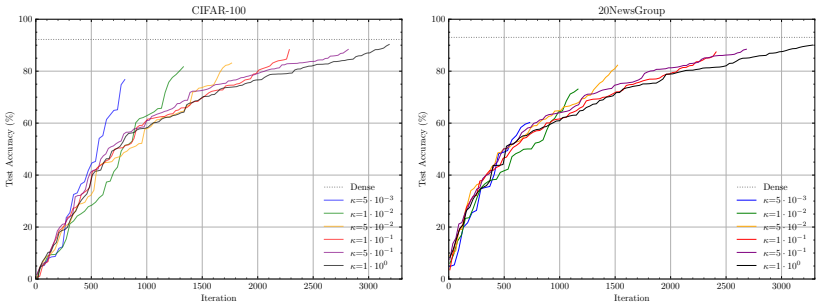

Figure 2: Performance of different κ at each iteration on CIFAR-100 and 20NewsGroup.

Figure 2 depicts individual runs of the algorithm across various values of κ, illustrating the corresponding test performance at each iteration. To accelerate the runtime, we leverage warm-starting within the bisection inner loop, initializing it with the optimal solution obtained from the preceding inner iteration. This strategy allows for a more efficient exploration of the solution space. Comparison between $\kappa = 1$, 0.1, and 0.5, with and without

warm-starting, is presented in Figure 3. Warm-starting yields an average speedup of 1.8x in convergence and reduces computational overhead.

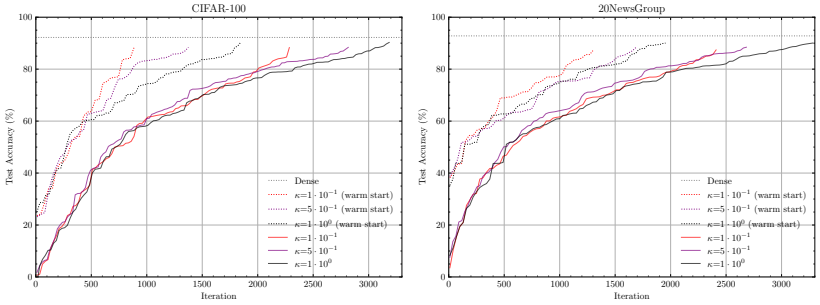

Figure 3: Performance of different κ at each iteration with warm starting.

In solving the problem (4), the input matrix $U \in \mathbb{R}^{p\times m}$ need not encompass the entire training dataset. To investigate the minimum sample size required for effectively training a sparse implicit model, we conducted experiments varying the number of samples. Figure 4 illustrates the performance of different κ values trained with partial data. As the percentage of total training samples increases, corresponding to higher m for the input matrix U, the dimension of M also increases. From Figure 4, it's evident that test performance initially improves with increasing training data. However, it eventually reaches a plateau, stabilizing at around 20% of the training data for CIFAR-100 and approximately 30% for 20NewsGroup. For very small values of κ, the model may be excessively sparse, hindering effective learning. Conversely, with sufficiently large κ, training can be streamlined by utilizing only a partial dataset. These results highlight the significance of the state matrix X as a high-quality representation, capable of capturing a substantial amount of underlying information. Consequently, it's feasible to train a model with significantly fewer training samples. Furthermore, Table 3 presents the computational speed-up achieved during training, highlighting the significant efficiency gains associated with utilizing partial datasets. This analysis presents the practical implications of our findings in terms of computational scalability and resource utilization.

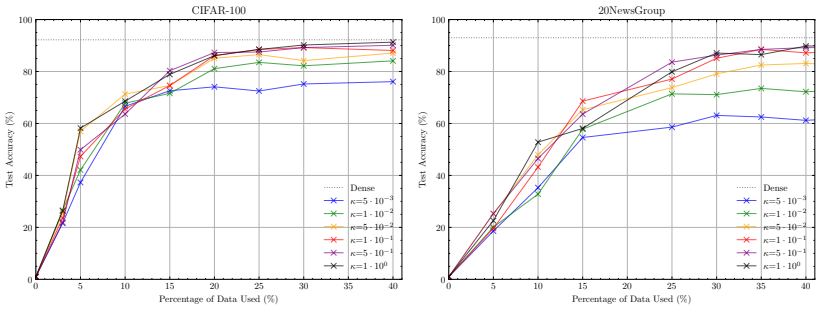

Figure 4: Performance of different κ trained with partial data.

Table 3: Training computational speed up by using warm-start and partial data matrix U.

	CIFAR-100		20NewsGroup	
κ	**Warm Start**	**20% Data**	**Warm Start**	**30% Data**
0.1	2.2x	9.1x	1.5x	8.7x
0.5	2.0x	7.8x	1.6x	8.4x
0.99	1.7x	7.5x	1.7x	7.4x

5. Discussion

In conclusion, this work introduces a novel paradigm in neural network sparsification—Implicit Model Sparsification. Departing from conventional techniques that often rely on complex network structures or specialized loss functions, our method offers a simplified approach. Implicit Model Sparsification is achieved through a straightforward least-squares problem, augmented with sparsity-inducing constraints or penalties. This simplicity enhances the applicability of the approach to a broad spectrum of neural network architectures. To ensure the scalability and practicality of our method, we have developed a parallel algorithm. This algorithm addresses the computational complexities inherent in transforming neural networks into implicit models, while preserving efficiency and effectiveness. Our experimental findings on CIFAR-100 and 20NewsGroup datasets show the remarkable efficacy of our approach, particularly its robust performance in scenarios involving high pruning rates. Furthermore, our investigation into the optimal sample size for training sparse implicit models reveals insightful trends. We observe that a moderate subset of the training data suffices to achieve competitive performance, highlighting the resilience and information-capturing capability of our approach. In essence, Implicit Model Sparsification offers a versatile and new paradigm for neural network compression, with promising implications for real-world applications through its simplicity, scalability, and effectiveness.

References

MOSEK ApS. *The MOSEK Optimizer API for Python 9.3.21*, 2022. URL https://docs.mosek.com/latest/pythonapi/index.html.

Amir H Ashouri, Tarek S Abdelrahman, and Alwyn Dos Remedios. Fast on-the-fly retraining-free sparsification of convolutional neural networks. *arXiv preprint arXiv:1811.04199*, 2018.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper/2019/file/01386bd6d8e091c2ab4c7c7de644d37b-Paper.pdf.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019b.

Matteo Cacciola, Antonio Frangioni, Xinlin Li, and Andrea Lodi. Deep neural networks pruning via the structured perspective regularization. *arXiv:2206.14056 [cs.LG]*, June 2022. URL https://arxiv.org/pdf/2206.14056.pdf.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Shaobing Chen and David Donoho. Basis pursuit. In *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 41–44. IEEE, 1994.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26, 2013.

Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. URL http://arxiv.org/abs/1708.04552.

Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021. doi: 10.1137/20M1358517. URL https://doi.org/10.1137/20M1358517.

Tianxiang Gao and Hongyang Gao. On the optimization and generalization of overparameterized implicit neural networks. *arXiv preprint arXiv:2209.15562*, 2022.

Yash Garg and K Selçuk Candan. isparse: Output informed sparsification of neural network. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pages 180–188, 2020.

Guillaume Garrigos and Robert M Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*, 2023.

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. *Advances in Neural Information Processing Systems*, 34:24247–24260, 2021.

Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems. *arXiv preprint arXiv:1009.4219*, 2010.

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. *Advances in Neural Information Processing Systems*, 33:11984–11995, 2020.

Yiming Hu, Siyang Sun, Jianquan Li, Jiagang Zhu, Xingang Wang, and Qingyi Gu. Multi-loss-aware channel pruning of deep networks. *2019 IEEE International Conference on Image Processing (ICIP)*, pages 889–893, 2019.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. *ECCV*, 2018.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 795–811. Springer, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Condat Laurent. Fast projection onto the simplex and the l1 ball. *Math. Prog*, 158:575–585, 2016.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Michael R Osborne, Brett Presnell, and Berwin A Turlach. A new approach to variable selection in least squares problems. *IMA journal of numerical analysis*, 20(3):389–403, 2000.

Hongwu Peng, Deniz Gurevin, Shaoyi Huang, Tong Geng, Weiwen Jiang, Orner Khan, and Caiwen Ding. Towards sparsification of graph neural networks. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, pages 272–279. IEEE, 2022.

Nicholas G Polson, James G Scott, and Brandon T Willard. Proximal algorithms in statistics and machine learning. 2015.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

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

Yi Sun, Xiaogang Wang, and Xiaoou Tang. Sparsifying neural network connections for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4856–4864, 2016.

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

Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of optimization*, 6(615-640):15, 2010.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR, 2013.

Yun Wang. *Feature screening for the lasso*. PhD thesis, Princeton University, 2015.

Appendix A. Proof of main results

A.1. Proof of Lemma 2

Recall that $\lambda_1 > \lambda_2$. Then by optimality condition we have

$$\begin{aligned}\frac{1}{2}\|Ma^*_{Q(\lambda_1)} - z\|^2 + \lambda_1\|a^*_{Q(\lambda_1)}\|_1 \leq& \frac{1}{2}\|Ma^*_{Q(\lambda_2)} - z\|^2 + \lambda_1\|a^*_{Q(\lambda_2)}\|_1\\ \frac{1}{2}\|Ma^*_{Q(\lambda_2)} - z\|^2 + \lambda_2\|a^*_{Q(\lambda_2)}\|_1 \leq& \frac{1}{2}\|Ma^*_{Q(\lambda_1)} - z\|^2 + \lambda_2\|a^*_{Q(\lambda_1)}\|_1\end{aligned}$$

Summing over the above two relations and re-arranging the terms, we have

$$(\lambda_1 - \lambda_2)(g(\lambda_1) - g(\lambda_2)) = (\lambda_1 - \lambda_2)(\|a^*_{Q(\lambda_1)}\|_1 - \|a^*_{Q(\lambda_2)}\|_1) \leq 0.$$

Dividing both sides by $\lambda_1 - \lambda_2 > 0$ completes the proof of monotonicity.

For the second condition, we know that $a^*_{Q(0)} = (M^\top M)^{-1}M^\top y$ and

$$\|a^*_{Q(0)}\|_1 = \|(M^\top M)^{-1}M^\top z\| \leq \|(M^\top M)^{-1}\| \cdot \|M^\top z\| \leq \frac{1}{m}\|M^\top z\|.$$

also we have

$$\begin{aligned}\frac{1}{2}\|z\|^2 &= \frac{1}{2}\|Ma - z - Ma\|^2\\ &= \frac{1}{2}\|Ma - z\|^2 - \langle Ma, Ma - z\rangle + \frac{1}{2}\|Ma\|^2\\ &= \frac{1}{2}\|Ma - z\|^2 + \langle Ma, z\rangle\\ &\leq \frac{1}{2}\|Ma - z\|^2 + \|M^\top z\|_\infty\|a\|_1.\end{aligned}$$

and this completes the proof.

A.2. Proof of Lemma 3

First by **Lemma 2** we know there exists λ_κ such that $g(\lambda_\kappa) = \kappa \in (0, \|(M^\top M)^{-1}M^\top z\|_1)$.

Then we verify that for any $a^*_{Q(\lambda_\kappa)} = \arg\min\left\{\frac{1}{2}\|Ma - z\|^2 + \lambda_\kappa\|a\|_1\right\}$,

$$a^*_{Q(\lambda_\kappa)} = \arg\min_{\|a\|_1 \leq \kappa}\left\{\frac{1}{2}\|Ma - z\|^2\right\}.$$

First recall that the optimality condition of $Q(\lambda_\kappa)$ tells that

$$\begin{aligned}0 \in \partial_{a=a^*_{Q(\lambda_\kappa)}}\left\{\tfrac{1}{2}\|Ma - z\|^2 + \lambda_\kappa\|a\|_1\right\} = M^\top(Ma^*_{Q(\lambda_\kappa)} - y)&\\ +\lambda_\kappa\partial(\|a^*_{Q(\lambda_\kappa)}\|_1)&\end{aligned}$$

Then we can plug $a^*_{Q(\lambda_\kappa)}, \lambda_\kappa$ into the optimality conditions of $P(\kappa)$ and verify that

$$\begin{aligned}\|a^*_{Q(\lambda_\kappa)}\|_1 \leq& \kappa\\ \lambda_\kappa \geq& 0\\ \lambda_\kappa(\kappa - \|a^*_{Q(\lambda_\kappa)}\|_1) =& 0\\ \partial_{a=a^*_{Q(\lambda_\kappa)}} \left\{\frac{1}{2}\|Ma - z\|^2 + \lambda_\kappa(\|a\|_1 - \kappa)\right\} \ni& 0,\end{aligned}$$

which completes the proof.

A.3. Proof of Lemma 4

Without loss of generality assume that $\lambda_1 > \lambda_2$. Then

$$\begin{aligned} h(a^*_{Q(\lambda_1)}) + \lambda_1\|a^*_{Q(\lambda_1)}\|_1 &\leq h(a^*_{Q(\lambda_2)}) + \lambda_1\|a^*_{Q(\lambda_2)}\|_1 \\ &= h(a^*_{Q(\lambda_2)}) + \lambda_2\|a^*_{Q(\lambda_2)}\|_1 \qquad (8)\\ &\quad + (\lambda_1 - \lambda_2)\|a^*_{Q(\lambda_2)}\|_1 \\ &\leq h(a^*_{Q(\lambda_2)}) + \lambda_2\|a^*_{Q(\lambda_2)}\|_1 \qquad (9)\\ &\quad + \sqrt{n}|\lambda_1 - \lambda_2|A. \end{aligned}$$

Exchanging the position of λ_1, λ_2, this completes the proof.

A.4. Proof of Theorem 6

Let $I(\gamma, \varepsilon)$ be the number of iterations for proximal gradient method to converge, given tolerate ε and initial optimality gap γ. We know, from **Lemma 3** that

$$I(\gamma, \varepsilon) = \mu^{-1}L\log(\gamma/\varepsilon).$$

Defining $\Delta = \|(M^\top M)^{-1}M^\top z\|_1 \cdot A$, we have, for iteration t, that

$$\gamma_t \leq 2^{-t}\sqrt{n}\Delta + \varepsilon_{t-1}$$

and

$$I(\gamma_t, \varepsilon_t) \leq \mu^{-1}L\log\left(\frac{2^{-t}\sqrt{n}\Delta + \varepsilon_{t-1}}{\varepsilon^\top}\right).$$

Taking $\varepsilon^\top \equiv \varepsilon, T = \log_2(1/\varepsilon)$, we have

$$
\begin{aligned}
K &= \sum_{t=1}^{\top} I(\gamma_t, \varepsilon_t) \\
&\leq \mu^{-1} L \sum_{t=1}^{\top} \log\left(\frac{\sqrt{n}\Delta}{\varepsilon} 2^{-t} + 1\right) \\
&\lesssim \mu^{-1} L \sum_{t=1}^{\top} \log\left(\frac{\sqrt{n}\Delta}{2^{t-T}} + 1\right) \\
&= \mathcal{O}(\log^2(1/\varepsilon))
\end{aligned}
$$

Appendix B. Sufficient Condition for Well-posedness

The forward pass of an implicit model relies on the fixed-point solution of the underlying equilibrium equation, while a backward pass requires one to differentiate this equation with respect to the model parameters (A, B, C, D). The solution to the equilibrium equation (E) does not necessarily exists nor be unique. We say that an equilibrium equation with activation map ϕ is well-posed if the following *well-posedness* condition is satisfied (El Ghaoui et al., 2021).

Theorem 7 (PF sufficient condition for well-posedness) *Assume that ϕ is strictly increasing. Then, A is well-posed for any such ϕ if $\lambda_{pf}(|A|) < 1$. Moreover, the solution x of equation (E) can be computed via the fixed point iterations $x \to \phi(Ax + b)$, with initial condition $x = 0$.*

Theorem 8 (Re-scaled implicit model) *Assume that ϕ is strictly increasing and positively homogeneous, i.e., $\phi(\alpha x) = \alpha\phi(x)$ for any $\alpha \geq 0$ and x. For a neural network $\mathcal{N}$ with its equivalent implicit form $(A_\mathcal{N}, B_\mathcal{N}, C_\mathcal{N}, D_\mathcal{N}, \phi)$, where $A_\mathcal{N}$ satisfies PF sufficient condition for well-posedness of Theorem (7), there exists a linearly-rescaled equivalent implicit model $(A'_\mathcal{N}, B'_\mathcal{N}, C'_\mathcal{N}, D'_\mathcal{N}, \phi)$ with $\|A'_\mathcal{N}\|_\infty < 1$ that gives the same output $\hat{Y}$ as the original $\mathcal{N}$ for any input U.*

Consider a standard layer-based neural network $\mathcal{N} : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \times \mathbb{R}$ with non-linear activation ϕ that is strictly increasing and maps input feature matrix $U \in \mathbb{R}^{p \times m}$ to outputs $Y = \mathcal{N}(U)$ via hidden layers. As shown in El Ghaoui et al. (2021), for such networks, there exists an equivalent implicit model, $(A_\mathcal{N}, B_\mathcal{N}, C_\mathcal{N}, D_\mathcal{N}, \phi)$ as in (1). Without loss of generality, we may re-scale the original weight matrices of $\mathcal{N}$ to obtain a strongly well-posed implicit model, $(A'_\mathcal{N}, B'_\mathcal{N}, C'_\mathcal{N}, D'_\mathcal{N}, \phi)$, by Theorem (8), in the sense that $\|A'_\mathcal{N}\|_\infty < 1$. This result also allows us to consider the convex constraint $\|A\|_\infty < 1$ as a sufficient condition as opposed to the non-convex PF sufficient condition, in light of the bound $\lambda_{\text{pf}}(A) \leq \|A\|_\infty$.