A Sequential Greedy Approach for Training Implicit Deep Models

Tanmay Gautam, Brendon G. Anderson, Somayeh Sojoudi, and Laurent El Ghaoui

***Abstract*— Recent works in deep learning have demonstrated impressive performance using "implicit deep models," wherein conventional architectures composed of forward-propagating, differentiable parametric layers are replaced by more expressive models composed of an implicitly defined fixed-point equation together with a prediction equation. Methods for training implicit deep models are currently restricted to end-to-end optimization, which relies on solving a matrix-variable fixed-point equation to compute the gradient and an expensive projection step at every iteration. In this work, we extend the idea of greedy layer-wise training, an approach found to yield state-of-the-art performance in conventional deep learning, to a sequential greedy training algorithm for implicit deep models with a strictly upper block triangular structure. We show that such implicit models can be viewed as generalized Dense Convolutional Networks (DenseNets), and thus inherit the parameter efficiency of DenseNets. For models trained with the Euclidean loss, we develop an alternating minimization subroutine for our sequential optimization algorithm, which consists of alternating between efficiently-solvable least squares problems and single hidden-layer training problems. Furthermore, we theoretically prove that training a non-strictly upper triangular ReLU implicit model is equivalent to training a strictly upper block triangular one, allowing for the application of our algorithm to even more general models. Experiments on smooth and nonsmooth function interpolation, and on MNIST and Fashion-MNIST classification tasks, show that our algorithm consistently converges to models that outperform both end-to-end implicit learning and conventional feed-forward models.**

I. Introduction

Over the past decades, the rise in computing power and available data has propelled deep learning to the forefront of machine learning research [1]. Deep learning has also found many applications in control and decision problems, including estimation for power systems, control of autonomous vehicles, and reinforcement learning for systems with uncertainty [2]–[4]. Current deep learning models are built upon the notion of hierarchical architectures, where input information is processed recursively through several forward-propagating, differentiable parametric layers [1]. Canonical examples of this type are standard feed-forward neural networks (FFNs) and convolutional neural networks (CNNs) commonly used to perform image classification [5], [6].

Recent work has proposed a new perspective wherein deep learning models can be analyzed based on implicit prediction rules [7]. This framework, termed "implicit deep learning," is based on a model consisting of a prediction equation and a fixed-point equilibrium equation in a state vector $x \in \mathbb{R}^n$:

$$\begin{aligned} \hat{y}(u) &= \mathbf{C}x + \mathbf{D}u, && \text{(prediction equation)} \\ x &= \phi(\mathbf{A}x + \mathbf{B}u), && \text{(fixed-point equation)} \end{aligned}$$

where $\phi\colon \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear activation map, $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$, $\mathbf{C} \in \mathbb{R}^{q\times n}$, and $\mathbf{D} \in \mathbb{R}^{q\times p}$ are model parameters, and where $u \in \mathbb{R}^p$ and $\hat{y} \in \mathbb{R}^q$ are respectively the input and output. The fixed-point equation can be readily interpreted as a discrete-time linear time-invariant (LTI) system composed with the nonlinearity ϕ evaluated at the system's equilibrium. The prediction equation is equivalent to the standard output equation seen in LTI systems.

The significance of implicit deep learning lies in its representational power: [7] illustrates how it encapsulates most of the current neural network architectures as special cases, including feed-forward, convolutional, and residual networks. This framework can be regarded as a more general model that offers greater capacity to possibly model novel prediction rules for deep learning that may not necessarily be tied to any notion of "network" or "layers" [7].

Previously, the training of implicit models has relied on an end-to-end optimization approach via (stochastic) projected gradient methods, where the gradient is found by implicitly differentiating through the fixed-point equation [7]. This end-to-end approach for dense implicit models suffers from some notable drawbacks. First, computing the gradient at each training step is nontrivial, as it requires solving a separate fixed-point equation in a matrix variable. The gradient step must be followed by a computationally cumbersome projection of the matrix variable $\mathbf{A}$ to ensure well-posedness [7]. Furthermore, optimizing over all model parameters with a global objective obscures the interpretability of the model, as we cannot gain insight into the contribution of each subset of the parameters. Finally, it is known from conventional deep learning that end-to-end optimization landscapes are highly chaotic when training large models [8].

A. Contributions

In this paper, we focus on the efficient training of implicit models with a strictly upper block triangular structure. We first motivate imposing this structure by relating strictly upper block triangular implicit models to generalized Dense Convolutional Networks (DenseNets) and in turn highlight some of the strengths of implicit models with the aforementioned structure via the advantages of DenseNets. Subsequently, we propose a novel sequential, greedy, blockwise training algorithm for implicit deep models with the assumed structure. Unlike end-to-end training, the approach is interpretable and does not rely on projection steps or

The authors are with the University of California, Berkeley and VinUniversity. Emails: {tgautam23, bganderson, sojoudi}@berkeley.edu, and laurent.eg@vinuni.edu.vn.

This work was supported by grants from AFOSR, ONR, and NSF.

implicit differentiation. We posit that this decomposition into blockwise training alleviates the chaotic optimization landscapes that arise when performing end-to-end training on large implicit models. We experimentally show that our sequential approach outperforms end-to-end training, which aligns with prior findings in the conventional deep learning literature claiming that end-to-end optimization is susceptible to poor local solutions. Finally, we theoretically prove that, for ReLU implicit models, the training for more general, non-strictly upper triangular models (with self-loops between feature blocks) is equivalent to the special case of strictly upper block triangular training, a result novel to the literature.

B. Related Works

a) Implicit Deep Learning: Implicit deep learning is largely inspired by the concurrent works [7], [9]. In [7], implicit models are shown to generalize most of the popular deep learning architectures, sufficient conditions are proven for the uniqueness of their predictions, and methods to assess their robustness are proposed. The paper [9] shows that implicit models are equivalent to infinite-depth feed-forward networks, and that they provide memory-efficient state-of-the-art performance for modeling sequential data. In [10], an alternative instantiation of an implicit layer is introduced by means of an ordinary differential equation (ODE). In this framework, termed "neural ODE," the output of the implicit layer is the solution to the underyling ODE. This is shown to be an expressive model class but requires solving an ordinary differential equation at test-time. Recent works have extended various aspects of the implicit deep learning framework, e.g., to model graph-structured data [11], to multiscale modeling for image classification [12], and to estimate their Lipschitz constants for the purposes of robustness analysis [13]. While implicit models have been studied in a variety of contexts, their current training procedures rely upon end-to-end optimization approaches.

b) Conventional Deep Learning: Deep CNNs have recently garnered a great deal of acclaim due to their success in the ImageNet competition [5], [14]–[16]. The emergence of deep architectures has moved the spotlight on a new set of challenges, including the vanishing gradient problem and parameter redundancy. In particular, architectures such as Residual Networks (ResNets) [14] and Highway networks [15] tackle the vanishing gradient problem by directly passing signals from one layer to the next using identity connections. This has been shown to improve both information and gradient flow in said deep networks. Moreover, techniques such as Stochastic Depth used in ResNets, where layers are randomly dropped during training, have also demonstrated that deep architectures are often plagued by parameter redundancy when having many layers [17]. In [16], the authors introduce the Dense Convolutional Network (DenseNet), which extends the notion of skip connections used in ResNets and Highway networks such that all layers are connected directly with their subsequent layers. As a maximal amount of information is shared between layers, DenseNets are known to have improved gradient propagation and parameter efficiency [16]. While [7] shows how CNNs and ResNets can be viewed as special cases of implicit models, we formalize this for DenseNets. This enables us to speak of the benefits of implicit models in terms of the benefits of DenseNets.

c) Layer-Wise Optimization: The sequential, blockwise training approach proposed in this work can be seen as a generalization of layer-wise optimization for traditional neural networks. The paper [18] introduces greedy layer-wise training for deep neural networks, wherein the parameters are optimized per-layer in a sequential manner to pre-train an initialization of the network, which is then used in end-to-end optimization. This approach is motivated by the hypothesis that end-to-end gradient-based optimization is susceptible to becoming stuck in poor local solutions, whereas an individual layer's sub-optimization has a more benign landscape, which is justified by the authors experimentally. The work [19] extends the greedy layer-wise approach to convolutional neural networks, and finds that the learned model outperforms the state-of-the-art end-to-end training methods on the large-scale CIFAR-10 and ImageNet datasets.

C. Notations

Throughout this work, we define an implicit model using parameters $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$, $\mathbf{C} \in \mathbb{R}^{q\times n}$, and $\mathbf{D} \in \mathbb{R}^{q\times p}$ with nonlinear activation map $\phi\colon \mathbb{R}^n \to \mathbb{R}^n$. The element-wise absolute value of $\mathbf{A}$ is denoted by $|\mathbf{A}|$. The Perron-Frobenius eigenvalue of $|\mathbf{A}|$, i.e., the real eigenvalue larger than the modulus of all other eigenvalues, is denoted by $\lambda_{\text{PF}}(|\mathbf{A}|)$. The input matrix with m input vectors $u \in \mathbb{R}^p$ is represented by $\mathbf{U} \in \mathbb{R}^{p\times m}$, and similarly the target matrix with m output vectors $y \in \mathbb{R}^q$ is represented by $\mathbf{Y} \in \mathbb{R}^{q\times m}$. The notation $\lfloor \alpha \rfloor$ represents the floor operator on scalar α. For a vector $x \in \mathbb{R}^n$ and natural number p, $\|x\|_p$ denotes the ℓ_p-norm of x, whereas for a matrix $\mathbf{M} \in \mathbb{R}^{m\times n}$, $\|\mathbf{M}\|_F$ denotes the Frobenius norm and $\|\mathbf{M}\|_p$ represents the ℓ_p-induced operator norm; $\|\mathbf{M}\|_p = \sup_{x\in\mathbb{R}^n\setminus\{0\}} \frac{\|\mathbf{M}x\|_p}{\|x\|_p}$. The rectified linear unit is $\mathrm{ReLU}(\cdot) = \max\{0, \cdot\}$, where, for the matrix $\mathbf{M}$, the maximum is taken element-wise. If $\mathbf{M} \in \mathbb{R}^{n\times n}$ is square, then we write $\mathrm{diag}(\mathbf{M})$ to mean the n-vector $(M_{11}, M_{22}, \ldots, M_{nn})$. Finally, $\mathcal{U}(a, b)$ denotes the uniform probability distribution with support $[a, b]$.

II. Background

A. Well-posedness of Implicit Models

The state x, characterized by the fixed-point equation, can be thought to represent the latent features extracted from the input [7]. In general, however, the fixed-point equation may not necessarily be well-posed in x—the activation map ϕ and matrix $\mathbf{A}$ must adhere to certain conditions to ensure the existence of a unique solution x to the fixed-point equation. Unique solutions to the fixed-point equation are desirable as this transfers to unique input-to-prediction mappings of the implicit model. The recent work [7] has introduced rigorous and numerically tractable conditions under which the fixed-point equation has a unique solution, which we recall in this section. In subsequent sections, we will show how such

conditions can be incorporated into the training problem as constraints to guarantee the well-posedness of the learned model. We begin with a few definitions.

Definition 1. A matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is said to be *well-posed with respect to* ϕ if, for all $b \in \mathbb{R}^n$, the equation

$$x = \phi(\mathbf{A}x + b) \tag{1}$$

has a unique solution $x \in \mathbb{R}^n$.

We write $\mathbf{A} \in \mathrm{WP}(\phi)$ to mean that $\mathbf{A}$ is well-posed with respect to ϕ.

Definition 2. A map $\phi\colon \mathbb{R}^n \to \mathbb{R}^n$ is called *component-wise non-expansive (CONE)* if $|\phi_i(x) - \phi_i(y)| \le |x_i - y_i|$ for all $x, y \in \mathbb{R}^n$ and all $i \in \{1, 2, \dots, n\}$.

Note that ReLU, leaky ReLU, sigmoid, and $\tanh$ activation maps are all CONE maps. CONE maps allow for a simple sufficient condition to ensure that the fixed-point equation is well-posed:

Theorem 1 (Well-Posedness of CONE Maps [7])**.** *Assume that ϕ is a CONE map. If $\lambda_{\mathrm{PF}}(|\mathbf{A}|) < 1$, then $\mathbf{A} \in \mathrm{WP}(\phi)$ and the fixed-point iteration $x(0) = 0,\ x(t+1) = \phi(\mathbf{A}x(t) + b),\ t \in \{0, 1, 2, \dots\}$, converges to the unique solution of* (1)*.*

We remark that the set $\mathcal{A}_{\mathrm{PF}} \coloneqq \{\mathbf{A} \in \mathbb{R}^{n\times n} : \lambda_{\mathrm{PF}}(|\mathbf{A}|) < 1\}$ is not convex, but the inequality $\lambda_{\mathrm{PF}}(|\mathbf{A}|) < \|\mathbf{A}\|_\infty$ implies that $\mathcal{A}_\infty \coloneqq \{\mathbf{A} \in \mathbb{R}^{n\times n} : \|\mathbf{A}\|_\infty < 1\}$ is a convex subset of $\mathcal{A}_{\mathrm{PF}}$, which is useful for efficiently treating the constraint $\mathbf{A} \in \mathrm{WP}(\phi)$ in the training problem. This work focuses on models where ϕ satisfies the CONE property.

B. Training Problem

We now turn our attention to formalizing the training problem for the implicit model. Suppose that we are given an input matrix $\mathbf{U} = \begin{bmatrix} u_1 & \cdots & u_m \end{bmatrix} \in \mathbb{R}^{p\times m}$ and a target matrix $\mathbf{Y} = \begin{bmatrix} y_1 & \cdots & y_m \end{bmatrix} \in \mathbb{R}^{q\times m}$. The prediction and fixed-point equation of the implicit model can then be written in matrix form as

$$\hat{\mathbf{Y}}(\mathbf{U}) = \mathbf{C}\mathbf{X} + \mathbf{D}\mathbf{U}, \tag{2}$$

$$\mathbf{X} = \phi(\mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{U}), \tag{3}$$

where $\hat{\mathbf{Y}}(\mathbf{U}) = \begin{bmatrix} \hat{y}(u_1) & \cdots & \hat{y}(u_m) \end{bmatrix} \in \mathbb{R}^{q\times m}$. The training problem is then formulated as

$$\begin{aligned} \min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{X}} \quad & \mathcal{L}(\mathbf{Y}, \mathbf{C}\mathbf{X} + \mathbf{D}\mathbf{U}) + \mathcal{P}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}) \\ \text{s.t.} \quad & \mathbf{X} = \phi(\mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{U}),\ \mathbf{A} \in \mathrm{WP}(\phi), \end{aligned} \tag{4}$$

where the loss function $\mathcal{L}$ is convex in its second argument, and $\mathcal{P}$ is an optional convex penalty function. As shown in Theorem 1, when ϕ is a CONE map, it suffices to replace the constraint $\mathbf{A} \in \mathrm{WP}(\phi)$ by the nonconvex constraint $\lambda_{\mathrm{PF}}(|\mathbf{A}|) < 1$. For efficient treatment, the nonconvex constraint can be relaxed to the convex constraint $\|\mathbf{A}\|_\infty < 1$.

III. Strictly Block Triangular Implicit Models

We begin by partitioning an implicit model of order n into $L > 1$ uniform parts as $n = n_1 + \cdots + n_L$, where $n_i = \lfloor \frac{n}{L} \rfloor$ for $i \in \{1, 2, \dots, L-1\}$ and $n_L = n - \sum_{i=1}^{L-1} n_i$. We may write the model matrices in terms of blocks associated with each part of the partition:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{L,L} & \mathbf{A}_{L,L-1} & \cdots & \mathbf{A}_{L,1} \\ \mathbf{A}_{L-1,L} & \mathbf{A}_{L-1,L-1} & \cdots & \mathbf{A}_{L-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{1,L} & \mathbf{A}_{1,L-1} & \cdots & \mathbf{A}_{1,1} \end{bmatrix},\ \mathbf{B} = \begin{bmatrix} \mathbf{B}_L \\ \mathbf{B}_{L-1} \\ \vdots \\ \mathbf{B}_1 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_L & \mathbf{C}_{L-1} & \cdots & \mathbf{C}_1 \end{bmatrix},\ \mathbf{X} = \begin{bmatrix} \mathbf{X}_L \\ \mathbf{X}_{L-1} \\ \vdots \\ \mathbf{X}_1 \end{bmatrix},$$

where $\mathbf{A}_{i,j} \in \mathbb{R}^{n_i\times n_j}$, $\mathbf{B}_i \in \mathbb{R}^{n_i\times p}$, $\mathbf{C}_i \in \mathbb{R}^{q\times n_i}$, and $\mathbf{X}_i \in \mathbb{R}^{n_i\times m}$ for all $i, j \in \{1, 2, \dots, L\}$. Next, we impose the following assumptions on our model.

Assumption 1. The activation map ϕ is a CONE map.

As mentioned earlier, most common activation maps used in deep learning, e.g., ReLU, are CONE maps, and therefore Assumption 1 is not restrictive. Moreover, this assumption allows the use of convex constraints to enforce the desirable well-posedness guarantee upon the model.

Assumption 2. The prediction equation has no feed-through from the input; $\mathbf{D} = 0$.

Since feed-forward neural networks with non-polynomial activation maps, e.g., ReLU maps, are universal function approximators [20], and implicit models recover feed-forward neural networks with $\mathbf{D} = 0$ and appropriate choices of $\mathbf{A}, \mathbf{B}, \mathbf{C}$, it holds that the implicit model remains a universal function approximator under Assumption 2.

Assumption 3. The matrix $\mathbf{A}$ is strictly upper block triangular, i.e., $\mathbf{A}_{i,j} = 0$ for all pairs (i, j) with $i \le j$.

Assumption 3 is not restrictive in practice, since an implicit model with such a matrix $\mathbf{A}$ is able to represent standard network architectures used today, e.g., feed-forward, convolutional, and residual networks [7]. In fact, feed-forward networks correspond to setting $\mathbf{A}_{i,j} = 0$ for all pairs (i, j) such that $j \neq i+1$, and therefore it is easily shown that our strictly upper block triangular model matrix allows for precisely $(L-2)n_1n_L + \frac{1}{2}(L-2)(L-3)n_1^2$ more nonzero parameters than standard feed-forward models of the same model order n. Furthermore, Assumption 3 is justified from a computational standpoint, since the condition $\mathbf{A} \in \mathrm{WP}(\phi)$ is trivially satisfied as ϕ is a CONE map, allowing us to remove this constraint from the training problem.

A. Relation to DenseNets

While [7] has already established how implicit models can encapsulate a significant subset of conventional architectures, we now highlight the relationship between implicit models with strictly upper block triangular structure and DenseNets as conceived in [16].

Figure 1 illustrates a computational graph detailing how input information is propagated through an implicit model under Assumptions 1–3. Note that this computational graph is characterized by a forward-propagating structure with an exhaustive set of $L(L+1)/2$ generalized, weighted skip connections linking each pair of feature blocks including the input. Differences to DenseNets, as proposed in [16], include that implicit models with said structure allow for arbitrary linear transformations (not only convolutions) between feature blocks as well as weighted skip connections throughout the model, including from the input to every feature block. This shows that strictly upper block triangular implicit models are a generalization of DenseNets.

As a result of this characterization, implicit models with an upper block triangular structure directly inherit many of the benefits observed in DenseNets. Due to the exhaustive set of connections between each feature block, the considered implicit models have maximal information flow. As a consequence, such models are also bound to have a greater parameter efficiency. Finally, since we connect each feature block directly to the output, the training procedure for such models will benefit from improved gradient propagation.

IV. Sequential Blockwise Training

Our approach is a greedy, sequential blockwise training method for strictly upper block triangular implicit models. The training problem under Assumptions 1-3 reduces to

$$\begin{aligned} \min_{\substack{\{\mathbf{A}_{i,j}\}_{1\le j<i\le L},\\ \{\mathbf{B}_i,\mathbf{C}_i,\mathbf{X}_i\}_{i=1}^{L}}} \quad & \mathcal{L}(\mathbf{Y}, \textstyle\sum_{j=1}^{L} \mathbf{C}_j\mathbf{X}_j) \\ \text{s.t.} \quad & \mathbf{X}_1 = \phi(\mathbf{B}_1\mathbf{U}), \\ & \mathbf{X}_i = \phi(\textstyle\sum_{j=1}^{i-1} \mathbf{A}_{i,j}\mathbf{X}_j + \mathbf{B}_i\mathbf{U}), \\ & i \in \{2,\dots,L\}. \end{aligned} \tag{5}$$

With the partition of n into L blocks, we may iteratively increment the model order of the implicit model by n_i and train each added block individually, while holding previously optimized fixed-point parameters constant and re-optimizing the auxiliary parameters. We formalize this approach in Algorithm 1. Due to the strictly upper block triangular structure assumed upon $\mathbf{A}$, the first subproblem (6) only optimizes over subblocks of the $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{X}$ matrices. The i^{th} subproblem optimizes over $\{\mathbf{A}_{i,j}\}_{j=1}^{i-1}$, $\mathbf{B}_i$, $\mathbf{X}_i$ and auxiliary blocks $\{\mathbf{C}_j\}_{j=1}^{i}$. Here feature blocks $\{\mathbf{X}_j\}_{j=1}^{i-1}$ are fixed and obtained as optimizers of the previous subproblems. The sequence of optimizations in Algorithm 1 occurs only once, so that the obtained subblocks can be concatenated to form the model matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. Figure 2 depicts a visual representation of the optimization sequence.

Remark 1. Our proposed approach can be applied to both regression and classification tasks. For regression, the convex loss can be chosen to be the mean-squared error (MSE) loss

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{2}\|\mathbf{Y} - \hat{\mathbf{Y}}\|_F^2, \tag{8}$$

while multi-class classification can be accommodated using a combination of the softmax and negative cross-entropy loss:

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = \log(\mathbf{1}^\top \times \exp(\hat{\mathbf{Y}}))\mathbf{1} - \operatorname{Tr}(\mathbf{Y}^\top\hat{\mathbf{Y}}). \tag{9}$$

Algorithm 1 Sequential Greedy Training for Implicit Models

Input: Input $\mathbf{U} \in \mathbb{R}^{p\times m}$, target $\mathbf{Y} \in \mathbb{R}^{q\times m}$.
Parameters: Model order $n > 1$, partition size $L > 1$.
Design choices: Loss $\mathcal{L}$, activation map ϕ.
Return: $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$, $\mathbf{C} \in \mathbb{R}^{q\times n}$.

begin training
1. Partition $n = n_1 + \cdots + n_L$ with $n_i = \lfloor \frac{n}{L} \rfloor$ for $i \in \{1, 2, \dots, L-1\}$ and $n_L = n - \sum_{i=1}^{L-1} n_i$.
2. Solve optimization

$$\min_{\mathbf{B}_1,\mathbf{C}_1,\mathbf{X}_1} \mathcal{L}(\mathbf{Y}, \mathbf{C}_1\mathbf{X}_1) \text{ s.t. } \mathbf{X}_1 = \phi(\mathbf{B}_1\mathbf{U}). \tag{6}$$

for $i \in \{2, 3, \dots, L\}$ **do**
3. Solve optimization

$$\begin{aligned} \min_{\substack{\{\mathbf{A}_{i,j}\}_{j=1}^{i-1},\mathbf{B}_i,\\ \{\mathbf{C}_j\}_{j=1}^{i},\mathbf{X}_i}} \quad & \mathcal{L}(\mathbf{Y}, \textstyle\sum_{j=1}^{i} \mathbf{C}_j\mathbf{X}_j) \\ \text{s.t.} \quad & \mathbf{X}_i = \phi(\textstyle\sum_{j=1}^{i-1} \mathbf{A}_{i,j}\mathbf{X}_j + \mathbf{B}_i\mathbf{U}). \end{aligned} \tag{7}$$

end for
end training

Remark 2. Each subproblem can be solved using gradient-based local search heuristics implemented with automatic differentiation software. Note that the i^{th} subproblem can be decomposed into a regression upon the target using both a feedforward component and direct linear (skip) contributions from the previous feature blocks. As such, this sequential approach avoids the need for implicit differentiation.

Remark 3. Even though our approach sets up a uniform partition of the model order n, this partition can be chosen arbitrarily. The reason the algorithm is initialized with a uniform partition is that the training complexity is equally distributed amongst all subproblems.

Remark 4. Inherent to our approach is the restriction of $\mathbf{A}$ as a strictly upper block triangular matrix. This assumption alleviates the requirement to project $\mathbf{A}$ onto the well-posedness ball at each iteration of the gradient-based optimization method. In particular, when considering the convex constraint $\|\mathbf{A}\|_\infty < 1$ sufficient for well-posedness, our method saves running a bisection method with complexity $O(n^3)$ at each projected gradient update.

Remark 5. Our approach puts less strain on the memory requirements during the training procedure as compared to the end-to-end optimization. For the former, the i^{th} subproblem only requires optimizing a subset of $n_i(\sum_{j=1}^{i-1} n_j + m + p) + q\sum_{j=1}^{i} n_j$ parameters, whereas the latter optimizes over $n(n+m+p+q)$ parameters simultaneously. In the case where we have a large order model and $n \gg n_i$, each optimization of the blockwise method needs to update significantly fewer parameters at a time than in the end-to-end method.

Remark 6. The greedy blockwise approach also lends itself to an increased interpretability of the implicit model param-

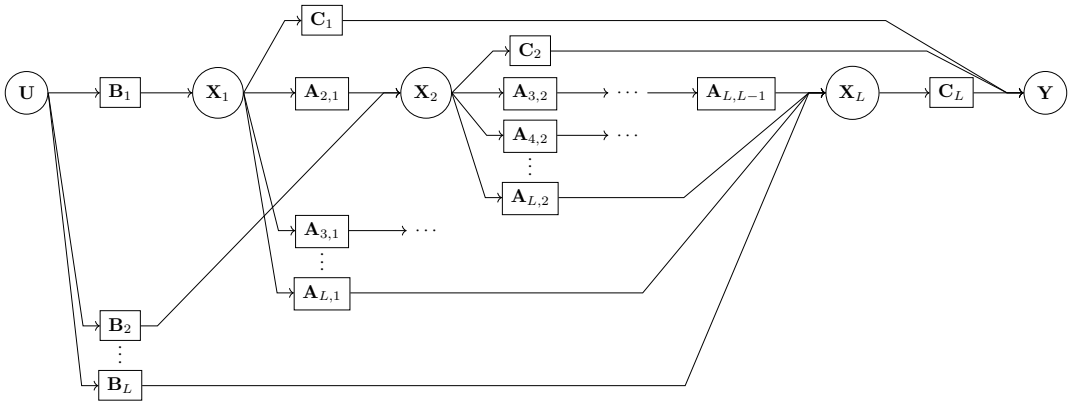

Fig. 1. Computational graph for an implicit model with a strictly upper block triangular $\mathbf{A}$ matrix.

$$\begin{bmatrix} \mathbf{X}_L \\ \mathbf{X}_{L-1} \\ \vdots \\ \mathbf{X}_2 \\ \mathbf{X}_1 \end{bmatrix} = \phi\left(\begin{bmatrix} 0 & \mathbf{A}_{L,L-1} & \cdots & \mathbf{A}_{L,2} & \mathbf{A}_{L,1} \\ 0 & 0 & \cdots & \mathbf{A}_{L-1,2} & \mathbf{A}_{L-1,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \mathbf{A}_{2,1} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{X}_L \\ \mathbf{X}_{L-1} \\ \vdots \\ \mathbf{X}_2 \\ \mathbf{X}_1 \end{bmatrix} + \begin{bmatrix} \mathbf{B}_L \\ \mathbf{B}_{L-1} \\ \vdots \\ \mathbf{B}_2 \\ \mathbf{B}_1 \end{bmatrix} \mathbf{U} \right)$$

$$\mathbf{Y} = \begin{bmatrix} \mathbf{C}_L & \mathbf{C}_{L-1} & \cdots & \mathbf{C}_2 & \mathbf{C}_1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_L \\ \mathbf{X}_{L-1} \\ \vdots \\ \mathbf{X}_2 \\ \mathbf{X}_1 \end{bmatrix}$$

Fig. 2. Illustration of the proposed sequential blockwise training scheme. Blocks in (light) green are optimized initially, whereas the blocks in (dark) blue are optimized towards the end of the algorithm. The auxiliary $\mathbf{C}$ blocks are re-optimized at every iteration.

eters. Within an end-to-end training approach of a dense implicit model, we lack insight into how the parameters are working together to minimize the loss. Our approach directly supervises the training for each subblock of parameters. In the first optimization, blocks $\mathbf{B}_1$, $\mathbf{C}_1$, and $\mathbf{X}_1$ attempt to directly regress upon the target. For the i^{th} optimization, the additional fixed-point parameters $\{\mathbf{A}_{i,j}\}_{j=1}^{i-1}$, $\mathbf{B}_i$, and $\mathbf{X}_i$ attempt to improve the regression by offering greater modeling capacity while auxiliary parameters $\{\mathbf{C}_j\}_{j=1}^{i}$ re-weight the contribution of each feature block $\{\mathbf{X}_j\}_{j=1}^{i}$.

A. Alternating Minimization for Implicit Models

While local search presents a valid means to solve the subproblems of Algorithm 1, we also propose an efficient alternating minimization heuristic for models trained under the squared Euclidean loss.

Assumption 4. The training problem uses the squared Euclidean loss (8).

When considering the training problem of implicit models with Euclidean loss, we can formulate the i^{th} subproblem in Algorithm 1 as follows:

$$\begin{aligned} \min_{\substack{\{\mathbf{A}_{i,j}\}_{j=1}^{i-1},\mathbf{B}_i,\\ \{\mathbf{C}_j\}_{j=1}^{i},\mathbf{X}_i}} \quad & \|\mathbf{Y} - \textstyle\sum_{j=1}^{i} \mathbf{C}_j\mathbf{X}_j\|_F^2 \\ \text{s. t.} \quad & \mathbf{X}_i = \phi(\textstyle\sum_{j=1}^{i-1} \mathbf{A}_{i,j}\mathbf{X}_j + \mathbf{B}_i\mathbf{U}). \end{aligned} \tag{10}$$

Rather than using a local search method to solve (10), we can leverage an alternating minimization heuristic. This subroutine is formalized in Algorithm 2. It decomposes each subproblem into alternating between solving a least squares problem and a single hidden-layer neural network training problem, each with global optimality guarantees. Each least squares problem (11) can be solved to global optimality with a computational complexity of $O\left((\sum_{j=1}^{i} n_j)^3\right)$, while there exist guarantees for solving the shallow training problem to (near) global optimality via local search heuristics [21], [22].

B. Extension to Non-Strict Upper Triangular ReLU Models

In this section, we relax Assumption 3 to allow for $\mathbf{A}$ to be upper triangular, i.e., we now assume that $\mathbf{A}_{i,j} = 0$ for all pairs (i, j) with $i < j$ and $\mathbf{A}_{i,i}$ is upper triangular for all i. This new model structure allows for self-loops at all of the feature blocks $\mathbf{X}_i$ (i.e., self-loops at the circular nodes in Figure 1). In this case, the i^{th} subproblem (7) in the blockwise sequential approach becomes

$$\begin{aligned} \min_{\substack{\{\mathbf{A}_{i,j}\}_{j=1}^{i},\mathbf{B}_i,\\ \{\mathbf{C}_j\}_{j=1}^{i},\mathbf{X}_i}} \quad & \mathcal{L}(\mathbf{Y}, \textstyle\sum_{j=1}^{i} \mathbf{C}_j\mathbf{X}_j) \\ \text{s. t.} \quad & \mathbf{X}_i = \phi(\textstyle\sum_{j=1}^{i} \mathbf{A}_{i,j}\mathbf{X}_j + \mathbf{B}_i\mathbf{U}), \\ & -1 < \operatorname{diag}(\mathbf{A}_{i,i}) < 1, \\ & (\mathbf{A}_{i,i})_{kl} = 0 \text{ for all } k > l, \end{aligned} \tag{13}$$

for $i \in \{1, 2, \ldots, L\}$, where the constraints $-1 < \operatorname{diag}(\mathbf{A}_{i,i}) < 1$ are linear constraints ensuring that the

Algorithm 2 Alternating Minimization Subroutine

Input: Features $\{\mathbf{X}_j\}_{j=1}^{i-1}$ from first $i-1$ subproblems.
Parameters: Number of alternating iterations T.
Return: $\{\mathbf{A}_{i,j}\}_{j=1}^{i-1}$, $\mathbf{B}_i$, $\mathbf{X}_i$, and $\{\mathbf{C}_j\}_{j=1}^{i}$.

begin subroutine
1. Initialize $\mathbf{X}_i$.
for $t \in \{1, 2, \ldots, T\}$ **do**
2. Solve least squares optimization

$$\min_{\{\mathbf{C}_j\}_{j=1}^{i}} \|\mathbf{Y} - \sum_{j=1}^{i} \mathbf{C}_j \mathbf{X}_j\|_F^2. \tag{11}$$

3. Solve single hidden-layer ReLU training problem

$$\begin{aligned} \min_{\tilde{\mathbf{A}}_i, \mathbf{X}_i} \ & \|\tilde{\mathbf{Y}} - \mathbf{C}_i \mathbf{X}_i\|_F^2 \\ \text{s.t. } & \mathbf{X}_i = \phi(\tilde{\mathbf{A}}_i \tilde{\mathbf{U}}_i), \end{aligned} \tag{12}$$

with $\tilde{\mathbf{Y}} = \mathbf{Y} - \sum_{j=1}^{i-1} \mathbf{C}_j \mathbf{X}_j$,

$$\tilde{\mathbf{A}}_i = \begin{bmatrix} \mathbf{A}_{i,i-1} & \cdots & \mathbf{A}_{i,1} & \mathbf{B}_i \end{bmatrix},$$

and

$$\tilde{\mathbf{U}}_i = \begin{bmatrix} \mathbf{X}_{i-1}^\top & \cdots & \mathbf{X}_1^\top & \mathbf{U}^\top \end{bmatrix}^\top.$$

end for
end subroutine

triangular matrix $\mathbf{A}$ is well-posed with respect to ϕ, as guaranteed by Theorem 1. The difficulty in the new problem (13) is the fact that the constraint on the variable $\mathbf{X}_i$ is now an implicit equality constraint. In general, this requires the use of implicit differentiation at each step of a gradient-based optimization algorithm, making each of the subproblems nontrivial to solve, and eliminating the immediate applicability of the alternating subroutine of Algorithm 2. Despite this challenge, we prove in Theorem 2 that for ReLU models, the feasible set of (13) is equivalently expressed in terms of an explicit equality constraint, making the i^{th} subproblem equivalent to a re-weighted subproblem under our original strictly upper block triangular assumption. For the sake of exposition, we assume that $n_j = 1$. We generalize the result to arbitrary n_j in Appendix I.

Theorem 2. *Assume that $\phi = \mathrm{ReLU}$ and $n_j = 1$ for all $j \in \{1, 2, \ldots, L\}$, and let $\mathbf{X}_i \in \mathbb{R}^{1 \times m}$. Then, there exist $\{\mathbf{A}_{i,j}\}_{j=1}^{i}, \mathbf{B}_i, \{\mathbf{C}_j\}_{j=1}^{i}$ that together with $\mathbf{X}_i$ are feasible for* (13) *if and only if there exist $\gamma_1, \ldots, \gamma_{i-1}, \lambda_1, \ldots, \lambda_p \in \mathbb{R}$ such that $\mathbf{X}_i = \mathrm{ReLU}\left(\sum_{j=1}^{i-1} \gamma_j \mathbf{X}_j + \sum_{k=1}^{p} \lambda_k \mathbf{U}_k\right)$, where $\mathbf{U}_k$ is the k^{th} row of $\mathbf{U}$.*

Proof. Notice that, since $n_j = 1$ for all j, every block of $\mathbf{A}$ is a scalar, so we write $\mathbf{A}_{i,j} = a_{ij}$. Similarly, denote the kth element of the row vector $\mathbf{B}_i$ by b_{ik}.

Suppose that $\{a_{ij}\}_{j=1}^{i}$, $\mathbf{B}_i$, and $\{\mathbf{C}_j\}_{j=1}^{i}$, together with $\mathbf{X}_i$, are feasible for (13). Then $-1 < a_{ii} < 1$ and

$$\mathbf{X}_i = \mathrm{ReLU}\left(\sum_{j=1}^{i} a_{ij} \mathbf{X}_j + \sum_{k=1}^{p} b_{ik} \mathbf{U}_k\right).$$

Let $l \in \{1, 2, \ldots, m\}$. If $X_{il} \neq 0$, then we have that

$$\begin{aligned} 0 < X_{il} &= \mathrm{ReLU}\left(\sum_{j=1}^{i} a_{ij} X_{jl} + \sum_{k=1}^{p} b_{ik} U_{kl}\right) \\ &= \sum_{j=1}^{i} a_{ij} X_{jl} + \sum_{k=1}^{p} b_{ik} U_{kl}. \end{aligned}$$

Hence, $X_{il} = \sum_{j=1}^{i-1} \frac{a_{ij}}{1-a_{ii}} X_{jl} + \sum_{k=1}^{p} \frac{1}{1-a_{ii}} b_{ik} U_{kl} = \mathrm{ReLU}\left(\sum_{j=1}^{i-1} \frac{a_{ij}}{1-a_{ii}} X_{jl} + \sum_{k=1}^{p} \frac{1}{1-a_{ii}} b_{ik} U_{kl}\right)$. On the other hand, if $X_{il} = 0$, then $\sum_{j=1}^{i} a_{ij} X_{jl} + \sum_{k=1}^{p} b_{ik} U_{kl} = \sum_{j=1}^{i-1} a_{ij} X_{jl} + \sum_{k=1}^{p} b_{ik} U_{kl} \leq 0$, so $\sum_{j=1}^{i-1} \frac{a_{ij}}{1-a_{ii}} X_{jl} + \sum_{k=1}^{p} \frac{b_{ik}}{1-a_{ii}} U_{kl} \leq 0$, which implies that again $X_{il} = \mathrm{ReLU}\left(\sum_{j=1}^{i-1} \frac{a_{ij}}{1-a_{ii}} X_{jl} + \sum_{k=1}^{p} \frac{b_{ik}}{1-a_{ii}} U_{kl}\right)$. Thus, it holds that

$$\mathbf{X}_i = \mathrm{ReLU}\left(\sum_{j=1}^{i-1} \frac{a_{ij}}{1-a_{ii}} \mathbf{X}_j + \sum_{k=1}^{p} \frac{b_{ik}}{1-a_{ii}} \mathbf{U}_k\right),$$

which proves the forward direction with $\gamma_j = \frac{a_{ij}}{1-a_{ii}}$ and $\lambda_k = \frac{b_{ik}}{1-a_{ii}}$.

Now, suppose there exist $\gamma_1, \ldots, \gamma_{i-1}, \lambda_1, \ldots, \lambda_p \in \mathbb{R}$ such that $\mathbf{X}_i = \mathrm{ReLU}\left(\sum_{j=1}^{i-1} \gamma_j \mathbf{X}_j + \sum_{k=1}^{p} \lambda_k \mathbf{U}_k\right)$. Then, let $\epsilon > 0$ and define $\{\mathbf{A}_{i,j}\}_{j=1}^{i} \subseteq \mathbb{R}, \mathbf{B}_i \in \mathbb{R}^{1 \times p}$ by $\mathbf{A}_{i,i} = a_{ii} := 1 - \epsilon$, $\mathbf{A}_{i,j} = (1 - a_{ii})\gamma_j$ for all $j \in \{1, 2, \ldots, i-1\}$, and $(\mathbf{B}_i)_k = (1 - a_{ii})\lambda_k$ for all $k \in \{1, 2, \ldots, p\}$. Let $\{\mathbf{C}_j\}_{j=1}^{i} \subseteq \mathbb{R}^{q \times 1}$ be arbitrary. Then, by construction $-1 < \mathrm{diag}(\mathbf{A}_{i,i}) = a_{ii} = 1 - \epsilon < 1$ and $(\mathbf{A}_{i,i})_{kl} = 0$ for all $k > l$. Reversing the steps of the forward direction proof also shows that our parameter choice yields $\mathbf{X}_i = \mathrm{ReLU}\left(\sum_{j=1}^{i} \mathbf{A}_{i,j} \mathbf{X}_j + \mathbf{B}_i \mathbf{U}\right)$. Thus, $\{\mathbf{A}_{i,j}\}_{j=1}^{i}, \mathbf{B}_i, \{\mathbf{C}_j\}_{j=1}^{i}$, together with $\mathbf{X}_i$, are feasible for (13). □

Theorem 2 shows that the implicit constraint in the non-strict training problem (13) may be equivalently replaced by an explicit constraint, avoiding the need to use implicit differentiation in solving the optimization for non-strict models.

V. Experiments

A. Nonlinear Function Interpolation via Regression

We consider the tasks of interpolating real, univariate, nonlinear, smooth and nonsmooth functions via regression. As illustrative examples, we consider the smooth function

$$f_1(u) = 5\cos(\pi u)\exp(-\tfrac{1}{2}|u|) \tag{14}$$

introduced in [7], and the nonsmooth function $f_2(u)$ formed by the product of a sawtooth waveform and a decaying exponential. The true functions $f_1(u)$ and $f_2(u)$ are presented in Figures 3 and 4, respectively. Training datasets were constructed by drawing 500 samples $\{(u_i, y_i)\}_{i=1}^{500}$ in

an i.i.d. manner where $u_i \sim \mathcal{U}(-5,5)$ and $y_i \sim \mathcal{U}(f_j(u_i)-1, f_j(u_i)+1)$ for $j \in \{1,2\}$. To mitigate stochasticity within the interpolation performance of the considered models, the training datasets were sampled 10 times and the average model performance is considered for comparison. The test datasets were formed by evaluating the true functions at 1000 uniformly spaced inputs $u \in [-5,5]$. As a comparison benchmark, a feed-forward network (FFN) with architecture (1-25-25-25-25-1) consisting of 4 hidden layers with 25 hidden units each was trained using the ADAM optimizer with step size 0.01 until convergence [23]. We measure the capacity of a model by summing up all of its parameters. For the considered FF architecture (with univariate inputs), the capacity is $25+3\times 25^2 = 1900$ parameters. We compare the average performance of this FFN with an implicit model of order $n = 30$. For end-to-end training, such a model has a capacity of $n(n+p+q) = 960$ parameters. The blockwise approach has approximately half the capacity due to the strictly block upper triangular assumption made on $\mathbf{A}$. In this experiment, the blockwise training approach used a partition of $L = 10$. Local search heuristics for solving the blockwise subproblems used the same optimizer configurations as was used for the FFN training, i.e., ADAM optimizer with a step size of 0.01. Visual illustrations of the average interpolation performance of the blockwise trained implicit models on the smooth and nonsmooth functions are shown in Figures 3 and 4. Tables I and II show the average test root-mean-square error (RMSE) as well as average training times (as run on a 2.6 GHz 6-Core CPU) of the different models on the two considered tasks. The blockwise trained implicit model significantly outperforms both the FFN and the end-to-end trained implicit model. The improved performance over the FFN comes even though the implicit model has a significantly lower capacity than the FFN. This empirically corroborates the parameter efficiency advantage inherent to implicit models. The fact that the blockwise approach also notably outperforms the end-to-end optimization suggests that our proposed training method is better able to exploit the parameter efficiency property. We also note a marked improvement in training time for the blockwise approach over the end-to-end optimization. This is due to the elimination of the well-posedness projections and fixed-point iterations.

TABLE I

AVERAGE TEST RMSE AND TRAINING TIMES FOR THE CONSIDERED IMPLICIT AND FF MODELS ON THE SMOOTH INTERPOLATION TASK.

Model	Training Time (s)	Test RMSE
Implicit (Blockwise, Local Search)	**4**	**0.29**
Implicit (End-to-End)	51	1.72
FFN	14	0.42

B. Classification Tasks

We next evaluate our algorithm on classification tasks by considering the MNIST and the Fashion-MNIST datasets [24]. The former considers handwritten digit classification, whereas the latter focuses on fashion product classification.

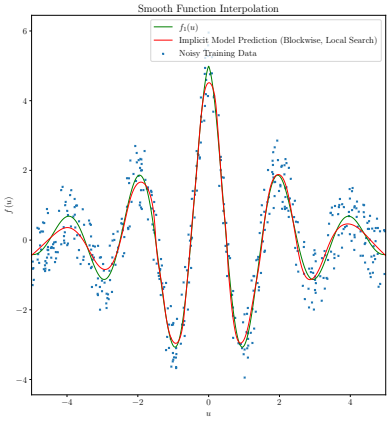

Fig. 3. Performance of implicit model trained via the blockwise algorithm on the smooth function interpolation task.

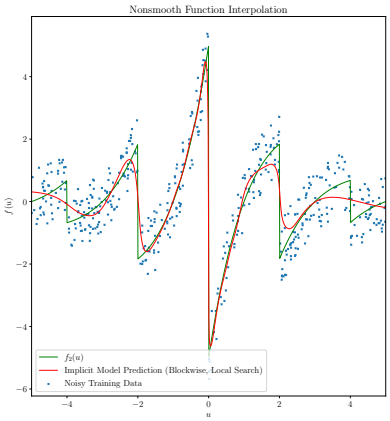

Fig. 4. Performance of implicit model trained via the blockwise algorithm on the nonsmooth function interpolation task.

For both datasets, the 28×28 pixel input images were reshaped into 784-dimensional input vectors. Moreover, we chose $m = 6 \times 10^4$ training samples and 10^4 test samples. As a comparison benchmark, a FFN with architecture (784-64-32-16-10) was trained using the ADAM optimizer with step size 0.001 for 200 epochs. The capacity of the FF architecture is $784\times 64+64\times 32+32\times 16+16\times 10 = 52896$ parameters. Our implicit model was initialized with an order $n = 32$ and for the case of our blockwise greedy algorithm we used $L = 16$ partitions (trained using local search). Each optimization for the implicit model was carried out for 200 epochs to keep consistency with the FFN baseline. For the end-to-end optimization the capacity of the model is $32\times 32+32\times 784+32\times 10 = 26432$. Similarly, with a strictly upper triangular structure, the capacity of the model used with blockwise training is approximately 26000 parameters. Table III shows the test accuracy (%) and training times for the considered models on both datasets.

The sequential blockwise trained implicit models outperform the models trained with end-to-end optimization as well as the FFN on both datasets. Across both datasets, the

TABLE II

AVERAGE TEST RMSE AND TRAINING TIMES FOR THE CONSIDERED IMPLICIT AND FF MODELS ON THE NONSMOOTH INTERPOLATION TASK.

Model	Training Time (s)	Test RMSE
Implicit (Blockwise, Local Search)	**5**	**0.52**
Implicit (End-to-End)	44	1.78
FFN	15	0.59

TABLE III

TEST ACCURACY AND TRAINING TIMES FOR THE CONSIDERED IMPLICIT AND FF MODELS ON THE CLASSIFICATION TASKS.

(a) MNIST Classification.

Model	Training Time (s)	Accuracy (%)
Implicit (Blockwise, Local Search)	**819**	**95.58**
Implicit (End-to-End)	1198	65.65
FFN	380	95.27

(b) Fashion-MNIST Classification.

Model	Training Time (s)	Accuracy (%)
Implicit (Blockwise, Local Search)	**791**	**86.9**
Implicit (End-to-End)	1368	58.06
FFN	379	86

sequential approach yields a significant computational speed up in the training time over the end-to-end optimization. Furthermore, as the implicit models were designed with significantly less parameters than the FFN, the results highlight the former's parameter efficiency which we were again able to exploit better using the sequential training approach.

VI. CONCLUSION

In this work, we introduce a sequential, greedy algorithm for training triangular implicit deep models in a blockwise fashion. We show how the corresponding subproblems can be decomposed into a subroutine that alternates between a least squares optimization and an easily solved single hidden-layer neural network training problem. We theoretically prove that, for the more general non-strictly triangular ReLU implicit models, the challenging implicit constraints can be equivalently replaced by more tractable explicit constraints, allowing for our algorithm to be applied to such models. Experiments on function interpolation, as well as MNIST and Fashion-MNIST classification tasks, show that our algorithm learns models with performance superior than end-to-end optimization and conventional feed-forward neural networks, with computation times lower than the end-to-end approach, making the proposed sequential algorithm a promising new approach for training implicit deep models.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[2] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2017.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[7] L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Tsai, "Implicit deep learning," *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 3, pp. 930–958, 2021.

[8] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Advances in neural information processing systems*, vol. 31, 2018.

[9] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[10] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[11] F. Gu, H. Chang, W. Zhu, S. Sojoudi, and L. El Ghaoui, "Implicit graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 984–11 995, 2020.

[12] S. Bai, V. Koltun, and J. Z. Kolter, "Multiscale deep equilibrium models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5238–5250, 2020.

[13] C. Pabbaraju, E. Winston, and J. Z. Kolter, "Estimating Lipschitz constants of monotone deep equilibrium models," in *International Conference on Learning Representations*, 2020.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[15] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015.

[16] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

[17] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 646–661.

[18] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, ser. NIPS'06. Cambridge, MA, USA: MIT Press, 2006, p. 153–160.

[19] E. Belilovsky, M. Eickenberg, and E. Oyallon, "Greedy layerwise learning can scale to ImageNet," 2019.

[20] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[21] Y. Tian, "An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis," 2017.

[22] T. Ergen and M. Pilanci, "Convex optimization for shallow neural networks," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2019, pp. 79–83.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[24] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *ArXiv*, vol. abs/1708.07747, 2017.

APPENDIX I
GENERALIZATION OF THEOREM 2

Below, we generalize Theorem 2 to the case where the partition sizes n_j may be larger than unity. The result shows

that, again, the implicit constraint on $\mathbf{X}_i$ may be replaced by equivalent explicit ones.

Theorem 3. *Assume that $\phi = \mathrm{ReLU}$, and let $\mathbf{X}_i \in \mathbb{R}^{n_i \times m}$. Then, there exists $\{\mathbf{A}_{i,j}\}_{j=1}^{i}, \mathbf{B}_i, \{\mathbf{C}_j\}_{j=1}^{i}$ that together with $\mathbf{X}_i$ are feasible for (13) if and only if, for all $k \in \{1, 2, \ldots, n_i\}$ and all $l \in \{1, 2, \ldots, m\}$, there exists $\gamma_{k1}^{(1)}, \ldots, \gamma_{kn_1}^{(1)} \in \mathbb{R}$, …, $\gamma_{k1}^{(i-1)}, \ldots, \gamma_{kn_{i-1}}^{(i-1)} \in \mathbb{R}$, $\gamma_{k(k+1)}^{(i)}, \ldots, \gamma_{kn_i}^{(i)} \in \mathbb{R}$, and $\lambda_{k1}, \ldots, \lambda_{kp} \in \mathbb{R}$ such that*

$$(\mathbf{X}_i)_{kl} = \mathrm{ReLU}\bigg(\sum_{r=k+1}^{n_i} \gamma_{kr}^{(i)} (\mathbf{X}_i)_{rl} + \sum_{j=1}^{i-1} \sum_{r=1}^{n_j} \gamma_{kr}^{(j)} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} \lambda_{kr} (\mathbf{U})_{rl} \bigg). \tag{15}$$

Proof. Suppose that $\{\mathbf{A}_{i,j}\}_{j=1}^{i}, \mathbf{B}_i, \{\mathbf{C}_j\}_{j=1}^{i}$, together with $\mathbf{X}_i$, are feasible for (13). Then $-1 < \mathrm{diag}(\mathbf{A}_{i,i}) < 1$, $(\mathbf{A}_{i,i})_{kl} = 0$ for all $k > l$, and

$$\mathbf{X}_i = \mathrm{ReLU}\left(\sum_{j=1}^{i} \mathbf{A}_{i,j} \mathbf{X}_j + \mathbf{B}_i \mathbf{U} \right).$$

Let $k \in \{1, 2, \ldots, n_i\}$ and $l \in \{1, 2, \ldots, m\}$. If $(\mathbf{X}_i)_{kl} \neq 0$, then we have that

$$\begin{aligned}
0 < (\mathbf{X}_i)_{kl} &= \mathrm{ReLU}\left(\sum_{j=1}^{i} \sum_{r=1}^{n_j} (\mathbf{A}_{i,j})_{kr} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} (\mathbf{B}_i)_{kr} (\mathbf{U})_{rl} \right) \\
&= \sum_{j=1}^{i} \sum_{r=1}^{n_j} (\mathbf{A}_{i,j})_{kr} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} (\mathbf{B}_i)_{kr} (\mathbf{U})_{rl} \\
&= \sum_{r=1}^{n_i} (\mathbf{A}_{i,i})_{kr} (\mathbf{X}_i)_{rl} + \sum_{j=1}^{i-1} \sum_{r=1}^{n_j} (\mathbf{A}_{i,j})_{kr} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} (\mathbf{B}_i)_{kr} (\mathbf{U})_{rl} \\
&= \sum_{r=k}^{n_i} (\mathbf{A}_{i,i})_{kr} (\mathbf{X}_i)_{rl} + \sum_{j=1}^{i-1} \sum_{r=1}^{n_j} (\mathbf{A}_{i,j})_{kr} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} (\mathbf{B}_i)_{kr} (\mathbf{U})_{rl},
\end{aligned}$$

so

$$\begin{aligned}
(\mathbf{X}_i)_{kl} &= \sum_{r=k+1}^{n_i} \frac{(\mathbf{A}_{i,i})_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{X}_i)_{rl} + \sum_{j=1}^{i-1} \sum_{r=1}^{n_j} \frac{(\mathbf{A}_{i,j})_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} \frac{(\mathbf{B}_i)_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{U})_{rl} \\
&= \mathrm{ReLU}\bigg(\sum_{r=k+1}^{n_i} \frac{(\mathbf{A}_{i,i})_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{X}_i)_{rl} + \sum_{j=1}^{i-1} \sum_{r=1}^{n_j} \frac{(\mathbf{A}_{i,j})_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} \frac{(\mathbf{B}_i)_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{U})_{rl} \bigg).
\end{aligned}$$

On the other hand, if $(\mathbf{X}_i)_{kl} = 0$, then

$$\begin{aligned}
0 &\geq \sum_{j=1}^{i} \sum_{r=1}^{n_j} (\mathbf{A}_{i,j})_{kr} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} (\mathbf{B}_i)_{kr} (\mathbf{U})_{rl} \\
&= \sum_{r=1}^{n_i} (\mathbf{A}_{i,i})_{kr} (\mathbf{X}_i)_{rl} + \sum_{j=1}^{i-1} \sum_{r=1}^{n_j} (\mathbf{A}_{i,j})_{kr} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} (\mathbf{B}_i)_{kr} (\mathbf{U})_{rl} \\
&= \sum_{r=k+1}^{n_i} (\mathbf{A}_{i,i})_{kr} (\mathbf{X}_i)_{rl} + \sum_{j=1}^{i-1} \sum_{r=1}^{n_j} (\mathbf{A}_{i,j})_{kr} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} (\mathbf{B}_i)_{kr} (\mathbf{U})_{rl},
\end{aligned}$$

so

$$0 \geq \sum_{r=k+1}^{n_i} \frac{(\mathbf{A}_{i,i})_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{X}_i)_{rl} + \sum_{j=1}^{i-1} \sum_{r=1}^{n_j} \frac{(\mathbf{A}_{i,j})_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} \frac{(\mathbf{B}_i)_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{U})_{rl},$$

which implies that again

$$(\mathbf{X}_i)_{kl} = \mathrm{ReLU}\bigg(\sum_{r=k+1}^{n_i} \frac{(\mathbf{A}_{i,i})_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{X}_i)_{rl} + \sum_{j=1}^{i-1} \sum_{r=1}^{n_j} \frac{(\mathbf{A}_{i,j})_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{X}_j)_{rl} + \sum_{r=1}^{p} \frac{(\mathbf{B}_i)_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}} (\mathbf{U})_{rl} \bigg).$$

Thus, (15) holds with $\gamma_{kr}^{(j)} = \frac{(\mathbf{A}_{i,j})_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}}$ and $\lambda_{kr} = \frac{(\mathbf{B}_i)_{kr}}{1 - (\mathbf{A}_{i,i})_{kk}}$, which proves the forward direction.

Now, suppose that, for all $k \in \{1, 2, \ldots, n_i\}$ and all $l \in \{1, 2, \ldots, m\}$, there exists $\gamma_{k1}^{(1)}, \ldots, \gamma_{kn_1}^{(1)} \in \mathbb{R}$,

$\ldots$, $\gamma_{k1}^{(i-1)}, \ldots, \gamma_{kn_{i-1}}^{(i-1)} \in \mathbb{R}$, $\gamma_{k(k+1)}^{(i)}, \ldots, \gamma_{kn_i}^{(i)} \in \mathbb{R}$, and $\lambda_{k1}, \ldots, \lambda_{kp} \in \mathbb{R}$ such that (15) holds. Then, let $\epsilon > 0$ and define $\mathbf{A}_{i,j} \in \mathbb{R}^{n_i \times n_j}$ for $j \in \{1, 2, \ldots, i\}$ and $\mathbf{B}_i \in \mathbb{R}^{n_i \times p}$ by $(\mathbf{A}_{i,i})_{kk} = 1 - \epsilon$ for all $k \in \{1, 2, \ldots, n_i\}$, $(\mathbf{A}_{i,i})_{kl} = 0$ for all $k > l$, $(\mathbf{A}_{i,i})_{kr} = (1 - (\mathbf{A}_{i,i})_{kk})\gamma_{kr}^{(i)}$ for all $k < r$, $(\mathbf{A}_{i,j})_{kr} = (1 - (\mathbf{A}_{i,i})_{kk})\gamma_{kr}^{(j)}$ for all $k \in \{1, 2, \ldots, n_i\}$, all $r \in \{1, 2, \ldots, n_j\}$, and all $j \in \{1, 2, \ldots, i-1\}$, and $(\mathbf{B}_i) = (1 - (\mathbf{A}_{i,i})_{kk})\lambda_{kr}$ for all $k \in \{1, 2, \ldots, n_i\}$ and all $r \in \{1, 2, \ldots, p\}$. Let $\{\mathbf{C}_j\}_{j=1}^{i} \subseteq \mathbb{R}^{q \times 1}$ be arbitrary. Then, it is clear by construction that $-1 < \operatorname{diag}(\mathbf{A}_{i,i}) = (1 - \epsilon)\mathbf{1}_{n_i} < 1$ (where $\mathbf{1}_{n_i}$ is the n_i-vector of all ones), and $(\mathbf{A}_{i,i})_{kl} = 0$ for all $k > l$. Reversing the steps of the forward direction proof also shows that our choice of parameters yields $\mathbf{X}_i = \operatorname{ReLU}\left(\sum_{j=1}^{i} \mathbf{A}_{i,j}\mathbf{X}_j + \mathbf{B}_i\mathbf{U}\right)$, so $\{\mathbf{A}_{i,j}\}_{j=1}^{i}, \mathbf{B}_i, \{\mathbf{C}_j\}_{j=1}^{i}$, together with $\mathbf{X}_i$, are feasible for (13). $\square$