Certified Invertibility in Neural Networks via Mixed-Integer Programming

Tianqi Cui TCUI3@JHU.EDU
Department of Chemical and Biomolecular Engineering, Johns Hopkins University, Baltimore, MD, USA

Thomas Bertalan TOM@TOMBERTALAN.COM
Department of Chemical and Biomolecular Engineering, Johns Hopkins University, Baltimore, MD, USA

George Pappas PAPPASG@SEAS.UPENN.EDU
Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA

Manfred Morari MORARI@SEAS.UPENN.EDU
Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA

Yannis Kevrekidis YANNISK@JHU.EDU
Department of Chemical and Biomolecular Engineering, Johns Hopkins University, Baltimore, MD, USA

Mahyar Fazlyab MAHYARFAZLYAB@JHU.EDU
Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD, USA

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Neural networks are known to be vulnerable to adversarial attacks, which are small, imperceptible perturbations that can significantly alter the network's output. Conversely, there may exist large, meaningful perturbations that do not affect the network's decision (excessive invariance). In our research, we investigate this latter phenomenon in two contexts: (a) discrete-time dynamical system identification, and (b) the calibration of a neural network's output to that of another network. We examine noninvertibility through the lens of mathematical optimization, where the global solution measures the "safety" of the network predictions by their distance from the non-invertibility boundary. We formulate mixed-integer programs (MIPs) for ReLU networks and L_p norms ($p = 1, 2, \infty$) that apply to neural network approximators of dynamical systems. We also discuss how our findings can be useful for invertibility certification in transformations between neural networks, e.g. between different levels of network pruning.

1. Introduction

Despite achieving high performance in various classification and regression tasks, neural networks do not always guarantee certain desired properties after training. Adversarial robustness is a well-known example, as neural networks can be overly sensitive to carefully designed input perturbations (Szegedy et al. (2013)). This intriguing property also holds in the reverse direction, where neural networks can be excessively insensitive to large perturbations in classification problems. This can cause two semantically different inputs (such as images) to be classified in the same category (Jacobsen et al. (2018)). Indeed, a fundamental trade-off exists between adversarial robustness and excessive invariance (Tramèr et al. (2020)), which is mathematically related to the noninvertibility of the input-output map defined by the neural network.

To address the issue of noninvertibility and excessive invariance, one can consider invertible-by-design architectures. Invertible neural networks (INNs) have been used to design generative models (Donahue and Simonyan (2019)), implement memory-saving gradient computation (Gomez et al. (2017)), and solve inverse problems (Ardizzone et al. (2018)). However, commonly used INN architectures suffer from exploding inverses. In this paper, we focus on certifying the (possible) non-invertibility of conventional neural networks after training. We specifically study two relevant invertibility problems: (i) local invertibility of neural networks, where we verify whether a dynamical system parameterized by a neural network is locally invertible around a certain input (or trajectory), and compute the largest region of local invertibility; and (ii) local invertibility of transformations between neural networks, where we certify whether two "equivalent" neural networks (e.g. resulting from different levels of pruning) can be transformed (or calibrated) to each other locally via an invertible map. We develop mathematical tools based on mixed-integer linear/quadratic programming for characterizing non-invertibility, which can be applied to neural network approximators of dynamical systems, as well as transformations between different neural networks.

Related Work Noninvertibility in neural networks was first studied in the 1990s (Gicquel et al. (1998); Rico-Martinez et al. (1993)). More recently, several papers have focused on the global invertibility property in neural networks, including works such as Chang et al. (2018); Teshima et al. (2020); Chen et al. (2018); MacKay et al. (2018); Jaeger (2014). The invertibility of neural networks has been analyzed (Behrmann et al. (2018)), and invertible architectures have been developed for applications such as generative modeling (Chen et al. (2019)), inverse problems (Ardizzone et al. (2019)), and probabilistic inference (Radev et al. (2020)). Some of these networks, such as RevNet (Gomez et al. (2017)), NICE (Dinh et al. (2015)), and real NVP (Dinh et al. (2017)), partition the input domains and use affine or coupling transformations as the forward pass, resulting in nonzero determinants and keeping the Jacobians (block-)triangular with nonzero diagonal elements. Others, like i-ResNet (Behrmann et al. (2019)), have no analytical forms for the inverse dynamics, yet their finite bi-Lipschitz constants can be derived. Both methods can guarantee global invertibility. A comprehensive analysis of these architectures can be found in Behrmann et al. (2021); Song et al. (2019). However, a theoretical understanding of the expressiveness of these architectures, as well as their universal approximation properties, is still incomplete. Compared to standard networks like multi-layer perceptrons (MLPs) or convolutional neural networks (CNNs), invertible neural networks (INNs) are computationally demanding. Neural ODE (Chen et al. (2018)) uses an alternative method to compute gradients for backward propagation, while i-ResNet (Behrmann et al. (2019)) has restrictions on the norm of every weight matrix to be enforced during the training process. In most cases, the input domain of interest is a small subset of the whole space. For example, the grey-scale image domain in computer vision problems is $[0,1]^{H\times W}$, where H and W are the height and width of the images; it is unnecessary to consider the entire$\mathbb{R}^{H\times W}$. We thus focus on *local invertibility*: how do we determine if our network is invertible on a given domain, and if not, how do we quantify noninvertibility?

2. Invertibility Certification of Neural Networks and of Transformations between them

Here we pose the verification of local invertibility of continuous functions as optimization problems. We then show that for ReLU networks, this leads to a mixed-integer linear/quadratic program. For an integer $q \geq 1$, we denote the L_q-ball centered at x_c by $\mathcal{B}_q(x_c, r) = \{x \in \mathbb{R}^n \mid \|x - x_c\|_q \leq r\}$ (the notation also holds when $q \to +\infty$).

2.1. Invertibility Certification of ReLU Networks via Mixed-Integer Programming

Problem 1 (Local Invertibility of NNs) *Given a neural network $f : \mathbb{R}^m \mapsto \mathbb{R}^m$ and a point $x_c \in \mathbb{R}^m$ in the input space, we want to find the largest radius $r > 0$ such that f is invertible on $\mathcal{B}_q(x_c, r)$, i.e., $f(x) \neq f(y)$ for all $x, y \in \mathcal{B}_q(x_c, r)$, $x \neq y$.* [1]

Another relevant problem is to verify whether, for a particular point, a nearby point exists with the same forward image. We formally state the problem as follows.

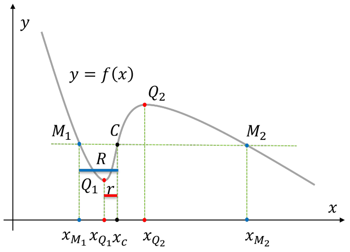

Figure 1: Illustration of problems 1 and 2 in one dimension.

Problem 2 (Pseudo Local Invertibility of NNs) *Given a neural network $f : \mathbb{R}^m \mapsto \mathbb{R}^m$ and a point $x_c \in \mathbb{R}^m$ in the input space, we want to find the largest radius $R > 0$ such that $f(x) \neq f(x_c)$ for all $x \in \mathcal{B}_q(x_c, R)$, $x \neq x_c$.*

If r and R are the optimal radii in Problems 1 and 2 respectively, we must have $r \leq R$. For Problem 1, the ball $\mathcal{B}_q(x_c, r)$ just "touches" the J_0 set (i.e. the set of points where $f' = 0$); for Problem 2, the ball $\mathcal{B}_q(x_c, R)$ extends to the "other" closest preimage of $f(x_c)$. Figure 1 illustrates both concepts in the one-dimensional case. For the scalar function $y = f(x)$ and around a particular input x_c, we show regions with local invertibility and pseudo invertibility. The points $Q_1 = (x_{Q_1}, y_{Q_1})$ and $Q_2 = (x_{Q_2}, y_{Q_2})$ are two closest turning points (elements of the J_0 set) to the point $C = (x_c, y_c)$; f is uniquely invertible (bi-Lipschitz) on the open interval (x_{Q_1}, x_{Q_2}), so that the optimal solution to Problem 1 is: $r = \min\{|x_{Q_1} - x_c|, |x_{Q_2} - x_c|\} = |x_{Q_1} - x_c|$. Noting that $M_1 = (x_{M_1}, y_{M_1})$ and $M_2 = (x_{M_2}, y_{M_2})$ are two closest points that have the same y-coordinate as the point $C = (x_c, y_c)$, the optimal solution to Problem 2 is $R = \min\{|x_{M_1} - x_c|, |x_{M_2} - x_c|\} = |x_{M_1} - x_c|$.

We now state our first result, posing the local invertibility of a function (such as a neural network) as a constrained optimization problem.

Theorem 1 (Local Invertibility of Continuous Functions) *Let $f \colon \mathbb{R}^m \to \mathbb{R}^m$ be a continuous function and $\mathcal{B} \subset \mathbb{R}^m$ be a compact set. Consider the following optimization problem,*

$$p^\star \leftarrow \max \quad \|x - y\| \quad \textit{subject to } x, y \in \mathcal{B}, \quad f(x) = f(y). \tag{1}$$

Then f is invertible on $\mathcal{B}$ if and only if $p^\star = 0$.

Theorem 2 (Pseudo Local Invertibility) *Let $f \colon \mathbb{R}^m \to \mathbb{R}^m$ be a continuous function and $\mathcal{B} \subset \mathbb{R}^m$ be a compact set. Suppose $x_c \in \mathcal{B}$. Consider the following optimization problem,*

$$P^\star \leftarrow \max \quad \|x - x_c\| \quad \textit{subject to } x \in \mathcal{B}, \quad f(x) = f(x_c). \tag{2}$$

Then we have $f(x) \neq f(x_c)$ for all $x \in \mathcal{B} \setminus \{x_c\}$ if and only if $P^\star = 0$.

Note that by adding the equality constraint $y = x_c$ to Problem (1), we obtain Problem (2). Hence, we will only focus on Problem (1) in the sequel.

[1] Here$f$ has the same domain/co-domain dimension. Our mixed-integer formulation does not require this assumption.

Mixed-Integer Formulation of Problem (1) We now show that for a given ball $\mathcal{B}_\infty(x_c, r)$ in the input space, and piecewise linear networks with ReLU activations, the optimization problem in (1) can be cast as an MILP. We start by noting that a single ReLU constraint $y = \max(0, x)$ with pre-activation bounds $\underline{x} \leq x \leq \bar{x}$ can be equivalently described by the following mixed-integer linear constraints (Tjeng et al. (2017)),

$$y = \max(0, x),\ \underline{x} \leq x \leq \bar{x} \iff \{y \geq 0,\ y \geq x, y \leq x - \underline{x}(1-t),\ y \leq \bar{x}t,\ t \in \{0,1\}\}, \quad (3)$$

where the binary variable $t \in \{0,1\}$ is an indicator of the activation function being active ($y = x$) or inactive ($y = 0$). Now consider an ℓ-layer feed-forward fully-connected ReLU network,

$$x^{(k+1)} = \max(W^{(k)}x^{(k)} + b^{(k)}, 0) \text{ for } k = 0, \cdots, \ell-1;\ f(x^{(0)}) = W^{(\ell)}x^{(\ell)} + b^{(\ell)}, \quad (4)$$

where $x^{(k)} \in \mathbb{R}^{n_k}$ ($n_0 = m$), $W^{(k)} \in \mathbb{R}^{n_{k+1} \times n_k}, b^{(k)} \in \mathbb{R}^{n_{k+1}}$ are the weight matrices and bias vectors of the affine layers. We denote $n = \sum_{k=1}^{\ell} n_k$ the total number of neurons. Suppose $l^{(k)}$ and $u^{(k)}$ are known elementwise lower and upper bounds on the input to the $(\ell+1)$-th activation layer, i.e., $l^{(k)} \leq W^{(k)}x^{(k)} + b^{(k)} \leq u^{(k)}$. Then the neural network equations are equivalent to a set of mixed-integer constraints as follows,

$$x^{(k+1)} = \max(W^{(k)}x^{(k)} + b^{(k)}, 0) \Leftrightarrow \begin{cases} x^{(k+1)} \geq W^{(k)}x^{(k)} + b^{(k)} \\ x^{(k+1)} \leq W^{(k)}x^{(k)} + b^{(k)} - l^{(k)} \odot (1_{n_{k+1}} - t^{(k)}) \\ x^{(k+1)} \geq 0, \quad x^{(k+1)} \leq u^{(k)} \odot t^{(k)}, \end{cases} \quad (5)$$

where $t^{(k)} \in \{0,1\}^{n_{k+1}}$ is a vector of binary variables for the $(k+1)$-th activation layer and $1_{n_{k+1}}$ denotes vector of all 1's in $\mathbb{R}^{n_{k+1}}$. We note that the element-wise pre-activation bounds $\{l^{(k)}, u^{(k)}\}$ can be precomputed by, for example, interval bound propagation or linear programming, assuming known bounds on the input of the neural network (Weng et al. (2018); Zhang et al. (2018); Hein and Andriushchenko (2017); Wang et al. (2018); Wong and Kolter (2018)). Since the state-of-the-art solvers for mixed-integer programming are based on branch & bound algorithms (Land and Doig (1960); Beasley (1996)), tight pre-activation bounds will allow the algorithm to prune branches more efficiently and reduce the total running time.

$$\begin{aligned}
&p^\star \leftarrow \max w \text{ subject to } \|x^{(0)} - x_c\|_\infty \leq r,\ \ \|y^{(0)} - x_c\|_\infty \leq r \\
&(\text{I}) : \begin{cases} (x^{(0)} - y^{(0)}) \leq w1_{n_0} \leq (x^{(0)} - y^{(0)}) + 4r(1_{n_0} - F) \\ -(x^{(0)} - y^{(0)}) \leq w1_{n_0} \leq -(x^{(0)} - y^{(0)}) + 4r(1_{n_0} - F') \\ F + F' \leq 1_{n_0}, 1_{n_0}^\top(F + F') = 1, F, F' \in \{0,1\}^{n_0} \end{cases} \\
&(\text{II}) : W^{(\ell)}x^{(\ell)} = W^{(\ell)}y^{(\ell)} \qquad\qquad (6) \\
&\quad \text{for } k = 0, \cdots, \ell - 1 : \\
&(\text{III}) : \begin{cases} x^{(k+1)} \geq W^{(k)}x^{(k)} + b^{(k)}, y^{(k+1)} \geq W^{(k)}y^{(k)} + b^{(k)} \\ x^{(k+1)} \leq W^{(k)}x^{(k)} + b^{(k)} - l^{(k)} \odot (1 - t^{(k)}), y^{(k+1)} \leq W^{(k)}y^{(k)} + b^{(k)} - l^{(k)} \odot (1 - t^{(k)}) \\ x^{(k+1)} \geq 0, y^{(k+1)} \geq 0, x^{(k+1)} \leq u^{(k)} \odot t^{(k)}, y^{(k+1)} \leq u^{(k)} \odot t^{(k)}; t^{(k)}, s^{(k)} \in \{0,1\}^{n_k+1}, \end{cases}
\end{aligned}$$

Having represented the neural network equations by mixed-integer constraints, it remains to encode the objective function $\|x^{(0)} - y^{(0)}\|$ as well as the set $\mathcal{B}$. We assume that $\mathcal{B}$ is an L_∞ ball around a given point x_c, i.e., $\mathcal{B} = \mathcal{B}_\infty(x_c, r)$. Furthermore, for the sake of space, we only consider L_∞ norms for the objective function. Specifically, consider the equality $w = \|x^{(0)} - y^{(0)}\|_\infty$. This equality can be encoded as mixed-integer linear constraints by introducing $2n_0$ mutually exclusive indicator vectors(F and F' each with n_0 coordinates). This would lead to the MILP in (6), where the set of constraints in (I) model the objective function $\|x^{(0)} - y^{(0)}\|_\infty$, and the set of constraints (III) encodes $x^{(k+1)} = \max(W^{(k)}x^{(k)} + b^{(k)}, 0)$ and $y^{(k+1)} = \max(W^{(k)}y^{(k)} + b^{(k)}, 0)$ which is exactly (5). The constraint (II) enforces $f(x^{(0)}) = f(y^{(0)})$ which can be inferred from (4). To see the correctness of (I), suppose $F_j = 1$ for some $j = 1, \cdots, n_0$. Then, we must have $F'_i = 0$ for $\forall i = 1, \cdots, n_0$ and $F_i = 0$ for $\forall i \neq j$. This implies $w = (x_j^{(0)} - y_j^{(0)}) \geq (x_i^{(0)} - y_i^{(0)})$ for $\forall i \neq j$, and $w \geq -(x_i^{(0)} - y_i^{(0)})$ for $\forall i$. A similar argument can be made when $F'_j = 1$ for some $j = 1, \cdots, n_0$. The optimization problem (6) has a total of $2(n_0 + n)$ integer variables.

Remark 3 *Using the ℓ_2 norm for both the objective function and the ball $\mathcal{B}_2(x_c, r)$, leads to a mixed-integer quadratic program (MIQP). However,* (6) *remains an MILP in the ℓ_1 norm case.*

Largest Region of Invertibility (Problem 1) For a fixed radius $r \geq 0$, the optimization problem (6) either verifies whether f is invertible on $\mathcal{B}_\infty(x_c, r)$ or it finds counter examples $x^{(0)} \neq y^{(0)}$ such that $f(x^{(0)}) = f(y^{(0)})$. Thus, we can find the maximal r by performing a bisection search on r.

To close this section, we consider the problem of invertibility certification in transformations between two functions (and in particular neural networks).

2.2. Invertibility Certification of Transformations between Neural Networks

Training two neural networks for the same regression or classification task practically never gives identical networks. Numerous criteria exist for comparing the performance of different models (e.g. accuracy in classification, or mean-squared loss in regression). Here we explore whether two different models *can be calibrated to each other* (leading to a *de facto* implicit function problem). Extending our analysis provides invertibility guarantees for the transformation from output of network 1 to output of network 2.

Problem 3 (Transformation Invertibility) *Given two functions $f_1, f_2 \colon \mathbb{R}^m \to \mathbb{R}^m$ (e.g. two neural networks) and a particular point $x_c \in \mathbb{R}^m$ in the input space, we would like to find the largest ball $\mathcal{B}_q(x_c, r)$ over which f_2 is a function of f_1.*

Theorem 4 *Let $f_1 \colon \mathbb{R}^m \to \mathbb{R}^n$, $f_2 \colon \mathbb{R}^m \to \mathbb{R}^n$ be two continuous functions and $\mathcal{B} \subset \mathbb{R}^m$ be a compact set. Then f_2 is a function of f_1 on $\mathcal{B}$ if and only if $p_{12}^\star = 0$, where*

$$p_{12}^\star \leftarrow \max \quad \|f_2(x^{(1)}) - f_2(x^{(2)})\| \quad \textit{subject to } x^{(1)}, x^{(2)} \in \mathcal{B}, \quad f_1(x^{(1)}) = f_1(x^{(2)}). \tag{7}$$

Similar to Problem 1, we can pose Problem 3 as a mixed-integer program. Furthermore, we can also define $p_{21}^\star$, whose zero value verifies whether f_1 is a function of f_2 over $\mathcal{B}$. It is straightforward that $p_{12}^\star = p_{21}^\star = 0$ if and only if f_2 is an invertible function of f_1.

3. Local Invertibility of Dynamical Systems and Neural Networks

Noninvertibility can lead to catastrophic consequences not only in classification but also in regression, particularly in dynamical systems prediction. The flow of smooth differential equations is invertible when it exists, yet traditional numerical integrators used to approximate them can be non-invertible. Neural network approximations of the corresponding map also suffer from this potential pathology. Here, we study non-invertibility in the context of dynamical systems predictions.

Continuous-time dynamical systems, in particular autonomous ordinary differential equations (ODEs) have the form $dX(t)/dt = f(X(t)), X(t = t_0) = X_0$, where $X(t) \in \mathbb{R}^m$ are the state variables of interest; $f : \mathbb{R}^m \mapsto \mathbb{R}^m$ relates the states to their time derivatives; $X_0 \in \mathbb{R}^m$ is the initial condition at t_0. If f is uniformly Lipschitz continuous in X and continuous in t, the Cauchy-Lipschitz theorem provides the existence and uniqueness of the solution.

In practice, we observe the states $X(t)$ at discrete points in time, starting at $t_0 = 0$. For a fixed timestep $\tau \in \mathbb{R}^+$, and $\forall n \in \mathbb{N}$, $t_n = n\tau$ denotes the n-th time stamp, and $X_n = X(t = t_n)$ the corresponding state values. Now we will have:

$$X_{n+1} := F(X_n) = X_n + \int_{t_n}^{t_{n+1}} f(X(t))dt;\ X_n = F^{-1}(X_{n+1}). \tag{8}$$

This equation also works as the starting point of many numerical ODE solvers.

For the time-one map in (8), the inverse function theorem provides a sufficient condition for its invertibility: If F is a continuously differentiable function from an open set $\mathcal{B}$ of $\mathbb{R}^m$ into $\mathbb{R}^m$, and the Jacobian determinant of F at p is nonzero, then F is invertible near p. Thus, if we define the *noninvertibility locus* as the set $J_0(F) = \{p \in \mathcal{B} : \det(\mathbf{J}_F(p)) = 0\}$; then the condition $J_0(F) = \emptyset$ guarantees global invertibility of F (notice that this condition is not necessary: the scalar function $F(X) = X^3$ provides a counterexample). If F is continuous over $\mathcal{B}$ but not everywhere differentiable, then the definition of J_0 set should be altered to:

$$J_0(F) = \{p \in \mathcal{B} : \forall N_0(p), \exists\, p_1, p_2 \in N_0(p), p_1 \neq p_2,\ \text{s.t. } \det(\mathbf{J}_F(p_1))\det(\mathbf{J}_F(p_2)) \leq 0\}\,. \tag{9}$$

Numerical Integrators are (often) Noninvertible Numerically approximating the integral in (8) can introduce noninvertibility in the transformation. A simple one-dimensional illustrative ODE example is $f(X) = X^2 + bX + c,\ X(t = 0) = X_0$, where $b, c \in \mathbb{R}$ are two fixed parameters. Although the analytical solution (8) is invertible, a forward-Euler discretization with step τ gives

$$X_{n+1} = F(X_n) = X_n + \tau(X_n^2 + bX_n + c) \Rightarrow \tau X_n^2 + (\tau b + 1)X_n + (\tau c - X_{n+1}) = 0. \tag{10}$$

Given a fixed X_{n+1}, Equation (10) is quadratic w.r.t. X_n; this determines the local invertibility of F based on $\Delta = (\tau b + 1)^2 - 4\tau(\tau c - X_{n+1})$: no real root if $\Delta < 0$; one real root with multiplicity 2 if $\Delta = 0$; and two distinct real roots if $\Delta > 0$. In practice, one uses small timesteps $\tau \ll 1$ for accuracy/stability, leading to the last case: there will always exist a solution X_n close to X_{n+1}, and a second preimage, far away from the region of our interest, and arguably physically irrelevant (to $X_n \to -\infty$ as $\tau \to 0$). On the other hand, as τ grows, the two roots move closer to each other, $J_0(F)$ moves close to the regime of our simulations, and noninvertibility can have visible implications on the predicted dynamics. Thus, choosing a small timestep in explicit integrators guarantees desirable accuracy, and simultaneously *practically* mitigates noninvertibility pathologies in the dynamics.

4. Numerical Experiments

We now present experiments with ReLU multi-layer perceptrons (MLPs) in regression problems, and also transformations between two ReLU networks. To solve the Mixed-integer programs we use Gurobi Optimization, LLC (2023). To find the pre-activation bounds, we use interval bound propagation.

1D Example We use a 1-10-10-1 randomly generated fully-connected neural network f with ReLU activations. We find the largest interval around the points $x = -1.8, -1, -0.3$ on which f is invertible (Problem 1), and the largest interval around the point $x = -1$ on which any other points inside the region will not map to $f(-1)$ (Problem 2). The results are plotted in the inset of Figure 2, where intervals in red and blue respectively represent the optimal solutions for the two problems. The computed largest certified radii are 0.157, 0.322, 0.214, and 0.553.

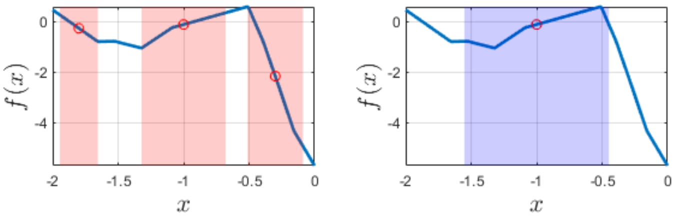

Figure 2: Solutions to Problem 1 (left, red) and Problem 2 (right, blue) for the MLP corresponding to a randomly-generated ReLU network (see text).

2D Example: the Brusselator Model The Brusselator (Tyson (1973)) is a two-variable (x, y) ODE system depending on parameters (a, b), that describes oscillatory dynamics in a theoretical chemical reaction scheme. We use its forward-Euler discretization

$$x_{n+1} = x_n + \tau(a + x_n^2 y_n - (b+1)x_n),\; y_{n+1} = y_n + \tau(bx_n - x_n^2 y_n). \tag{11}$$

Rearranging the equation of y_{n+1} to solve for y_n in (11) and substituting it into the one of x_{n+1} we obtain:

$$\tau(1-\tau)x_n^3 + \tau(\tau a - x_{n+1} - y_{n+1})x_n^2 + (\tau b + \tau - 1)x_n + (x_{n+1} - \tau a) = 0. \tag{12}$$

Equation (12) is a cubic for x_n given (x_{n+1}, y_{n+1}) when $\tau \neq 1$. By varying the parameters a, b and τ, we see the past states $(x_n, y_n)^T$ (also called "inverses" or "preimages") may be multi-valued, so that this discrete-time system is, in general, noninvertible. We fix $a = 1$ and consider how inverses will be changing (a) with b for fixed $\tau = 0.15$; and (b) with τ, for fixed $b = 2$.

In general, the neural network we are interested in is a mapping from 3D to 2D: $(x_{n+1}, y_{n+1})^T \approx \mathcal{N}(x_n, y_n; p)^T$, where $p \in \mathbb{R}$ is the parameter. The network dynamics will be parameter-dependent if we set $p \equiv b$, or timestep-dependent if $p \equiv \tau$. Considering the first layer of a MLP:

$$W^{(0)} \begin{bmatrix} x_n \\ y_n \\ p \end{bmatrix} + b^{(0)} = (W^{(0)}(e_1 + e_2)) \begin{bmatrix} x_n \\ y_n \end{bmatrix} + (pW^{(0)}e_3 + b^{(0)}), \tag{13}$$

where $e_{1,2,3} \in \mathbb{R}^3$ are indicator vectors. For fixed p our network $\mathcal{N}$ can be thought of as an MLP mapping from $\mathbb{R}^2$ to $\mathbb{R}^2$, by slightly modifying the weights and biases in the first linear layer. Here, we trained two separate MLPs, with b and τ dependence respectively.

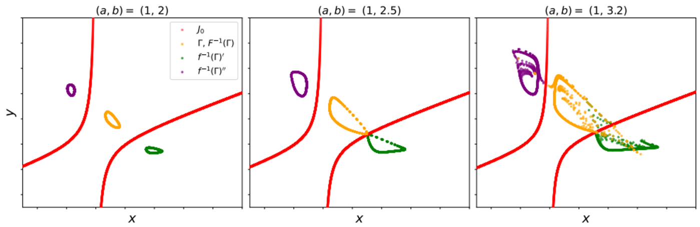

Figure 3: Attractors and their multiple inverses for several parameter values of the Brusselator model. Notice the relation of the J_0 curves and the "extra" preimages. When the attractor starts interacting with the J_0 and these extra preimages, the dynamic behavior degenerates quantitatively and qualitatively.

Parameter-Dependent Inverses We start with a brief discussion of the dynamics and noninvertibility in the ground-truth system (see Figure 3). Consider an initial state located on the Brusselator attracting invariant circle (IC, in orange); we know this has at least one preimage *also on this IC*. In Figure 3 we see that every point on the IC has three preimages: one still on the IC, and two additional inverses (in green and purple); after one iteration, all three loops map to the orange one, and then remain forward invariant. The phase space *folds* along the two branches of the J_0 curve (shown in red). For lower values of b (left), these three closed loops do not intersect each other. As b increases the (orange) attractor will become tangent to (center), and subsequently intersect J_0 (right), leading to mixing of the preimages. At this point the predicted dynamics become nonphysical (beyond just inaccurate).

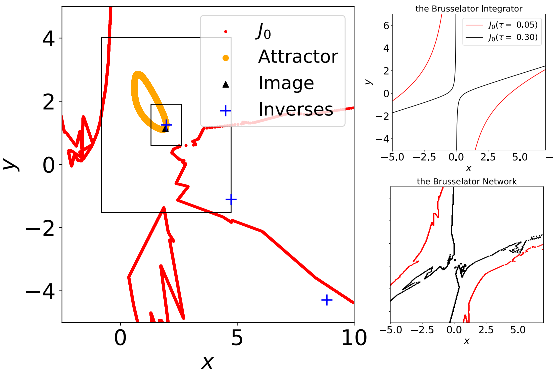

Figure 4: **Left:** illustration of our solution to Problems 1 and 2 for the Brusselator network. For a random reference point on the attractor, we show the neighborhoods found by our algorithms. They clearly find the closest point on the J_0 curve / the closest "extra preimage" of the point of interest. **Right**: plots of J_0 curves at different τ, for both the Euler integrator **(Top)** and our Brusselator ReLU network **(Bottom)**. Small timesteps lead to progressively remote J_0 curves. Notice also the piecewise linear nature of the J_0 curve for the ReLU network; its accurate computation is an interesting problem by itself.

After convergence of training, we employ our algorithm to obtain noninvertibility certificates for the resulting MLP, and plot results of $b = 2.1$ in the left subfigure of Figure 4. In Figure 4, we arbitrarily select one representative point, marked by a triangle ($\triangle$), on the attractor (the orange invariant circle); a nearby inverse *also* on the attractor, the *primal* inverse, is marked by a cross ($+$). Our algorithm will produce two regions for this point, one for each of our problems (squares of constant L_∞ distance in 2D). As a sanity check, we also compute the J_0 sets (the red point), as well as a few additional inverses, beyond the primal ones with the help of numerical root solver and automatic differentiation (Baydin et al. (2017)). Clearly, the smaller square neighborhood "just hits" the J_0 curve, while the larger one extends to the closest nonprimal inverse of the attractor.

Timestep-Dependent Inverses In the right two subfigures of Figure 4, we explore the effect of varying the time horizon τ. We compare a single Euler step of the ground truth ODE to the MLP approximating the same flowmap, and find that, in both, smaller time horizons lead to larger regions of invertibility.

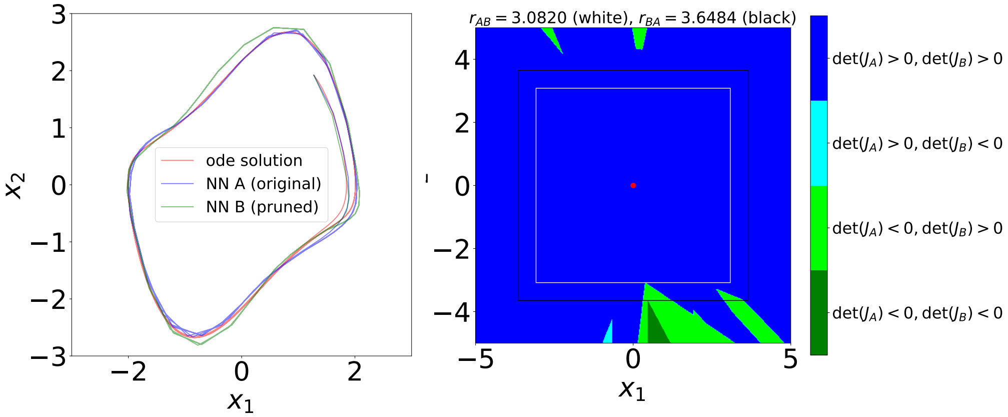

Figure 5: Left: Trajectories of the ODE solution for the Van der Pol system (red), and their discrete-time neural network approximations (blue and green). All three trajectories begin at the same initial state; the ODE solution is smooth (continuous-time), the other two use straight lines between consecutive states (discrete-time). However, it is clear all three systems have nearby attractors, indicating good performance of the network and its pruned version. Right: visualization of MILP computation results, along with the sign of the Jacobian values of the networks on the grid points of the input domain. Here, the center of the region is marked red, while the white and black boundaries quantify the region of mappability between outputs of network A and network B.

Network Transformation Example: Learning the Van der Pol Equation Here, to test our algorithm on network transformation problem 3, we trained two networks on the same regression task. Our data comes from the 2D Van der Pol equation $dx_1/dt = x_2, dx_2/dt = \mu(1 - x_1^2)x_2 - x_1$, where the input and output are the initial and final states of 1000 solution trajectories with time duration 0.2 for $\mu = 1$, when a stable limit cycle exists. The initial states are uniformly sampled in the region $[-3, 3] \times [-3, 3]$. The neural network A used to learn the time series is a 2-32-32-2 MLP, while the neural network B is a sparse version of A, where half of the weight entries are pruned (set to zero) based on Zhu and Gupta (2018). To visualize the performances of the networks, two trajectories generated by respectively iterating the network functions for fixed times from a given initial state have been plotted in the left subplot of Figure 5. The ODE solution trajectory starting at the same initial state with same time duration is also shown. We see that both network functions A and B exhibit long-term oscillations, though the shapes of the attractors have small visual differences from the true ODE solution (the red curve).

These two network functions were then used to test the correctness of the algorithm for the problem 3. Here we chose the center points $x_c = (0, 0)^T$, computed and plotted the mappable regions for two subcases (see right subfigure of Figure 5): the output of network B is a function of the output of network A (the square with white bounds centered at the red point, radius 3.0820), and vice versa (the square with black bounds centered at the red point, radius 3.6484). For validation we also computed the Jacobian values of network A and network B on every grid point of the input domain, and shown that the white square touches the J_0 curve of network A, while the black square touches the J_0 curve of network B. Inside the black square the Jacobian of network B remains positive, so that network B is invertible (i.e. the existence of the mapping from $f_B(x)$ to x, or equivalently, $f_B^{-1}(x)$); therefore we can find the mapping from $f_B(x)$ to $f_A(x)$ by composing the mapping from $f_B(x)$ to x and the mapping from x to $f_A(x)$ (the function $f_A(x)$ itself). The size of the white square can be similarly rationalized, validating our computation.

Sparsity	40 %			50 %			60 %		
Network B	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9
r_{AB}	3.0820	3.0820	3.0820	3.0820	3.0820	3.0820	3.0820	3.0820	3.0820
r_{BA}	3.4609	3.1055	3.8555	3.6484	2.6523	3.8203	3.6328	3.9727	4.5547

Table 1: The radii of the mappable regions between the original network A and its pruned versions B.

As a sanity check, we consructed eight more pruned networks; two of them have 50% sparsity (networks B_5 and B_6), three have 40% sparsity (networks B_1, B_2 and B_3) and the others have 60% sparsity (networks B_7, B_8 and B_9). Above, we discussed network B_4. For each pruned network, we computed the radii of the regions of interest (aka r_{AB} and r_{BA}). The results are listed in Table 1. All pruned networks $\{B_i\}$ share the same radii r_{AB}, consistent with the invertibility of A itself. Since $r_A = 3.0820$, A is invertible in the ball we computed, and the existence of the mapping $y_A \mapsto y_B$ by composition of $y_A \mapsto x$ and $x \mapsto y_B$. In our work the input and output dimensions are the same (e.g. $m = n$ in Problem 3); this condition is not restrictive, and our algorithm can be possibly extended to classification problems, where in general $m \gg n$.

5. Conclusions

In this paper, we addressed the issue of noninvertibility that arises in discrete-time dynamical systems and neural networks performing time-series related tasks. We highlighted the potential pathological consequences of such noninvertibility, which extend beyond prediction inaccuracies and affect the predicted dynamics of the networks. Moreover, we extended our analysis to transformations between different neural networks and formulated three problems that provide a quantifiable assessment of local invertibility for any arbitrarily selected input. For functions such as MLPs with ReLU activations, we formulated these problems as mixed-integer programs and performed experiments on regression tasks; we also extended our algorithm to Resnets.

In future work, we aim to develop structure-exploiting methods that can globally solve these mixed-integer programs more efficiently for larger networks. Additionally, given the linearity of convolution and average pooling operations and the piecewise linearity of max pooling, we plan to adapt our algorithm to convolutional neural networks like AlexNet (Krizhevsky et al. (2017)) and VGG (Simonyan and Zisserman (2015)). Our successful application of the algorithm to ResNet architectures (He et al. (2016)) holds promise for applicability to recursive architectures (Lu et al. (2018); E (2017)) such as fractal networks (Larsson et al. (2017)), poly-inception networks (Zhang et al. (2016)), and RevNet (Gomez et al. (2017)). Furthermore, we are working on making the algorithm practical for continuous differentiable activations such as tanh or Swish (Ramachandran et al. (2017)), and other piecewise activations such as Gaussian Error Linear Units (GELUs, Hendrycks and Gimpel (2016)). Finally, we are particularly interested in exploring the case where the input and output domains have different dimensions, such as in classifiers.

References

Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

Lynton Ardizzone, Jakob Kruse, Sebastian J. Wirkert, D. Rahner, Eric W. Pellegrini, R. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. Analyzing inverse problems with invertible neural networks. *ArXiv*, abs/1808.04730, 2019.

Atılım Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.*, 18(1):5595–5637, January 2017. ISSN 1532-4435.

J. E. Beasley, editor. *Advances in Linear and Integer Programming*. Oxford University Press, Inc., USA, 1996. ISBN 0198538561.

Jens Behrmann, Sören Dittmer, Pascal Fernsel, and P. Maass. Analysis of invariance and robustness via invertibility of relu-networks. *ArXiv*, abs/1806.09730, 2018.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Joern-Henrik Jacobsen. Invertible residual networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 573–582. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/behrmann19a.html.

Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger Grosse, and Joern-Henrik Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1792–1800. PMLR, 13–15 Apr 2021. URL http://proceedings.mlr.press/v130/behrmann21a.html.

Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2811–2818. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16517.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6572–6583, Red Hook, NY, USA, 2018. Curran Associates Inc.

. Full text is available at: https://arxiv.org/abs/2301.11783.

Ricky TQ Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1906.02735.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015. URL http://arxiv.org/abs/1410.8516.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *Advances in neural information processing systems*, 32, 2019.

Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 3 2017. doi: 10.1007/s40304-017-0103-z. Dedicated to Professor Chi-Wang Shu on the occasion of his 60th birthday.

N. Gicquel, J.S. Anderson, and I.G. Kevrekidis. Noninvertibility and resonance in discrete-time neural networks for time-series processing. *Physics Letters A*, 238(1):8–18, 1998. ISSN 0375-9601. doi: https://doi.org/10.1016/S0375-9601(97)00753-6. URL https://www.sciencedirect.com/science/article/pii/S0375960197007536.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pages 2266–2276, 2017.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. *arXiv preprint arXiv:1811.00401*, 2018.

H. Jaeger. Controlling recurrent neural networks by conceptors. *ArXiv*, abs/1403.3369, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL https://doi.org/10.1145/3065386.

A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1910129.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3282–3291, Stockholm, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/lu18d.html.

Matthew MacKay, Paul Vicol, Jimmy Ba, and Roger Grosse. Reversible recurrent neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9043–9054, Red Hook, NY, USA, 2018. Curran Associates Inc.

Stefan T. Radev, Ulf K. Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2020. doi: 10.1109/TNNLS.2020.3042395.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Swish: a self-gated activation function. *CoRR*, 2017. URL http://arxiv.org/abs/1710.05941v1.

R. Rico-Martinez, I.G. Kevrekidis, and R.A. Adomaitis. Noninvertibility in neural networks. In *IEEE International Conference on Neural Networks*, pages 382–386 vol.1, 1993. doi: 10.1109/ICNN.1993.298587.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

Yang Song, Chenlin Meng, and Stefano Ermon. Mintnet: Building invertible neural networks with masked convolutions. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/70a32110fff0f26d301e58ebbca9cb9f-Paper.pdf.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators, 2020.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Florian Tramèr, Jens Behrmann, Nicholas Carlini, Nicolas Papernot, and Jörn-Henrik Jacobsen. Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations. In *International Conference on Machine Learning*, pages 9561–9571. PMLR, 2020.

John J. Tyson. Some further studies of nonlinear oscillations in chemical systems. *The Journal of Chemical Physics*, 58(9):3919–3930, 1973. doi: 10.1063/1.1679748. URL https://doi.org/10.1063/1.1679748.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, pages 6367–6377, 2018.

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295, 2018.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems*, pages 4939–4948, 2018.

Xingcheng Zhang, Zhizhong Li, Chen Change Loy, and Dahua Lin. Polynet: A pursuit of structural diversity in very deep networks. *arXiv preprint arXiv:1611.05725*, 2016.

Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Sy1iIDkPM.