SIAM J. MATH. DATA SCI.
Vol. 3, No. 3, pp. 930–958

Implicit Deep Learning*

Laurent El Ghaoui†, Fangda Gu†, Bertrand Travacca‡, Armin Askari†, and Alicia Tsai†

Abstract. Implicit deep learning prediction rules generalize the recursive rules of feedforward neural networks. Such rules are based on the solution of a fixed-point equation involving a single vector of hidden features, which is thus only implicitly defined. The implicit framework greatly simplifies the notation of deep learning, and opens up many new possibilities in terms of novel architectures and algorithms, robustness analysis and design, interpretability, sparsity, and network architecture optimization.

Key words. deep learning, deep equilibrium models, Perron–Frobenius theory, fixed-point equations, robustness, adversarial attacks

AMS subject classifications. 690C26, 49M99, 65K10, 62M45, 26B10

DOI. 10.1137/20M1358517

1. Introduction.

1.1. Implicit prediction rules.

In this paper, we consider a new class of deep learning models that are based on implicit prediction rules. Such rules are not obtained via a recursive procedure through several layers, as in current neural networks. Instead, they are based on solving a fixed-point equation in some single "state" vector $x \in \mathbb{R}^n$. Precisely, for a given input vector u, the predicted vector is

$$\hat{y}(u) = Cx + Du \qquad \text{(prediction equation)}, \tag{1.1a}$$

$$x = \phi(Ax + Bu) \qquad \text{(equilibrium equation)}, \tag{1.1b}$$

where $\phi : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear vector map (the "activation" map), and matrices A, B, C, D contain model parameters. Figure 1 provides a block-diagram view of an implicit model, to be read from right to left, so as to be consistent with matrix-vector multiplication rules.

We can think of the vector $x \in \mathbb{R}^n$ as a "state" corresponding to n "hidden" features that are extracted from the inputs, based on the so-called equilibrium equation (1.1b). In general, that equation cannot be solved in closed-form, and the model above provides x only *implicitly*. This equation is not necessarily well-posed, in the sense that it may not admit a solution, let alone a unique one; we discuss this important issue of well-posedness in section 2.

For notational simplicity only, our rule does not contain any bias terms; we can easily account for those by considering the vector $(u, 1)$ instead of u, thereby increasing the column

*Received by the editors August 7, 2020; accepted for publication (in revised form) June 18, 2021; published electronically September 16, 2021.
https://doi.org/10.1137/20M1358517
Funding: This work was partially supported by sumup.ai, the National Science Foundation, Total S.A., the Pacific Earthquake Engineering Research Center, and Berkeley Artificial Intelligence Laboratory (BAIR).

†EECS and IEOR Departments, UC Berkeley, Berkeley, CA 94720 USA (elghaoui@berkeley.edu, gfd18@berkeley.edu, armin.askari@gmail.com, aliciatsai@berkeley.edu).
‡CEE Department, UC Berkeley, Berkeley, CA 94720 USA (bertrand.travacca@berkeley.edu).

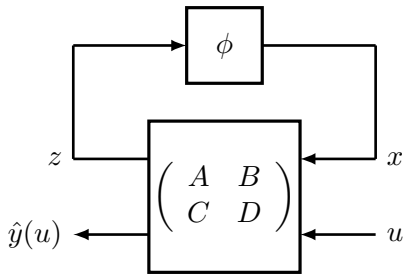

Figure 1. *A block-diagram view of an implicit model.*

dimension of B by one.

Perhaps surprisingly, as seen in section 3, the implicit framework includes most current neural network architectures as special cases. Implicit models are a much wider class: they present much more capacity, as measured by the number of parameters for a given dimension of the hidden features; also, they allow for cycles in the network, which is not permitted under the current paradigm of deep networks.

Implicit rules open up the possibility of using novel architectures and prediction rules for deep learning, which are not based on any notion of "network" or "layers," as is classically understood. In addition, they allow one to consider rigorous approaches to challenging problems in deep learning, including robustness analysis, sparsity, and interpretability, and feature selection.

1.2. Contributions and paper outline. Our contributions in this paper, and its outline, are as follows.

- *Well-posedness and composition* (section 2): In contrast with standard deep networks, implicit models may not be well-posed, in the sense that the equilibrium equation may have no or multiple solutions. We establish rigorous and numerically tractable conditions for implicit rules to be well-posed. These conditions are then used in the training problem, guaranteeing the well-posedness of the learned prediction rule. We also discuss the composition of implicit models, via cascade connections, for example.
- *Implicit models of neural networks* (section 3): We provide details on how to represent a wide variety of neural networks as implicit models, building on the composition rules of section 2.
- *Robustness analysis* (section 4): We describe how to analyze the robustness properties of a given implicit model, deriving bounds on the state under input perturbations, and generating adversarial attacks. We also discuss which penalties to include in the training problem so as to encourage robustness of the learned rule.
- *Interpretability, sparsity, compression, and deep feature selection* (section 5): Here we focus on finding appropriate penalties to use in order to improve properties such as model sparsity, or obtain feature selection. We also discuss the impact of model errors.
- *Training problem: formulations and algorithms* (section 6): Informed by our previous findings, we finally discuss the corresponding training problem. Following the work of [21] and [32], we represent activation functions using so-called Fenchel divergences to relax the training problem into a more tractable form. We discuss several algorithms,

including stochastic projected gradients, Frank–Wolfe, and block-coordinate descent. Section 7 provides a few experiments supporting the theory put forth in this paper. Finally, section 8 is devoted to prior work and references.

1.3. Notation. For a matrix U, $|U|$ (resp., U_+) denotes the matrix with the absolute values (resp., positive part) of the entries of U. For a vector v, we denote by $\mathbf{diag}(v)$ the diagonal matrix formed with the entries of v; for a square matrix V, $\mathbf{diag}(V)$ is the vector formed with the diagonal elements of V. The notation $\mathbf{1}$ refers to the vector of ones, with size inferred from context. The Hadamard (componentwise) product between two n-vectors x, y is denoted $x \odot y$. We use $s_k(z)$ to denote the sum of the largest k entries of a vector z. For a matrix A, and integers $p, q \geq 1$, we define the induced norm

$$\|A\|_{p\to q} = \max_{\xi} \ \|A\xi\|_q \ : \ \|\xi\|_p \leq 1.$$

The case when $p = q = \infty$ corresponds to the l_∞-induced norm of A, also known as its *max-row-sum norm*:

$$\|A\|_\infty := \max_i \sum_j |A_{ij}|.$$

We denote the set $\{1, \ldots, L\}$ compactly as $[L]$. For an n-vector partitioned into L blocks, $z = (z_1, \ldots, z_L)$, with $z_l \in \mathbb{R}^{n_l}$, $l \in [L]$, with $n_1 + \cdots + n_L = n$, we denote by $\eta(z)$ the L-vector of norms:

$$\eta(z) := (\|z_1\|_{p_1}, \ldots, \|z_L\|_{p_L})^\top. \tag{1.2}$$

Finally, any square, nonnegative matrix M admits a real eigenvalue that is larger than the modulus of any other eigenvalue; this nonnegative eigenvalue is the so-called *Perron–Frobenius (PF) eigenvalue* [36] and is denoted $\lambda_{\mathrm{PF}}(M)$.

2. Well-posedness and composition.

2.1. Assumptions on the activation map. We restrict our attention to activation maps ϕ that obey a "Blockwise LIPschitz" (BLIP) continuity condition. This condition is satisfied for most popular activation maps and arises naturally when "composing" implicit models (see subsection 2.4). Precisely, we assume that:

1. *Blockwise:* The map ϕ acts in a blockwise fashion, that is, there exists a partition of n, i.e., $n = n_1 + \cdots + n_L$, such that for every vector partitioned into the corresponding blocks $z = (z_1, \ldots, z_L)$ with $z_l \in \mathbb{R}^{n_l}$, $l \in [L]$, we have $\phi(z) = (\phi_1(z_1), \ldots, \phi_L(z_L))$ for appropriate maps $\phi_l \ : \ \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$, $l \in [L]$.
2. *Lipschitz:* For every $l \in [L]$, the maps ϕ_l are Lipschitz-continuous with constant $\gamma_l > 0$ with respect to the l_{p_l}-norm for some integer $p_l \geq 1$:

$$\forall\, u, v \in \mathbb{R}^{n_l} \ : \ \|\phi_l(u) - \phi_l(v)\|_{p_l} \leq \gamma_l \|u - v\|_{p_l}.$$

In the remainder of the paper, we refer to such maps with the acronym BLIP, omitting the dependence on the underlying structure information (integers n_l, p_l, γ_l, $l \in [L]$). We shall

consider a special case, referred to as COmponentwise Non-Expansive (CONE) maps, when $n_l = 1$, $\gamma_l = 1$, $l \in [L]$. Such CONE maps satisfy

$$\forall\, u, v \in \mathbb{R}^n \;:\; |\phi(u) - \phi(v)| \leq |u - v|, \tag{2.1}$$

with inequality and absolute value taken componentwise. Examples of CONE maps include the ReLU (defined as $\phi(\cdot) = \max(0, \cdot)$) and its "leaky" variants (tanh and sigmoid), each applied componentwise to a vector input. Our model also allows for maps that do not operate componentwise, such as the softmax function, which operates on an n-vector z as

$$z \to \mathrm{SoftMax}(z) := \left(\frac{e^{z_i}}{\sum_i e^{z_j}} \right)_{i \in [n]}. \tag{2.2}$$

The softmax map is 1-Lipschitz-continuous with respect to the l_1-norm [18].

2.2. Well-posed matrices. We consider the prediction rule (1.1a) with input point $u \in \mathbb{R}^p$ and predicted output vector $\hat{y}(u) \in \mathbb{R}^q$. The equilibrium equation (1.1b) does not necessarily have a well-defined, unique solution x. In order to ensure this, we assume that the $n \times n$ matrix A satisfies the following well-posedness property.

Definition 2.1 (Well-posedness property). *The $n \times n$ matrix A is said to be well-posed for ϕ (in short, $A \in \mathrm{WP}(\phi)$) if, for any n-vector b, the equation in $x \in \mathbb{R}^n$,*

$$x = \phi(Ax + b), \tag{2.3}$$

has a unique solution.

There are many classes of matrices that satisfy the well-posedness property. As seen next, strictly upper-triangular matrices are well-posed with respect to any activation map that acts componentwise; such a class arises when modeling feedforward neural networks as implicit models, as seen in subsection 3.2.

2.3. Tractable sufficient conditions for well-posedness. Our goal now is to understand how we can constrain A to have the well-posedness property, in a numerically tractable way.

We assume that ϕ is a BLIP map, as defined in subsection 2.1. We partition the A matrix according to the tuple $(n_1, \ldots, n_L)$, into blocks $A_{ij} \in \mathbb{R}^{n_i \times n_j}$, $1 \leq i, j \leq L$, and define an $L \times L$ matrix of induced norms $N(A, \gamma) \in \mathbb{R}_+^{L \times L}$, with elements for $l, h \in [L]$ given by

$$(N(A,\gamma))_{ij} := \gamma_i \|A_{ij}\|_{p_j \to p_i} = \gamma_i \max_{\xi} \; \|A_{ij}\xi\|_{p_i} \;:\; \|\xi\|_{p_j} \leq 1. \tag{2.4}$$

In the case of CONE maps, the vector γ is all ones, and we have simply $N(A, \gamma) = |A|$.

The sufficient condition stated next is based on the contraction mapping theorem [41, p. 83].

Theorem 2.2 (PF sufficient condition for well-posedness for BLIP activation). *Assume that ϕ satisfies the BLIP condition, as defined in subsection 2.1. Then, A is well-posed with respect to ϕ if*

$$\lambda_{\mathrm{PF}}(N(A, \gamma)) < 1, \tag{2.5}$$

where $N(A,\gamma)$ is the matrix of induced norms defined in (2.4). Then, for any n-vector b, the solution to (2.3) can be computed via the fixed-point iteration:

$$x(0) = 0, \quad x(t+1) = \phi(Ax(t) + b), \quad t = 0, 1, 2, \ldots. \tag{2.6}$$

When ϕ is a CONE map, the PF condition (2.5) reduces to $\lambda_{\mathrm{PF}}(|A|) < 1$.

Proof. Let $b \in \mathbb{R}^n$. Our first step is to establish that for the Picard iteration (2.6), we have, for every $t \geq 1$,

$$\eta(x(t+1) - x(t)) \leq N(A,\gamma)\eta(x(t) - x(t-1)).$$

Here, η is a vector of norms, as defined in (1.2). For every $l \in [L]$, $t \geq 0$,

$$\begin{aligned}
[\eta(x(t+1) - x(t))]_l &= \|\phi_l([Ax(t) + b]_l) - \phi_l([Ax(t-1) + b]_l)\|_{p_l} \quad \text{[using (2.6)]} \\
&\leq \gamma_l \|[A(x(t) - x(t-1))]_l\|_{p_l} = \gamma_l \Big\| \sum_{h \in [L]} A_{lh}(x(t) - x(t-1))_h \Big\|_{p_l} \\
&\leq \gamma_l \sum_{h \in [L]} \|A_{lh}\|_{p_h \to p_l} \|x_h(t) - x_h(t-1)\|_{p_h} = [N(A,\gamma)\eta(x(t) - x(t-1))]_l,
\end{aligned}$$

which establishes the desired bound, where $M := N(A,\gamma)$.

Assume now that $\lambda_{\mathrm{PF}}(M) < 1$, as posited in the theorem. Then, from the Perron–Frobenius theorem, $I - M$ is nonsingular and all the other (possibly complex) eigenvalues λ of $N(A,\gamma)$ satisfy $|\lambda| \leq \lambda_{\mathrm{PF}}(N(A,\gamma)) < 1$. We prove existence of a solution to the equilibrium equation by showing that the sequence of Picard iterates is Cauchy: For every $t, \tau \geq 0$,

$$\eta(x(t+\tau) - x(t)) \leq \sum_{k=t}^{t+\tau} M^k \eta(x(1) - x(0)) \leq M^t \sum_{k=0}^{\tau} M^k \eta(x(1) - x(0)) \leq M^t w,$$

where $w \in \mathbb{R}^L_+$ is defined by

$$w := \sum_{k=0}^{+\infty} M^k \eta(x(1) - x(0)) = (I - M)^{-1} \eta(x(1) - x(0)).$$

As M^t converges to 0 irrespective of τ, the sequence of Picard iterates is Cauchy, and therefore the sequence $\{x(t)\}$ converges to $x \in \mathbb{R}^n$ and $x = \phi(Ax + b)$. The above proves the existence of a solution.

To prove unicity, consider $x^1, x^2 \in \mathbb{R}^n_+$ two solutions to the equation. Using the hypotheses in the theorem, we have, for any $k \geq 1$,

$$\eta(x^1 - x^2) \leq M\eta(x^1 - x^2) \leq M^k \eta(x^1 - x^2).$$

The fact that $M^k \to 0$ as $k \to +\infty$ then establishes unicity. ∎

Remark 2.3. The fixed-point iteration (2.6) has linear convergence; each iteration is a matrix-vector product, and hence the complexity of solving the equilibrium equation is comparable to that of a forward pass through a network of similar size.

Remark 2.4. The PF condition $\lambda_{\mathrm{PF}}(N(A,\gamma)) < 1$ is not convex in A, but the convex condition $\|N(A,\gamma)\|_\infty < 1$ is sufficient, in light of the bound $\|M\|_\infty \geq \lambda_{\mathrm{PF}}(|M|)$, and is valid for any square matrix M.

Remark 2.5. The PF condition of Theorem 2.2 is conservative. For example, a triangular matrix A is well-posed with respect to the ReLU and if and only if $\mathbf{diag}(A) < \mathbf{1}$, a consequence of the upcoming Theorem 2.7. The corresponding equilibrium equation can then be solved via the backward recursion

$$x_n = \frac{(b_n)_+}{1 - A_{nn}}, \quad x_i = \frac{1}{1 - A_{ii}}\big(b_i + \sum_{j>i} A_{ij}x_j\big)_+, \quad i = n-1, \dots, 1.$$

Such a matrix does not necessarily satisfy the PF condition; we can have in particular $A_{11} < -1$, which implies $\lambda_{\mathrm{PF}}(|A|) > 1$.

Remark 2.6. The well-posedness property is invariant under row and column permutation, provided ϕ acts componentwise. Precisely, if A is well-posed with respect to a componentwise CONE map ϕ, then for any $n \times n$ permutation matrix P, $PAP^\top$ is well-posed with respect to ϕ. The PF sufficient condition is also invariant under row and column permutations. A similar statement can be made for the more general BLIP case.

2.4. Composition of implicit models. Implicit models can be easily composed via matrix algebra. Sometimes, the connection preserves well-posedness, thanks to the following result.

Theorem 2.7 (Well-posedness of block-triangular matrices, componentwise activation). *Assume that the activation map ϕ acts componentwise. The upper block-triangular matrix*

$$A := \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

with $A_{ii} \in \mathbb{R}^{n_i \times n_i}$, $i = 1, 2$, is well-posed with respect to ϕ if and only if its diagonal blocks A_{11}, A_{22} are.

Proof. Express the equation $x = \phi(Ax + b)$ as

$$x_1 = \phi(A_{11}x_1 + A_{12}x_2 + b_1), \quad x_2 = \phi(A_{22}x_2 + b_2),$$

where $b = (b_1, b_2)$, $x = (x_1, x_2)$, with $b_i \in \mathbb{R}^{n_i}$, $x_i \in \mathbb{R}^{n_i}$, $i = 1, 2$. Here, since ϕ acts componentwise, we use the same notation ϕ in the two equations.

Now assume that A_{11} and A_{22} are well-posed with respect to ϕ. Since A_{22} is well-posed for ϕ, the second equation has a unique solution x_2^*; plugging $x_2 = x_2^*$ into the second equation, and using the well-posedness of A_{11}, we see that the first equation has a unique solution in x_1; hence A is well-posed.

To prove the converse direction, assume that A is well-posed. The second equation above must have a unique solution x_2^*, irrespective of the choice of b_2; hence A_{22} must be well-posed. To prove that A_{11} must be well-posed, too, set $b_2 = 0$, b_1 arbitrary, leading to the system

$$x_1 = \phi(A_{11}x_1 + A_{12}x_2 + b_1), \quad x_2 = \phi(A_{22}x_2).$$

Since A_{22} is well-posed for ϕ, there is a unique solution x_2^* to the second equation; the first equation then reads $x_1 = \phi(A_{11}x_1 + b_1 + A_{12}x_2^*)$. It must have a unique solution for any b_1; hence A_{11} is well-posed. ∎

This result establishes the fact stated previously, that when ϕ is the ReLU, an upper-triangular matrix $A \in \mathrm{WP}(\phi)$ if and only if $\mathbf{diag}(A) < \mathbf{1}$. A similar result holds with the lower block-triangular matrix

$$A := \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix},$$

where $A_{12} \in \mathbb{R}^{n_1 \times n_2}$ is arbitrary. It is possible to extend this result to activation maps ϕ that satisfy the block Lipschitz continuity (BLIP) condition, in which case we need to assume that the partition of A into blocks is consistent with that of ϕ. As seen later, this feature arises naturally when composing implicit models from well-posed blocks.

Theorem 2.8 (Well-posedness of block-triangular matrices, blockwise activation). *Assume that the matrix A can be written as*

$$A := \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

with $A_{ii} \in \mathbb{R}^{n_i \times n_i}$, $i = 1, 2$, and ϕ acts blockwise accordingly, in the sense that there exist two maps ϕ_1, ϕ_2 such that $\phi((z_1, z_2)) = (\phi_1(z_1), \phi_2(z_2))$ for every $z_i \in \mathbb{R}^{n_i}$, $i = 1, 2$. Then A is well-posed with respect to ϕ if and only if for $i = 1, 2$, A_{ii} is well-posed with respect to ϕ_i.

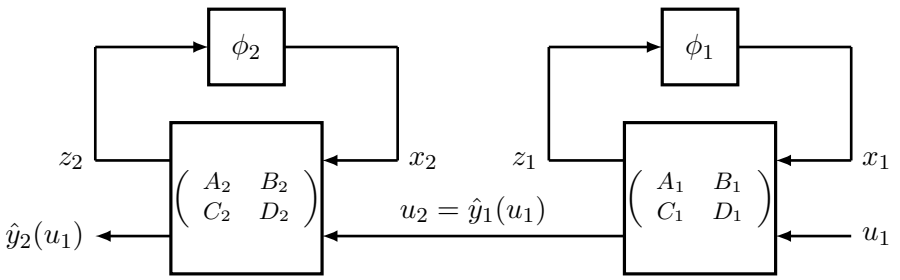

Figure 2. *Cascade connection of two implicit models.*

Using the above results, we can preserve well-posedness of implicit models via composition. For example, given two models with matrix parameters (A_i, B_i, C_i, D_i) and activation functions ϕ_i, $i = 1, 2$, we can consider a "cascaded" prediction rule (visualized in Figure 2):

$$\hat{y}_2 = C_2x_2 + D_2u_2 \text{ where } u_2 = \hat{y}_1 = C_1x_1 + D_1u_1, \text{ where } x_i = \phi_i(A_ix_i + B_iu_i), \quad i = 1, 2.$$

The above rule can be represented as (1.1a), with $x = (x_2, x_1)$, $\phi((z_2, z_1)) = (\phi_2(z_2), \phi_1(z_1))$, and

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{cc|c} A_2 & B_2C_1 & B_2D_1 \\ 0 & A_1 & B_1 \\ \hline C_2 & D_2C_1 & D_2D_1 \end{array}\right).$$

Due to Theorem 2.8, the cascaded rule is well-posed for the componentwise map with values $\phi(z_1, z_2) = (\phi_1(z_1), \phi_2(z_2))$ if and only if each rule is.

A similar result holds if we put two or more well-posed models in parallel and do a (weighted) sum of the outputs. With the above notation, setting $\hat{y}(u_1, u_2) = \hat{y}_1(u_1) + \hat{y}_2(u_2)$ leads to a new implicit model that is also well-posed. Other possible connections include concatenation, i.e., $\hat{y}(u) = (\hat{y}_1(u), \hat{y}_2(u))$, and affine transformations (a special case of cascade connection where one of the systems has no activation). We leave the details to the reader.

In both cascade and parallel connections, the triangular structure of the matrix A of the composed system ensures that the PF sufficient condition for well-posedness is satisfied for the composed system if and only if it holds for each subsystem.

Multiplicative connections in general are not Lipschitz-continuous unless the inputs are bounded. Precisely, consider two activation maps ϕ_i that are Lipschitz-continuous with constant γ_i and are bounded, with $|\phi_i(v)| \leq c_i$ for every v, $i = 1, 2$; then, the multiplicative map

$$(u_1, u_2) \in \mathbb{R}^2 \to \phi(u) = \phi_1(u_1)\phi_2(u_2)$$

is Lipschitz-continuous with respect to the l_1-norm, with constant $\gamma := \max\{c_2\gamma_1, c_1\gamma_2\}$. Such connections arise in the context of attention units in neural networks, which use (bounded) activation maps such as tanh.

Finally, feedback connections are also possible. Consider two well-posed implicit systems:

$$y_i = C_i x_i + D_i u_i, \quad x_i = \phi_i(A_i x_i + B u_i), \quad i = 1, 2.$$

Now let us connect them in a feedback connection: The combined system is described by the implicit rule (1.1), where $u_1 = u + y_2$, $u_2 = y_1 = y$. The feedback system is also an implicit model of the form (1.1), with appropriate matrices (A, B, C, D), and activation map acting blockwise: $\phi(z_1, z_2) = (\phi_1(z_1), \phi_2(z_2))$ and state (x_1, x_2). In the simplified case when $D_1 = D_2 = 0$, the feedback connection has the model matrix

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{cc|c} A_1 & B_1C_2 & B_1 \\ B_2C_1 & A_2 & 0 \\ \hline C_1 & 0 & 0 \end{array}\right).$$

Note that the connection is not necessarily well-posed, even if both subsystems are.

2.5. Scaling implicit models. Assume that the activation map ϕ is componentwise nonexpansive (CONE) and positively homogeneous, as is the ReLU or its leaky version. Consider an implicit model of the form (1.1), and assume that it satisfies the PF sufficient condition for well-posedness of Theorem 2.2: $\lambda_{\mathrm{PF}}(|A|) \leq \kappa$, where $0 \leq \kappa < 1$ is given. Then, there is another implicit model with the same activation map ϕ and matrices (A', B', C', D'), which has the same prediction rule (i.e., $\hat{y}'(u) = \hat{y}(u)\ \forall u$) and satisfies $\|A'\|_\infty < 1$.

This result is a direct consequence of the following expression of the PF eigenvalue as an optimally scaled l_∞-norm, known as the Collatz–Wielandt formula (see [36, p. 666]):

$$\lambda_{\mathrm{PF}}(|A|) = \inf_S \|S^{-1}|A|S\|_\infty \ : \ S = \mathbf{diag}(s), \quad s > 0. \tag{2.7}$$

The above expression implies that the condition $\lambda_{\mathrm{PF}}(|A|) < 1$ guarantees the existence of a diagonal state scaling operator S such that $\|S^{-1}|A|S\|_\infty < 1$; in fact such a scaling can be obtained via fixed-point iterations, based on the formula $s = (I - |A|)^{-1}\mathbf{1}$. The new model matrices are then $A' = S^{-1}AS$, $B' = S^{-1}B$, $C' = CS$, $D' = D$.

In a training problem, this result allows us to consider the convex constraint $\|A\|_\infty < 1$ in lieu of its Perron–Frobenius eigenvalue counterpart. This result also allows us to rescale any given implicit model, such as one derived from deep neural networks, so that the norm condition is satisfied; we will exploit this in our robustness analyses in section 4.

3. Implicit models of deep neural networks.

A large number of deep neural networks can be modeled as implicit models, including convolutional and recurrent networks, attention units, residual connections, etc.

3.1. Well-posedness.

Thanks to the composition rules of subsection 2.4, it suffices to model individual layers, since a neural network is just a cascade connection of such layers. The block-Lipschitz structure of the activation map, and the strictly triangular structure of the matrix A, then emerge naturally as the result of composing the layers in a "cascade" fashion. This implies that the implicit models we obtain are well-posed, as the corresponding Perron–Frobenius eigenvalue of the matrix $N(A, \gamma)$ defined in (2.4) is zero, since $N(A, \gamma)$ is then strictly triangular.

We may always assume that the resulting implicit model satisfies the stronger norm condition for well-posedness mentioned in Remark 2.4. For example, in the case of a CONE map ϕ, the stronger condition $\|A\|_\infty < 1$ can always be obtained by appropriately scaling the weight matrices of the network's layers and by using a scaled version for the state vector x.

3.2. Dense feedforward networks.

We now illustrate the construction of an implicit model for the following fully dense feedfoward network, leaving the construction for other types of networks, including convolutional and recurrent networks, to the supplementary material (idl_supplement.pdf [local/web 584KB]). Consider the following prediction rule, with $L > 1$ fully connected layers:

$$\hat{y}(u) = W_L x_L, \quad x_{l+1} = \phi_l(W_l x_l), \quad x_0 = u. \tag{3.1}$$

Here $W_l \in \mathbb{R}^{n_{l+1} \times n_l}$ and $\phi_l : \mathbb{R}^{n_{l+1}} \to \mathbb{R}^{n_{l+1}}$, $l = 1, \ldots, L$, are given weight matrices and activation maps, respectively. We can express the above rule as (1.1a), with $x = (x_L, \ldots, x_1)$,

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{cccc|c} 0 & W_{L-1} & \ldots & 0 & 0 \\ & 0 & \ddots & \vdots & \vdots \\ & & \ddots & W_1 & 0 \\ & & & 0 & W_0 \\ \hline W_L & 0 & \ldots & 0 & 0 \end{array}\right), \tag{3.2}$$

and an appropriately defined blockwise activation function ϕ, defined as operating on an n-vector $z = (z_L, \ldots, z_1)$ as $\phi(z) = (\phi_L(z_L), \ldots, \phi_1(z_1))$. Due to the strictly upper-triangular structure of A, the system is well-posed, irrespective of A; in fact the equilibrium equation $x = \phi(Ax + Bu)$ is easily solved via backward block substitution, which corresponds to a

simple forward pass through the network. Note that choosing the state vector to list the hidden variables in the natural order, instead of the reverse one, would lead to a strictly *lower* triangular matrix A.

4. Robustness. In this section, our goal is to analyze the robustness properties of a given implicit model (1.1a). We seek to bound the state, output, and loss function, under unknown-but-bounded inputs. This robustness analysis task is of interest in itself for diagnosis or for generating adversarial attacks. It will also inform our choice of appropriate penalties or constraints in the training problem. We assume that the activation map is BLIP, and that the matrix A of the implicit model satisfies the sufficient conditions for well-posedness outlined in Theorem 2.2.

4.1. Input uncertainty models. We assume that the input vector is uncertain, and only known to belong to a given set $\mathcal{U} \subseteq \mathbb{R}^p$. Our results apply to a wide variety of sets $\mathcal{U}$; we will focus on the following two cases.

The first case corresponds to a box:

$$\mathcal{U}^{\text{box}} := \left\{u \in \mathbb{R}^p \ : \ |u - u^0| \leq \sigma_u\right\}. \tag{4.1}$$

Here, the p-vector $\sigma_u > 0$ is a measure of componentwise uncertainty affecting each data point, while u^0 corresponds to a vector of "nominal" inputs. The following variant limits the number of changes in the vector u:

$$\mathcal{U}^{\text{card}} := \left\{u \in \mathbb{R}^p \ : \ |u - u^0| \leq \sigma_u, \ \ \mathbf{Card}(u - u^0) \leq k\right\}, \tag{4.2}$$

where **Card** denotes the cardinality (number of nonzero components) in its vector argument, and $k < p$ is a given integer.

4.2. Box bounds on the state vector. Assume first that ϕ is a CONE map, and that the input is only known to belong to the box set $\mathcal{U}^{\text{box}}$ (4.1). We seek to find componentwise bounds on the state vector x, of the form $|x - x^0| \leq \sigma_x$, with x and x^0 the unique solution to $\xi = \phi(A\xi + Bu)$ and $\xi = \phi(A\xi + Bu^0)$, respectively, and $\sigma_x > 0$. We have

$$|x - x^0| = |\phi(Ax + Bu) - \phi(Ax^0 + Bu^0)| \leq |A||x - x^0| + |B(u - u^0)|,$$

which shows in particular that

$$\|x - x^0\|_\infty \leq \|A\|_\infty \|x - x^0\|_\infty + \|B(u - u^0)\|_\infty;$$

hence, provided $\|A\|_\infty < 1$, we have

$$\|x - x^0\|_\infty \leq \frac{\||B|\sigma_u\|_\infty}{1 - \|A\|_\infty}. \tag{4.3}$$

With the cardinality constrained uncertainty set $\mathcal{U}^{\text{card}}$ (4.2), we obtain

$$\|x - x^0\|_\infty \leq \frac{\delta}{1 - \|A\|_\infty}, \ \ \delta := \max_{1 \leq i \leq n} \ s_k(\sigma_u \odot |B|^\top e_i), \tag{4.4}$$

with e_i the ith unit vector in $\mathbb{R}^n$, and s_k the sum of the top k entries in a vector.

We may refine the analysis above with a "box" (componentwise) bound. When ϕ is a blockwise Lipschitz (BLIP) map, our result involves the matrix of norms $N(A,\gamma)$ defined in (2.4), as well as a similar matrix defined for the input matrix B: Decomposing B into blocks $B = (B_{li})_{l\in[L],i\in[p]}$, with $B_{li} \in \mathbb{R}^{n_l}$, $l \in [L]$, we define the $L \times p$ matrix of norms

$$N(B,\gamma) := \gamma_l(\|B_{li}\|_{p_l})_{l\in[L],\, i\in[p]}. \tag{4.5}$$

Theorem 4.1 (Box bound on the vector norms of the state, BLIP map). *Assume that ϕ is BLIP, and the corresponding sufficient well-posedness condition of Theorem 2.2 is satisfied. Then, $I - N(A,\gamma)$ is invertible, and*

$$\eta(x - x^0) \le (I - N(A,\gamma))^{-1} N(B,\gamma)\sigma_u, \tag{4.6}$$

where the vector of norms function $\eta(\cdot)$ is defined as in (1.2).

Proof. For every $l \in [L]$,

$$\begin{aligned}
[\eta(x - x^0)]_l &\le \|[\phi(A(x - x^0) + B(u - u^0)b)]_l\|_{p_l} \\
&\le \gamma_l \|\sum_{h\in[L]} A_{lh}(x - x^0)_h\|_{p_l} + \gamma_l \|\sum_{i\in[p]} B_{li}(u - u^0)_i\|_{p_l} \\
&\le \gamma_l \sum_{h\in[L]} \|A_{lh}\|_{p_h\to p_l}\eta(x - x^0)_h + \gamma_l \sum_{i\in[p]} \|B_{li}\|_{p_l}|u - u^0|_i \\
&\le [N(A,\gamma)\eta(x - x^0)]_l + [N(B,\gamma)|u - u^0|]_l,
\end{aligned}$$

which establishes the desired bound. ∎

Note that the box bound can be computed via fixed-point iterations. For example, when ϕ is a CONE map, we solve $(I - |A|)\sigma_x = |B|\sigma_u$ as the limit point of the fixed-point iteration

$$\sigma_x(0) = 0, \quad \sigma_x(t+1) = |A|\sigma_x(t) + |B|\sigma_u, \quad t = 0, 1, 2, \ldots,$$

which converges since $\lambda_{\mathrm{PF}}(|A|) < 1$.

4.3. Bounds on the output and the sensitivity matrix. The above allows us to analyze the effect of input noise on the output vector y. Let us assume that the function ϕ satisfies the CONE (componentwise non-expansiveness) condition (2.1). In addition, we assume that the stronger condition for well-posedness, $\|A\|_\infty < 1$, is satisfied. (As noted in subsection 2.5, we can always rescale the model so as to ensure that property, provided that $\lambda_{\mathrm{PF}}(|A|) < 1$.) For the implicit prediction rule (1.1), we then have

$$\forall\, u, u^0 \;:\; \|\hat{y}(u) - \hat{y}(u^0)\|_\infty \le \rho\|u - u^0\|_\infty, \text{ where } \rho := \frac{\|B\|_\infty\|C\|_\infty}{1 - \|A\|_\infty} + \|D\|_\infty.$$

The above shows that the prediction rule is Lipschitz-continuous, with a constant bounded above by ρ. This result motivates the use of the $\|\cdot\|_\infty$-norm as a penalty on model matrices A, B, C, D, for example, via a convex penalty that bounds the Lipschitz constant above:

$$\rho \le \frac{1}{2}\frac{\|B\|_\infty^2 + \|C\|_\infty^2}{1 - \|A\|_\infty} + \|D\|_\infty. \tag{4.7}$$

We can refine this analysis with the following theorem, applicable to the case when ϕ is blockwise Lipschitz (BLIP). Decomposing C into column blocks $C = (C_1, \ldots, C_L)$, with $C_l \in \mathbb{R}^{q \times n_l}$, $l \in [L]$, we define the matrix of (dual) norms

$$N(C) := (\|C_{il}\|_{p_l^*})_{l \in [L],\, i \in [q]},$$

where $p_l^* := 1/(1 - 1/p_l)$, $l \in [L]$. Also recall the corresponding matrix of norms related to A in (2.4) and B in (4.5).

Theorem 4.2 (Box bound on the output, BLIP map). *Assume that ϕ is a BLIP map, and that the sufficient condition for well-posedness $\lambda_{\mathrm{PF}}(N(A, \gamma)) < 1$ is satisfied. Then, $I - N(A, \gamma)$ is invertible, and*

$$\forall\, u, u^0 \;:\; |\hat{y}(u) - \hat{y}(u^0)| \le S|u - u^0|, \tag{4.8}$$

where the (nonnegative) $q \times p$ matrix

$$S := N(C)(I - N(A, \gamma))^{-1} N(B, \gamma) + |D|$$

is a "sensitivity matrix" of the implicit model with a BLIP map. In the case of a CONE map, the sensitivity matrix is given by

$$S = |C|(I - |A|)^{-1}|B| + |D|.$$

Proof. For given $i \in [q]$, we have

$$\begin{aligned} &|\hat{y}(u) - \hat{y}(u^0)|_i \\ &\quad \le \Big| \sum_{l \in [L]} C_{il}(x - x^0)_l \Big| + (|D||u - u^0|)_i \le \sum_{l \in [L]} \|C_{il}\|_{p_l^*} \eta (x - x^0)_l + (|D||u - u^0|)_i. \end{aligned}$$

■

The sensitivity matrix can be computed via fixed-point iterations that are guaranteed to converge, thanks to the well-posedness assumption in the theorem. In the case of CONE maps, the iterations involve finding the limit point X_∞ of

$$X(t+1) = |A|X(t) + |B|, \quad t = 0, 1, 2, \ldots,$$

and setting $S = |C|X_\infty + |D|$. Our refined analysis suggests a "natural" penalty to use during the training phase in order to improve robustness, namely $\|S\|_\infty$.

4.4. Linear programming (LP) relaxation for CONE maps. The previous bounds do not provide a direct way to generate an adversarial attack, that is, a feasible point $u \in \mathcal{U}$ that has maximum impact on the state. In some cases it may be possible to refine our previous box bounds via an LP relaxation, which has the advantage of suggesting a specific adversarial attack. Here we restrict our attention to the ReLU activation: $\phi(z) = \max(z, 0) = z_+$, which is a CONE map.

We consider the problem

$$p^* := \max_{x, u \in \mathcal{U}} \sum_{i \in [n]} f_i(x_i) \;:\; x = z_+, \;\; z = Ax + Bu, \;\; |x - x^0| \le \sigma_x, \tag{4.9}$$

where f_i's are arbitrary functions. Setting $f_i(\xi) = (\xi - x_i^0)^2$, $i \in [n]$, leads to the problem of finding the largest discrepancy (measured in l_2-norm) between x and x^0; setting $f_i(\xi) = -\xi$, $i \in [n]$, finds the minimum l_1-norm state. By construction, our bound can only improve on the previous state bound, in the sense that

$$p^* \leq \sum_{i\in[n]} \max_{|\alpha|\leq 1} f_i(x_i^0 + \alpha\sigma_{x,i}).$$

Our result is expressed for general sets $\mathcal{U}$, based on the support function $\sigma_{\mathcal{U}}$ with values for $b \in \mathbb{R}^p$ given by

$$\sigma_{\mathcal{U}}(b) := \max_{u\in\mathcal{U}} b^\top u. \tag{4.10}$$

Note that this function depends only on the convex hull of the set $\mathcal{U}$. In the case of the box set $\mathcal{U}^{\text{box}}$ given in (4.1), there is a convenient closed-form expression:

$$\sigma_{\mathcal{U}^{\text{box}}}(b) = b^\top u^0 + \sigma_u^\top |b|.$$

Likewise for the cardinality-constrained set $\mathcal{U}^{\text{card}}$ defined in (4.2), we have

$$s_{\mathcal{U}_k}(b) = \max_{u\in\mathcal{U}^{\text{card}}} b^\top u = b^\top u^0 + s_k(\sigma_u \odot |b|),$$

where s_k is the sum of the top k entries of its vector argument—a convex function.

The only coupling constraint in (4.9) is the affine equation, which suggests the following relaxation.

Theorem 4.3 (LP bound on the state). *An upper bound on the objective of problem (4.9) is given by*

$$p^* \leq \overline{p} := \min_{\lambda} \sigma_{\mathcal{U}}(B^\top\lambda) + \sum_{i\in[n]} g_i(\lambda_i, (A^\top\lambda)_i),$$

where $s_{\mathcal{U}}$ is the support function defined in (4.10), and g_i, $i \in [n]$, are the convex functions

$$g_i \ : \ (\alpha,\beta) \in \mathbb{R}^2 \to g_i(\alpha,\beta) := \max_{\zeta \,:\, |\zeta_+ - x_i^0| \leq \sigma_{x,i}} f_i(\zeta_+) - \alpha\zeta + \beta\zeta_+, \quad i \in [n].$$

If the functions g_i, $i \in [n]$, are closed, we have the bidual expression

$$\overline{p} := \max_{x,\, u\in\mathcal{U}} \ -\sum_{i\in[n]} g_i^*(-(Ax+Bu)_i, x_i),$$

where g_i^* is the conjugate of g_i, $i \in [n]$.

Proof. We have

$$\begin{aligned} p^* \leq \overline{p} &:= \min_{\lambda} \max_{x,\, u\in\mathcal{U}} \sum_{i\in[n]} f_i(x_i) + \lambda^\top(Ax+Bu-z) \ : \ x = z_+, \ \ |x - x^0| \leq \sigma_x \\ &= \min_{\lambda} \max_{u\in\mathcal{U}} \lambda^\top Bu + \max_{z \,:\, |z^+ - x^0| \leq \sigma_x} \sum_{i\in[n]} (f_i(z_i^+) + (A^\top\lambda)_i z_i^+ - \lambda_i z_i) \\ &= \min_{\lambda,\, \mu = A^\top\lambda} \left(\max_{u\in\mathcal{U}} \lambda^\top Bu\right) + \sum_{i\in[n]} g_i(\lambda_i, \mu_i), \end{aligned}$$

which establishes the first part of the theorem. If we further assume that the functions g_i, $i \in [n]$, are closed, then strong duality holds, so that

$$\overline{p} = \min_{\lambda,\,\mu} \max_{x,\,u \in \mathcal{U}} \lambda^\top B u + x^\top (A^\top \lambda - \mu) + \sum_{i \in [n]} g_i(\lambda_i, \mu_i) = \max_{x,\,u \in \mathbf{Co}\mathcal{U}} - \sum_{i \in [n]} g_i^*(-(Ax + Bu)_i, x_i),$$

where g_i^* is the conjugate of g_i, $i \in [n]$. ∎

In the case when $f_i(\xi) = c_i \xi$, $i \in [n]$, where $c \in \mathbb{R}^n$ is given, it turns out that our relaxation, when expressed in bidual form, has a natural look:

$$p^* \leq \overline{p} = \max_{x,\,u \in \mathcal{U}} c^\top x \;:\; x \geq Ax + Bu, \;\; x \geq 0, \;\; |x - x^0| \leq \sigma_x.$$

When the cardinality of changes in the input is constrained in the set $\mathcal{U}^{\mathrm{card}}$, the bound takes the form

$$\begin{aligned} p^* \leq \overline{p} = \max_{x,\,u} c^\top x \;:\; & x \geq Ax + Bu, \;\; x \geq 0, \;\; |x - x^0| \leq \sigma_x, \\ & \|\,\mathbf{diag}(\sigma_u)^{-1}(u - u^0)\|_1 \leq k, \;\; |u - u^0| \leq \sigma_u. \end{aligned}$$

5. Sparsity and model compression. In this section, we examine the role of sparsity and low-rank structure in implicit deep learning, specifically in the model parameter matrix

$$M := \begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

We assume that the activation map ϕ is a CONE map; most of our results can be generalized to BLIP maps.

Since a CONE map acts componentwise, the prediction rule (1.1a) is invariant under permutations of the state vector, in the sense that, for any $n \times n$ permutation matrix, the matrix $\mathbf{diag}(P, I) M \,\mathbf{diag}(P^\top, I)$ represents the same prediction rule as M given above.

Various kinds of sparsity of M can be encouraged in the training problem, with appropriate penalties. For example, we can use penalties that encourage many elements in M to be zero; the advantage of such "elementwise" sparsity is, of course, computational, since sparsity in matrices A, B, C, D will allow for computational speedups at test time. Another interesting kind of sparsity is rank sparsity, which refers to the case when model matrices are low-rank.

Next, we examine the benefits of row (or column) sparsity, which refers to the fact that entire rows (or columns) of a matrix are zero. Note that column sparsity in a matrix N can be encouraged with a penalty in the training problem, of the form

$$\mathcal{P}(N) = \sum_i \|N e_i\|_\alpha,$$

where $\alpha > 1$. Row sparsity can be handled via $\mathcal{P}(N^\top)$.

5.1. Feature selection. We may use the implicit model to select features. Any zero column in the matrix $(B^\top, D^\top)^\top$ means that the corresponding element in an input vector does not play any role in the prediction rule. We may thus use a column-norm penalty in the training problem, in order to encourage such a sparsity pattern:

$$\mathcal{P}(B,D) = \sum_{j=1}^{p} \left\| \begin{pmatrix} B \\ D \end{pmatrix} e_j \right\|_\alpha, \tag{5.1}$$

with $\alpha > 1$.

5.2. Dimension reduction via row and column sparsity. Assume that the matrix A is row-sparse. Without loss of generality, using permutation invariance, we can assume that M writes

$$M = \begin{pmatrix} A_{11} & A_{12} & B_1 \\ 0 & 0 & B_2 \\ C_1 & C_2 & D \end{pmatrix},$$

where A_{11} is a square matrix of order $n_1 < n$. We can then decompose x accordingly, as $x = (x_1, x_2)$ with $x_1 \in \mathbb{R}^{n_1}$, and the above implies $x_2 = \phi(B_2 u)$. The prediction rule for an input $u \in \mathbb{R}^p$ then writes

$$\hat{y}(u) = C_1 x_1 + Du, \quad x_1 = \phi(A_{11} x_1 + A_{12}\phi(B_2 u) + B_1 u).$$

The rule only involves x_1 as a true hidden feature vector. In fact, the row sparsity of A allows for a computational speedup, as we simply need to solve a fixed-point equation for the vector with reduced dimensions, x_1.

Further assume that (A, B) is row-sparse. Again without loss of generality we may put M in the above form, with $B_2 = 0$. Then the prediction rule can be written

$$\hat{y}(u) = C_1 x_1 + Du, \quad x_1 = \phi(A_{11} x_1 + B_1 u).$$

This means that the dimension of the state variable can be fully reduced, to $n_1 < n$. Thus, row sparsity of (A, B) allows for a reduction in the dimension of the prediction rule.

When (A, B) is column-sparse, we obtain similar speedups: We only need to solve for x_1, and x_2 can be directly expressed as a closed-form function of x_1. We leave the details to the reader.

Now assume that $(A^\top, C^\top)^\top$ is column-sparse. Again, the prediction rule does not need x_2 at all, so that the computation of the latter vector can be entirely avoided. This means that the dimension of the state variable can be fully reduced, to $n_1 < n$. Thus, column sparsity of $(A^\top, C^\top)^\top$ allows for a reduction in the dimension of the prediction rule.

To summarize, row or column sparsity of A allows for a computational speedup; if the corresponding rows of B (resp., columns of C) are zero, then the prediction rule involves only a vector of reduced dimensions.

5.3. Rank sparsity. Assume that the matrix A is rank $k \ll n$, and that a corresponding factorization is known: $A = LR^\top$, with $L, R \in \mathbb{R}^{n\times k}$. In this case, for any p-vector u, the equilibrium equation $x = \phi(Ax + Bu)$ can be written as $x = \phi(Lz + Bu)$, where $z := R^\top x$.

Hence, we can obtain a prediction for a given input u via the solution of a low-dimensional fixed-point equation in $z \in \mathbb{R}^k$:

$$z = R^\top \phi(Lz + Bu).$$

It can be shown that when ϕ is a CONE map, the above rule is well-posed if $\lambda_{\mathrm{PF}}(|R|^T|L|) < 1$. Once a solution z is found, we simply set the prediction to be $\hat{y}(u) = C\phi(Lz + Bu) + Du$.

At test time, if we use fixed-point iterations to obtain our predictions, then the computational savings brought about by the low-rank representation of A can be substantial, with a per-iteration cost going from $O(n^2)$ to $O(kn)$ if we use the above equation.

5.4. Model error analysis. The above analysis suggests replacing a given model matrix A with a low-rank or sparse approximation, denoted A^0. The resulting state error can then be bounded as follows. Assume that $|A - A^0| \le E$, where $E \ge 0$ is a known upper bound on the componentwise error in A. The following theorem provides *relative* error bounds on the state, provided the perturbed system satisfies the well-posedness condition $\lambda_{\mathrm{PF}}(|A|) < 1$. As before, we denote by x^0, x the (unique) solutions to the unperturbed and perturbed equilibrium equations $\xi = \phi(A^0\xi + Bu)$ and $\xi = \phi(A\xi + Bu)$, respectively.

Theorem 5.1 (Effect of errors in A). *Assume that ϕ is a CONE map, and that $\lambda_{\mathrm{PF}}(|A^0 + E|) < 1$. Then*

$$|x - x^0| \le (I - (|A^0 + E|))^{-1} E x_0. \tag{5.2}$$

Proof. We have

$$|x - x^0| \le |Ax - Ax^0| = |A^0(x - x^0) + E(x - x^0) + Ex^0| \le |A^0 + E||x - x^0| + |E||x^0|.$$

Applying a technique similar to that employed in the proof of Theorem 4.1, we obtain the desired relative error bounds. ∎

6. Training implicit models.

6.1. Setup. We are now given an input data matrix $U = [u_1, \dots, u_m] \in \mathbb{R}^{p\times m}$ and response matrix $Y = [y_1, \dots, y_m] \in \mathbb{R}^{q\times m}$, and seek to fit a model of the form (1.1a), with A well-posed with respect to ϕ, which we assume to be a BLIP map. We note that the rule (1.1a), when applied to a collection of inputs $(u_i)_{1\le i\le m}$, can be written in matrix form as

$$\hat{Y}(U) = CX + DU, \text{ where } X = \phi(AX + BU),$$

where $U = [u_1, \dots, u_m] \in \mathbb{R}^{p\times m}$, and $\hat{Y}(U) = [\hat{y}(u_1), \dots, \hat{y}(u_m)] \in \mathbb{R}^{q\times m}$.

We consider a training problem of the form

$$\min_{A,B,C,D,X} \mathcal{L}(Y, CX + DU) + \mathcal{P}(A, B, C, D) \ : \ X = \phi(AX + BU), \ \ A \in \mathrm{WP}(\phi). \tag{6.1}$$

In the above, $\mathcal{L}$ is a loss function, assumed to be convex in its second argument, and $\mathcal{P}$ is a convex penalty function, which can be used to enforce a given (linear) structure (such as A strictly upper block triangular) on the parameters and/or encourage their sparsity.

In practice, we replace the well-posedness condition by the sufficient PF condition of Theorem 2.2. As argued in subsection 2.5, the latter can be further replaced without loss of

generality with an easier-to-handle (convex) norm constraint; in the case of CONE maps, this condition is $\|A\|_\infty \leq \kappa$, where $\kappa \in (0,1)$ is a hyperparameter. In the more general case of BLIP maps, we can use a similar norm constraint $\|N(A,\gamma)\|_\infty \leq \kappa$, where $N(A,\gamma)$ is defined in (2.4).

Examples of loss functions. For regression tasks, we may use the squared Euclidean loss: for $Y, \hat{Y} \in \mathbb{R}^{q\times m}$,

$$\mathcal{L}(Y,\hat{Y}) := \frac{1}{2}\|Y-\hat{Y}\|_F^2.$$

For multiclass classification, a popular loss is a combination of negative cross-entropy with the softmax: for two q-vectors $y, \hat{y}$, with $y \geq 0$, $y^\top \mathbf{1} = 1$, we define

$$\mathcal{L}(y,\hat{y}) = -y^\top \log\left(\frac{e^{\hat{y}}}{\sum_{i=1}^q e^{\hat{y}_i}}\right) = \log\left(\sum_{i=1}^q e^{\hat{y}_i}\right) - y^\top \hat{y}.$$

We can extend the definition to matrices by summing the contribution of all columns, each corresponding to a data point: for $Y, Z \in \mathbb{R}^{q\times m}$,

$$\mathcal{L}(Y,\hat{Y}) = \sum_{j=1}^m \log\left(\sum_{i=1}^q e^{\hat{Y}_{ij}}\right) - \sum_{j=1}^m \sum_{i=1}^q Y_{ij}\hat{Y}_{ij} = \log(\mathbf{1}^\top \exp(\hat{Y}))\mathbf{1} - \mathbf{Tr}\, Y^\top \hat{Y}, \tag{6.2}$$

where both the log and the exponential functions apply componentwise.

Examples of penalty functions. Beyond well-posedness, the penalty can be used to encourage desired properties of the model. For robustness, the convex penalty (4.7) can be used. We may also use an l_∞-norm penalty on the sensitivity matrices S appearing in Theorem 4.2. For feature selection, an appropriate penalty may involve the block norms (5.1); sparsity of the model matrices can be similarly handled with ordinary l_1-norm penalties on the elements of the model matrices (A,B,C,D).

6.2. Gradient methods. Assuming that ϕ is a CONE map for simplicity, we consider the problem

$$\min_{A,B,C,D,X} \mathcal{L}(Y, CX+DU) + \mathcal{P}(A,B,C,D) \;:\; X = \phi(AX+BU), \;\; \|A\|_\infty \leq \kappa, \tag{6.3}$$

where $\kappa < 1$ is given. We assume that the map ϕ is differentiable. We can solve (6.3) using (stochastic) projected gradient descent, by differentiating through the equilibrium equation. It turns out that this differentiation requires solving an equilibrium equation involving a matrix variable, which, thanks to well-posedness, can be very efficiently solved via fixed-point iterations.

Computing gradients. Considering a mini-batch of size 1 first, we define $\hat{y} = Cx + Du$, $z = Ax + Bu$. We have

$$\begin{pmatrix} \nabla_A \mathcal{L} & \nabla_B \mathcal{L} \\ \nabla_C \mathcal{L} & \nabla_D \mathcal{L} \end{pmatrix} = \begin{pmatrix} \nabla_z \mathcal{L} \\ \nabla_{\hat{y}} \mathcal{L} \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix}^\top,$$

where $\nabla_{\hat{y}}\mathcal{L}$ is easy to compute, and $\nabla_z\mathcal{L}$ is obtained via implicit differentiation:

$$\nabla_z\mathcal{L} = \left(\frac{\partial\mathcal{L}}{\partial x}\cdot\frac{\partial x}{\partial z}\right)^\top, \quad \frac{\partial\mathcal{L}}{\partial x} = \frac{\partial\mathcal{L}}{\partial\hat{y}}\cdot\frac{\partial(Cx+Du)}{\partial x},$$
$$\frac{\partial x}{\partial z} = \frac{\partial\phi(z)}{\partial z} + \frac{\partial\phi(Ax+Bu)}{\partial x}\cdot\frac{\partial x}{\partial z} = (I-\Phi A)^{-1}\Phi,$$

where $\Phi := \frac{\partial\phi(z)}{\partial z}$ is a diagonal matrix. Since ϕ is a CONE map, we have $\|\Phi\|_\infty \le 1$; since the current matrix A satisfies the norm condition $\|A\|_\infty < 1$, the inverse of the matrix $(I-\Phi A)$ exists. The gradient of the loss function $\nabla_{\hat{y}}\mathcal{L}$ can be easily computed, and we have

$$\nabla_z\mathcal{L} = \left(C(I-\Phi A)^{-1}\Phi\right)^\top \nabla_{\hat{y}}\mathcal{L}. \tag{6.4}$$

Thanks to well-posedness, the gradient $\nabla_z\mathcal{L}$ is the unique solution to the following equilibrium equation in vector v:

$$v = \Phi\left(A^\top v + C^\top\nabla_{\hat{y}}\mathcal{L}\right). \tag{6.5}$$

Turning to the case of a mini-batch of, say, size b, the main effort in computing the gradient consists in solving matrix equations in an $n\times b$ matrix V:

$$V = \Psi\circ\left(A^\top V + C^\top G\right),$$

where the columns of G contain the gradients of the loss with respect to $\hat{y}$, and Ψ is a matrix whose columns contain the derivatives of the activation map, both evaluated at a specific training point. Due to the fact that A satisfies the PF sufficient condition for well-posedness with respect to ϕ, the equation above has a unique solution; the matrix V can be computed as the limit point of the convergent fixed-point iterations

$$V(t+1) = \Psi\circ\left(A^\top V(t) + C^\top L\right), \quad t = 0,1,2,\ldots. \tag{6.6}$$

Projection step. In order to handle the well-posedness constraint $\|A\|_\infty \le \kappa$, the projected gradient method requires a projection at each step. This step corresponds to a subproblem of the form

$$\min_A \ \|A-A^0\|_F \ : \ \|A\|_\infty \le \kappa, \tag{6.7}$$

with matrix A^0 given. The above problem is decomposable across rows, leading to n subproblems of the form

$$\min_a \ \frac{1}{2}\|a-a_i^0\|_2^2 \ : \ \|a\|_1 \le \kappa,$$

where $a_i^0 \in \mathbb{R}^n$ is the ith row of A^0. The problem cannot be solved in closed form, but a bisection method can be applied to the dual:

$$p^* = \max_{\lambda\ge 0} \ -\kappa\lambda + \sum_{i\in[n]} s_i(\lambda),$$

where, for $\lambda \geq 0$ given,

$$s_i(\lambda) := \min_{\xi} \frac{1}{2}(\xi - a_i^0)^2 + \lambda|\xi|, \quad i \in [n].$$

A subgradient of the objective is

$$g_i(\lambda) := -\kappa + \sum_{i\in[n]} \max(|a_i^0| - \lambda, 0), \quad i \in [n].$$

Observe that $p^* \geq 0$; hence at optimum,

$$0 \leq \lambda \leq \frac{1}{\kappa} \sum_{i\in[n]} s(\lambda, a_i^0) \leq \lambda^{\max} := \frac{1}{2\kappa}\|a_i^0\|_2^2.$$

The bisection can be initialized with the interval $\lambda \in [0, \lambda^{\max}]$.

Returning to the original problem (6.7), we see that all the iterations can be expressed in a "vectorized" form, where updates for the different rows of A are done in parallel. The dual variables corresponding to each row are collected in a vector $\lambda \in \mathbb{R}^n$. We initialize the bisection with a vector interval $[\lambda_l, \lambda_u]$, with $\lambda^l = 0$, $\lambda_i^u = \frac{1}{2}\|a_i^0\|_2^2/\kappa$, $i \in [n]$. We update the current vector interval as follows:

1. Set $\lambda = (\lambda_l + \lambda_u)/2$.
2. Form a vector $g(\lambda)$ containing the subgradients corresponding to each row, evaluated at λ_i, $i \in [n]$:
$$g(\lambda) = -\kappa \mathbf{1} + (|A^0| - \lambda \mathbf{1}^T)_+^T \mathbf{1}.$$
3. For every $i \in [n]$, reset $\lambda_i^u = \lambda_i$ if $g_i(\lambda) > 0$, and $\lambda_i^l = \lambda_i$ if $g_i(\lambda) \leq 0$.

6.3. Block-coordinate descent methods. Block-coordinate descent methods use cyclic updates, optimizing one matrix variable at a time, fixing all the other variables. Such methods are easier to apply to a so-called Fenchel formulation of the problem, which is equivalent to problem (6.1):

$$\min_{A,B,C,D,X} \mathcal{L}(Y, CX + DU) + \mathcal{P}(A, B, C, D) \ : \ F_\phi(X, AX + BU) \leq 0, \quad \|N(A, \gamma)\|_\infty \leq \kappa,$$

where F_ϕ is the so-called Fenchel divergence adapted to ϕ [21], applied columnwise to matrix inputs. In the case of the ReLU activation, for two given vectors x, z of the same size, we have

$$F_\phi(x, z) := \frac{1}{2}x \odot x + \frac{1}{2}z_+ \odot z_+ - x \odot z \text{ if } x \geq 0, \tag{6.8}$$

with $\odot$ the componentwise multiplication. We can then define $F_\phi(X, Z)$, with X, Z two matrix inputs having the same number of columns, by summing over these columns. As seen in [21], a large number of popular activation maps can be expressed similarly.

The BCD methods are particularly interesting when the updates require solving convex problems. For instance, considering the training problem (6.3), given X, the problem is convex in the model matrices A, B, C, D. Then, given the model matrices, finding X consists in a feasibility problem that can be solved with fixed-point iterations.

We may also consider a relaxed form of the problem:

$$\min_{A,B,C,D,X} \mathcal{L}(Y, CX + DU) + \mathcal{P}(A, B, C, D) + \lambda^\top F_\phi(X, AX + BU), \quad \|N(A, \gamma)\|_\infty \leq \kappa,$$

where $\lambda > 0$ is an n-vector parameter. Here, all the updates involve solving convex problems, as shown in [21], since the Fenchel divergence, for most of activation maps, is biconvex in its two arguments—meaning that given the first argument fixed, it is convex in the second, and vice versa. We refer to [21, 47] for more on Fenchel divergences in the context of implicit deep learning and neural networks, and illustrate and give more detailed examples of such methods in the next section.

7. Numerical experiments.

7.1. Learning real nonlinear functions via regression. We start by illustrating the ability of the gradient method, as presented in subsection 6.2, to learn the parameters of the implicit model, with well-posedness enforced via a max-row-sum norm constraint, $\|A\|_\infty \leq 0.5$. We aim at learning a real function; as an example we focus on

$$f(u) = 5\cos(\pi u)\exp\Big(-\frac{1}{2}|u|\Big).$$

We select the input u_i, $i \in \{1, \ldots, m\}$, uniformly at random between -5 and 5 with $m = 200$; we add a random noise to the output, $y(u) = f(u) + w$ with w taken uniformly at random between -1 and 1, and hence the standard deviation for $y(u)$ is $1/\sqrt{(3)} \simeq 0.57$. We consider an implicit model of order $n = 75$. We learn the parameters of the model by doing only two successive block updates: first, we update (A, B) using stochastic projected gradient descent, the gradient being obtained with the implicit chain rule described in subsection 6.2. The root-mean-square error (RMSE) across iterations for this block update is shown in Figure 3. After this first update we achieve an RMSE of 1.77. Second, we update (C, D) using linear regression. After this update we achieve an RMSE of 0.56. For comparison purposes, we also train a neural network with three hidden layers of width $n/3 = 25$ using ADAM, mini-batches, and a tuned learning rate. We run ADAM until convergence. We get an RMSE $= 0.65$, which is slightly above that of our implicit model. This first simple experiment shows the ability to fit nonlinear functions with implicit models as illustrated in Figure 4.

7.2. Comparison with neural networks. In this section we compare the performance of implicit models with that of neural networks. Experiments on both synthetic datasets and real datasets are conducted. The experiment results show that implicit models have the potential of matching or exceeding the performance of neural networks in various settings. To simplify the experiments, we do not apply any specific network structure or regularization during training. When it comes to training neural networks we will always use ADAM with a grid-search tuned step-size. Similarly, we will always use stochastic projected gradient descent as detailed in subsection 6.2. The choice of the number of hidden features n for implicit models is always aligned with the neural network architecture for fair comparison as explained in subsection 3.2.

7.2.1. Synthetic datasets. We consider two synthetic classification datasets: one generated from a neural network and another from an implicit model. For each dataset, we then

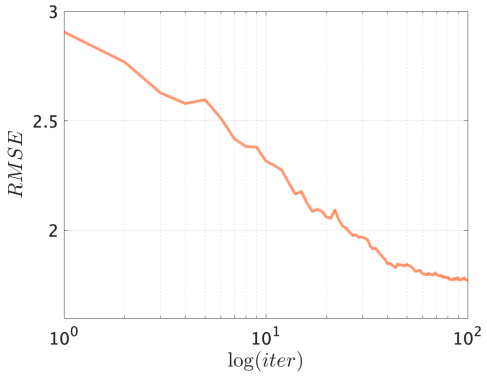

Figure 3. *RMSE across projected gradient iterations for the (A, B) block update.*

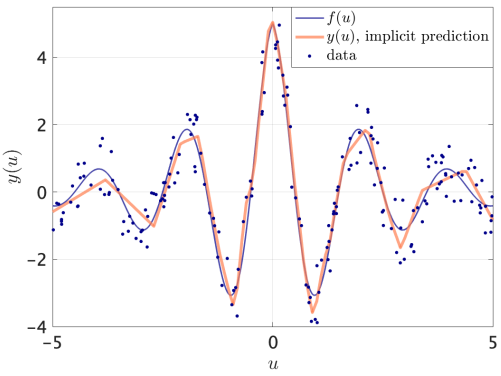

Figure 4. *Implicit prediction $y(u)$ comparison with $f(u)$.*

aim at fitting both an implicit model and a neural network. Each data point in the datasets contains an input $u \in \mathbb{R}^5$ and output $y \in \mathbb{R}^2$. The model architectures and data generation details are deferred to the supplementary materials (idl_supplement.pdf [local/web 584KB]).

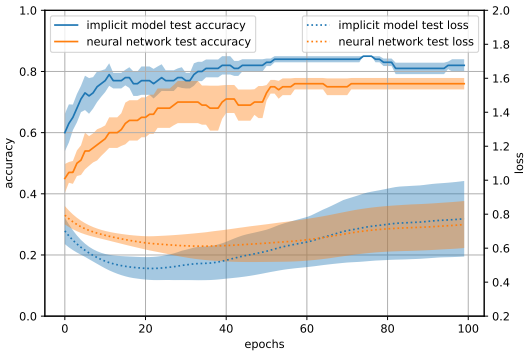

Figure 5. *Performance comparison on a synthetic dataset generated from a neural network. Average best accuracy, implicit:* 0.85*; neural network:* 0.76*. The curves are generated from* 5 *different runs with the lines marked as mean, and regions marked as the standard deviation over the runs.*

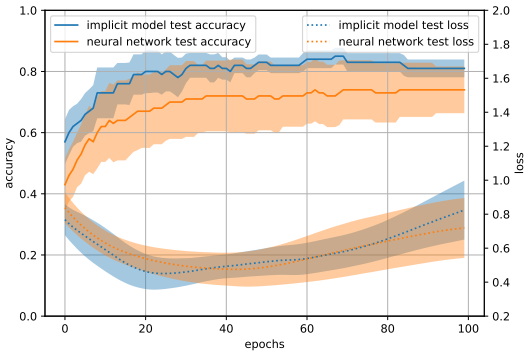

Figure 6. *Performance comparison on a synthetic dataset generated from an implicit model. Average best accuracy, implicit:* 0.85*; neural networks:* 0.74*. The curves are generated from* 5 *different runs with the lines marked as mean, and regions marked as the standard deviation over the runs.*

We find that the implicit model outperforms neural networks in both synthetic experiments. See Figure 5 and Figure 6 for details. This may be explained by the increased modeling capacity of the implicit model, given similar parameter size, with respect to its neural network counterpart as mentioned in subsection 1.1.

7.2.2. Real-world datasets. We continue to compare the performance of the implicit model with that of neural networks in real-world settings. For this purpose, we pick two real-world datasets, the hand-written digit classification dataset MNIST and the German Traffic Sign Recognition Benchmark (GTSRB) dataset. The performance is given in Figure 7 and

Figure 8. Similar to what we observed with synthetic datasets, the implicit model is capable of matching and even outperforming classical neural networks.

Figure 7. *Performance comparison on MNIST. Average best accuracy, implicit:* 0.976*; neural networks:* 0.972*. The curves are generated from* 5 *different runs with the lines marked as mean, and regions marked as the standard deviation over the runs.*

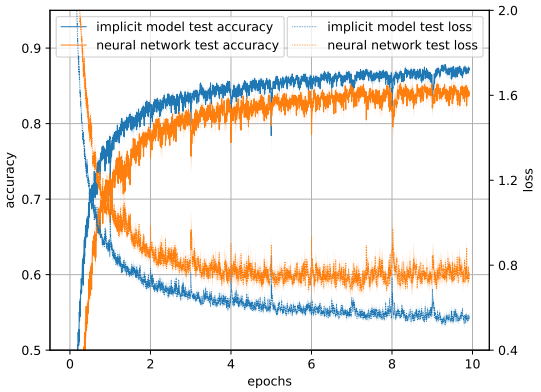

Figure 8. *Performance comparison on GTSRB. Average best accuracy, implicit:* 0.874*; neural networks:* 0.859*. The curves are generated from* 5 *different runs with the lines marked as mean, and regions marked as the standard deviation over the runs.*

7.3. Adversarial attack.

7.3.1. Attack via the sensitivity matrix. Our analysis above highlights the use of the *sensitivity matrix* as a measure for robustness. In this section, we show how the sensitivity matrix can be used to generate effective attacks on two public datasets, MNIST and CIFAR-10. Examples of the sensitivity matrix are shown in the supplementary material (idl_supplement.pdf [local/web 584KB]). We compare our method against commonly used gradient-based attacks [19, 38]. In this experiment, we consider two models: (1) feedforward network trained on the MNIST dataset (98% clean accuracy), and (2) ResNet-20 [25] trained on the CIFAR-10 dataset (92% clean accuracy).

We compare our method with commonly used gradient-based attacks. Precisely, for a given function F (prediction rule) learned by a deep neural network, a benign sample $u \in \mathbb{R}^p$, and the target y associated with u, we compute the gradient of the function F with respect to the given sample u, $\nabla_u F(u, y)$. We then take the absolute value of the gradient as an indication of which input features an adversary should perturb, similar to the saliency map technique [44, 38]. The absolute value of the gradient can be seen as a "local" version of the sensitivity matrix; however, unlike the gradient, the sensitivity matrix does not depend on the input data, making it a more general measurement of robustness for any given model.

Table 1 presents the experimental results of an adversarial attack using the sensitivity matrix and the absolute value of the gradient on MNIST and CIFAR-10. For the sensitivity matrix attack, we start from perturbing the input features that have the highest values according to the sensitivity matrix. For the gradient-based attack, we do the same according to the absolute value of the gradient. We perturb the input features into small random values.

Our experiments show that the sensitivity matrix attack is as effective as the gradient-based attack, while being very simple to implement.

Interestingly, our attack does not rely on any input samples that the gradient-based attack needs. An adversary with the model parameters could easily craft adversarial samples using the sensitivity matrix. In the absence of access to the model parameters, an adversary can rely on the principle of *tranferability* [33] and train a surrogate model to obtain the sensitivity matrix. Figure 9 displays adversarial images generated using the sensitivity matrix. An interesting case is to use the sensitivity matrix to generate a sparse attack as seen in Figure 9.

Table 1

Experimental results of attack success rate against percentage of perturbed inputs on MNIST and CIFAR-10 (10000 samples from test set).

	Sensitivity matrix attack		Gradient-based attack	
% of perturbed inputs	MNIST	CIFAR-10	MNIST	CIFAR-10
0.1%	1.01%	3.04%	2.42%	1.75%
1%	13.41%	10.16%	26.92%	6.66%
10%	70.67%	36.21%	74.90%	33.18%
20%	89.82%	57.01%	87.10%	52.57%
30%	90.22%	67.45%	89.82%	66.59%

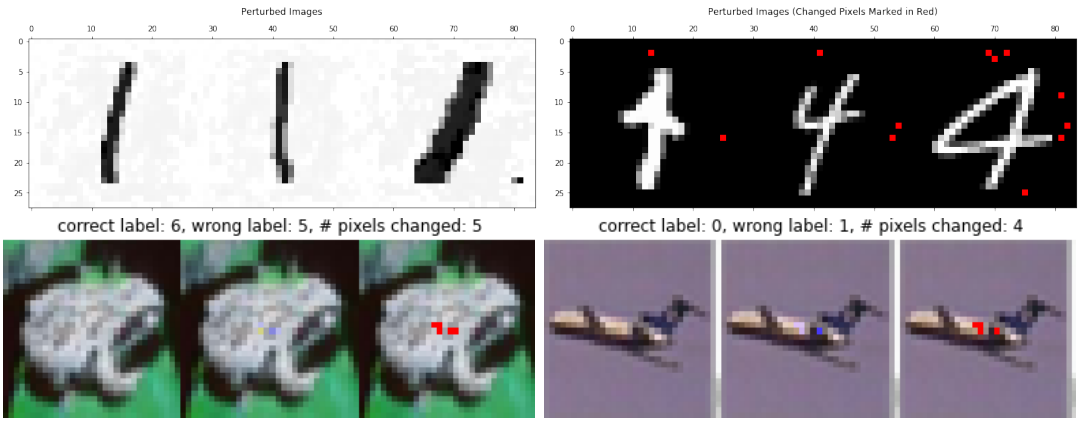

Figure 9. *Top: Adversarial samples from MNIST. On the left are dense attacks with small perturbations, and on the right are sparse attacks with random perturbations (perturbed pixels are marked in red). Bottom: Example sparse attack on CIFAR-10. The left images are cleaned images, the middle ones are perturbed images, and the right ones mark the perturbed pixels in red for higher visibility.*

7.3.2. Attack with LP relaxation for CONE maps. Although one can use the sensitivity matrix to generate an effective adversarial example, one may wish to perform a more sophisticated attack by exploiting the weakness of an individual data point. This can be done by considering the LP relaxation (Theorem 4.3), which has the advantage of generating a specific adversarial example for a given input data. The experiment in this section is again on MNIST and CIFAR-10 images. Here, the problem outlined in (4.9) is solved by LP relaxation, with

the function $f_i(\xi) = (\xi - x_i^0)^2$. The optimization problem then finds a perturbed image that leads to the largest discrepancy between the perturbed state x and the nominal state x^0. Figure 10 shows five example images. The perturbed images generated by the LP relaxation appear quite similar to the original images; however, the model fails to predict these otherwise correctly predicted images.

Our framework also allows for sparse adversarial attack by adding a cardinality constraint. Figure 11 shows three examples of perturbed images under nonsparse and sparse attack. Images on the left are the results of nonsparse attack, and those on the right are the results of sparse attack. The model fails to predict the label correctly under both conditions. These results illustrate how the implicit prediction rules can be used to generate powerful adversarial attacks. It is also useful for adversarial training as a large number of adversarial examples can be generated using the technique and added back to the training data.

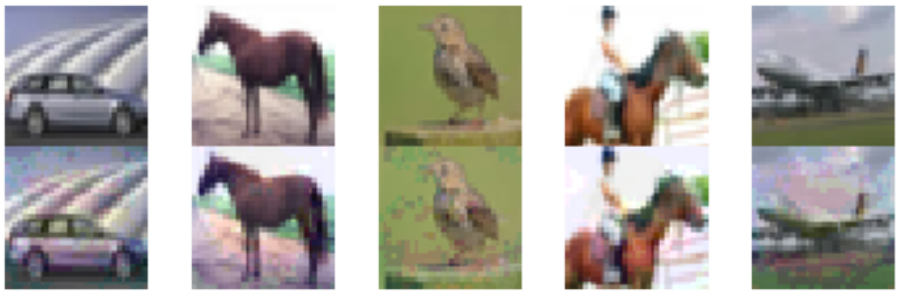

Figure 10. *Example attack on CIFAR dataset. Top: Clean data. Bottom: Perturbed data.*

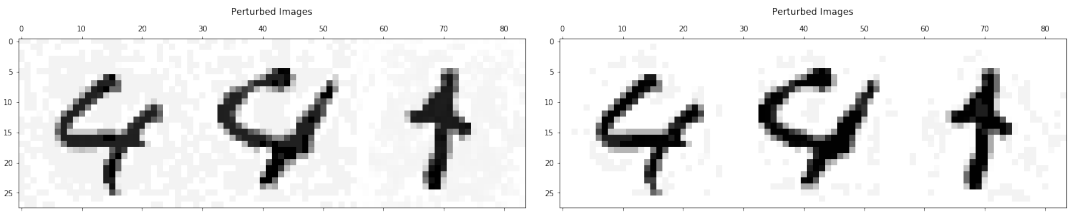

Figure 11. *Example attack on MNIST dataset. Left: Nonsparse attack. Right: Sparse attack.*

8. Prior work.

8.1. In implicit deep learning. Recent works have considered versions of implicit models and demonstrated their potential in deep learning. In the pioneering work by Bai and collaborators [6, 27, 7, 22] the authors demonstrated empirical success of an entirely implicit framework, which they call Deep Equilibrium Models. They present a general form of implicit model based on an implicit equation of the form

$$z_{1:T}^{[i+1]} = f_\theta(z_{1:T}^{[i]}; u_{1:T}), \quad z_{1:T}^{[0]} = 0,$$

where i is the layer index; $z_{1:T}^{[i]}$ is the hidden sequence of length T at layer i; $u_{1:T} = [u_1, \ldots, u_T] \in \mathbb{R}^{T \times p}$ is the input sequence, where $u_i \in \mathbb{R}^p$ and $T \in \mathbb{N}$; and f_θ is some nonlinear

transformation. This formulation represents the class of *weight-tied* sequence models, where the same transformation f_θ is used for all layers, reminiscent of recurrent neural networks. The authors show that any deep network can be represented by this weight-tied representation, akin to the reformulation in subsection 3.2.

The models are then trained using quasi-Newton methods, and gradients are computed using the implicit function theorem. The main difference with our approach lies in the fact that the above-mentioned work focuses on obtaining empirical results in the context of natural language processing and computer vision, whereas our work focuses on theoretical foundations such as well-posedness and robustness.

In the more recent work [49], the authors use the same structure as ours (1.1b) and do provide results pertaining to well-posedness. The difference with our approach there lies in the assumptions made on the activation function ϕ. Instead of the BLIP (blockwise Lipschitz) assumption, the authors propose that the activation should be a proximal operator for some convex function g, of the form

$$\phi(x) = \arg\min_z \frac{1}{2}\|x - z\|^2 + g(z).$$

Note that the ReLU activation can be represented as a proximal operator with $g(z) = I(z \geq 0)$, where I is the indicator function. The authors then observe that under this assumption on ϕ, a condition for well-posedness is

$$(1 - m)I \succeq \frac{A + A^\top}{2} \tag{8.1}$$

with $m > 0$. This condition is different from ours, but the two are not equivalent. As an example, we can choose ϕ to be the ReLU and $A = -2I$. A does not satisfy our condition of well-posedness, since $\lambda_{\mathrm{PF}}(A) = 2$, but it does satisfy (8.1) for $m = 1$. Conversely, for the choice

$$A = \begin{bmatrix} 0.5 & 0 \\ 2 & -0.5 \end{bmatrix}$$

we have $\lambda_{\mathrm{PF}}(A) = 0.5 < 1$, but the eigenvalues of $\frac{1}{2}(A + A^\top)$ are $\{\pm\frac{\sqrt{5}}{2}\}$; therefore (8.1) is not satisfied. The authors then show how to compute a solution to the equilibrium equation using splitting techniques for monotone operators, mainly the forward-backward and Peaceman–Rachford algorithms, which serve the same purpose as the Picard iterations we propose. Finally, the authors also use a form of implicit differentiation using these algorithms to learn the parameters of the model.

Prior to the implicit frameworks, some authors have used implicit types of methods in model design. The paper [11] uses implicit methods to solve and construct a general class of models known as neural ordinary differential equations, while [14] uses implicit models to construct a differentiable physics engine that enables gradient-based learning and high sample efficiency. Furthermore, many papers explore the concept of integrating implicit models with modern deep learning methods in a variety of ways. For example, [48] shows promise in integrating logical structures into deep learning by incorporating a semidefinite programming (SDP) layer into a network in order to solve a (relaxed) MAXSAT problem; see also [48]. In [1]

the authors propose including a model predictive control as a differentiable policy class for deep reinforcement learning, which can be seen as a kind of implicit architecture. In [2] the authors introduce implicit layers where the activation is the solution of some quadratic programming problem, and in [15] the authors incorporate a stochastic optimization formulation for an end-to-end learning task, in which the model is trained by differentiating the solution of a stochastic programming problem.

8.2. In robust control. Our approach is rooted in the field of robust control analysis and design. The idea of analyzing (linear) dynamical systems subject to uncertainty via optimization-based approaches has a long history; most relevant to our approach are the landmark references [13, 26], which delineate an approach, based on linear programming, that focuses on the so-called l_∞-to-l_∞ gain of a dynamical system; it employs a technique that embeds nonlinearities in so-called sector bounds, and uses corresponding relaxations. Our results pertaining to sensitivity matrices directly parallel that kind of analysis. Also relevant is the more recent work [51, 52], which analyzes stability of a system controlled by a neural network, and obtains the region of attraction for such a system using a "state-space" representation for the neural network similar to ours.

8.3. In lifted models. In implicit learning, there is usually no way to express the state variable in closed-form, which makes the task of computing gradients with respect to model parameters challenging. Thus, a natural idea in implicit learning is to keep the state vector as a variable in the training problem, resulting in a higher-dimensional (or "lifted") expression of the training problem. The idea of lifting the dimension of the training problem in (nonimplicit) deep learning by introducing "state" variables has been studied in a variety of works [46, 4, 21, 17, 53, 54, 9, 32]. Lifted models are trained using block coordinate descent methods, the Alternating Direction Method of Multipliers, or iterative, non-gradient-based methods. In this work, we introduce a novel aspect of lifted models, namely the possibility of defining a prediction rule implicitly.

8.4. In robustness analysis of neural networks. The issue of robustness in deep learning is generating quite a bit of attention, due to the fact that many deep learning models suffer from the lack of robustness. Prior relevant works have demonstrated that deep learning models are vulnerable to adversarial attacks [19, 28, 38, 29]. The work [40] explores SDP relaxations to the attack problem. The vulnerability issue of deep learning models has motivated the study of corresponding defense strategies [35, 39, 40, 20, 43, 50, 12]. However, many of the defense strategies have since been shown to be ineffective [5, 8], suggesting the need for theoretical understanding of robustness evaluations for deep learning models. In this work, we formalize the robustness analysis of deep learning via the lens of the implicit model. A large number of deep learning architectures can be modeled using implicit prediction rules, making our robustness evaluation a versatile analysis tool.

8.5. In sparsity, compression, and deep feature selection. Sparsity and compression, which are well understood in classical settings, have found their place in deep learning and are an active branch of research. Early work in pruning dates back to as early as the 1990s [31, 24] and has since gained interest. In [45], the authors show that randomly dropping units (i.e., increasing the sparsity level of the network or compressing the network) reduces overfitting

and improves the generalization performance of networks. Recently, more sophisticated ways of pruning networks have been proposed in an effort to reduce the overall size of the model, while retaining or accepting a modest decrease in accuracy: a nonextensive list of works includes [55, 37, 23, 42, 3, 30, 10, 34, 16].

Acknowledgments. The authors would like to thank the anonymous reviewers, as well as Murat Arcak, Zico Kolter, and Mert Pilanci, for their useful comments and suggestions.

REFERENCES

[1] B. Amos, I. D. Jimenez Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter, *Differentiable MPC for end-to-end planning and control*, in Advances in Neural Information Processing Systems, 2018, pp. 8289–8300.

[2] B. Amos and J. Z. Kolter, *OptNet: Differentiable optimization as a layer in neural networks*, in Proceedings of the 34th International Conference on Machine Learning, PMLR 70, JMLR.org, 2017, pp. 136–145.

[3] S. Anwar, K. Hwang, and W. Sung, *Structured pruning of deep convolutional neural networks*, ACM J. Emerging Technologies Comput. Systems (JETC), 13 (2017), 32.

[4] A. Askari, G. Negiar, R. Sambharya, and L. El Ghaoui, *Lifted Neural Networks*, preprint, https://arxiv.org/abs/1805.01532, 2018.

[5] A. Athalye, N. Carlini, and D. A. Wagner, *Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples*, in Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, 2018, Proc. Mach. Learn. Res. 80, PMLR, 2018, pp. 274–283.

[6] S. Bai, J. Z. Kolter, and V. Koltun, *Deep Equilibrium Models*, preprint, https://arxiv.org/abs/1909.01377, 2019.

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

[8] N. Carlini and D. A. Wagner, *Adversarial examples are not easily detected: Bypassing ten detection methods*, in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec'17, Dallas, TX, 2017, ACM, 2017, pp. 3–14.

[9] M. Carreira-Perpinan and W. Wang, *Distributed optimization of deeply nested systems*, in Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, S. Kaski and J. Corander, eds., Reykjavik, Iceland, 2014, Proc. Mach. Learn. Res. 33, PMLR, pp. 10–19, http://proceedings.mlr.press/v33/carreira-perpinan14.html.

[10] S. Changpinyo, M. Sandler, and A. Zhmoginov, *The Power of Sparsity in Convolutional Neural Networks*, preprint, https://arxiv.org/abs/1702.06257, 2017.

[11] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, *Neural ordinary differential equations*, in Advances in Neural Information Processing Systems 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., Curran Associates, Inc., 2018, pp. 6571–6583, http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf.

[12] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, *Certified Adversarial Robustness via Randomized Smoothing*, preprint, https://arxiv.org/abs/arXiv:1902.02918, 2019.

[13] M. A. Dahleh and I. J. Diaz-Bobillo, *Control of Uncertain Systems: A Linear Programming Approach*, Prentice-Hall, Inc., 1994.

[14] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, *End-to-end differentiable physics for learning and control*, in Advances in Neural Information Processing Systems 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., Curran Associates, Inc., 2018, pp. 7178–7189, http://papers.nips.cc/paper/7948-end-to-end-differentiable-physics-for-learning-and-control.pdf.

[15] P. Donti, B. Amos, and J. Z. Kolter, *Task-based end-to-end model learning in stochastic optimization*, in Advances in Neural Information Processing Systems, 2017, pp. 5484–5494.

[16] U. Evci, F. Pedregosa, A. Gomez, and E. Elsen, *The Difficulty of Training Sparse Neural Networks*, preprint, https://arxiv.org/abs/1906.10732, 2019.
[17] V. Ganapathiraman, Z. Shi, X. Zhang, and Y. Yu, *Inductive two-layer modeling with parametric Bregman transfer*, in International Conference on Machine Learning, PMLR, 2018, pp. 1636–1645.
[18] B. Gao and L. Pavel, *On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning*, preprint, https://arxiv.org/abs/1704.00805, 2017.
[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, 2015, Conference Track Proceedings; available online at http://arxiv.org/abs/1412.6572.
[20] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. A. Mann, and P. Kohli, *On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models*, preprint, http://arxiv.org/abs/1810.12715, 2018.
[21] F. Gu, A. Askari, and L. El Ghaoui, *Fenchel lifted networks: A Lagrange relaxation of neural network training*, in International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 3362–3371.
[22] F. Gu, H. Chang, W. Zhu, S. Sojoudi, and L. El Ghaoui, *Implicit graph neural networks*, in Advances in Neural Information Processing Systems 33, 2020.
[23] S. Han, H. Mao, and W. J. Dally, *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*, preprint, https://arxiv.org/abs/1510.00149, 2015.
[24] B. Hassibi, D. G. Stork, and G. J. Wolff, *Optimal Brain Surgeon and general network pruning*, in IEEE International Conference on Neural Networks, IEEE, 1993, pp. 293–299.
[25] 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.
[26] M. H. Khammash, *Stability and Performance Robustness of Discrete-Time Systems with Structured Uncertainty*, Ph.D. thesis, Rice University, Houston, TX, 1990.
[27] J. Kolter, *private communication with A. Askari*, 2019.
[28] A. Kurakin, I. J. Goodfellow, and S. Bengio, *Adversarial examples in the physical world*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 2017, Workshop Track Proceedings, OpenReview.net, 2017, https://openreview.net/forum?id=HJGU3Rodl.
[29] A. Kurakin, I. J. Goodfellow, and S. Bengio, *Adversarial Machine Learning at Scale*, preprint, https://arxiv.org/abs/1611.01236, 2017.
[30] V. Lebedev and V. Lempitsky, *Fast ConvNets using group-wise brain damage*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2554–2564.
[31] Y. LeCun, J. S. Denker, and S. A. Solla, *Optimal brain damage*, in Advances in Neural Information Processing Systems, 1990, pp. 598–605.
[32] J. Li, C. Fang, and Z. Lin, *Lifted proximal operator machines*, in Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 4181–4188.
[33] Y. Liu, X. Chen, C. Liu, and D. Song, *Delving into transferable adversarial examples and black-box attacks*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 2017, Conference Track Proceedings, OpenReview.net, 2017, https://openreview.net/forum?id=Sys6GJqxl.
[34] C. Louizos, M. Welling, and D. P. Kingma, *Learning Sparse Neural Networks through L_0 Regularization*, preprint, https://arxiv.org/abs/1712.01312, 2017.
[35] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, in International Conference on Learning Representations, 2018, https://openreview.net/forum?id=rJzIBfZAb.
[36] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.
[37] S. Narang, E. Elsen, G. Diamos, and S. Sengupta, *Exploring Sparsity in Recurrent Neural Networks*, preprint, https://arxiv.org/abs/1704.05119, 2017.
[38] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, *The limitations of deep learning in adversarial settings*, in IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, 2016, IEEE, 2016, pp. 372–387, https://doi.org/10.1109/EuroSP.2016.36.
[39] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, *Distillation as a defense to adversarial perturbations against deep neural networks*, in IEEE Symposium on Security and Privacy, SP 2016,

San Jose, CA, 2016, IEEE Computer Society, 2016, pp. 582–597, https://doi.org/10.1109/SP.2016.41.

[40] A. Raghunathan, J. Steinhardt, and P. S. Liang, *Semidefinite relaxations for certifying robustness to adversarial examples*, in Advances in Neural Information Processing Systems, 2018, pp. 10877–10887.

[41] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*, Interdiscip. Appl. Math. 10, Springer Science & Business Media, 2013.

[42] A. See, M.-T. Luong, and C. D. Manning, *Compression of Neural Machine Translation Models via Pruning*, preprint, https://arxiv.org/abs/1606.09274, 2016.

[43] U. Shaham, Y. Yamada, and S. Negahban, *Understanding adversarial training: Increasing local stability of neural nets through robust optimization*, Neurocomputing, 307 (2018), pp. 195–204, https://doi.org/10.1016/j.neucom.2018.04.027.

[44] K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep inside convolutional networks: Visualising image classification models and saliency maps*, in 2nd International Conference on Learning Representations, ICLR 2014, Banff, Canada, 2014, Y. Bengio and Y. LeCun, eds., Workshop Track Proceedings; available online at http://arxiv.org/abs/1312.6034.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, J. Mach, Learn. Res., 15 (2014), pp. 1929–1958.

[46] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, *Training neural networks without gradients: A scalable ADMM approach*, in International Conference on Machine Learning, 2016, pp. 2722–2731.

[47] B. Travacca, L. El Ghaoui, and S. Moura, *Implicit optimization: Models and methods*, in 2020 59th IEEE Conference on Decision and Control (CDC), 2020, pp. 408–415, https://doi.org/10.1109/CDC42340.2020.9304169.

[48] P.-W. Wang, P. Donti, B. Wilder, and Z. Kolter, *SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver*, in Proceedings of the 36th International Conference on Machine Learning, K. Chaudhuri and R. Salakhutdinov, eds., Proc. Mach. Learn. Res. 97, Long Beach, CA, 2019, PMLR, pp. 6545–6554, http://proceedings.mlr.press/v97/wang19e.html.

[49] E. Winston and J. Z. Kolter, *Monotone operator equilibrium networks*, in Advances in Neural Information Processing Systems 33, 2020, pp. 10718–10728.

[50] E. Wong and J. Z. Kolter, *Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope*, preprint, https://arxiv.org/abs/1711.00851, 2017.

[51] H. Yin, P. Seiler, and M. Arcak, *Stability Analysis Using Quadratic Constraints for Systems with Neural Network Controllers*, preprint, https://arxiv.org/abs/2006.07579, 2020.

[52] H. Yin, P. Seiler, M. Jin, and M. Arcak, *Imitation Learning with Stability and Safety Guarantees*, preprint, https://arxiv.org/abs/2012.09293, 2020.

[53] J. Zeng, T. T.-K. Lau, S. Lin, and Y. Yao, *Global Convergence of Block Coordinate Descent in Deep Learning*, preprint, https://arxiv.org/abs/1803.00225, 2018.

[54] Z. Zhang and M. Brand, *Convergent block coordinate descent for training Tikhonov regularized deep neural networks*, in Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., 2017, pp. 1719–1728.

[55] M. Zhu and S. Gupta, *To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression*, preprint, https://arxiv.org/abs/1710.01878, 2017.