3-LAYER NEURAL NETWORKS ARE UNIVERSAL APPROXIMATORS FOR FUNCTIONALS AND FOR CONTROL STRATEGIES

Vladik Kreinovich\textsuperscript{1} and Ongard Sirisaengtaksin\textsuperscript{1,2}

\textsuperscript{1}Computer Science Department
University of Texas at El Paso, El Paso, TX 79968
\textsuperscript{2}Department of Computer and Mathematical Sciences
University of Houston-Downtown, Houston, TX 77002

Abstract
It has been proved that 3-layer neural networks can approximate any continuous function with any given accuracy (Hecht-Nielsen, Cybenko, Funahashi, Hornik, Stinchcombe, White). These theorems mean, in particular, that for a plant whose state can be described by finitely many parameters, for an arbitrary control \( u = f(x_1, \ldots, x_n) \), and for an arbitrary accuracy \( \varepsilon > 0 \), we can implement this control with a given accuracy using 3-layer neural networks (consisting of ideal neurons).

Recently, it has also been shown that we can approximate an arbitrary function with a neural network that consists of realistic hardware neurons. Namely, it is possible for an arbitrary continuous function \( f \), and a real number \( \varepsilon > 0 \), to produce a design (scheme) of a neural network and the necessary implementation accuracy \( \delta \) in such a way that even if all the neurons are built with this accuracy \( \delta \), the in-out characteristics of the resulting network is \( \varepsilon \)-close to \( f \).

In the present paper, we prove two generalizations of these results:

- First, we prove that a 3-layer neural network can also approximate the control of distributed systems (i.e., plants whose state requires infinitely many parameters to describe).

- Second, we prove that not only a 3-layer neural network can approximate a control of a specific plant, but one can design a universal neural network controller, that inputs a description of a plant, the desired objective, and generates an appropriate control.

In mathematical terms, we prove that 3-layer networks can approximate arbitrary continuous functionals.
1. INTRODUCTION

1.1. Neural networks are often used for control design
For non-linear systems, it is often very difficult to find an explicit analytical expression for a control that satisfies certain given goals. It is even more difficult to design a control that is optimal in some reasonable sense (e.g., uses the minimal amount of fuel, attains its goals in the shortest time, etc). One of the methods that (in many cases) helps to design such a control is to train a neural network in such a way that for a given input $\vec{x}$, its output is close to the ideal control value $u$.

1.2. What are the limits of this methodology?
Sometimes this method helps, sometimes it does not. If after, say, 3000 iterations the network is still not appropriately trained, does it mean that it cannot be trained in principle, or that we were not sufficiently patient (and after more iterations, we would have got the desired control)?

1.3. Neural networks are universal approximators
An answer to this question was given by several authors who proved that 3-layer neural networks can approximate any continuous function $f(\vec{x})$ with any given accuracy (Hecht-Nielsen, 1987; Cybenko, 1989; Funahashi, 1989; Hornik et al 1989; Hecht-Nielsen, 1990). These results are extremely valuable for control: they show that for a plant whose state can be described by finitely many parameters, for an arbitrary control $u(\vec{x})$, and for an arbitrary accuracy $\varepsilon > 0$, we can implement this control with a given accuracy using 3-layer neural networks. In other words, these results mean that, in principle, an arbitrary control can be obtained by using a 3-layer neural network.

This result covers many important control problems, but still it does not cover all of them. Namely, the following control problems are not covered by this classical result:

- First, it does not cover the control of the distributed systems (i.e., plants that require infinitely many parameters to describe their current state). Can we apply a 3-layer neural network to control them?

- Second, the classical result show that any given control of a given plant can be implemented as a neural network controller. But the goals of control theory are usually more ambitious that controlling one plant: the main goal is to design universal control methods, so that we would be able to input the description of a plant, the desired objective, and generate an appropriate control. Classical approximation theorem shows that for each plant, the resulting control can be implemented by a neural network. But can we approximate the general method? I.e., can we design a universal neural controller, that would transform the description of a plant and an objective into an actual control strategy?

1.4. Known neural approximation results are based on idealized neurons. Actual hardware neurons are not precise
We said that an arbitrary control can be “in principle” obtained by using a 3-layer neural network, because these results are based on the assumption that we can design neurons with precisely known in-out characteristics $s(x)$. In reality, however, it is
technically impossible to design a hardware device whose in-out characteristics \( \tilde{s}(x) \) precisely coincides with a prescribed function \( s(x) \). What we can guarantee is as follows: we can fix an accuracy \( \delta > 0 \), and the biggest possible signal \( X \), and guarantee that the in-out characteristics \( \tilde{s}(x) \) of a manufactured (hardware) neuron belongs to an interval \( [s(x) - \delta, s(x) + \delta] \) for all \( x \) from \(-X\) to \( X\).

1.5. Actual hardware neurons are universal approximators

With actual hardware neurons (Sirisaengtaksin, 1993; Sirisaengtaksin & Kreinovich, 1993), it is possible for an arbitrary continuous function \( f \), and a real number \( \varepsilon > 0 \), to produce a design (scheme) of a neural network and the necessary accuracy \( \delta \) in such a way that if we build all the neurons with this accuracy \( \delta \), the in-out characteristics of the resulting network is \( \varepsilon \)–close to \( f \).

In the present paper, we analyze both problems, and give positive answers to both questions: yes, neural networks can approximate an arbitrary control of a distributed system, and an arbitrary universal control strategy. In purely mathematical terms, our results state that 3-layer neural networks can approximate arbitrary continuous functionals. Furthermore, these approximations can be extended to the case of actual hardware neurons, i.e., the approximation properties will still be there if a 3-layer neural network is built with actual hardware neuron whose in-out characteristics slightly different from the desired mathematical function.

1.6. The structure of the paper

The structure of the paper is as follows: in Section 2, we present the above-mentioned classical result. In Section 3, we explain its importance for control theory: namely, it shows that an arbitrary “realistic” control can be approximated by a neural network with an arbitrary accuracy. In Section 4, we show that this approximation result is true if we consider networks of hardware neurons (for which the parameters of the neurons and their connections can be guaranteed only with some accuracy). In Section 5, we describe the control problems that are not directly covered by these results. In Sections 6 through 8, we show that neural networks can be used for these problems as well. In Section 9, the same is shown for neural networks that consist of hardware neurons.

1.7. A remark about the proofs

For reader’s convenience, all the proofs are given in Section 10. These results follow from approximation theorem proved in (Hornik et al, 1989) and other above-cited papers. For a trained mathematician, their proofs are straightforward and reasonably routine. However, for the convenience of the readers who are interested in control and/or neural network aspects of these results, and who are not professional mathematicians, we decided to describe these proofs in all necessary details.

This paper subsumes a technical report (Kreinovich & Sirisaengtaksin, 1992).

2. CLASSICAL RESULT: NEURAL NETWORKS ARE UNIVERSAL APPROXIMATORS FOR FUNCTIONS

Let us first recall the classical result that neurons are universal approximators for functions.

**Definition 1.** Suppose that a monotonic continuous function \( s(x) : R \to (0,1) \) is given such that

\[
\lim_{x \to -\infty} s(x) = 0, \quad \lim_{x \to \infty} s(x) = 1.
\]
We say that a function \( f(x_1, ..., x_n) : \mathbb{R}^n \rightarrow \mathbb{R}^p \) is representable by a 3-layer neural network if

\[
    f_l(x_1, ..., x_n) = \sum_{k=1}^{K} \beta_{kl} s\left(\sum_{i=1}^{n} w_{ki} x_i + b_k\right)
\]

for some integer \( K \), and real numbers \( \beta_{kl}, w_{ki}, \) and \( b_k \), where \( 1 \leq k \leq K \), \( 1 \leq i \leq n \), and \( 1 \leq l \leq p \).

**Comment.**

- This definition describes a network with 3 layers: an input layer, a hidden layer, and an output layer. The input layer consists of \( n \) neurons that read \( n \) input values \( x_1, ..., x_n \). The hidden layer consists of \( K \) neurons that input \( x_i \) and generate the signals \( y_k = s(\sum_{i=1}^{n} w_{ki} x_i + b_k) \), \( 1 \leq k \leq K \). Finally, the output layer consists of \( p \) linear neurons that combine the signals \( y_k \) into the outputs \( \sum_{k} \beta_{kl} y_k \).

- In Definition 1, parameters \( \beta_{kl}, w_{ki}, \) and \( b_k \), can be arbitrary real numbers (of any sign). This definition describes a model of a neuron that is most widely used in artificial neural networks (Hecht-Nielsen, 1990). This model is an oversimplified description of a biological neuron. For more biologically relevant neurons, see, e.g., (Carpenter & Grossberg, 1991; Levine, 1991).

  In particular, biological neural networks have more than 3 layers. In this paper, we restrict ourselves to 3-layer networks only. The reason is that in the majority of applications to control and to numerical computations, time is a crucial factor. The more layers we have, the bigger the processing time. Therefore, 3-layer networks guarantee the fastest data processing.

  Other models used in artificial neural networks include radial basis functions, for which the output of a neuron is \( y = s(\sqrt{\sum x_i^2}) \) for some function \( s \) (see, e.g., (Poggio & Girosi, 1990; Powell, 1992)).

- It is known (Cybenko, 1989; Funahashi, 1989; Hornik et al, 1989) that thus defined neural networks are universal approximators for functions. Namely, the following result is true:

**Definition 2.** Suppose that \( S \subset \mathbb{R}^n \) is a set, and \( \varepsilon > 0 \) is a real number. We say that functions \( f \) and \( g \) from \( S \) to \( \mathbb{R}^p \) are \( \varepsilon \)-close on \( S \) if \( |f_l(x_1, ..., x_n) - g_l(x_1, ..., x_n)| \leq \varepsilon \) for all \( (x_1, ..., x_n) \in S \) and all \( l \) from 1 to \( p \).

**THEOREM.** (Hornik, Stinchcombe, White) Assume that \( n \) and \( p \) are positive integers, \( S \subset \mathbb{R}^n \) is a compact set, \( f : S \rightarrow \mathbb{R}^p \) is continuous, and \( \varepsilon > 0 \) is a real number. Then, there exists a function \( f(x_1, ..., x_n) \) that is representable by a 3-layer neural network and that is \( \varepsilon \)-close to \( f \) on \( S \).

**Comment.** The number \( K \) of neurons in the hidden layer, that is necessary to get an \( \varepsilon \)-approximation to \( f \), depends on \( f \) and on \( \varepsilon \). In general, the better accuracy we seek (i.e., the smaller \( \varepsilon > 0 \) we take), the more neurons we need.
3. THIS CLASSICAL RESULT IS VERY IMPORTANT FOR CONTROL THEORY

The theorem that we just described is very important for applications of neural networks to control. Namely, suppose that we consider a plant, whose state can be described by finitely many parameters $x_1, ..., x_n$, and whose possible controls can be characterized by $p$ control parameters $u_1, ..., u_p$. For such a system, to design a control means to find a way to describe proper control values $u_i$ for each state of the plane $(x_1, ..., x_n)$. In mathematical terms, a control strategy is a function from $\mathbb{R}^n$ to $\mathbb{R}^p$.

In real-life situations, for every physical parameter $x_i$, there is an a priori bound on its value: velocity cannot take the value that exceed the speed of light, position cannot take the values that exceed the size of the area that we are analyzing, etc. If we denote by $X_i$ the biggest possible value of $|x_i|$, then we must consider only the values $x_i \in [-X_i, X_i]$, and therefore, only the states $(x_1, ..., x_n)$ from a compact set

$$S = [-X_1, X_1] \times [-X_2, X_2] \times ... \times [-X_n, X_n]$$

are physically possible. Therefore, we can consider only functions from $S$ to $\mathbb{R}^p$.

In non-linear control theory (see, e.g., (Mohler, 1991)), for many reasonable objectives, the optimal control is not continuous (the so-called “bang-bang” control is a typical example of this phenomenon). However, when it comes to practical implementations, we have to take into consideration the fact that every real-life device, no matter how fast it is, produces a continuous change of parameters: we cannot immediately change the position, cannot change the velocity in no time (with infinite acceleration), and even the change in the electric current (that occurs, e.g., when we switch something on or off) is continuous. Therefore, any real-life (hardware) control is a continuous function from $S$ to $\mathbb{R}^p$.

If we take into consideration restriction related to hardware (in particular, continuity), then the problem of finding an optimal control for a given hardware device will be formulated as that of optimizing an objective function under the condition that a continuous function $f : S \rightarrow \mathbb{R}^p$ satisfies given (hardware-related) restrictions. As a result, if we take implementation restrictions into consideration, the “ideal” (i.e., optimal) control will be continuous.

Since we are talking about a hardware control, and hardware devices cannot be absolutely precise, we cannot guarantee that the actual control will precisely coincide with the function $f : S \rightarrow \mathbb{R}^p$ that is prescribed by (thus restricted) optimization. However, by imposing more and more strict restrictions on the quality of the hardware parts, we can guarantee that the resulting hardware control is as close to the theoretical one as possible.

In view of that, the above-formulated theorem states that for an arbitrary “ideal” control $f : S \rightarrow \mathbb{R}^p$, and for an arbitrary $\varepsilon > 0$, there exists a neural network that computes $f$ with accuracy $\varepsilon$. In other words, whatever ideal control we start with, we can always train a neural network so that it will produce the results that will be close to the given ideal control as possible.
4. ADDITIONAL PROBLEMS ARISE FOR HARDWARE NEURONS

4.1. Hardware neurons are not precise, so this result is not directly applicable to them

In the above approximation theorem, we design a neural network that approximates \( f \) if the neurons are precisely described by the function \( s(x) \). But if we want to implement the neural network in hardware, then, of course, the characteristics of the actual hardware neurons will be only approximately equal to \( s(x) \). In view of that, it is necessary to design an approximating network so that it will provide the desired approximation for actual (non-precise) hardware neurons as well. In this section, following (Sirisaengtaksin & Kreinovich, 1993), we will show that such a design is possible.

4.2. Motivation: neurons can be only approximately implemented in hardware

4.2.1 In-out characteristic of a neuron can be only approximately implemented in hardware

It is technically impossible to design a hardware device whose in-out characteristics \( \tilde{s}(x) \) coincides (precisely coincides) with a prescribed function \( s(x) \). What we can expect is that whatever accuracy \( \delta > 0 \) we require, it will be possible to create a hardware neuron whose response for all \( x \) that do not exceed some level \( X \) is \( \delta \)-close to \( s(x) \)

\[
|\tilde{s}(x) - s(x)| \leq \delta.
\]

This can be done by tuning the device for all values \( x \) from this interval \((-X, X)\).

4.2.2 An actual hardware neuron can only process signals of limited size \((|x| \leq X \text{ for some } X)\)

It is, of course, impossible, to tune for all possible \( x \), for one reason that the ability of the existing testers to generate big signals is limited. Even if we overcome this difficulty by making an improvement each time, then, in finite time, we can still generate only finitely many signals \( X_1, ..., X_N \), and so there is no way to directly check our hardware neuron for \( |x| > \max(|X_1|, ..., |X_N|) \).

We can check a neuron indirectly: by designing a hardware neuron in such a way that any signal \( x > X \) simply unleashes a generator that generates an output signal that is “equal” to 1 (in reality, we cannot make it actually equal 1, but we can make it \( \delta \)-close to 1). Likewise, we can design \(-X\) as another threshold so that for \( x < -X \), the generated signal is close to 0.

To guarantee the continuity of the resulting piece-wise defined transformation, we must choose \( \delta \) and \( X \) in such a way that \( s(-X) \leq \delta \) and \( s(X) \geq 1 - \delta \). We will call such values \( \delta \) and \( X \) consistent.

Thus, although we cannot design a hardware neuron that gives precisely \( s(x) \), for every consistent pair of values \( \delta \) and \( X \), we can (in principle) design a neuron whose output \( \tilde{s}(x) \) satisfies the following inequalities: \( |	ilde{s}(x) - s(x)| \leq \delta \) for \( x \in [-X, X] \), \( |	ilde{s}(x)| \leq \delta \) for \( x < -X \), and \( |1 - \tilde{s}(x)| \leq \delta \) for \( x > X \).

4.2.3 In hardware, the values of the coefficients can be implemented only approximately

Likewise, it is impossible to get the precise values of the coefficients \( \beta_{kl} \), \( w_{ki} \) and \( b_k \), so, we can only guarantee these values with some precision \( \delta > 0 \).
4.3. Definitions and the main result

**Definition 3.** Assume that \( s(x) \) is a monotonic continuous function \( s : R \to (0, 1) \) such that

\[
\lim_{x \to -\infty} s(x) = 0, \quad \lim_{x \to \infty} s(x) = 1,
\]

and \( K, n \) and \( p \) are positive integers. By a design of a 3-layer neural network (or design for short) we mean a tuple \( \mu = (\{\beta_{kl}\}, \{w_{ki}\}, \{b_k\}, X, \delta) \), where:

- \( \beta_{kl}, w_{ki}, \) and \( b_k \) are real numbers, \( 1 \leq k \leq K, 1 \leq i \leq n, 1 \leq l \leq p, \) and
- \( X \) and \( \delta \) are positive real numbers such that \( s(-X) \leq \delta \) and \( s(X) \geq 1 - \delta \) (real numbers \( X \) and \( \delta \) that satisfy these inequalities will be called consistent).

The pair \((X, \delta)\) will be called a manufacturing accuracy of a design \( \mu \).

**Definition 4.** A function \( \tilde{s} : R \to R \) is called an implementation of a neuron with a manufacturing accuracy \((X, \delta)\) if for \( x \in [-X, X] \), \(|\tilde{s}(x) - s(x)| \leq \delta\); for \( x < -X \), \(|\tilde{s}(x)| \leq \delta\); and for \( x > X \), \(|\tilde{s}(x) - 1| \leq \delta\).

**Definition 5.** By an implementation of a design \( \mu = (\{\beta_{kl}\}, \{w_{ki}\}, \{b_k\}, X, \delta) \) we mean a function

\[
\tilde{f}_l(x_1, ..., x_n) = \sum_{k=1}^{K} \tilde{\beta}_{kl} \tilde{s}_k(\sum_{i=1}^{n} \tilde{w}_{ki} x_i + \tilde{b}_k),
\]

where each function \( \tilde{s}_k \) is an implementations of a neuron with manufacturing accuracy \((X, \delta)\), and \( \tilde{\beta}_{kl}, \tilde{w}_{ki} \) and \( \tilde{b}_k \) satisfy the following inequalities:

\[
|\tilde{\beta}_{kl} - \beta_{kl}| \leq \delta, \quad |\tilde{w}_{ki} - w_{ki}| \leq \delta, \quad \text{and} \quad |\tilde{b}_k - b_k| \leq \delta.
\]

**Definition 6.** Suppose that \( S \subset R^n \). We say that a function \( f : S \to R^n \) is \( \varepsilon \)-approximated by a design \( \mu \) if, for every implementation \( \tilde{f} \) of this design, \( |f_l(x) - \tilde{f}_l(x)| \leq \varepsilon \) for all \( x \in S \).

**THEOREM** (Sirisaengtaksin & Kreinovich, 1993). Assume that \( n \) and \( p \) are positive integers, \( S \subset R^n \) is a compact set, \( f : S \to R^n \) is continuous, and \( \varepsilon > 0 \) is a real number. Then, there exists a design \( \mu \) that \( \varepsilon \)-approximates \( f(x) \).

5. THE PREVIOUS APPROXIMATION RESULTS DO NOT COVER ALL POSSIBLE CONTROL SITUATIONS

The above-given theorems cover many important control problems, but they do not cover all of them. Let us explain why.

- First, not all the systems can be described by finitely many parameters. For example, when we control the processes in a chemical reactor, then, to make a proper control decision, we must know the temperature \( T \), density \( \rho \), and other characteristics in all the points \( u \) inside a reactor. So, in this case, to describe a state of the plant, we must know the functions \( T(u), \rho(u), \) etc. In case we are controlling a plant that consist of several reactors, we need to know the functions \( T(u), \rho(u), \ldots \), for \( u \) from the first reactor \( U_1 \), and similar functions to describe other reactors \( U_2, \ldots \). Such systems are called systems with distributed parameters. Can we use neural networks to control them?
• Second, in the above theorems, we are talking about controlling a plant under one specific control objective. In real life, we have different plants and different control objectives. Therefore, the objective of control theory is not only to provide optimal control of specific plants under specific criteria, but also to develop general methods that would help to control an arbitrary plant under an arbitrary objective. So, we would like a neural network to serve as a universal controller in the following sense: we provide it with the description of the system, with an optimality criterion, and it will generate the optimal control for this very system under this very criterion.

6. NEURONS CAN APPROXIMATE CONTROLS OF DISTRIBUTED SYSTEMS: MAIN RESULT

6.1. How to describe a state of a distributed system

Definition 7. Suppose that $n$ and $m$ are non-negative integers, and $U_1, ..., U_m$ are compact metric spaces with metrics $\rho_1, ..., \rho_m$. By a state we mean a tuple $s = (x_1, ..., x_n, f_1, ..., f_m)$, where $x_i$ are real numbers, and $f_j$ is a continuous function from $U_j$ to $R$.

Let us recall the standard denotation:

Denotation 1. By $C(U_j)$ we will denote the set of all continuous functions from $U_j$ to $R$. Topology on $C(U_j)$ is determined by a $C$–metric $\|f - g\| = \sup_{u \in U_j} |f(u) - g(u)|$.

Denotation 2. The set $S$ of all states will be denoted by $R^n \times C(U_1) \times ... \times C(U_m)$. Let us define the following metric on $S$:

$$\rho(s, \tilde{s}) = \max(|x_1 - \tilde{x}_1|, ..., |x_n - \tilde{x}_n|, \|f_1 - \tilde{f}_1\|, ..., \|f_n - \tilde{f}_m\|),$$

where $s = (x_1, ..., x_n, f_1, ..., f_m)$ and $\tilde{s} = (\tilde{x}_1, ..., \tilde{x}_n, \tilde{f}_1, ..., \tilde{f}_m)$.

Definition 8. By a set of all physically possible states we will mean a compact set $S \subset S$. States from $S$ will be called physically possible.

Remark. These definitions are different from the ones that a reader may encounter in the majority of the papers on distributed parameter control. The main reason is that we are trying to be realistic, in particular, we are trying to take into consideration the hardware-related restrictions on the control strategies. The motivation for this definition is as follows: in real life, as we have already mentioned, all physical quantities are bounded. The coordinates of a controlled object (plant) are bounded by the area where we can control; its velocity is bounded by the velocity of light; etc. So, for an arbitrary physical quantity, its values are bounded, and the values of its derivatives are bounded by some number $M$. So, the functions $f_j(u)$ that correspond to physically possible states, are uniformly bounded, and their derivatives are uniformly bounded.

So, we can use the known description of compact sets in a space $C(U)$ of continuous functions from a compact space $U$ to $R$. Namely, according to Arzelá-Ascoli Theorem (see, e.g., (Sprecher, 1987)), a set of functions $F \subseteq C(U)$ is pre-compact if and only if it is uniformly bounded (i.e., there exists a number $M$ that bound all of them, i.e., such that $|f(u)| \leq M$ for all $u \in U$ and for all $f \in F$), and uniformly
Definition 9. Suppose that an integer \( p \) is given. Elements \( u = (y_1, \ldots, y_p) \) will be called \textit{control values}. On \( \mathbb{R}^p \), we will define a metric \( \|y - \tilde{y}\| = \max|y_i - \tilde{y}_i| \). By a \textit{control strategy} we mean a continuous function \( J : S \rightarrow \mathbb{R}^p \) that assigns to every physically possible state a control value \( u \).

Definition 10. Suppose that \( \varepsilon > 0 \). We say that control strategies \( J \) and \( \tilde{J} \) are \( \varepsilon \)-\textit{close} if for all \( s \in S \), \( \|J(s) - \tilde{J}(s)\| \leq \varepsilon \).

Definition 11. We say that a control strategy \( \tilde{J} \) is \textit{representable by a 3-layer neural network} if
\[
\tilde{J}(x_1, \ldots, x_n, f_1, \ldots, f_m) = F(x_1, \ldots, x_n, f_1(u^{(1)}_1), \ldots, f_1(u^{(N_1)}_1), \ldots, f_m(u^{(1)}_m), \ldots, f_m(u^{(N_m)}_m))
\]
for some elements \( u^{(1)}_1 \in U_1, \ldots, u^{(N_1)}_1 \in U_1, \ldots, u^{(1)}_m \in U_m, \ldots, u^{(N_m)}_m \in U_m \), and for some function \( F \) that is representable by a 3-layer neural network.

Comment. So, the neural network inputs the values \( x_i \) and \( f_j(u^{(i)}_j) \), and generates the control values \( y_1, \ldots, y_p \).

6.3. The main result

THEOREM 1. For an arbitrary control strategy \( J \), and for an arbitrary \( \varepsilon > 0 \), there exists a control strategy \( \tilde{J} \) that is representable by a 3-layer neural network, and that is \( \varepsilon \)-close to \( J \).

Remark. We formulated this theorem in control terms. However, the same approximations may be important in other applications, in Appendix, we give the reformulation of this theorem and of several future results in general mathematical terms.

7. TAKING LOCATION ERROR INTO CONSIDERATION

Remark. Theorem 1 corresponds to the case when we can measure the values of \( f_j : U_j \rightarrow \mathbb{R} \) precisely in the given points \( u^{(i)}_j \). In real life, we can only guarantee this with some finite accuracy \( \varepsilon_u \). Let us prove that if we take this inaccuracy into consideration, 3-layer neurons will still be universal approximators.

Definition 12. Suppose that \( \varepsilon > 0 \) and \( \varepsilon_u > 0 \). We say that for a location error \( \varepsilon_u \), a control strategy \( J \) is \( \varepsilon \)-\textit{approximable} by a 3-layer neural network if for every \( j \) from 1 to \( m \), there exist elements \( u^{(1)}_j, \ldots, u^{(N_j)}_j \in U_j \), and there exists a function \( F \) that is representable by a 3-layer neural network such that for all \( s = (x_1, \ldots, x_n, f_1, \ldots, f_m) \in S \), and for any \( \tilde{u}^{(i)}_j \) such that \( \rho_j(u^{(i)}_j, \tilde{u}^{(i)}_j) \leq \varepsilon_u \), we have
\[
\|J(s) - F(x_1, \ldots, x_n, f_1(u^{(1)}_1), \ldots, f_1(u^{(N_1)}_1), \ldots, f_m(u^{(1)}_m), \ldots, f_m(u^{(N_m)}_m))\| \leq \varepsilon.
\]
THEOREM 2. For an arbitrary control strategy $J$, and for an arbitrary $\varepsilon > 0$, there exists a positive real number $\varepsilon_u$ such that for location error $\varepsilon_u$, $J$ is $\varepsilon$-approximable by a 3-layer neural network.

8. NEURAL NETWORKS CAN APPROXIMATE UNIVERSAL CONTROL STRATEGIES

Remark. By a universal control strategy we mean a method, that, given:

- a description of a plant (i.e., the description of how its states are changed if we apply this or that control);
- an objective, i.e., a function of a trajectory that we are trying to minimize, and
- a current state,

describes what control to apply. We already know how to describe a state, so let us show how to describe the plant’s dynamics and our objectives.

The definitions that we come up with are slightly different from the ones that a reader may encounter in the majority of the papers on control theory. The main difference is that (like in the case of distributed control), we are trying to be realistic, in particular, we are trying to take into consideration the hardware-related restrictions on the control strategies. Therefore, we restrict ourselves to continuous strategies only, and we assume that the set of all physically possible control strategies is compact.

Specific motivations that explain why this particular definition is different from the usual ones, will be provided after each definition.

8.1. How to describe the dynamics

Definition 13. By the set of physically possible control values we mean a compact set $C \subset \mathbb{R}^p$. Control values from the set $C$ will be called physically possible.

Remark. The motivation for this definition is the same as for Definition 8.

Definition 14. Suppose that $S \subset \mathbb{R}^n \times C(U_1) \times \ldots \times C(U_m)$ is the set of physically possible states, and $C$ is the set of physically possible control values. By a dynamics we mean a tuple $d = (g_1, \ldots, g_n, h_1, \ldots, h_m)$ of continuous functions, where $g_i : S \times C \to \mathbb{R}$, and $h_j : U_j \times S \times C \to \mathbb{R}$.

Remark. Dynamics describes how a state evolves in time: if we are in the state $s = (x_1, \ldots, x_n, f_1, \ldots, f_m) \in S$, and apply control $y \in C$, then $dx_i/dt = g_i(s, y)$ and $df_j(u)/dt = h_j(u, s, y)$.

Definition 15. On the set $\mathcal{D} = C(S \times C)^n \times C(U_1 \times S \times C) \times \ldots \times C(U_m \times S \times C)$ of all dynamics we can define a metric as follows: if $d = (g_1, \ldots, g_n, h_1, \ldots, h_m)$, and $\tilde{d} = (\tilde{g}_1, \ldots, \tilde{g}_n, \tilde{h}_1, \ldots, \tilde{h}_m)$, then

$$\rho(d, \tilde{d}) = \max_i \|g_i - \tilde{g}_i\|, \max_j \|h_j - \tilde{h}_j\|,$$

where $\| \cdot \|$ denotes a $C$-metric.

Definition 16. By a set of physically possible dynamics we mean a compact set $D \subset \mathcal{D}$. Elements of $D$ will be called physically possible dynamics.
The reason for this definition is the same as above: in real life situations, the rate with which we can change the state is limited (\(|g_i| \leq M\)), and the difference between the rates in nearby points is also limited. So, the set of all physically possible dynamics is uniformly bounded and equicontinuous, hence, compact.

8.2. How to describe trajectories and objectives

Definition 17. Suppose that \(S \subset R^n \times C(U_1) \times ... \times C(U_m)\) is the set of physically possible states, and \(T_0 > 0\) is a real number. By a trajectory we mean a tuple \(d = (G_1,...,G_n,H_1,...,H_m)\) of continuous functions, where \(G_i : [0,T_0] \rightarrow R\), and \(H_j : [0,T_0] \times U_j \rightarrow R\).

Remark. The meaning of these functions is as follows: for an arbitrary moment of time \(t \in [0,T_0]\), the state \(s(t)\) is equal to \((G_1(t),...,G_n(t),H_1(u,t),...,H_m(u,t))\).

Definition 18. On the set \(T = C([0,T_0])^n \times C([0,T_0] \times U_1) \times ... \times C([0,T_0] \times U_m)\) of all trajectories we can define a metric as follows: if \(t = (G_1,...,G_n,H_1,...,H_m)\), and \(\tilde{t} = (\tilde{G}_1,...,\tilde{G}_n,\tilde{H}_1,...,\tilde{H}_m)\), then

\[
\rho(t,\tilde{t}) = \max (\max_i \|G_i - \tilde{G}_i\|, \max_j \|H_j - \tilde{H}_j\|),
\]

where \(\|\cdot\|\) denotes a \(C-\)metric.

Definition 19. By a set of physically possible trajectories we mean a compact set \(T \subset \mathcal{T}\). Elements of \(T\) will be called physically possible trajectories.

Remark. Physically possible trajectories are limited in what the states can be, and in the rate with which the states change. Therefore, the set of all physically possible trajectories is compact.

Definition 20. By an objective we mean a continuous mapping \(o : T \rightarrow R\).

Remark. In other words, an objective is a function that describes a quality of each physically possible trajectory. It can be time that was necessary to reach the goal, or the total fuel spent, etc. The goal of the control theory is to maintain the trajectory with the smallest possible value of an objective.

Definition 21. On the set \(\mathcal{O} = C(T)\) of all objectives we can define a \(C-\)metric.

Definition 22. By a set of physically meaningful objectives we mean a compact set \(\mathcal{O} \subset \mathcal{O}\). Elements of \(\mathcal{O}\) will be called physically meaningful objectives.

8.3. What is a universal control strategy and how it can be represented by a neural network

Definition 23. By a universal control strategy we mean a continuous mapping \(u : D \times \mathcal{O} \times S \rightarrow R^p\).

Remarks.
1. In other words, a universal control strategy takes a description of the dynamics, an objective, and a current state, and generates control values that take into consideration what plant we are controlling and what is our objective. Every general method of control theory is an example of a universal control strategy: for example, linearization methods, etc.
2. The notion of a universal control strategy is a reasonable theoretical concept that formalizes the main objective of control theory. However, to avoid misunderstanding, we must remark that currently, for most interesting problems, no methods that implement universal strategies are known. In other words, for these problems, there is no analytical (or algorithmic) methodology that would help us to find an optimal control based on given descriptions of a plant and of an objective. Therefore, it is desirable to design such a universal strategy by using known examples of optimal control to train a neural network.

Before we start the training, it would be nice to know whether such a training is at all possible, i.e., whether there exist a neural network whose results are close to the ones that are generated by the (unknown) universal control strategy. Since we do not know that strategy, the only way to guarantee that such a neural network exists for the strategy that we are interested in, is to prove that an approximating neural network exists for an arbitrary universal control strategy. This is exactly what we are going to prove in this Section.

**Definition 24.** Suppose that $\varepsilon > 0$. We say that universal control strategies $u$ and $\tilde{u}$ are $\varepsilon-$close if for all $d \in D$, $o \in O$, and $s \in S$, $\|u(d, o, s) - \tilde{u}(d, o, s)\| \leq \varepsilon$.

**Definition 25.** We say that a universal control strategy $\tilde{u}$ is representable by a $3-$layer neural network if

$$\tilde{u}(d, o, s) = \tilde{u}((g_1, ..., g_n, h_1, ..., h_m), o, (x_1, ..., x_n, f_1, ..., f_m)) = F(\{g_i(s^{(k)}), c^{(k)}\}, \{h_j(u_j(k), s^{(k)}, c^{(k)})\}, \{o(t^{(k)})\}, \{x_i\}, \{f_j(u_j^{(k)})\})$$

for some states $s^{(k)}$, control values $c^{(k)} \in C$, elements $u_j^{(k)} \in U_j$, trajectories $t^{(k)}$, and for some function $F$ that is representable by a 3-layer neural network.

**Comment.** So, the neural network inputs the sample values of $g_i$ and $h_k$ for some states and controls, the values of the objective for several trajectories, and the values of $x_i$ and $f_j(u_j^{(k)})$, and generates the control values $y_1, ..., y_p$.

8.4. Result: an arbitrary universal control strategy can be approximated by a neural network

**THEOREM 3.** For an arbitrary universal control strategy $u$, and for an arbitrary $\varepsilon > 0$, there exists a universal control strategy $\tilde{u}$ that is representable by a 3-layer neural network, and that is $\varepsilon-$close to $u$.

**Remark.** This result means that whatever general method of solving control problems someone invents, there is always a possibility to implement this method by a 3-layer neural network. This network is universal in the sense that it will work for an arbitrary plant and an arbitrary objective.

9. HARDWARE NEURONS CAN APPROXIMATE CONTROLS OF DISTRIBUTED SYSTEMS AND UNIVERSAL CONTROLLERS

**Remark.** A design that is not sensitive to parameters of a neuron, can be proposed for more general approximation theorems 1, 2 and 3. Namely, the following is true:
Definition 26. We say that a control strategy $J$ is $\varepsilon-$approximated by a design $\mu$, if, for some elements $u_1^{(1)} \in U_1, \ldots, u_1^{(N_1)} \in U_1, \ldots, u_m^{(1)} \in U_m, \ldots, u_m^{(N_m)} \in U_m$, and for every implementation $F$ of this design, a control strategy

$$\tilde{J}(x_1, \ldots, x_n, f_1, \ldots, f_m) = \tilde{F}(x_1, \ldots, x_n, f_1(u_1^{(1)}), \ldots, f_1(u_1^{(N_1)}), \ldots, f_m(u_m^{(1)}), \ldots, f_m(u_m^{(N_m)}))$$

is $\varepsilon-$close to $J$.

**THEOREM 4.** For an arbitrary control strategy $J$, and for an arbitrary $\varepsilon > 0$, there exists a design $\mu$ that $\varepsilon$-approximates $J$.

Definition 27. Suppose that $\varepsilon > 0$ and $\varepsilon_u > 0$. We say that for a location error $\varepsilon_u$, a control strategy $J$ is $\varepsilon-$approximated by a design $\mu$ (or that a design $\mu$ $\varepsilon-$approximates $J$) if for every $j$ from 1 to $m$, there exist elements $u_j^{(1)}, \ldots, u_j^{(N_j)} \in U_j$, such that for every implementation $\tilde{F}$ of the design $\mu$, for all $s = (x_1, \ldots, x_n, f_1, \ldots, f_m) \in S$, and for any $\tilde{u}_j^{(i)}$ such that $\rho_j(u_j^{(i)}, \tilde{u}_j^{(i)}) \leq \varepsilon_u$, we have

$$|J(s) - \tilde{F}(x_1, \ldots, x_n, f_1(\tilde{u}_1^{(1)}), \ldots, f_1(\tilde{u}_1^{(N_1)}), \ldots, f_m(\tilde{u}_m^{(1)}), \ldots, f_m(\tilde{u}_m^{(N_m)}))| \leq \varepsilon.$$

**THEOREM 5.** For an arbitrary control strategy $J$, and for an arbitrary $\varepsilon > 0$, there exists a positive real number $\varepsilon_u$ and a design $\mu$ such that for location error $\varepsilon_u$, $\mu$ $\varepsilon$-approximates $J$.

Definition 28. We say that a universal control strategy $u$ is $\varepsilon-$approximated by a design $\mu$, if for some states $s^{(k)}$, control values $c^{(k)} \in C$, elements $u_j^{(k)} \in U_j$, and trajectories $t^{(k)}$, and for any implementation $\tilde{F}$ of the design $\mu$, the strategy

$$\tilde{u}(d, o, s) = \tilde{u}((g_1, \ldots, g_n, h_1, \ldots, h_m), o, (x_1, \ldots, x_n, f_1, \ldots, f_m)) =$$

$$\tilde{F}([g_i(s^{(k)}, c^{(k)})], [h_j(u_j^{(k)}, s^{(k)}, c^{(k)})], [o(t^{(k)})], \{x_1\}, \{f_j(u_j^{(k)})\})$$

is $\varepsilon-$close to $u$.

**THEOREM 6.** For an arbitrary universal control strategy $u$, and for an arbitrary $\varepsilon > 0$, there exists a design $\mu$ that $\varepsilon$-approximates $u$.

10. PROOFS

10.1. General comment
To the best of our knowledge, all the proofs are original in the sense that these theorems have never been proved before. This does not mean, of course, that all the ideas of these proofs are new. We are using more or less standard techniques of mathematical approximation theory. So, readers who are knowledgeable in the related mathematics, can browse through them without going through the details that for them may seem more or less routine. However, we decided to give the detailed proofs so that any reader (and not only a mathematician) will be able to follow each proof step by step.

10.2. Proofs of Theorems 1 and 2

1°. Let us first show that it is sufficient to prove Theorem 2, and then Theorem 1 will follow.
Indeed, if we prove Theorem 2, this means that there exists a function $F$ that is representable by a 3-layer neural network, and a number $\varepsilon_u > 0$ such that for an arbitrary state $s$, if $\rho_j(u_j^{(i)}, \tilde{u}_j^{(i)}) \leq \varepsilon_u$, then

$$|J(s) - F(x_1, \ldots, x_n, f_1(\tilde{u}_1^{(i)}), \ldots, f_1(u_1^{(N_1)}), \ldots, f_m(\tilde{u}_m^{(1)}), \ldots, f_m(u_m^{(N_m)}))| \leq \varepsilon.$$ 

In particular, this inequality is true for $\tilde{u}_j^{(i)} = u_j^{(i)}$. Therefore, we have the formulation of Theorem 1.

In view of this remark, in the following text we will prove Theorem 2.

2°. Let us first give an outline of the proof, to make it easier for a reader to follow its logic:

- First, we find the states $s_1, \ldots, s_M$ that form a sufficiently dense subset of $S$.
- Then, we find the values $u_j^{(1)}, \ldots, u_j^{(N_j)} \in U_j$ such that if $\tilde{u}_j^{(i)}$ are $\varepsilon_u$-close to $u_j^{(i)}$, then the error of reconstructing a state $s = (x_1, \ldots, x_n, f_1, \ldots, f_m)$ from the values $f_j(\tilde{u}_j^{(i)})$ is sufficiently small.
- Third, we define an auxiliary function $J^a$ in such a way: for the states $s_q = (\{x_i\}, \{f_j\})$, we want to have $J^a(\{x_i\}, \{f_j(u_j^{(i)})\}) \approx J(s_q)$; then, we use interpolation to define $J^a(s)$ for all $s$.
- Finally, we approximate the resulting function $J^a$ by a neural network.

3°. Before we start proving, let us introduce some denotations.

3.1°. We want to guarantee that $|y_l - \tilde{y}_l| \leq \varepsilon$ for all $l$ from 1 to $p$, where $y_l$ is the control that we want to approximate (i.e., $J_l(s)$), and $\tilde{y}_l$ is a control that is generated by a desired neural network. These inequalities can be rewritten as $\|y - \tilde{y}\| \leq \varepsilon$, where we denoted $\|y - \tilde{y}\| = \max_l |y_l - \tilde{y}_l|$.

3.2°. Let us denote the desired set of points $\{u_j^i\}$, where $1 \leq j \leq m$ and $1 \leq i \leq N_j$, by $V$. Correspondingly, the set of values $\{\tilde{u}_j^i\}$ will be denoted by $\tilde{V}$.

We are interested in the points $\tilde{u}_j^i$ such that $|\tilde{u}_j^i - u_j^i| \leq \varepsilon_u$ for all $i, j$. This inequality can be rewritten as $d_V(V, \tilde{V}) \leq \varepsilon_u$, where

$$d_V(V, \tilde{V}) = \max_{1 \leq j \leq m} \max_{1 \leq i \leq N_j} |u_j^i - \tilde{u}_j^i|$$

is a metric on the set of all such sequences.

3.3°. For each state $s$, and each set of points $\tilde{V}$, by $\pi_{\tilde{V}}(s)$ we will denote the sequence of values $\{x_i\}, \{f_j(\tilde{u}_j^i)\}$. In particular, if we take $\tilde{V} = V$, we get $\pi_V(s) = (\{x_i\}, \{f_j(u_j^i)\})$.

Since $\pi_{\tilde{V}}(s)$ is a sequence of real numbers, we can define the distance $d(\pi_{\tilde{V}}(s), \pi_V(s))$ between two such sequences $\pi_{\tilde{V}}(s) = (\{x_i\}, \{f_j(\tilde{u}_j^i)\})$ and $\pi_V(s) = (\{\tilde{x}_i\}, \{\tilde{f}_j(\tilde{u}_j^i)\})$ as

$$d(\pi_{\tilde{V}}(s), \pi_V(s)) = \max_{1 \leq i \leq n} |x_i - \tilde{x}_i|, \max_{1 \leq j \leq m} \max_{1 \leq i \leq N_j} |f_j(u_j^i) - \tilde{f}_j(u_j^i)||. $$
Let us show that for \( d(\pi_V(s), \pi_V(\tilde{s})) \), we get the following inequality:
\[
d(\pi_V(s), \pi_V(\tilde{s})) \leq \rho(s, \tilde{s}).
\]
Indeed, \( \rho(s, \tilde{s}) \) was defined as the maximum of the values \( |x_i - \tilde{x}_i| \) and \( |f_j(u_j) - \tilde{f}_j(u_j)| \) for all \( i, j \) and all \( u_j \in U_j \). So, \( \rho(s, \tilde{s}) \geq |x_i - \tilde{x}_i| \) and \( \rho(s, \tilde{s}) \geq |f_j(u_j) - \tilde{f}_j(u_j)| \). On the other hand, \( d(\pi_V(s), \pi_V(\tilde{s})) \) was defined as the biggest of the values \( |x_i - \tilde{x}_i| \) and \( |f_j(u_j) - \tilde{f}_j(u_j)| \) for all \( i, j \) and for \( u_j = u_j^{(1)}, \ldots, u_j^{(N_s)} \). Since \( \rho(s, \tilde{s}) \) is not smaller than any of these values, it is not smaller than the biggest of them. So, we conclude that \( \rho(s, \tilde{s}) \geq d(\pi_V(s), \pi_V(\tilde{s})) \).

4th. In this Section, we will discuss \( J^a \).

First, in 4.1st, we will use the above-defined denotations to formulate what we want from \( J^a \). Then (in 4.2st), we give an informal motivation for our future choice of \( J^a \). In Section 4.3st, we give an explicit expression for \( J^a \) in terms of \( s_q, \ldots \), and in the last subsection (4.4st), we explain what conditions these \( s_q, \ldots \), must satisfy so that the resulting \( J^a \) would satisfy the desired inequality. In the next Section 5th, these conditions will be used to choose \( \{s_q\}, \ldots \).

4.1st. First, let us use the denotations from 3st to explain what exactly we want from \( J^a \).

We want to guarantee that for every state \( s \in S \), and for every \( \tilde{V} \) such that \( d_V(V, \tilde{V}) \leq \varepsilon_u \), we have \( ||J(s) - J^a(\pi_V(s))|| \leq \varepsilon/2 \). If we guarantee that, then we will be able to prove the Theorem: namely, we will choose a function \( F \) that is representable by a 3-layer neural network, and that approximates \( J^a \) with an accuracy \( \varepsilon/2 \) (i.e., \( ||F(\pi_V(s)) - J^a(\pi_V(s))|| \leq \varepsilon/2 \)), and then we would get the desired inequality
\[
||J(s) - F(\pi_V(s))|| \leq ||J(s) - J^a(\pi_V(s))|| + ||J^a(\pi_V(s)) - F(\pi_V(s))|| \leq \varepsilon/2 + \varepsilon/2 = \varepsilon.
\]

4.2st. Let us now give an informal motivation of our future choice of \( J^a \).

Suppose that we have already chosen \( \{s_q\}_{1 \leq q \leq M}, V, \) and \( \varepsilon_u \). The value of \( J^a \) depends only on \( \pi_V(s) \). According to 4.1st, we would like to choose \( J^a \) in such a way that if \( s = s_q \), then \( J^a(\pi_V(s_q)) \approx J(s_q) \). Since \( \tilde{V} \) is close to \( V \), the value of \( \pi_V(s_q) \) is close to \( \pi_V(s) \). So, to guarantee that \( J^a(\pi_V(s_q)) \approx J(s_q) \), we can assume that \( J^a(\pi_V(s_q)) \approx J(s_q) \). So, as a first approximation to \( J^a \), we can take a function that is defined in \( M \) points \( \pi_V(s_1), \ldots, \pi_V(s_M) \), and whose value in \( \pi_V(s_q) \) is equal to \( J(s_q) \).

A natural way to interpolate this function is to take for every other \( \pi_V(s) \), a convex combination of these values \( J^a(\pi_V(s)) = \sum_q \alpha_q(\pi_V(s)) J(s_q) \), so that the coefficient \( \alpha_q \) at \( J(s_q) \) is different from 0 only if \( \pi_V(s) \) is sufficiently close to \( \pi_V(s_q) \). A natural way to do that is to choose a positive real number \( \varepsilon_s \), and to take a coefficient \( \alpha_q \) proportional to \( \max\{0, \varepsilon_s - d(\pi_V(s_q), \pi_V(s))\} \). So, we are ready to describe an explicit expression for \( J^a \).

4.3st. Let us describe \( J^a \).

Suppose that we have already chosen \( \{s_q\}_{1 \leq q \leq M}, V, \varepsilon_u > 0 \) and \( \varepsilon_s > 0 \). Then, we want to choose the following \( J^a \):
\[
J^a(\pi_V(s)) = \frac{\sum_{i=1}^{M} \phi(\pi_V(s_q), \pi_V(s)) J(s_q)}{\sum_{i=1}^{M} \phi(\pi_V(s_q), \pi_V(s))},
\]
where we denoted \( \phi(\pi_V(s_q), \pi_V(s)) = \max\{0, \varepsilon_s - d(\pi_V(s_q), \pi_V(s))\} \).

4.4st. Let us find out, what conditions we must impose on \( \{s_q\}_{1 \leq q \leq M}, V, \varepsilon_u > 0 \), and \( \varepsilon_s > 0 \), so that \( J^a \) is everywhere defined and satisfies the desired inequality
\[
||J(s) - J^a(\pi_V(s))|| \leq \varepsilon/2.
\]
4.4.1°. Let us first describe when \( J^a \) is everywhere defined.

According to the above definition of \( J^a \) as a fraction, it is defined if the denominator of the fraction is different from 0. It is true if for each \( s \), for which there exists an \( s_q \) such that \( d(\pi_V(s), \pi_V(s_q)) < \varepsilon_s \).

We can guarantee this inequality, if \( \tilde{V} \) is sufficiently close to \( V \), and \( s \) is sufficiently close to \( s_q \). In other words, since \( d(\pi_V(s), \pi_V(s_q)) \leq d(\pi_V(s), \pi_V(s)) + d(\pi_V(s), \pi_V(s_q)) \), we can guarantee the above inequality if we manage to guarantee the inequalities \( d(\pi_V(s), \pi_V(s)) \leq \varepsilon_s/3 \) and \( d(\pi_V(s), \pi_V(s_q)) \leq \varepsilon_s/3 \).

The first of these inequalities must be used to determine \( \varepsilon_u \). As for the second, since we know (see 3.3°) that \( d(\pi_V(s), \pi_V(\tilde{s})) \leq \rho(s, \tilde{s}) \), it is sufficient to guarantee that for every \( s \), there exists a \( s_q \) for which \( \rho(s, s_q) \leq \varepsilon_s/3 \), i.e., that \( \{s_q\} \) form a \((\varepsilon_s/3)\)-net for \( S \).

4.4.2°. Now, let us analyze when \( J^a \) satisfies the desired inequality. The value \( J^a(\pi_V(s)) \) is a convex combination of the values \( J(s_q) \). So, to guarantee that for every \( l \) from 1 to \( p \), the values \( J^a(\pi_V(s)) \) belong to the desired interval \( [J_l(s) - \varepsilon/2, J_l(s) + \varepsilon/2] \), it is sufficient to guarantee that \( J_l(s_q) \in [J_l(s) - \varepsilon/2, J_l(s) + \varepsilon/2] \) for all \( q \) for which \( d(\pi_V(s), \pi_V(s_q)) \leq \varepsilon_s \).

We have already decided to choose \( \varepsilon_u \) in such a way that \( d(\pi_V(s), \pi_V(s)) \leq \varepsilon_s/3 \) for every state \( s \in S \). Therefore, if \( d(\pi_V(s), \pi_V(s_q)) \leq \varepsilon_s \), then

\[
d(\pi_V(s), \pi_V(s_q)) \leq d(\pi_V(s), \pi_V(s)) + d(\pi_V(s), \pi_V(s_q)) \leq \varepsilon_s/3 + \varepsilon_s = (4/3)\varepsilon_s.
\]

According to our outline, we must choose \( V \) in such a way that the error of reconstructing \( s \) from \( \pi_V(s) \) is small. In other words, we would like to choose \( V \) in such a way that for every \( s, \tilde{s} \in S \), \( \rho(s, \tilde{s}) \leq d(\pi_V(s), \pi_V(\tilde{s})) + \varepsilon_r \) for some \( \varepsilon_r > 0 \).

If we choose such a \( V \), then, from \( d(\pi_V(s), \pi_V(s_q)) \leq (4/3)\varepsilon_s \), we can conclude that \( \rho(s, s_q) \leq (4/3)\varepsilon_s + \varepsilon_r \).

We want to guarantee the inequality \( \|J(s) - J(s_q)\| \leq \varepsilon/2 \). Therefore, we must choose \( \varepsilon_s \) and \( \varepsilon_r \) in such a way that if \( \rho(s, \tilde{s}) \leq (4/3)\varepsilon_s + \varepsilon_r \), then \( \|J(s) - J(\tilde{s})\| \leq \varepsilon/2 \).

4.4.3°. Summarizing, we arrive at the following conclusions:

- We must choose \( \varepsilon_s \) and \( \varepsilon_r \) in such a way that if \( \rho(s, \tilde{s}) \leq (4/3)\varepsilon_s + \varepsilon_r \), then
  \[
  \|J(s) - J(\tilde{s})\| \leq \varepsilon/2.
  \]
- Then, we must choose the states \( s_q \) so that they form a \((\varepsilon_s/3)\)-net for \( S \).
- After that, we must choose \( V \) so that for every \( s, \tilde{s} \in S \), \( \rho(s, \tilde{s}) \leq d(\pi_V(s), \pi_V(\tilde{s})) + \varepsilon_r \).
- Finally, we must choose \( \varepsilon_u \) so that if \( d_V(V, \tilde{V}) \leq \varepsilon_u \), then \( d(\pi_V(s), \pi_V(s)) \leq \varepsilon_s/3 \).

5°. Let us now use the conditions from 4.4.3° to choose \( s_q, V, \varepsilon_u \), etc.

5.1°. Let us first choose \( \varepsilon_r \) and \( \varepsilon_s \). According to 4.4.2°, we must choose these two parameters in such a way that if \( \rho(s, \tilde{s}) \leq (4/3)\varepsilon_s + \varepsilon_r \), then \( \|J(s) - J(\tilde{s})\| \leq \varepsilon/2 \). A functional \( J \) is continuous on a compact set \( S \). Therefore, it is uniformly continuous on \( S \). This means, in particular, that there exists a \( \delta > 0 \) such that if \( \rho(s, \tilde{s}) \leq \delta \), and \( s, \tilde{s} \in S \), then \( |J(s) - J(\tilde{s})| \leq \varepsilon/2 \). Let us fix such a \( \delta \).

So, if we choose \( \varepsilon_s \) and \( \varepsilon_r \) in such a way that \((4/3)\varepsilon_s + \varepsilon_r \leq \delta \), then the desired condition will be true.
Let us choose them so that \( \varepsilon_r = \delta/2 \) and \( (4/3)\varepsilon_s = \delta/2 \), i.e., \( \varepsilon_s = (3/8)\delta \).

5.2°. Let us now choose \( \{s_q\} \).

According to 4.4.1°, the states \( \{s_q\} \) must form a \( (\varepsilon_s/3) \)-net for \( S \). Here, \( \varepsilon_s/3 = \delta/8 \). Since \( S \) is compact, there exists a finite \( (\delta/8) \)-net in \( S \). So, let us choose one of these nets as the desired set \( s_1, \ldots, s_M \).

5.3°. Let us now choose the values \( V = \{u_j^{(i)}\} \).

According to 4.4.3°, we must choose them in such a way that for every \( s, \tilde{s} \in S \), \( \rho(s, \tilde{s}) \leq d(\pi_V(s), \pi_V(\tilde{s})) + \varepsilon_r \).

5.3.1°. How can we guarantee this inequality?

Before we provide the reader (in the next subsection) with our choice, we would like to explain this choice informally.

If for two states \( s \) and \( \tilde{s} \), the values of the corresponding functions \( f_j \) and \( \tilde{f}_j \) are close in the points \( u_j^{(i)} \), then the only reason why they can differ for \( u \neq u_j^{(i)} \) is because the values \( f_j(u) \) and \( \tilde{f}_j(u) \) change drastically when we go from \( u_j^{(i)} \) to \( u \). So, the natural way to prevent such a drastic change is to choose \( u_j^{(i)} \) in such a way that the values of the functions \( f_j(u) \) will not change much when we change \( u \) to a nearest point \( u_j^{(i)} \). Since the functions \( f_j \) are continuous, the closer \( u \) is to \( u_j^{(i)} \), the smaller is the difference, So, we must choose \( u_j^{(i)} \) with the continuity of \( f_j \) in mind.

5.3.2°. Let us now describe our choice. The states \( s \) that constitute the set \( S \) are tuples \((x_1, \ldots, x_n, f_1, \ldots, f_m)\). Let us denote \( x_i \) by \( \pi_i(s) \), and \( f_j \) by \( \pi_j^f(s) \). The mappings (projections) \( \pi_i \) and \( \pi_j^f \) are continuous. If a function \( f \) is continuous, then the image \( f(K) \) of a compact set \( S \) is also compact. Therefore, the sets \( \pi_i(K) \) and \( \pi_j^f(K) \) are compact. The set \( \pi_j^f(K) \) consists of all the functions \( f_j \) that correspond to states \( s \in S \).

So, for each \( j \), the set \( \pi_j^f(K) \) is a compact subset of \( C(U_j) \) (where \( U_j \) is compact). According to Arzelá-Ascoli Theorem (see, e.g., (Sprecher, 1987)), this means that the functions from \( \pi_j^f(K) \) are equicontinuous, i.e., for every \( \alpha > 0 \), there exists a \( \beta_j > 0 \) such that if \( \rho(u, \tilde{u}) \leq \beta_j \), and \( f_j \in \pi_j^f(K) \), then \( |f_j(u) - f_j(\tilde{u})| \leq \alpha \). In particular, such \( \beta_j \) exist for \( \alpha = \varepsilon_r/2 = \delta/4 \) (we will see in a moment why namely \( \delta/4 \) is needed here). Let us choose these \( \beta_j \).

Since \( U_j \) is compact, there exists a finite \( \beta_j \)-net for \( U_j \), i.e., a finite set \( u_j^{(1)}, \ldots, u_j^{(N_j)} \) of points from \( U_j \) such that for every \( u \in U_j \) there exists a \( i \leq N_j \) for which \( \rho(u, u_j^{(i)}) \leq \beta_j \).

5.3.3°. Let us prove that for this choice, for every \( s, \tilde{s} \in S \), \( \rho(s, \tilde{s}) \leq d(\pi_V(s), \pi_V(\tilde{s})) + \delta/2 \).

Indeed, suppose that \( s \) and \( \tilde{s} \) are states from \( S \). Let us denote \( d(\pi_V(s), \pi_V(\tilde{s})) \) by \( d \). According to the definition of the metric \( d \), we can conclude that \( |x_i - \tilde{x}_i| \leq d \) and \( |f_j(u_j^{(i)}) - \tilde{f}_j(u_j^{(i)})| \leq d \) for all \( i \) and \( j \). The desired inequality \( \rho(s, \tilde{s}) \leq d + \delta/2 \) means that \( |x_i - \tilde{x}_i| \leq d + \delta/2 \) and \( ||f_j - \tilde{f}_j|| \leq d + \delta/2 \), i.e., \( |f_j(u) - \tilde{f}_j(u)| \leq d + \delta/2 \) for an arbitrary \( u \in U_j \).

The first inequality \( (|x_i - \tilde{x}_i| \leq d + \delta/2) \) easily follows from the given inequalities: namely, \( |x_i - \tilde{x}_i| \leq d < d + \delta/2 \). So, it is sufficient to prove that \( |f_j(u) - \tilde{f}_j(u)| \leq d + \delta/2 \).

Let us prove it. Take an arbitrary \( u \in U_j \). Since \( u_j^{(i)} \) form a \( \beta_j \)-net, there exists an \( i \) for which \( \rho(u, u_j^{(i)}) \leq \beta_j \). Since both \( f_j \) and \( \tilde{f}_j \) correspond to the states from \( S \), from our choice of \( \beta_j \), we can conclude that \( |f_j(u) - \tilde{f}_j(u_j^{(i)})| \leq \delta/4 \) and
\[|\tilde{f}_j(u) - \tilde{f}_j(u_j^{(i)})| \leq \delta/4. \text{ Therefore,} \]
\[|f_j(u) - \tilde{f}_j(u)| \leq |f_j(u) - f_j(u_j^{(i)})| + |f_j(u_j^{(i)}) - \tilde{f}_j(u_j^{(i)})| + |\tilde{f}_j(u_j^{(i)}) - \tilde{f}_j(u)| \leq \delta/4 + d + \delta/4 = d + \delta/2. \]

So, this choice of \( V \) really guarantees the desired inequality. 5.4°. Let us now choose \( \varepsilon_u \).

As we have already mentioned in 4.4.3°, we must choose \( \varepsilon_u \) in such a way that if \( d_V(V, \tilde{V}) \leq \varepsilon_u \), then \( d(\pi_{\tilde{V}}(s), \pi_V(s)) \leq \varepsilon_u/3 \). According to the definitions of \( d \) and \( d_V \), this means that if \( \rho(u_j^{(i)}, \tilde{u}_j^{(i)}) \leq \varepsilon_u \), then \( |f_j(u_j^{(i)}) - f_j(\tilde{u}_j^{(i)})| \leq \varepsilon_u/3 = \delta/8 \).

Just like in 5.3.2°, we conclude that since \( S \) is compact, the functions \( f_j \) are equicontinuous, and for each \( j \) from 1 to \( m \), there exists a \( \gamma_j \) such that if \( \rho(u_j, \tilde{u}_j) \leq \gamma_j \), then \( |f_j(u_j) - f_j(\tilde{u}_j)| \leq \delta/8 \).

So, it is sufficient to take \( \varepsilon_u = \min_j \gamma_j \).

6°. For the above-described choice of \( s_q, V \), and \( \varepsilon_u \), we satisfy the conditions from 4.4.3° and therefore, the function
\[ J^a(\pi_{\tilde{V}}(s)) = \frac{\sum_{i=1}^M \phi(\pi_V(s_j), \pi_{\tilde{V}}(s))J(s)}{\sum_{i=1}^M \phi(\pi_V(s_j), \pi_{\tilde{V}}(s))} \]
is everywhere defined and satisfies the inequality \( \| J(s) - J^a(\pi_{\tilde{V}}(s)) \| \leq \varepsilon/2 \). So, to complete the proof, it is necessary to approximate \( J^a \) by a neural network.

6.1°. To be able to apply the classical approximation theorem, we must show that this function \( J^a \) is a continuous function on a compact. Let us prove it.

The expression \( \phi(\pi_V(s), \pi_{\tilde{V}}(s)) = \max\{0, \delta/2 - d(\pi_V(s), \pi_{\tilde{V}}(s))\} \) is continuous. The functions \( J^a \) is obtained from these functions by addition, multiplication and division, hence it is also continuous. Let us now prove that its domain is a compact set. This domain coincides with the set of all expressions \( \pi_{\tilde{V}}(s) \) for \( s \in S \). From the above inequality \( d(\pi_V(s), \pi_{\tilde{V}}(s)) \leq \rho(s, \tilde{s}) \), it follows that \( \pi_V(s) \) is continuous on \( S \). The values \( \tilde{V} \) are finite sequences of elements of compacts \( U_j \), so we can define a topology on the set of all such \( \tilde{V} \), and in this topology, the set of all possible \( \tilde{V} \) is also compact. Arguments like the ones in 5.4°, prove that \( \pi_{\tilde{V}}(s) \) is a continuous function of \( \tilde{V} \). So, the function \( \pi \) that transform \( V \) and \( s \) into \( \pi_{\tilde{V}}(s) \), is a continuous function on a compact set. Therefore, its image is also compact, and this image is precisely the domain of \( J^a \).

6.2°. Now, we are ready to apply the classical approximation theorem and thus conclude our proof.

Namely, we have a continuous function \( J^a \) on a finite-dimensional compact \( \pi_{\tilde{V}}(K) \). According to the classical approximation theorem, there exists a function \( F \) from \( \pi_{\tilde{V}}(K) \) into \( R \) that is representable by a 3-layer neural network, and that is \((\varepsilon/2)\)—close to \( J^a \). In other words, \[ |J^a_l(\pi_{\tilde{V}}(s)) - F_l(\pi_{\tilde{V}}(s))| \leq \varepsilon/2 \text{ for all } s, l, \text{ and } \tilde{V}. \]

For this function \( F \), the following inequality is true:
\[ |J_l(s) - F_l(\pi_{\tilde{V}}(s))| \leq |J_l(s) - J^a_l(\pi_{\tilde{V}}(s))| + |J^a_l(\pi_{\tilde{V}}(s)) - F_l(\pi_{\tilde{V}}(s))| \leq \varepsilon/2 + \varepsilon/2 = \varepsilon. \]
Q.E.D.
10.3. Proof of Theorem 1′
Theorem 1′ is a reformulation of Theorem 1 in purely mathematical terms, so, having proved Theorem 1, we actually proved Theorem 1′ as well.

10.4. Proof of Theorem 3
Theorem 3 follows from Theorem 1′: Indeed, a universal control strategy is defined as a continuous function from $\mathcal{K} = D \times O \times S$ to $\mathbb{R}^q$. This set $\mathcal{K}$ is a compact subset of $\mathcal{K} = D \times O \times S = (C(S \times C)^n \times C(U_1 \times S \times C) \times \ldots \times C(U_m \times S \times C)) \times C(T) \times (\mathbb{R}^n \times C(U_1) \times \ldots \times C(U_m))$. The set $\mathcal{K}$ is of the type that we considered in Theorem 1′, with compact sets $S \times C, \ldots, S \times C, U_1 \times S \times C, \ldots, U_m \times S \times C, U_1, \ldots, U_m$ playing the role of $U_i$. So, according to Theorem 1′, any function from a compact set $\mathcal{K} \subset \mathcal{K}$ can be approximated by a 3-layer neural network. Q.E.D.

10.5. Proofs of Theorems 4–6
These proofs are similar to the proofs of Theorems 1–3, with the only difference that to approximate a function by a 3-layer neural network, we should use the result from (Sirisaengtaksin & Kreinovich, 1993) (formulated in Section 4) instead of a classical approximation result (Hornik et al, 1989) (see Section 2).

11. CONCLUSIONS
Classical results from (Hecht-Nielsen, 1987; Cybenko, 1989; Funahashi, 1989; Hornik et al 1989; Hecht-Nielsen, 1990). prove that 3-layer neural networks can approximate any continuous function with a given accuracy. These theorems mean that for a plant whose state can be described by finitely many parameters, for an arbitrary control $f$, and for an arbitrary accuracy $\varepsilon > 0$, we can implement this control with a given accuracy using 3-layer neural networks.

This result covers many important control problems, but still it does not cover all of them.

We prove two generalizations of this result, that show that neural networks can be used in the most general control situations. Namely, we proved the following two results:

- Neural networks can approximate the control of the distributed systems (i.e., plants that require infinitely many parameters to describe their current state).

- One can design a universal neural network controller: we input the description of a plant, the desired objective, and generate an $\varepsilon$–optimal control.

In mathematical terms, we prove that 3-layer networks can approximate arbitrary continuous functionals.

We also prove that these approximations can be implemented in a design that is not sensitive to the parameters of the neurons, i.e., the approximation properties will still be there if we take into consideration the fact that the input-output characteristics of an actual hardware neuron can be slightly different from the desired mathematical function.

ACKNOWLEDGEMENTS
This work was supported in part by NSF Grant No. CDA-9015006, NSF Research Opportunity Award (for O.S.), NASA Research Grant No. 9-482, and a grant from
the Institute for Materials and Manufacturing Management. The authors are greatly thankful to Sergey Aityan, James Buckley, Yi-Chieh Chang, Bart Kosko, Arnold Neumaier, and Ronald Yager for valuable discussions, to Kurt Hornik, Maxwell Stinchcombe and Halbert White for their important suggestions and comments on neural approximations, and to the anonymous referee for valuable comments.

REFERENCES

Sirisaengtaksin, O., & V. Kreinovich (1993). Neural networks that are not sensitive to the imprecision of hardware neurons, Interval Computations (to appear).

APPENDIX A: REFORMULATION OF THE APPROXIMATION RESULTS IN GENERAL MATHEMATICAL TERMS

A.1. Neurons are universal approximators for functionals

Definition 10’. Assume that \( n, m \) and \( p \) are positive integers, \( U_1, \ldots, U_m \) are compact metric spaces, \( S \subset \mathbb{R}^n \times C(U_1) \times \ldots \times C(U_m) \). Suppose that \( \varepsilon > 0 \). We say that control functionals \( J : S \to \mathbb{R}^p \) and \( \tilde{J} : S \to \mathbb{R}^p \) are \( \varepsilon \)-close if for all \( s \in S \), \( \| J(s) - \tilde{J}(s) \| \leq \varepsilon \).

Definition 11’. Assume that \( n, m \) and \( p \) are positive integers, \( U_1, \ldots, U_m \) are compact metric spaces, \( S \subset \mathbb{R}^n \times C(U_1) \times \ldots \times C(U_m) \). We say that a functional \( \tilde{J} : S \to \mathbb{R}^p \) is representable by a 3-layer neural network if

\[
\tilde{J}(x_1, \ldots, x_n, f_1, \ldots, f_m) = F(x_1, \ldots, x_n, f_1(u_1^{(1)}), \ldots, f_1(u_1^{(N_1)}), \ldots, f_m(u_m^{(1)}), \ldots, f_m(u_m^{(N_m)}))
\]
for some elements $u^{(1)}_1 \in U_1, ..., u^{(N_1)}_1 \in U_1, ..., u^{(1)}_m \in U_m, ..., u^{(N_m)}_m \in U_m$, and for some function $F$ that is representable by a 3-layer neural network.

**THEOREM 1’.** Assume that $n, m$ and $p$ are positive integers, $U_1, ..., U_m$ are compact metric spaces, $S \subset \mathbb{R}^n \times C(U_1) \times ... \times C(U_m)$ is a compact set, $J : S \to \mathbb{R}^p$ is continuous, and $\varepsilon > 0$ is a real number. Then, there exists a functional $\tilde{J}$ that is representable by a 3-layer neural network, and that is $\varepsilon$-close to $J$ on $S$.

**A.2. Taking location error into consideration**

**Definition 12’.** Assume that $n, m$ and $p$ are positive integers, $S \subset \mathbb{R}^n \times C(U_1) \times ... \times C(U_m)$ and $J : S \to \mathbb{R}^p$ is a mapping. Suppose also that $\varepsilon > 0$ and $\varepsilon_u > 0$ are real numbers. The value $\varepsilon_u$ will be called a location error. We say that for location error $\varepsilon_u$, a mapping $J$ is $\varepsilon$-approximable by a 3-layer neural network if for every $j$ from 1 to $m$, there exist values $u^{(1)}_j, ..., u^{(N_j)}_j \in U_j$, and there exists a function $F$ that is representable by a 3-layer neural network such that for all $(x_1, ..., x_n, f_1, ..., f_m) \in S$, and for any $\tilde{u}^{(i)}_j$ such that $\rho_j(u^{(i)}_j, \tilde{u}^{(i)}_j) \leq \varepsilon_u$, we have

$$\| J(s) - F(x_1, ..., x_n, f_1(u^{(1)}_1), ..., f_1(u^{(N_1)}_1), ..., f_m(u^{(1)}_m), ..., f_m(u^{(N_m)}_m)) \| \leq \varepsilon.$$ 

**THEOREM 2’.** Assume that $n, m$ and $p$ are positive integers, $S \subset \mathbb{R}^n \times C(U_1) \times ... \times C(U_m)$ is a compact set, $J : S \to \mathbb{R}^p$ is continuous, and $\varepsilon > 0$ is a real number. Then, there exists a real number $\varepsilon_u > 0$ such that for location error $\varepsilon_u$, $J$ is $\varepsilon$-approximable by a 3-layer neural network.

**A.3. Hardware neurons can approximate functionals**

**Definition 26’.** Assume that $n, m$ and $p$ are positive integers, $U_1, ..., U_m$ are compact metric spaces, $S \subset \mathbb{R}^n \times C(U_1) \times ... \times C(U_m)$. We say that a functional $J : S \to \mathbb{R}^p$ is $\varepsilon$-approximated by a design $\mu$, if for some elements $u^{(1)}_1 \in U_1, ..., u^{(N_1)}_1 \in U_1, ..., u^{(1)}_m \in U_m, ..., u^{(N_m)}_m \in U_m$, and for every implementation $\tilde{F}$ of this design, the functional

$$\tilde{J}(x_1, ..., x_n, f_1, ..., f_m) = \tilde{F}(x_1, ..., x_n, f_1(u^{(1)}_1), ..., f_1(u^{(N_1)}_1), ..., f_m(u^{(1)}_m), ..., f_m(u^{(N_m)}_m))$$

is $\varepsilon$-close to $J$.

**THEOREM 4’.** Assume that $n, m$ and $p$ are positive integers, $U_1, ..., U_m$ are compact metric spaces, $S \subset \mathbb{R}^n \times C(U_1) \times ... \times C(U_m)$ is a compact set, $J : S \to \mathbb{R}^p$ is continuous, and $\varepsilon > 0$ is a real number. Then, there exists a design $\mu$ that $\varepsilon$-approximates $J$.

**Definition 27’.** Assume that $n, m$ and $p$ are positive integers, $S \subset \mathbb{R}^n \times C(U_1) \times ... \times C(U_m)$ and $J : S \to \mathbb{R}^p$ is a mapping. Suppose also that $\varepsilon > 0$ and $\varepsilon_u > 0$ are real numbers. The value $\varepsilon_u$ will be called a location error. We say that for location error $\varepsilon_u$, a mapping $J$ is $\varepsilon$-approximated by a design $\mu$, if for every $j$ from 1 to $m$, there exist values $u^{(1)}_j, ..., u^{(N_j)}_j \in U_j$, such that for any implementation $\tilde{F}$ of $\mu$, for all $(x_1, ..., x_n, f_1, ..., f_m) \in S$, and for any $\tilde{u}^{(i)}_j$ such that $\rho_j(u^{(i)}_j, \tilde{u}^{(i)}_j) \leq \varepsilon_u$, we have

$$| J(s) - \tilde{F}(x_1, ..., x_n, f_1(u^{(1)}_1), ..., f_1(u^{(N_1)}_1), ..., f_m(u^{(1)}_m), ..., f_m(u^{(N_m)}_m)) | \leq \varepsilon.$$
THEOREM 5’. Assume that \( n, m \) and \( p \) are positive integers, \( S \subset R^n \times C(U_1) \times \ldots \times C(U_m) \) is a compact set, \( J : S \to R^p \) is continuous, and \( \varepsilon > 0 \) is a real number. Then, there exists a real number \( \varepsilon_u > 0 \) and a design \( \mu \) such that for location error \( \varepsilon_u, \mu \varepsilon\text{-approximates } J \).