An approach to reachability analysis for feed-forward ReLU neural networks

Alessio Lomuscio, Lalit Maganti
Imperial College London, London, UK
{a.lomuscio, lalit.maganti13}@imperial.ac.uk

Abstract

We study the reachability problem for systems implemented as feed-forward neural networks whose activation function is implemented via ReLU functions. We draw a correspondence between establishing whether some arbitrary output can ever be outputed by a neural system and linear problems characterising a neural system of interest. We present a methodology to solve cases of practical interest by means of a state-of-the-art linear programs solver. We evaluate the technique presented by discussing the experimental results obtained by analysing reachability properties for a number of benchmarks in the literature.

我们研究实现为前馈神经网络的系统的可达性问题，该系统的激活功能是通过ReLU函数实现的。 我们在确定神经系统是否可以输出任意输出与表征感兴趣的神经系统的线性问题之间建立了对应关系。 我们提出了一种通过最先进的线性程序求解器解决实际问题的方法。 我们通过讨论通过分析文献中许多基准的可达性特性而获得的实验结果来评估所提出的技术。

1 Introduction

Over the past ten years, there has been growing interest in trying to verify formally the correctness of AI systems. This has been compounded by recent public calls for the development of "responsible" and "verifiable AI" [27]. Indeed, since the development of ever more complex and pervasive AI systems including autonomous vehicles, the need for higher guarantees of correctness for the systems has intensified. Formal verification is one of the techniques used in other areas of Computer Science, including hardware and automatic flight control systems, to debug systems and certify their correctness. It is therefore expected that formal methods will contribute to provide guarantees that AI systems behave as intended.

在过去的十年中，人们对尝试正式验证AI系统的正确性越来越感兴趣。 最近公开呼吁开发“ 负责任的 ”和“ 可验证的人工智能 ”使这一问题更加复杂[27]。的确，由于开发了包括自动驾驶汽车在内的越来越复杂和普及的AI系统，因此对系统正确性的更高保证的需求日益增加。 形式验证是计算机科学其他领域（包括硬件和自动飞行控制系统）中用于调试系统并验证其正确性的技术之一。 因此，可以预期的是，形式化方法将有助于确保AI系统的行为符合预期。

In the area of multi-agent systems (MAS) there already has been considerable activity aimed at verifying MAS formally. In one line efficient model checkers for finite state MAS against expressive AI-based specifications, such as those based on epistemic logic, have been developed [12, 20, 7]. Abstraction techniques have also been put forward to verify infinite state MAS [19] and approaches for parameterised verification for MAS and swarms have been introduced [15, 14, 16]. In a different strand of work, theorem proving approaches have been tailored to MAS [1, 28], and techniques for the direct verification of MAS programs have been put forward [4].

While significant results have been achieved in these lines, their object of study is a system that is given either via a traditional programming language or a MAS-oriented programming language. It is however expected that machine learning technology will provide the backbone for a wide range of AI applications, including robotics, autonomous systems, and AI decision making systems. With the few exceptions discussed below, at present there is no methodology for the verification of systems based on neural networks. This paper aims to make a contribution on this topic.

尽管在这些方面已经取得了显著成果，但他们的研究对象是通过传统编程语言或面向MAS的编程语言提供的系统。 但是，预计机器学习技术将为各种AI应用程序（包括机器人技术，自治系统和AI决策系统）提供基础。 除了下面讨论的少数例外，目前还没有用于验证基于神经网络的系统的方法。 本文旨在对此主题做出贡献。

To begin this investigation, we consider feed-forward neural-networks (FFNNs) only, possibly with several layers, where the activation function is governed by ReLU functions [9, 24]. We consider specifications concerning safety only and, in particular, we study reachability. The method we present enables us to check whether a particular output, perhaps representing a bug, is ever produced by a given neural-network. While the target specifications are comparatively simpler than present research in MAS and reactive systems, reachability remains of paramount importance in program analysis as it enables the identification of simple errors.

为了开始这项研究，我们仅考虑前馈神经网络（FFNN），可能具有多层结构，其中激活函数由ReLU函数控制[9，24]。 我们仅考虑与安全性相关的规范，尤其是研究可达性。 我们介绍的方法使我们能够检查给定的神经网络是否曾经产生过特定的输出(可能代表错误)。尽管目标规范比MAS和反应系统中的当前研究相对简单，但可达性在程序分析中仍然是最重要的，因为它可以识别简单错误。

The rest of the paper is organised as follows. In Section 2, we fix the notation on ReLU-based FFNNs and mixed integer linear programs. In Section 3, we present an encoding of neural-networks

本文的其余部分安排如下。在第2节中，我们将介绍一些基于ReLU的FFNN和MILP上的符号标记。在第3节中，我们将根据线性规划对神经网络进行编码，并根据相应的线性规划问题将FFNN的可达性形式关联起来。在第4节中，我们将技术应用于文献中的极点平衡问题，并确定相应FFNN的安全特征。在第5节中，我们通过介绍在各种大小的FFNN上获得的实验结果来评估该方法的可伸缩性。在第6节中，我们将讨论进一步的工作以作为总结。

in terms of linear programs and formally relate reachability on FFNNs in terms of a corresponding linear programming problem. In Section 4, we apply the technique to a pole-balancing problem from the literature and identify safety features of the corresponding FFNNs. In Section 5, we evaluate the scalability of the approach by presenting experimental results obtained on FFNNs of various sizes. We conclude in Section 6 by discussing further work.

如上所述，AI系统验证中，过去和现在的很多研究都涉及对实际程序或传统的有限状态模型的分析，这些模型代表基于时间或基于AI的规范的AI系统。尽管目的与本文相同，但研究的对象本质上是不同的。相反，关于FFNN的许多文献都与训练和性能有关，而没有涉及形式化验证问题。当前用于检查网络的正确性的技术依赖于测试数据集，这是基于概率性的，这显然是不完备的。以下是我们意识到的神经网络形式验证的几个异常。

Related Work. As stated above, much of the past and present research on verification of AI systems concerns the analysis of actual programs or traditional finite-state models representing AI systems against temporal or AI-based specifications. While the aims are the same as those in this paper, the object of study is intrinsically different. In contrast, much of the literature on FFNNs is concerned with training and performance and does not address the formal verification question. Currently techniques used for checking the correctness of networks rely on test datasets which are probablistic and thus obviously incomplete. The few exceptions for formal verification of neural networks that we are aware of are the following.

[17] advocates the use of safety specifications to validate neural networks. The work here presented on reachability partially falls within the types 3 and 4 of safety which they discuss. While the broad direction of the work is in line with what pursued here, no actual verification method is discussed. A method for finding adversarial inputs for ReLU feed forward networks through the use of linear programming was proposed in [3]. However, the LP encoding proposed there is tailored to adversarial inputs and cannot be applied to reachability. Moreover, finding adversarial inputs can be thought of as a special case of reachability where the input set is constrained with respect to a specific input; thus the formulation here proposed is more general. Related to this, a method for finding adversarial inputs using a layer-by-layer approach and employing SMT solvers was recently proposed [11]. This technique supports any activation function, not just ReLU as we do here. However, because the focus is on adversarial inputs, as before, the method seems not immediately generilisable to solving reachability on feed-forward networks. A methodology for the analysis of ReLU feed-forward networks, conducted independently from this research, was made available on ArXiv some time before this submission [13]. While the aims seem largely in line as those presented here, there is no formal correspondence presented between reachability analysis and linear programs as we do here. Moreover, the underlying techniques proposed are different. While their method is based on SMT-solving, we only use linear programming here. Linear programming is used in [13] as a comparison against SMT, but there is no mention of any optimisation on the LP engine. In contrast, we here focus on an efficient LP translation and handle floating point operations in an optimised manner. Also the scenarios studied are different from ours and are not released for a comparison. Judging from the experimental results presented for the LP comparison, the LP technique used in [13] performs significantly less efficiently than what we develop here. For example, their analysis is only shown with up to 300 ReLU constraints whereas our technique is able to handle more than 500 with ease. An in-depth comparison of the performance of the two different techniques will not be possible until all data in [13] are released.

2 Preliminaries

In what follows we fix the notation on the key concepts to be used in the rest of the paper.

Feed-forward neural networks (FFNN) are the simplest class of neural networks [10]. Their main distinguishing feature is the lack of cycles. 没有循环

Definition 1 [Feed-Forward Neural Network] Let V be a set of vertices, 顶点 $E \subseteq V \times V$ be a set of edges, $w : E \to \mathbb{R}$ be a *edge weight function* and $b : V \to \mathbb{R}$ be a *bias function*. A weighted, directed, acyclic graph $N = (V, E, w, b)$ is a *feed-forward neural network* if the following properties hold:

1. For $j = 1 \dots n$ there exists an ordered collection 有序的容器 of ordered sets 有序集合 $L^{(j)} \subseteq V$ known as 称为层 *layers* such that $V = \biguplus_{j=1}^{n} L^{(j)}$ and for a vertex $v \in L^{(j)}$ $j < n$, the set of endpoints for edges originating from v is equal to $L^{(j+1)}$ and for a vertex $v \in L^{(n)}$, the set of endpoints from v is equal to the empty set.

2. Each $L^{(j)}, j \geq 2$ is associated with a function $\sigma^{(j)} : \mathbb{R} \to \mathbb{R}$ known as the *activation function*.

In this paper we consider only *ReLU* activation functions. This function has grown in popularity over the past few years in feed-forward networks, replacing the sigmoid and tanh functions; this is due to the improvement in convergence to the function being approximated when training the network. It is defined as $\text{ReLU}(x) = \max(x, 0)$.

在本文中，我们仅考虑ReLU激活函数。 在过去的几年中，此功能已在前馈网络中流行，取代了S型和tanh功能。 这是由于训练网络时对近似函数收敛的改进。

While the network itself is a weighted graph with bias represented as a function, we can define the traditional vector and matrix representation of bias and weights respectively.

虽然网络本身是一个以函数表示的带bias加权图，但我们可以分别定义偏差和权重的传统矢量和矩阵表示。

Definition 2 [Bias and weights of layer] For each layer $L^{(k)},\ k \geq 2$ of a neural network N, we define the *bias vector* to be the vector $b^{(k)} \in R^m$ with $m = |L^{(k)}|$ defined elementwise as: $b_i^{(k)} = b(L_i^{(k)})$. We further define the *weight matrix* to be the matrix $W^{(k)} \in R^{m \times n}$ with $m = |L^{(k)}|$ and $n = |L^{(k-1)}|$ defined elementwise as: $W_{ij}^{(k)} = w(L_j^{(k-1)}, L_i^{(k)})$.

The main use of neural networks is as universal function approximators. For the purposes of this paper, we define the function computed by a network.

Definition 3 [Function computed by network] Let N be a neural network and $2 \leq i \leq k$ with k the number of layers in N. Then, layer $L^{(i)}$ defines a function $f^{(i)} : \mathbb{R}^m \to \mathbb{R}^n$ known as a *computed function* where $m = |L^{(i-1)}|$, $n = |L^{(i)}|$ and defined as $f^{(i)}(x) = \sigma^{(i)}(W^{(i)}x + b^{(i)})$ Further, N defines a function $f : \mathbb{R}^m \to \mathbb{R}^n$ known as the *computed function* where $m = |L^{(1)}|$, $n = |L^{(k)}|$ and defined as: $f(x) = f^{(k)}(f^{(k-1)}(...(f^{(2)}(x))))$

Linear Programming is an optimisation technique where the objective function and constraints are linear. Efficient algorithms exist to solve linear programming problems efficiently [30]. For the purposes of this paper consider mixed integer linear programs; these are linear programs which contain both real and integer variables.

线性规划是一种优化技术，其中目标函数和约束是线性的。存在有效的算法来有效地解决线性规划问题[30]。为了本文的目的，考虑混合整数线性规划，它是同时包含实数和整数变量的线性规划。

Definition 4 [Mixed Integer Linear Programs] A function $f(x_1, ..., x_n)$ is said to be *linear* if for some $c \in \mathbb{R}^N$, we have $f(x_1, ..., x_n) = \sum_{i=1}^{N} c_i x_i$.

For any linear function $f(x_1, ..., x_n)$, and any $b \in \mathbb{R}$, $f(x_1, ..., x_n) = b$, $f(x_1, ..., x_n) \leq b$ and $f(x_1, ..., x_n) \geq b$ are said to be *linear constraints*. 对于f，f=b,f<b,f>b都称为线性约束

线性规划（LP）是一个数学优化问题，其中目标函数是线性的，并且对目标函数变量的约束是线性的。

A *linear program* (LP) is a mathematical optimisation problem where the the objective function is linear and the constraints on the variables of the objective function are linear.

混合整数线性规划是一种线性程序，它允许要求变量为整数的约束（即xi∈Z类型的约束）。

A *mixed integer linear program* is a linear program which allows for constraints which require variables to be integer, i.e., constraints of the type $x_i \in \mathbb{Z}$.

In this paper we use *Gurobi* linear programming solver [8]. Gurobi has good performance and can be used on a wide range of problems [22].

3 Verifying Reachability for FFNN

In this paper we focus on reachability analysis, a particular aspect of safety analysis.

在本文中，我们专注于可达性分析，这是安全性分析的特定方面。

In general terms reachability analysis consists on finding whether a certain state (or set of states) of a system can be reached given a fixed set of initial states of the system. Reachability analysis is commonly used to identify bugs in software systems, e.g., whether mutual exclusion is enforced in concurrent applications [21].

互斥

一般而言，可到达性分析包括查找在给定的一组固定初始状态下是否可以达到系统的某个状态（或一组状态）。 可达性分析通常用于识别软件系统中的错误，例如在并发应用程序中是否强制执行互斥[21]。

To apply this concept to neural networks, we treat our fixed set of initial states to be a fixed set of *input vectors* and we attempt to find out whether any vector in a set of *output vectors* can be computed by the network from a vector in the input set.

为了将此概念应用到神经网络，我们将确定的初始状态集视为确定的输入向量集，并尝试找出是否可以从输入集合中的某一个向量，由神经网络计算出，输出向量中的任何向量。

Definition 5 [Reachability for FFNN.] Suppose N is an FFNN with computing function $f : \mathbb{R}^m \to \mathbb{R}^n$ with $m = |L^{(1)}|$, $n = |L^{(k)}|$, where k is the number of layers in N.

Let $I \subseteq \mathbb{R}^m$ and $O \subseteq \mathbb{R}^n$. We say that O is *reachable from* I using N if $\exists y \in O, \exists x \in I, f(x) = y$.

如果存在x属于I，存在y属于O，使得f(x)=y,我们就称O在N的作用下是由I可达的。

虽然定义5给出了一般情况，但在这里我们关注于通过Rn上的有限数量的线性约束表示的输入和输出集。

While Definition 5 presents the general case, we here focus on input and output sets that are representable via a finite number of linear constraints on $\mathbb{R}^n$. Observe this still enables us to capture a

请注意，这仍然使我们能够捕获大量的系统，因为所有的线性等式和不等式都是允许的，例如，在[13]中对ACAS网络进行的验证使用了可线性定义的输入和输出集。

large number of systems since all linear equalities and disequalities are allowed e.g. verification of ACAS networks performed on in [13] uses linearly definable input and output sets.

Definition 6 [Linearly Definable Set.] Let $S \subseteq \mathbb{R}^n$. We say that S is linearly defineable if there exists a finite set of linear constraints C_S such that $S = \{x \in \mathbb{R}^n \mid x \text{ satifies every constraint in } C_S\}$. We define C_S to be the constraint set of S.

我们现在证明，对于一个具有ReLU激活函数的神经网络，建立可达性可以被改写为求解一个相应的线性规划问题，该线性规划由所讨论的神经网络的线性编码产生。

We now show that estabilishing reachability for a neural network with ReLU activation functions can be rephrased into solving a corresponding linear program resulting from the linear encoding of the neural network in question.

尽管过去已经提出了针对单个神经元的非形式化线性编码[13]，但我们在这里介绍的是一种采用逐层方法进行操作的形式化编码。 此外，它仅使用单个二进制变量； 正如我们稍后演示的那样，这对于实际应用中的效率至关重要。

While informal linear encodings for individual neurons have been proposed in the past [13], the one we present here is a formal one which operates on a layer by layer approach. Moreover, it only utilises a single binary variable; as we demonstrate later, this is important for efficiency in practical applications.

Definition 7 [Linear Encoding for FFNN.] Let N be an FFNN and $2 \leq i \leq k$ with k the number of layers in N. Suppose further $x^{(i-1)}$ and $x^{(i)}$ are vectors of real (LP) variables representing the input and output of layer i respectively and $\delta^{(i)}$ a vector of binary (LP) variables. Then, the set of *linear constraints encoding layer* i, (with a ReLU activation function) is defined as:

$$C_i = \{x_j^{(i)} \geq W_j^{(i)} x^{(i-1)} + b_j^{(i)}, x_j^{(i)} \leq W_j^{(i)} x^{(i-1)} + b_j^{(i)} + M\delta_j^{(i)},$$
$$x_j^{(i)} \geq 0, x_j^{(i)} \leq M(1 - \delta_j^{(i)}) \mid j = 1 \ldots |L^{(i)}|\}$$

where M a "sufficiently large" constant.

We can define the set of *linear constraints encoding the network* as $C = \cup_{i=2}^{k} C_i$.

By means of this encoding, we can reduce reachability analysis to solving a linear program defined on these constraints.

通过这种编码，我们可以减少可达性分析，以解决基于这些约束定义的线性程序。

Definition 8 [LP encoding reachability for FFNN.] Let N be an FFNN, C its encoding as per Definition 7, and $I \subseteq \mathbb{R}^m$ (respectively, $O \subseteq \mathbb{R}^n$) be a set of linearly definable network inputs (outputs, respectively).

The *linear program encoding the reachability* of O from I through N is given by the objective function $z = 0$ and constraints $C_{reach} = C_{in} \cup C \cup C_{out}$, with the sign of the variables unconstrained, where

- C_{in} is a constraint set for I defined on the same variables used in the encoding of the input to the second layer of N (i.e. $x_j^{(1)}$ for $j = 1 \ldots m$), and
- C_{out} is a constraint set for O defined on the same variables used in the encoding of the output of the last layer of N (i.e. $x_j^{(k)}$ for $j = 1 \ldots n$).

Theorem 1 [Equivalence between reachability analysis and corresponding LP problems.] Suppose N is an FFNN on linearly definable input I and output O. Let L be the corresponding linear problem encoding the reachability of O from I through N (Definition 8).

Then, O is reachable from I through N if and only if the linear program L has a feasible solution $\mathbf{x}$.

Proof sketch. $\Longrightarrow$ We have $\exists x \in I, \exists y \in O$ with $f(x) = y$. Then, for the assignment of LP variables $x_j^{(1)} \to x_j$ and $x_j^{(i)} \to f^{(i)}(\ldots(f^{(2)}(x_j)))$, we have that the LP is feasible.

$\Longleftarrow$ We have $\mathbf{x}$ is a feasible solution for L. Let $x_j = \mathbf{x}_j^{(1)}$ and $y_j = \mathbf{x}_j^{(k)}$. Then, we have $x \in I, y \in O$ and $y = f(x)$ by definition of the LP.

4 Verifying a Neural Controller for the Inverted Cart Problem

We now use the methodology developed in the previous section to verify several reachability specifications on the well-studied inverted pendulum controller problem [2].

Inverted pendulum on cart problem (IPCP). The system is composed by a cart moving along a frictionless track with bounds at either ends of the track. Attached to the centre of the cart through the use of a frictionless and unactuated joint is a pole. The pole acts as an inverted pendulum.

The problem can be expressed in control terms by using two state variables and their derivatives [2].

- Position of the cart on the track, denoted by x and bounded by ± 2.4.
- Speed of the cart, denoted by $\dot{x}$ and unbounded.
- Angle of the pendulum (counter-clockwise), denoted by θ and bounded by $\pm 15°$.
- Angular velocity of the pendulum (counter-clockwise), denoted by $\dot{\theta}$ and unbounded.

The output of the controller at every discrete time step is a signal for the application of a force of $+10N$ or $-10N$ (where the direction is aligned with x). Intuitively, the aim of the controller is to balance the pendulum on the cart for as long as possible while, at the same time, remaining both in the bounds of the track and in the bounds of the angle of the pendulum.

A neural network controller for the IPCP. Due to its inherent non-linearity, the IPCP traditional controller methods cannot be used to solve the problem. Reinforcement learning methods can be used to derive a neural network that can be used as a controller for the system. To create a neural controller we used the deep learning library KERAS [5] combined with the Q-learning library KERAS-RL [25].

The precise details of the resulting network and the training data are not relevant for what follows (the supplementary material reports the details). After training the resulting deep, feed-forward neural network obtained can be described as follows:

- The input layer consists of 4 input nodes, one for each of the variables of the system.
- The resulting three hidden layers consist of 16 nodes; each layer uses the ReLU activation function.
- The output layer consists of 2 nodes denoting the "q-value" of moving left and right respectively. The higher the q-value, the higher is the predicted reward for the action. The output layer does not use a ReLU function as is standard for most networks.

Reachability via Linear Programming. Having derived a neural-network controller for the IPCP we now proceed to analyse it in terms of reachability properties. While the theoretical encoding of the problem is presented in Section 3, to solve the resulting problem via an automatic solver, we need to address the resulting issues in terms of floating point approximations.

在推导出IPCP的神经网络控制器后，我们现在着手分析它的可到达性特性。虽然在第3节中介绍了问题的理论编码，但为了通过自动求解器来解决所产生的问题，我们需要用浮点近似来解决所产生的问题。

Floating point arithmetic. For the encoding to be correct, the constraints present in the resulting LP must take into account a safe level of floating point precision and use tolerances when defining the links between the layers. Not doing so may render the analysis to be unsound. A system may be assessed to be safe (i.e., unwanted regions of the output may be shown to be unreachable); but this could be the result of by approximations (roundings or truncations) due to the underlying floating point arithmetic.

浮点运算的问题。为了使编码正确，结果LP中存在的约束条件必须考虑到安全的浮点精度水平，并在定义各层之间的联系时使用容差。如果不这样做，可能会使分析不健全。一个系统可能被评估为安全的（即，输出的不需要的区域可能被显示为无法到达）；但这可能是由于底层浮点算术的近似（四舍五入或截断）的结果。

To address this issue we use we use "epsilon" terms when encoding the network. These are terms used to link the layers of the network to allow for small perturbations when invoking the linear program solver. In combination with this, we change the objective function to minimise the sum of these epsilon terms, instead of simply using the constant 0.

为了解决这个问题，我们在对网络进行编码时使用了 "epsilon"术语。这些术语用于连接网络的各层，以便在调用线性程序求解器时允许小的扰动。结合这一点，我们改变目标函数，使这些epsilon项的总和最小化，而不是简单地使用常数0。

Formally, for each layer, the constraint set changes as follows:

$$\begin{aligned} C_i = \{ & x_j^{(i)} \geq W_j^{(i)} x^{(i-1)} + b_j^{(i)} - \epsilon_j^{(i)}, x_j^{(i)} \leq W_j^{(i)} x^{(i-1)} + b_j^{(i)} + M\delta_j^{(i)} + \epsilon_j^{(i)}, \\ & x_j^{(i)} \geq 0, x_j^{(i)} \leq M(1 - \delta_j^{(i)}) \mid j = 1...|L^{(i)}| \} \end{aligned}$$

where ϵ_j are non-negative variables. Correspondingly, when encoding a neural network as linear program, we change the objective function to be $z = \sum_{i=2}^{k} \sum_{j=1}^{|L^{(i)}|} \epsilon_j^{(i)}$, which we then aim to minimise.

This entails that we aim to find an exact solution if possible but, if one exists with a small epsilon sum, we can still accept it if it is within the tolerance of the underlying floating point arithmetic. We

这意味着我们的目标是尽可能找到一个精确的解，但如果存在一个小的epsilon和，我们仍然可以接受它，如果它是在底层浮点算术的容差范围内。我们通过再增加一个形式为K的约束条件来实现，其中t是容差项。

do this by adding a further constraint of the form $\sum_{i=2}^{k} \sum_{j=1}^{|L^{(i)}|} \epsilon_j^{(i)} \leq t$, where t is the tolerance term.

In practice, for current computers we can take this to be $1e^{-6}$ which is one order of magnitude larger than the machine epsilon for 32-bit floating point numbers. We adopted this value in our experiments. However, when binary inputs are used, a larger tolerance value may be required because of the techniques used by solvers. We adopted a value of $1e^{-4}$ for these problems.

在实践中，对于当前的计算机，我们可以将其取为1e-6，这比32位浮点数的机器epsilon大一个数量级。我们在实验中采用了这个值。但是，当使用二进制输入时，由于求解器使用的技术，可能需要更大的容差值。我们对这些问题采用了1e-4的值。

Reachability specifications and results. In view of the encoding discussed above we can now proceed to verify the behaviour of the neural-network trained for the IPCP. We consider the following specifications where in each case S is a tupe of form $(x, \dot{x}, \theta, \dot{\theta})$.

1. Is it ever the case that $Q(S, 10) \not\geq Q(S, -10) + 100$ where $S = (0, 0, -5, 0)$? Intuitively, this says that force of $10N$ has a Q-reward of at least 100 units greater than $-10N$ for the given state S. This expresses the fact the controller attempts (in the strongest possible sense within the bounds of the problem) to move the cart to the right when the pendulum is leaning to the right and all other factors are unimportant.

2. Is it ever the case that $Q(S, 10) \not\geq Q(S, -10) + 100$ where $S \in \{(x, \dot{x}, \theta, \dot{\theta}) \mid |x| \leq 0.5, |\dot{x}| \leq 0.2, -5 \leq \theta \leq -4, |\dot{\theta}| \leq 0.1\}$? This is the same specification as above but to be checked on a larger states of configurations.

3. Is it ever the case that $Q(S, 10) \not\geq Q(S, -10) + 10$ where $S \in \{(x, \dot{x}, \theta, \dot{\theta}) \mid x \leq -2, |\dot{x}| \leq 0.2, -2 \leq \theta \leq -1.5, |\dot{\theta}| \leq 0.25\}$? This represents the fact that the controller attempts to move the cart to the right when the pendulum is not at risk from falling over but the cart is almost out of left hand side track bound.

4. Is it ever the case that $Q(S, 10) \not\geq Q(S, -10) + 10$ where $S \in \{(x, \dot{x}, \theta, \dot{\theta}) \mid x \leq -2, |\dot{x}| \leq 0.4, -2 \leq \theta \leq 1, 0 \leq \dot{\theta} \leq 0.1\}$? This is a relaxation of the above specification to analyse a larger set of configurations.

To analyse the specifications above we compiled the network for the IPCP into an LP as described above. We then used Gurobi [8] to solve the corresponding LP problems. A solution found by Gurobi corresponds to the fact that there is a configuration can be found solving the problem. In this case, the specification can be met by the system for the values found. If a solution cannot be found, since by Theorem 1 the method is complete, we conclude that no input in the space analysed can produce the output checked.

为了分析上述规范，我们将IPCP的网络编译成如上所述的LP。然后我们使用Gurobi[8]来解决相应的LP问题。由Gurobi找到的解对应的是，可以找到一个配置来解决这个问题。在这种情况下，对于找到的值，系统可以满足规范。如果不能找到解，因为由定理1，该方法是完整的，我们的结论是，在所分析的空间中，没有任何输入可以产生所检查的输出。

Doing as above we promptly obtained the following results (full benchmarks are reported in the next section).

1. No solution could be found satisfying the constraints on the input and output of the network. We conclude that our synthesised controller does strongly prefer to apply $10N$ to balance the pendulum in those circumstances.

2. As above no solution could be found. Again, we conclude that in all the region explored the behaviour of the synthesised controller is correct.

3. Again, no solution could be found satisfying both the inputs and the outputs. This indicates that the controller attempts to return the cart to the centre of the track when it is one side of the track within the range of parameters above.

4. The solver reported the solution $x = (-2.0, -0.4, -0.15, 0.1)$ (approximation shown) for the problem above. This shows that the controller applies the force in what is, intuitively, the incorrect direction when the configuration of the system is as above. Note also that in this situation the angle of the pendulum would also suggest an application of the force in the positive direction.

The analysis conducted above shows that the synthesised controller does not satisfy our specifications as put forward. The values found by the solver can be fed to the neural network to confirm the result. We have effectively found a “bug” in the synthesised controller by using a formal encoding into an LP problem.

上面的分析表明，合成控制器不符合我们提出的规格。 求解器找到的值可以输入到神经网络以确认结果。 通过使用形式化编码到LP问题中，我们已经在合成控制器中有效地发现了一个“ 错误 ”。

We stress the importance of this result which enabled us to find an error in the resulting network in under 1s. Comparable techniques, e.g., testing are in incomplete and may take considerably longer to identify the need for further training.

我们强调此结果的重要性，这使我们能够在不到1秒的时间内发现结果网络中的错误。 可比较的技术（例如测试）尚不完善，可能需要更长的时间才能确定是否需要进一步培训。

5 Experimental Results

We now report on the experimental results obtained by using the technique presented in the previous sections on a number of feed-forward networks. The experiments were run on an Intel Core i7-4790 CPU (3.600GHz, 8 cores) running Linux kernel 4.4 upon which we invoked Gurobi version 7.0. The benchmarks are shown only to evaluate the scalability of the approach, not as validation for the corresponding problems. Indeed, for some of the problems below (Reuters and MNIST), reachability analysis is not applicable. Moreover, since, as discussed in the Introduction, no other approach exists for reachability analysis, we are unable to compare our results to other techniques.

More details on all of these experiments as well as the networks and the code used to perform verification can be found in the supplementary material of this paper.

Problem	Layer Sizes	Vars (Continuous, Binary)	Time (s)
Inv. Pen. 1	4, 16, 16, 16, 2	108, 31	>0.01
Inv. Pen. 2		140, 39	0.02
Inv. Pen. 3		142, 41	0.03
Inv. Pen. 4		143, 42	0.04
Mountain Car 1	2, 50, 190, 3	406, 117	0.06
Mountain Car 2		404, 115	0.04
Mountain Car 3		404, 117	0.02
Mountain Car 4		407, 118	0.04
Pendulum 1	3, 32, 32, 32, 1	154, 41	0.05
Pendulum 2		154, 41	0.02
Pendulum 3		154, 41	0.02
Pendulum 4		161, 48	0.65
Acrobot 1	6, 64, 168, 3	579, 162	0.48 (3 problems)
Acrobot 2		569, 152	0.66 (3 problems)
Acrobot 3		590, 173	3.31
Acrobot 4		609, 192	47.28
Reuters	1000, 512, 46	1546, 526	40.30
MNIST	4608, 128, 10	5002, 128	8.54

Table 1: Experimental results for the three neural networks described. Vars column refers to the number of variables in the LP: both continuous and binary.

Inverted pendulum. The neural network's description for this scenario, including its size and architecture, is described in Section 4. As reported in Table 5, the method here proposed solved each reachability query in less than 1 sec. We checked several other reachability specifications, here not reported, to evaluate the performance degradation as a function of the specification. We could not find specifications that could not be solved in under 1 sec. We conclude that the methodology presented can be used to evaluate any reachability problem for the IPCP controller as here synthesised.

Acrobot, Pendulum and Mountain Car. These are well-known problems from classic control theory [29, 6, 23]. As with the inverted pendulum problem, we trained agent networks which solve these problems and evaluated specifications using our toolkit. For the acrobot and pendulum problems, the networks rely on non-linear trigonometric functions; we generate a piece-wise linear approximation for these.

The specifications verified in the Pendulum and Mountain Car benchmarks but with networks which are both larger and have different layer sizes.

Similarly to the case above, the time taken to perform verification was less than 1 sec. Differently from the pendulum case, the specifications verified for the acrobot problem are much more complex (mirroring the problem itself); consequently the specifications took up to 45 sec to be verified. The third and fourth specifications for acrobot are especially demanding as they require a full search of a large part of a relatively high dimension state space to find a solution. Taking this into account, we believe the performance to be more than acceptable.

Reuters text classification. This neural network is intended to solve the problem of classifying articles from their content [26]. The network is composed of a binary input layer followed by a hidden layer with a ReLU activation function followed by the output layer. The structure of network is thus rather shallow, but contains a large number of input and hidden neurons.

To evaluate the performance of the approach, we fixed all but 50 of the inputs to known values and attempted to carry out reachability analysis on the remaining inputs. We also neglected the use of the softmax function to reduce the complexity resulting from its use.

We were able to replicate the existence of an input for a (pre-softmax) output of the network for which we knew that a binary input existed. We were able to solve the LP problem and find the corresponding values in just under 45 secs. As above, considering the size of the hidden layers and the number of binary variables present in the corresponding problem, we find the performance to be attractive.

MNIST Image Recognition. This neural network was put forward to perform image recognition on the MNSIT handwritten numeric digits dataset [18]. The network is composed of a convolutional part with several filters and a max-pooling layer followed by a hidden feed-forward layer and an output layer.

We only considered the hidden and output layers of the network as these are the feed-forward portion of the network. As in the previous example the size of these layers is between 10^2 and 10^3 nodes. As in the previous experiment we attempted to find an input for some output for which we know an input exists. As before, we did not use softmax function when encoding the output.

The toolkit was able to find the input in just over 2 secs. Given the size of the network and the number of variables in the corresponding problem, we again evaluate the performance positively. Comparing the last two scenarios, we conjecture that the large increase of binary variables in the problem caused by the binary constraints on the input creates the large performance gap between the Reuters dataset and MNIST.

In summary the results suggest that the methodology developed, when paired with the optimisations here studied, can solve the reachability problem for several neural networks of interest. In particular we were able to solve reachability analysis for deep nets of 3 layers of significant size. The experiments demonstrate that performance depends on a number of factors the most important being the size of the state space searched, the number of variables (especially binary variables), and the number of constraints.

6 Conclusions

In this paper we have observed that while there is increasing awareness that future society critical AI systems will need to be verifiable, all of the major verification techniques fail to target neural networks. Yet, neural networks are presently forecast to drive most future AI systems. We have attempted to begin to fill this gap by providing a methodology for studying reachability analysis in feed forward neural networks. Specifically, we have drawn a formal correspondence between reachability in a neural network and an associated linear programming problem. We have presented how to circumvent problems caused by floating point arithmetic and optimise the corresponding linear programming problem. The experimental results shown demonstrate that the method can solve reachability for networks of significant size.

Much work remains to do in this direction. We intend to study recurrent networks and develop alternative techniques to solve the reachability problem in recurrent networks. Also we intend to apply the results of this work to synthesised controllers in engineering.

在本文中，我们观察到，虽然人们越来越意识到未来社会关键的人工智能系统需要可验证，但所有主要的验证技术都没有针对神经网络。然而，目前预测神经网络将驱动大多数未来的人工智能系统。我们试图通过提供一种研究前馈神经网络中可到达性分析的方法来开始填补这一空白。具体来说，我们已经在神经网络中的可达性和相关的线性编程问题之间建立了形式上的对应关系。我们介绍了如何规避浮点运算引起的问题，并优化相应的线性编程问题。所展示的实验结果表明，该方法可以解决相当规模的网络的可达性问题。

在这个方向上还有很多工作要做。我们打算研究循环网络，开发替代技术来解决循环网络中的可达性问题。同时我们打算将这项工作的结果应用于工程中的综合控制器。

References

[1] N. Alechina, M. Dastani, F. Khan, B .Logan, and J-J. Meyer. Using theorem proving to verify properties of agent programs. In *Specification and verification of multi-agent systems*, pages 1–33. Springer, 2010.

[2] A. Barto, R. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[3] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi. Measuring neural net robustness with constraints. *CoRR*, abs/1605.07262, 2016. ILP

[4] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying multi-agent programs by model checking. *Autonomous Agents and Multi-Agent Systems*, 12(2):239–256, 2006.

[5] C. François. Keras, 2015. URL https://github.com/fchollet/keras. https://github.com/fchollet/keras.

[6] K. Furuta, M. Yamakita, and S. Kobayashi. Swing up control of inverted pendulum. In *Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON'91., 1991 International Conference on*, pages 2193–2198. IEEE.

[7] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *Proceedings of 16th International Conference on Computer Aided Verification (CAV04)*, volume 3114 of *Lecture Notes in Computer Science*, pages 479–483. Springer, 2004.

[8] Inc. Gurobi Optimization. Gurobi optimizer reference manual. http://www.gurobi.com, 2016.

[9] S. Haykin. *Neural Networks and Learning Machines*. Pearson Education, 2011. ISBN 9780133002553.

[10] S.S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999. ISBN 9780139083853.

[11] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. *CoRR*, abs/1610.06940, 2016.

[12] M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Woźna, and A. Zbrzezny. Verics 2007 - a model checker for knowledge and real-time. *Fundamenta Informaticae*, 85(1):313–328, 2008.

[13] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. *ArXiv e-prints*, February 2017.

[14] P. Kouvaros and A. Lomuscio. Verifying emergent properties of swarms. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI15)*, pages 1083–1089. AAAI Press, 2015.

[15] P. Kouvaros and A. Lomuscio. Parameterised verification for multi-agent systems. *Artificial Intelligence*, 234:152–189, 2016.

[16] P. Kouvaros and A. Lomuscio. Formal verification of opinion formation in swarms. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS16)*, pages 1200–1209. IFAAMAS, 2016.

[17] Z. Kurd and T. Kelly. Establishing safety criteria for artificial neural networks. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 163–169. Springer, 2003.

[18] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

[19] A. Lomuscio and J. Michaliszyn. Verification of multi-agent systems via predicate abstraction against ATLK specifications. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS16)*, 2016.

[20] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. *Software Tools for Technology Transfer*, 2015. doi: 10.1007/s10009-015-0378-x. URL http://dx.doi.org/10.1007/s10009-015-0378-x. http://dx.doi.org/10.1007/s10009-015-0378-x.

[21] J. Magee and J. Kramer. *Concurrency: state models & Java programs*. Worldwide series in computer science. Wiley, 2006. ISBN 9780470093559. URL https://books.google.co.uk/books?id=CpJQAAAAMAAJ.

[22] H. Mittelmann. Benchmarks for optimization software, 2016. URL http://plato.asu.edu/bench.html. http://plato.asu.edu/bench.html.

[23] A. Moore. Efficient memory-based learning for robot control. Technical Report UCAM-CL-TR-209, University of Cambridge, Computer Laboratory, November 1990. URL `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-209.pdf`.

[24] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[25] M. Plappert. keras-rl. https://github.com/matthiasplappert/keras-rl, 2016.

[26] Reuters. Reuters-21578 dataset, 1987. URL `http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`.

[27] S. J. Russell, D. Dewey, and M. Tegmark. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, 36(4), 2015.

[28] S. Shapiro. *Specifying and verifying multiagent systems using the cognitive agents specification language (CASL)*. University of Toronto, 2005.

[29] R. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press, 1996.

[30] W. Winston. *Operations research: applications and algorithms*. Number v. 1. Duxbury Press, 1987.