

Universal approximation of set-valued maps and application to control

Francisco Periago

Universidad Politécnica de Cartagena

<https://multisimo.com/fpe/> - <https://github.com/fperiago>
joint work with Carlos García-Cervera (UCSB, USA); Mathieu Kessler (UPCT, Spain) and Pablo Pedregal (UCLM, Spain)

Congreso Bienal de la Real Sociedad Matemática Española

Alicante, January 19-23, 2026

Motivation and Main Goal



Consider the control system

$$\begin{cases} y'(t) = Ay(t) + Bu(t), & t > 0 \\ y(0) = y^0 \in X, \end{cases} \quad (1)$$

where

- $A : D(A) \subset X \rightarrow X$ is an operator,
- $y = y(t) \in D(A)$ is the state variable,
- $u = u(t) \in \mathcal{U}$ is the control variable, and
- B is an admissible control operator

Consider the control system

$$\begin{cases} y'(t) = Ay(t) + Bu(t), & t > 0 \\ y(0) = y^0 \in X, \end{cases} \quad (1)$$

where

- $A : D(A) \subset X \rightarrow X$ is an operator,
- $y = y(t) \in D(A)$ is the state variable,
- $u = u(t) \in \mathcal{U}$ is the control variable, and
- B is an admissible control operator

Examples of set- valued maps and operators arising in this context

Consider the control system

$$\begin{cases} y'(t) = Ay(t) + Bu(t), & t > 0 \\ y(0) = y^0 \in X, \end{cases} \quad (1)$$

where

- $A : D(A) \subset X \rightarrow X$ is an operator,
- $y = y(t) \in D(A)$ is the state variable,
- $u = u(t) \in \mathcal{U}$ is the control variable, and
- B is an admissible control operator

Examples of set- valued maps and operators arising in this context

- Reachability set at time T :

$$\mathbb{R}^n \ni y^0 \Rightarrow \mathbb{R}^n, \quad F_T(y^0) = \mathcal{R}(T, y^0) := \left\{ y(T; y^0, u(\cdot)) \mid u(\cdot) \in \mathcal{U} \right\}$$

Consider the control system

$$\begin{cases} y'(t) = Ay(t) + Bu(t), & t > 0 \\ y(0) = y^0 \in X, \end{cases} \quad (1)$$

where

- $A : D(A) \subset X \rightarrow X$ is an operator,
- $y = y(t) \in D(A)$ is the state variable,
- $u = u(t) \in \mathcal{U}$ is the control variable, and
- B is an admissible control operator

Examples of set- valued maps and operators arising in this context

- Reachability set at time T :

$$\mathbb{R}^n \ni y^0 \Rightarrow \mathbb{R}^n, \quad F_T(y^0) = \mathcal{R}(T, y^0) := \left\{ y(T; y^0, u(\cdot)) \mid u(\cdot) \in \mathcal{U} \right\}$$

- Solution map for a controllability problem

$$X \ni y^0 \Rightarrow \mathcal{U}, \quad \mathcal{G}(y^0) = \left\{ u(\cdot; y^0) \in \mathcal{U} \mid y(T) = y^T \right\}$$

Consider the control system

$$\begin{cases} y'(t) = Ay(t) + Bu(t), & t > 0 \\ y(0) = y^0 \in X, \end{cases} \quad (1)$$

where

- $A : D(A) \subset X \rightarrow X$ is an operator,
- $y = y(t) \in D(A)$ is the state variable,
- $u = u(t) \in \mathcal{U}$ is the control variable, and
- B is an admissible control operator

Examples of set- valued maps and operators arising in this context

- Reachability set at time T :

$$\mathbb{R}^n \ni y^0 \Rightarrow \mathbb{R}^n, \quad F_T(y^0) = \mathcal{R}(T, y^0) := \left\{ y(T; y^0, u(\cdot)) \mid u(\cdot) \in \mathcal{U} \right\}$$

- Solution map for a controllability problem

$$X \ni y^0 \Rightarrow \mathcal{U}, \quad \mathcal{G}(y^0) = \left\{ u(\cdot; y^0) \in \mathcal{U} \mid y(T) = y^T \right\}$$

Goal: develop a data-driven method to learn this class of set-valued functions and operators

A first partial answer

Strategy

- 1 Prove the existence of a suitable selection of the set-valued map
- 2 Learn such a selection, which is single-valued, by using Deep Operator Networks (DeepONet)

Strategy

- 1 Prove the existence of a suitable selection of the set-valued map
- 2 Learn such a selection, which is single-valued, by using Deep Operator Networks (DeepONet)

Theorem (Michael)

Let F be a lower semicontinuous set-valued map with closed, convex values from a compact metric space X to a Banach space Y . It does have a continuous selection.

This is Th. 9.1.2 in



J.-P. Aubin, H. Frankowska, *Set-valued analysis*, Vol. 2 of Systems & Control: Foundations & Applications, Birkhäuser Boston, 1990

Strategy

- 1 Prove the existence of a suitable selection of the set-valued map
- 2 Learn such a selection, which is single-valued, by using Deep Operator Networks (DeepONet)

Theorem (Michael)

Let F be a lower semicontinuous set-valued map with closed, convex values from a compact metric space X to a Banach space Y . It does have a continuous selection.

This is Th. 9.1.2 in



J.-P. Aubin, H. Frankowska, *Set-valued analysis*, Vol. 2 of Systems & Control: Foundations & Applications, Birkhäuser Boston, 1990

For controllability problems, **continuity is a consequence of observability**

$$\|u\| \leq \|y^0\|$$

For the sake of clarity, we focus on the control problem

$$\left\{ \begin{array}{ll} y_{tt} - y_{xx} = 0, & \text{in } (0, 1) \times (0, 2) \\ y(x, 0) = y^0(x), & \text{on } (0, 1) \\ y_t(x, 0) = y^1(x) & \text{on } (0, 1) \\ y(0, t) = 0, & \text{on } (0, 2) \\ y(1, t) = u(t) & \text{on } (0, 2) \\ y(x, 2) = y_t(x, 2) = 0, & \text{on } (0, 1). \end{array} \right.$$

For the sake of clarity, we focus on the control problem

$$\left\{ \begin{array}{ll} y_{tt} - y_{xx} = 0, & \text{in } (0, 1) \times (0, 2) \\ y(x, 0) = y^0(x), & \text{on } (0, 1) \\ y_t(x, 0) = y^1(x) & \text{on } (0, 1) \\ y(0, t) = 0, & \text{on } (0, 2) \\ y(1, t) = u(t) & \text{on } (0, 2) \\ y(x, 2) = y_t(x, 2) = 0, & \text{on } (0, 1). \end{array} \right.$$

Goal: approximate the solution map

$$\begin{aligned} \mathcal{G} : \quad L^2(0, 1) \times H^{-1}(0, 1) &\rightarrow L^2(0, 2) \\ (y^0, y^1) &\mapsto \mathcal{G}(y^0, y^1) := u \end{aligned}$$

where u is the **unique** control of minimal L^2 -norm.

For the sake of clarity, we focus on the control problem

$$\left\{ \begin{array}{ll} y_{tt} - y_{xx} = 0, & \text{in } (0, 1) \times (0, 2) \\ y(x, 0) = y^0(x), & \text{on } (0, 1) \\ y_t(x, 0) = y^1(x) & \text{on } (0, 1) \\ y(0, t) = 0, & \text{on } (0, 2) \\ y(1, t) = u(t) & \text{on } (0, 2) \\ y(x, 2) = y_t(x, 2) = 0, & \text{on } (0, 1). \end{array} \right.$$

Goal: approximate the solution map

$$\begin{aligned} \mathcal{G} : \quad L^2(0, 1) \times H^{-1}(0, 1) &\rightarrow L^2(0, 2) \\ (y^0, y^1) &\mapsto \mathcal{G}(y^0, y^1) := u \end{aligned}$$

where u is the **unique** control of minimal L^2 -norm.

$$\|u\|_{L^2(0,2)} \leq C \left(\|y^0\|_{L^2(0,1)} + \|y^1\|_{H^{-1}(0,1)} \right)$$

■ Dataset

Fix a set of **sensor points** $\{x_1, x_2, \dots, x_m\} \subset [0, 1]$. Then,

$$(y^0, y^1) \mapsto (y^0(x_1), y^0(x_2), \dots, y^0(x_m); y^1(x_1), y^1(x_2), \dots, y^1(x_m)) \equiv y^{\text{initial}}$$

■ Dataset

Fix a set of **sensor points** $\{x_1, x_2, \dots, x_m\} \subset [0, 1]$. Then,

$$(y^0, y^1) \mapsto (y^0(x_1), y^0(x_2), \dots, y^0(x_m); y^1(x_1), y^1(x_2), \dots, y^1(x_m)) \equiv y^{\text{initial}}$$

Take $t \in [0, 2]$.

■ Dataset

Fix a set of **sensor points** $\{x_1, x_2, \dots, x_m\} \subset [0, 1]$. Then,

$$(y^0, y^1) \mapsto (y^0(x_1), y^0(x_2), \dots, y^0(x_m); y^1(x_1), y^1(x_2), \dots, y^1(x_m)) \equiv y^{\text{initial}}$$

Take $t \in [0, 2]$. **Features** are

$$\{(y_\ell^{\text{initial}}; t_\ell), \quad 1 \leq \ell \leq N\}$$

■ Dataset

Fix a set of **sensor points** $\{x_1, x_2, \dots, x_m\} \subset [0, 1]$. Then,

$$(y^0, y^1) \mapsto (y^0(x_1), y^0(x_2), \dots, y^0(x_m); y^1(x_1), y^1(x_2), \dots, y^1(x_m)) \equiv y^{\text{initial}}$$

Take $t \in [0, 2]$. **Features** are

$$\{(y_\ell^{\text{initial}}; t_\ell), \quad 1 \leq \ell \leq N\}$$

and **labels**

$$\{u_\ell = u(y_\ell^{\text{initial}}; t_\ell), \quad 1 \leq \ell \leq N\},$$

■ Dataset

Fix a set of **sensor points** $\{x_1, x_2, \dots, x_m\} \subset [0, 1]$. Then,

$$(y^0, y^1) \mapsto (y^0(x_1), y^0(x_2), \dots, y^0(x_m); y^1(x_1), y^1(x_2), \dots, y^1(x_m)) \equiv y^{\text{initial}}$$

Take $t \in [0, 2]$. **Features** are

$$\{(y_\ell^{\text{initial}}; t_\ell), \quad 1 \leq \ell \leq N\}$$

and **labels**

$$\{u_\ell = u(y_\ell^{\text{initial}}; t_\ell), \quad 1 \leq \ell \leq N\},$$

■ Hypothesis space: the neural network DeepONet

$$\mathcal{N}(\theta; (y^{\text{initial}}(x_j); t)) := \sum_{k=1}^p \sum_{i=1}^n c_i^k \sigma \left(\sum_{j=1}^m \xi_{ij}^k y^{\text{initial}}(x_j) + \theta_i^k \right) \cdot \sigma(w_k \cdot t + \eta_k)$$

$\theta = (c_i^k, \xi_{ij}^k, \theta_i^k, w_k, \eta_k)$ is the set of parameters of the net.

■ Dataset

Fix a set of **sensor points** $\{x_1, x_2, \dots, x_m\} \subset [0, 1]$. Then,

$$(y^0, y^1) \mapsto (y^0(x_1), y^0(x_2), \dots, y^0(x_m); y^1(x_1), y^1(x_2), \dots, y^1(x_m)) \equiv y^{\text{initial}}$$

Take $t \in [0, 2]$. **Features** are

$$\{(y_\ell^{\text{initial}}; t_\ell), \quad 1 \leq \ell \leq N\}$$

and **labels**

$$\{u_\ell = u(y_\ell^{\text{initial}}; t_\ell), \quad 1 \leq \ell \leq N\},$$

■ Hypothesis space: the neural network DeepONet

$$\mathcal{N}(\theta; (y^{\text{initial}}(x_j); t)) := \sum_{k=1}^p \sum_{i=1}^n c_i^k \sigma \left(\sum_{j=1}^m \xi_{ij}^k y^{\text{initial}}(x_j) + \theta_i^k \right) \cdot \sigma(w_k \cdot t + \eta_k)$$

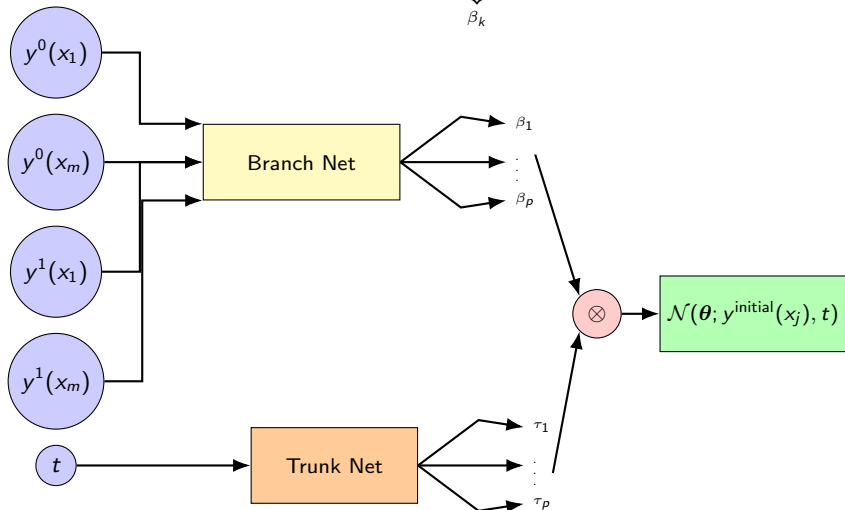
$\theta = (c_i^k, \xi_{ij}^k, \theta_i^k, w_k, \eta_k)$ is the set of parameters of the net.

■ Loss function: Mean Squared Error (MSE)

$$\text{MSE}(\theta) = \frac{1}{N} \sum_{\ell=1}^N |\mathcal{N}(\theta; (y_\ell^{\text{initial}}; t_\ell)) - u_\ell|^2$$

DeepONet's architecture

$$\mathcal{N}(\theta; (y^{\text{initial}}(x_j); t)) := \underbrace{\sum_{k=1}^p \sum_{i=1}^n c_i^k \sigma \left(\sum_{j=1}^m \xi_{ij}^k y^{\text{initial}}(x_j) + \theta_i^k \right)}_{\beta_k} \cdot \underbrace{\sigma(w_k \cdot t + \eta_k)}_{\tau_k}$$



Approximation scheme

$$\begin{array}{ccc} X & \xrightarrow{\mathcal{G}} & Y \\ \downarrow \mathcal{E} & & \uparrow \mathcal{R} \\ \mathbb{R}^{2m} & \xrightarrow{\mathcal{A}} & \mathbb{R}^p \end{array}$$

Approximation scheme

$$\begin{array}{ccc} X & \xrightarrow{\mathcal{G}} & Y \\ \downarrow \mathcal{E} & & \uparrow \mathcal{R} \\ \mathbb{R}^{2m} & \xrightarrow{\mathcal{A}} & \mathbb{R}^p \end{array}$$

- $\mathcal{E} : (y^0, y^1) \mapsto y^{initial}$ is an *encoder*

Approximation scheme

$$\begin{array}{ccc} X & \xrightarrow{\mathcal{G}} & Y \\ \downarrow \mathcal{E} & & \uparrow \mathcal{R} \\ \mathbb{R}^{2m} & \xrightarrow{\mathcal{A}} & \mathbb{R}^p \end{array}$$

- $\mathcal{E} : (y^0, y^1) \mapsto y^{initial}$ is an *encoder*
- $\mathcal{A} : y^{initial} \mapsto \beta_k(y^{initial})$ is a finite dimensional *approximation operator*

Approximation scheme

$$\begin{array}{ccc} X & \xrightarrow{\mathcal{G}} & Y \\ \downarrow \mathcal{E} & & \uparrow \mathcal{R} \\ \mathbb{R}^{2m} & \xrightarrow{\mathcal{A}} & \mathbb{R}^p \end{array}$$

- $\mathcal{E} : (y^0, y^1) \mapsto y^{initial}$ is an *encoder*
- $\mathcal{A} : y^{initial} \mapsto \beta_k(y^{initial})$ is a finite dimensional *approximation operator*
- $\mathcal{R} : \beta_k(y^{initial}) \mapsto \sum_{k=1}^p \beta_k(y^{initial}) \tau_k(t)$ is a *reconstruction operator*.

Thus,

$$\mathcal{G} \approx \mathcal{R} \circ \mathcal{A} \circ \mathcal{E}.$$

Where does this architecture come from ?



Theorem (Universal Approximation Theorem for Functions)

Suppose that $K \subset \mathbb{R}^d$ is compact, $U \subset C(K)$ is compact, and $\sigma \in C^\infty(\mathbb{R})$ is not a polynomial. Then, for any $\varepsilon > 0$ there exist a positive integer n , real numbers $\theta_i, \omega_i \in \mathbb{R}^n$, independent of $f \in U$, and constants $c_i = c_i(f)$ depending on f , such that

$$|f(x) - \sum_{i=1}^n c_i \sigma(\omega_i \cdot x + \theta_i)| < \varepsilon$$

holds for all $x \in K$ and $f \in U$. Moreover, each $c_i(f)$ is a continuous linear functional defined on U .

Theorem (Universal Approximation Theorem for Functions)

Suppose that $K \subset \mathbb{R}^d$ is compact, $U \subset C(K)$ is compact, and $\sigma \in C^\infty(\mathbb{R})$ is not a polynomial. Then, for any $\varepsilon > 0$ there exist a positive integer n , real numbers $\theta_i, \omega_i \in \mathbb{R}^n$, independent of $f \in U$, and constants $c_i = c_i(f)$ depending on f , such that

$$|f(x) - \sum_{i=1}^n c_i \sigma(\omega_i \cdot x + \theta_i)| < \varepsilon$$

holds for all $x \in K$ and $f \in U$. Moreover, each $c_i(f)$ is a continuous linear functional defined on U .

Theorem (Universal Approximation Theorem for Functionals)

... for a continuous functional $f : V \rightarrow \mathbb{R}$ we have that

$$|f(y) - \sum_{i=1}^n c_i \sigma \left(\sum_{j=1}^m \xi_{ij} y(x_j) + \theta_i \right)| < \varepsilon, \quad \text{for all } y \in V.$$

Where does this architecture come from ?



... for an operator,

$$\mathcal{G}(y)(t) \approx \sum_{k=1}^n c_k (\mathcal{G}(y)) \sigma(\omega_k \cdot t + \theta_k),$$

where $c_k (\mathcal{G}(y))$ are continuous functionals.

Where does this architecture come from ?

... for an operator,

$$\mathcal{G}(y)(t) \approx \sum_{k=1}^n c_k (\mathcal{G}(y)) \sigma(\omega_k \cdot t + \theta_k),$$

where $c_k (\mathcal{G}(y))$ are continuous functionals.

Theorem (Universal Approximation Theorem for Operators)

Suppose that σ is not a polynomial, X is a Banach space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are compact sets, V is a compact set in $C(K_1)$, and $\mathcal{G} : V \rightarrow C(K_2)$ is a continuous operator. Then for any $\varepsilon > 0$, there are a positive integers n, p , m sensor points $x_1, x_2, \dots, x_m \in K_1$, and real constants $c_i^k, \theta_i^k, \xi_{ij}^k, \eta_k$ such that

$$|\mathcal{G}(y)(t) - \sum_{k=1}^p \sum_{i=1}^n c_i^k \sigma \left(\sum_{j=1}^m \xi_{ij}^k y(x_j) + \theta_i^k \right) \sigma(\omega_k \cdot t + \zeta_k)| < \varepsilon, \quad \forall y \in V, t \in K_2$$



Chen, T., Chen, H.: *Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems.* **IEEE Transactions on Neural Networks** 6(4), 911-917, 1995.

- Operator $\mathcal{G} : X \rightarrow Y$

- Operator $\mathcal{G} : X \rightarrow Y$
- Probability measure $\mu \in \mathcal{P}_2(X)$

- Operator $\mathcal{G} : X \rightarrow Y$
- Probability measure $\mu \in \mathcal{P}_2(X)$
- Encoding operator $\mathcal{E} : X \rightarrow \mathbb{R}^m$

- Operator $\mathcal{G} : X \rightarrow Y$
- Probability measure $\mu \in \mathcal{P}_2(X)$
- Encoding operator $\mathcal{E} : X \rightarrow \mathbb{R}^m$

How to sample continuous functions for the initial data (y^0, y^1) ?

- Operator $\mathcal{G} : X \rightarrow Y$
- Probability measure $\mu \in \mathcal{P}_2(X)$
- Encoding operator $\mathcal{E} : X \rightarrow \mathbb{R}^m$

How to sample continuous functions for the initial data (y^0, y^1) ?

$$\mu \sim \sum_{j=1}^{\infty} \sqrt{\lambda_j} \xi_j(\omega) \varphi_j(x) \quad (2)$$

- $\xi_j \sim \mathcal{N}(0, 1)$ i.i.d. standard Gaussian variables
- (λ_j, φ_j) are the eigenvalues and normalized eigenfunctions of the operator

$$\mathcal{C}(\phi)(x) = \int_0^1 C(x, x') \phi(x') dx', \quad C(x, x') = \sigma^2 \exp\left(-\frac{|x - x'|^2}{2\ell^2}\right)$$

- Operator $\mathcal{G} : X \rightarrow Y$
- Probability measure $\mu \in \mathcal{P}_2(X)$
- Encoding operator $\mathcal{E} : X \rightarrow \mathbb{R}^m$

How to sample continuous functions for the initial data (y^0, y^1) ?

$$\mu \sim \sum_{j=1}^{\infty} \sqrt{\lambda_j} \xi_j(\omega) \varphi_j(x) \quad (2)$$

- $\xi_j \sim \mathcal{N}(0, 1)$ i.i.d. standard Gaussian variables
- (λ_j, φ_j) are the eigenvalues and normalized eigenfunctions of the operator

$$\mathcal{C}(\phi)(x) = \int_0^1 C(x, x') \phi(x') dx', \quad C(x, x') = \sigma^2 \exp\left(-\frac{|x - x'|^2}{2\ell^2}\right)$$

Thus, we truncate (2) and sample the Gaussian random variables.

$\{\varphi_j\}$ is an orthonormal basis of $L^2(0, 1)$ (Mercer's Theorem)

Putting all together, the **training dataset** takes the form

$$\begin{bmatrix} X_{\text{train}} & \vdots & y_{\text{train}} \end{bmatrix} = \begin{bmatrix} y_{1,1}^{\text{initial}} & \cdots & y_{1,2m}^{\text{initial}} & \vdots & u_1(t_1) \\ \cdots & \cdots & \cdots & \vdots & \cdots \\ y_{1,1}^{\text{initial}} & \cdots & y_{1,2m}^{\text{initial}} & \vdots & u_1(t_\ell) \\ \hline \cdots & \cdots & \cdots & \vdots & \cdots \\ \cdots & \cdots & \cdots & \vdots & \cdots \\ \hline y_{N,1}^{\text{initial}} & \cdots & y_{N,2m}^{\text{initial}} & \vdots & u_N(t_1) \\ \cdots & \cdots & \cdots & \vdots & \cdots \\ y_{N,1}^{\text{initial}} & \cdots & y_{N,2m}^{\text{initial}} & \vdots & u_N(t_\ell) \end{bmatrix}$$

Error analysis and the curse of dimensionality



When we talk about error, what are we talking about?

When we talk about error, what are we talking about?

- **Approximation error:** This is the distance between the Hypothesis space and the operator $\mathcal{G}$ to be approximated, i.e., if $\mathcal{F}$ is a fixed space of DeepONets, then

$$\mathcal{N}_{\mathcal{F}} = \arg \min_{\mathcal{N} \in \mathcal{F}} \mathcal{L}(\mathcal{N}) := \int_{L^2(D)} \int_U |\mathcal{G}(y)(t) - \mathcal{N}_{\mathcal{F}}(y)(t)|^2 dt d\mu(y),$$

then **approximation error** is

$$\mathcal{E}_{\text{approx}} := \sqrt{\mathcal{L}(\mathcal{N}_{\mathcal{F}})}.$$

When we talk about error, what are we talking about?

- **Approximation error:** This is the distance between the Hypothesis space and the operator $\mathcal{G}$ to be approximated, i.e., if $\mathcal{F}$ is a fixed space of DeepONets, then

$$\mathcal{N}_{\mathcal{F}} = \arg \min_{\mathcal{N} \in \mathcal{F}} \mathcal{L}(\mathcal{N}) := \int_{L^2(D)} \int_U |\mathcal{G}(y)(t) - \mathcal{N}_{\mathcal{F}}(y)(t)|^2 dt d\mu(y),$$

then **approximation error** is

$$\mathcal{E}_{\text{approx}} := \sqrt{\mathcal{L}(\mathcal{N}_{\mathcal{F}})}.$$

- **Generalization (or estimation) error:** We approximate $\mathcal{N}_{\mathcal{F}}$ by using a specific (training) dataset $\mathcal{T}$, and a **empirical loss**. So, we get

$$\mathcal{N}_{\mathcal{T}} = \arg \min_{\mathcal{N} \in \mathcal{F}} \mathcal{L}_M(\mathcal{N}) := \frac{|U|}{M} \sum_{j=1}^M |\mathcal{G}(y_j)(t_j) - \mathcal{N}(y_j)(t_j)|^2$$

Thus, $\mathcal{E}_{\text{gener}} := (\mathcal{L}(\mathcal{N}_{\mathcal{T}}) - \mathcal{L}(\mathcal{N}_{\mathcal{F}}))^{1/2}$.

When we talk about error, what are we talking about?

- **Approximation error:** This is the distance between the Hypothesis space and the operator $\mathcal{G}$ to be approximated, i.e., if $\mathcal{F}$ is a fixed space of DeepONets, then

$$\mathcal{N}_{\mathcal{F}} = \arg \min_{\mathcal{N} \in \mathcal{F}} \mathcal{L}(\mathcal{N}) := \int_{L^2(D)} \int_U |\mathcal{G}(y)(t) - \mathcal{N}_{\mathcal{F}}(y)(t)|^2 dt d\mu(y),$$

then **approximation error** is

$$\mathcal{E}_{\text{approx}} := \sqrt{\mathcal{L}(\mathcal{N}_{\mathcal{F}})}.$$

- **Generalization (or estimation) error:** We approximate $\mathcal{N}_{\mathcal{F}}$ by using a specific (training) dataset $\mathcal{T}$, and a **empirical loss**. So, we get

$$\mathcal{N}_{\mathcal{T}} = \arg \min_{\mathcal{N} \in \mathcal{F}} \mathcal{L}_M(\mathcal{N}) := \frac{|U|}{M} \sum_{j=1}^M |\mathcal{G}(y_j)(t_j) - \mathcal{N}(y_j)(t_j)|^2$$

Thus, $\mathcal{E}_{\text{gener}} := (\mathcal{L}(\mathcal{N}_{\mathcal{T}}) - \mathcal{L}(\mathcal{N}_{\mathcal{F}}))^{1/2}$.

- **Optimization error:** The empirical loss is highly nonlinear, non-convex so that we compute a local minimum $\mathcal{N}_M$ of $\mathcal{L}_M$. Optimization error is then

$$\mathcal{E}_{\text{optim}} := \|\mathcal{N}_M - \mathcal{N}_{\mathcal{T}}\|^{1/2}.$$

$$\mathcal{E}_{\text{total}} = \mathcal{E}_{\text{approx}} + \mathcal{E}_{\text{gener}} + \mathcal{E}_{\text{optim}}$$

Definition (Curse of dimensionality)

For a given $\varepsilon > 0$, let $\mathcal{N}_\varepsilon$ be a DeepONet such that $\mathcal{E}_{\text{approx}} < \varepsilon$, and

$$\text{size}(\mathcal{N}_\varepsilon) \sim \mathcal{O}\left(\varepsilon^{-\vartheta_\varepsilon}\right) \quad \text{for some } \vartheta_\varepsilon \geq 0.$$

Our DeepONet approximation, with underlying measure μ , is said to *incur a curse of dimensionality* if $\lim_{\varepsilon \rightarrow 0} \vartheta_\varepsilon = +\infty$ and *breaks the curse of dimensionality* if $\lim_{\varepsilon \rightarrow 0} \vartheta_\varepsilon = \bar{\vartheta} < +\infty$.

$$\mathcal{E}_{\text{total}} = \mathcal{E}_{\text{approx}} + \mathcal{E}_{\text{gener}} + \mathcal{E}_{\text{optim}}$$

Definition (Curse of dimensionality)

For a given $\varepsilon > 0$, let $\mathcal{N}_\varepsilon$ be a DeepONet such that $\mathcal{E}_{\text{approx}} < \varepsilon$, and

$$\text{size}(\mathcal{N}_\varepsilon) \sim \mathcal{O}\left(\varepsilon^{-\vartheta_\varepsilon}\right) \quad \text{for some } \vartheta_\varepsilon \geq 0.$$

Our DeepONet approximation, with underlying measure μ , is said to *incur a curse of dimensionality* if $\lim_{\varepsilon \rightarrow 0} \vartheta_\varepsilon = +\infty$ and *breaks the curse of dimensionality* if $\lim_{\varepsilon \rightarrow 0} \vartheta_\varepsilon = \bar{\vartheta} < +\infty$.

the learning of Lipschitz functions suffers from the curse of dimensionality.



Yarotsky, D.: *Optimal approximation of continuous functions by very deep relu networks*. **Conference on Learning Theory. PMLR, 639-649, 2018.**

In our setting, m is the number of sensors for the encoding operator $u \mapsto \mathcal{E}(u) = (u(x_1), \dots, u(x_m))$.

However, for some classes of linear and nonlinear operators, the DeepONet approximation may break the curse of dimensionality for **approximation error**.



Lanthaler, S., Mishra, S., Karniadakis, G.E.: *Error estimates for DeepONets: a deep learning framework in infinite dimensions*. **Trans. Math. Appl.** 6(1), 1–144, 2022.

Some examples are:

However, for some classes of linear and nonlinear operators, the DeepONet approximation may break the curse of dimensionality for **approximation error**.



Lanthaler, S., Mishra, S., Karniadakis, G.E.: *Error estimates for DeepONets: a deep learning framework in infinite dimensions*. **Trans. Math. Appl.** **6**(1), 1–144, 2022.

Some examples are:

- holomorphic mappings $[-1, 1]^N \rightarrow V$, with V a Banach space,

However, for some classes of linear and nonlinear operators, the DeepONet approximation may break the curse of dimensionality for **approximation error**.



Lanthaler, S., Mishra, S., Karniadakis, G.E.: *Error estimates for DeepONets: a deep learning framework in infinite dimensions*. **Trans. Math. Appl.** 6(1), 1–144, 2022.

Some examples are:

- holomorphic mappings $[-1, 1]^N \rightarrow V$, with V a Banach space,
- some PDE operators like
 - 1 parametric elliptic PDEs: $\mathcal{G} : a \mapsto y$, where y solves

$$-\nabla \cdot (a \nabla y) = f$$

However, for some classes of linear and nonlinear operators, the DeepONet approximation may break the curse of dimensionality for **approximation error**.



Lanthaler, S., Mishra, S., Karniadakis, G.E.: *Error estimates for DeepONets: a deep learning framework in infinite dimensions*. **Trans. Math. Appl.** **6**(1), 1–144, 2022.

Some examples are:

- holomorphic mappings $[-1, 1]^N \rightarrow V$, with V a Banach space,
- some PDE operators like

- 1 parametric elliptic PDEs: $\mathcal{G} : a \mapsto y$, where y solves

$$-\nabla \cdot (a \nabla y) = f$$

- 2 parabolic nonlinear PDEs: $\mathcal{G} : y^0 \mapsto y(T)$, where y solves

$$\begin{cases} \frac{\partial y}{\partial t} = \Delta y + f(y) \\ y(0) = y^0 \end{cases}$$

However, for some classes of linear and nonlinear operators, the DeepONet approximation may break the curse of dimensionality for **approximation error**.



Lanthaler, S., Mishra, S., Karniadakis, G.E.: *Error estimates for DeepONets: a deep learning framework in infinite dimensions*. **Trans. Math. Appl.** **6**(1), 1–144, 2022.

Some examples are:

- holomorphic mappings $[-1, 1]^N \rightarrow V$, with V a Banach space,
- some PDE operators like

- 1 parametric elliptic PDEs: $\mathcal{G} : a \mapsto y$, where y solves

$$-\nabla \cdot (a \nabla y) = f$$

- 2 parabolic nonlinear PDEs: $\mathcal{G} : y^0 \mapsto y(T)$, where y solves

$$\begin{cases} \frac{\partial y}{\partial t} = \Delta y + f(y) \\ y(0) = y^0 \end{cases}$$

- 3 scalar conservation laws: $\mathcal{G} : y^0 \mapsto y(T)$, where y solves

$$\begin{cases} \partial_t y + \partial_x (f(y)) = 0 \\ y(0) = y^0 \end{cases}$$

However, for some classes of linear and nonlinear operators, the DeepONet approximation may break the curse of dimensionality for **approximation error**.



Lanthaler, S., Mishra, S., Karniadakis, G.E.: *Error estimates for DeepONets: a deep learning framework in infinite dimensions*. **Trans. Math. Appl.** 6(1), 1–144, 2022.

Some examples are:

- holomorphic mappings $[-1, 1]^N \rightarrow V$, with V a Banach space,
- some PDE operators like

- 1 parametric elliptic PDEs: $\mathcal{G} : a \mapsto y$, where y solves

$$-\nabla \cdot (a \nabla y) = f$$

- 2 parabolic nonlinear PDEs: $\mathcal{G} : y^0 \mapsto y(T)$, where y solves

$$\begin{cases} \frac{\partial y}{\partial t} = \Delta y + f(y) \\ y(0) = y^0 \end{cases}$$

- 3 scalar conservation laws: $\mathcal{G} : y^0 \mapsto y(T)$, where y solves

$$\begin{cases} \partial_t y + \partial_x (f(y)) = 0 \\ y(0) = y^0 \end{cases}$$

- bounded linear operators $\mathcal{G} : L^2(D) \rightarrow L^2(U)$

Theorem (DeepONet approximation of controllability systems)

Consider the control system for $x \in \Omega \subset \mathbb{R}^d$ and $t > 0$:

$$\begin{cases} y' + Ay = Bu \\ y(0) = y^0 \end{cases}$$

where u is such that $y(T; u) = 0$.

Consider the operator $\mathcal{G} : y^0 \rightarrow u$. Then, for any fixed $\varepsilon > 0$ and $\sigma > 0$, there exists a DeepONet $\mathcal{N} = (\beta, \tau)$ achieving an approximation error

$$\mathcal{E}_{approx} = \|\mathcal{G} - \mathcal{N}\|_{L^2(\mu)} \lesssim \varepsilon$$

and such that

$$\text{size}(\beta) \lesssim \log(\varepsilon^{-1}), \quad \text{depth}(\beta) \lesssim 1, \quad p \sim \log(\varepsilon^{-1})^d, \quad m \sim \log(\varepsilon^{-1})^{2d+\sigma}.$$



García-Cervera, C.J.; Kessler, M.; Pedregal, P.; Periago, F.:

Approximation of Set-Valued Maps through Neural Networks, and Application to Control. **Submitted.**

Estimates for generalization error for DeepONet

Boundedness assumption: there exists $\psi : L^2(D) \rightarrow [0, +\infty[$ such that

$$|\mathcal{G}(y)(t)| \leq \psi(y), \quad \sup_{\theta \in [-B, B]^{d_\theta}} |\mathcal{N}_\theta(y)(t)| \leq \psi(y), \quad \forall y \in L^2(D), \forall t \in U,$$

and there exists $C, \kappa > 0$ such that

$$\psi(y) \leq C (1 + \|y\|_{L^2(D)})^\kappa.$$

Estimates for generalization error for DeepONet

Boundedness assumption: there exists $\psi : L^2(D) \rightarrow [0, +\infty[$ such that

$$|\mathcal{G}(y)(t)| \leq \psi(y), \quad \sup_{\theta \in [-B, B]^{d_\theta}} |\mathcal{N}_\theta(y)(t)| \leq \psi(y), \quad \forall y \in L^2(D), \forall t \in U,$$

and there exists $C, \kappa > 0$ such that

$$\psi(y) \leq C (1 + \|y\|_{L^2(D)})^\kappa.$$

Observability implies $|\mathcal{G}(y)(t)| \leq \psi(y)$

Estimates for generalization error for DeepONet

Boundedness assumption: there exists $\psi : L^2(D) \rightarrow [0, +\infty[$ such that

$$|\mathcal{G}(y)(t)| \leq \psi(y), \quad \sup_{\theta \in [-B, B]^{d_\theta}} |\mathcal{N}_\theta(y)(t)| \leq \psi(y), \quad \forall y \in L^2(D), \forall t \in U,$$

and there exists $C, \kappa > 0$ such that

$$\psi(y) \leq C (1 + \|y\|_{L^2(D)})^\kappa.$$

Observability implies $|\mathcal{G}(y)(t)| \leq \psi(y)$

Lipschitz continuity assumption: there exists $\Phi : L^2(D) \rightarrow [0, +\infty[$ such that

$$|\mathcal{N}_\theta(y)(t) - \mathcal{N}_{\theta'}(y)(t)| \leq \Phi(y) \|\theta - \theta'\|_{\ell^\infty}, \quad \forall y \in L^2(D), \forall t \in U,$$

and there exists $C, \kappa > 0$ such that

$$\Phi(y) \leq C (1 + \|y\|_{L^2(D)})^\kappa.$$

Estimates for generalization error for DeepONet

Boundedness assumption: there exists $\psi : L^2(D) \rightarrow [0, +\infty[$ such that

$$|\mathcal{G}(y)(t)| \leq \psi(y), \quad \sup_{\theta \in [-B, B]^{d_\theta}} |\mathcal{N}_\theta(y)(t)| \leq \psi(y), \quad \forall y \in L^2(D), \forall t \in U,$$

and there exists $C, \kappa > 0$ such that

$$\psi(y) \leq C (1 + \|y\|_{L^2(D)})^\kappa.$$

Observability implies $|\mathcal{G}(y)(t)| \leq \psi(y)$

Lipschitz continuity assumption: there exists $\Phi : L^2(D) \rightarrow [0, +\infty[$ such that

$$|\mathcal{N}_\theta(y)(t) - \mathcal{N}_{\theta'}(y)(t)| \leq \Phi(y) \|\theta - \theta'\|_{\ell^\infty}, \quad \forall y \in L^2(D), \forall t \in U,$$

and there exists $C, \kappa > 0$ such that

$$\Phi(y) \leq C (1 + \|y\|_{L^2(D)})^\kappa.$$

Theorem (Bound for generalization error (Lanthaler et al. 2022))

Let $\mathcal{N}_\mathcal{F}$ and $\mathcal{N}_\mathcal{T}$ be as before. If the above two assumptions hold, then

$$\mathbb{E} [\mathcal{L}(\mathcal{N}_\mathcal{T}) - \mathcal{L}(\mathcal{N}_\mathcal{F})] \leq \frac{C}{\sqrt{N}} \left(1 + Cd_\theta \log \left(CB\sqrt{N} \right) \right)^{2\kappa + \frac{1}{2}} \quad (3)$$

where the constant $C = C(\mu, \psi, \Phi)$ is independent of B and d_θ .

Data generation. Initial conditions are computed by sampling a Gaussian random field with kernel

$$C(x, x') = \exp\left(-\frac{|x - x'|^2}{2\ell}\right), \quad x, x' \in (0, 1),$$

Data generation. Initial conditions are computed by sampling a Gaussian random field with kernel

$$C(x, x') = \exp\left(-\frac{|x - x'|^2}{2\ell}\right), \quad x, x' \in (0, 1),$$

DeepONet size. We use the ReLU activation function, trunk depth/width = 2/40 and branch depth/width = 2/40.

Data generation. Initial conditions are computed by sampling a Gaussian random field with kernel

$$C(x, x') = \exp\left(-\frac{|x - x'|^2}{2\ell}\right), \quad x, x' \in (0, 1),$$

DeepONet size. We use the ReLU activation function, trunk depth/width = 2/40 and branch depth/width = 2/40.

Training process. Adam optimizer with learning rate 0.01. Initialization of the parameters is carried out with Glorot uniform.

Data generation. Initial conditions are computed by sampling a Gaussian random field with kernel

$$C(x, x') = \exp\left(-\frac{|x - x'|^2}{2\ell}\right), \quad x, x' \in (0, 1),$$

DeepONet size. We use the ReLU activation function, trunk depth/width = 2/40 and branch depth/width = 2/40.

Training process. Adam optimizer with learning rate 0.01. Initialization of the parameters is carried out with Glorot uniform.

Implementation. DeepONet: learning operators [Lu, Li, Pang, Zhang, Karniadakis: Nat. Mach. Intell., 2021]

<https://github.com/lululxvi/deepxde>

Codes available at: <https://github.com/fperiago/deeponet>

An academic example with an explicit solution

$$\left\{ \begin{array}{ll} y_{tt} - y_{xx} = 0, & \text{in } (0, 1) \times (0, 2) \\ y(x, 0) = y^0(x), & \text{on } (0, 1) \\ y_t(x, 0) = y^1(x) & \text{on } (0, 1) \\ y(0, t) = 0, & \text{on } (0, 2) \\ y(1, t) = u(t) & \text{on } (0, 2) \\ y(x, 2) = y_t(x, 2) = 0, & \text{on } (0, 1). \end{array} \right.$$

An academic example with an explicit solution

$$\left\{ \begin{array}{ll} y_{tt} - y_{xx} = 0, & \text{in } (0, 1) \times (0, 2) \\ y(x, 0) = y^0(x), & \text{on } (0, 1) \\ y_t(x, 0) = y^1(x) & \text{on } (0, 1) \\ y(0, t) = 0, & \text{on } (0, 2) \\ y(1, t) = u(t) & \text{on } (0, 2) \\ y(x, 2) = y_t(x, 2) = 0, & \text{on } (0, 1). \end{array} \right.$$

The operator to be approximated is

$$\begin{aligned} \mathcal{G} : \quad L^2(0, 1) \times H^{-1}(0, 1) &\rightarrow L^2(0, 2) \\ (y^0, y^1) &\mapsto \mathcal{G}(y^0, y^1) := u \end{aligned}$$

An academic example with an explicit solution

$$\left\{ \begin{array}{ll} y_{tt} - y_{xx} = 0, & \text{in } (0, 1) \times (0, 2) \\ y(x, 0) = y^0(x), & \text{on } (0, 1) \\ y_t(x, 0) = y^1(x) & \text{on } (0, 1) \\ y(0, t) = 0, & \text{on } (0, 2) \\ y(1, t) = u(t) & \text{on } (0, 2) \\ y(x, 2) = y_t(x, 2) = 0, & \text{on } (0, 1). \end{array} \right.$$

The operator to be approximated is

$$\begin{aligned} \mathcal{G} : \quad L^2(0, 1) \times H^{-1}(0, 1) &\rightarrow L^2(0, 2) \\ (y^0, y^1) &\mapsto \mathcal{G}(y^0, y^1) := u \end{aligned}$$

where

$$u(t) = \begin{cases} \frac{1}{2}y^0(1-t) + \frac{1}{2}\int_{1-t}^1 y^1(s) ds, & 0 \leq t \leq 1 \\ -\frac{1}{2}y^0(t-1) + \frac{1}{2}\int_{t-1}^1 y^1(s) ds, & 1 < t \leq 2 \end{cases}$$

is the **unique** control of minimal L^2 -norm.

Numerical experiments: the wave equation

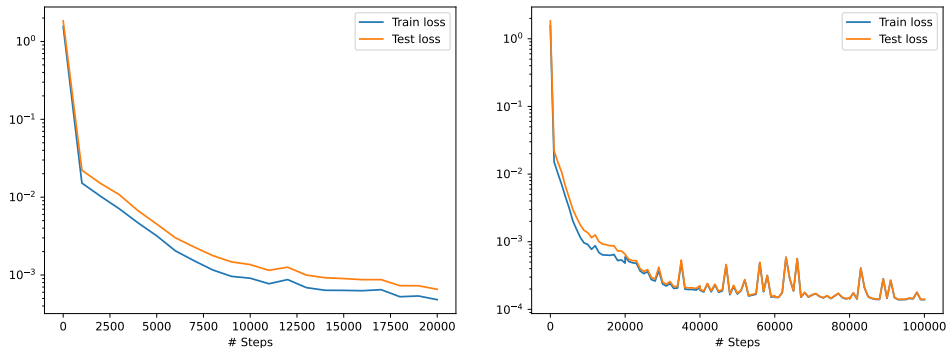


Figure: Loss history for correlation lengths of 0.25 for position and 0.125 for velocity. Number of sensor points = 51. Sample functions: (Left) 100 and (right) 10000.

$$\text{Test error} \sim 1.5 \times 10^{-6}$$

Numerical experiments: the wave equation

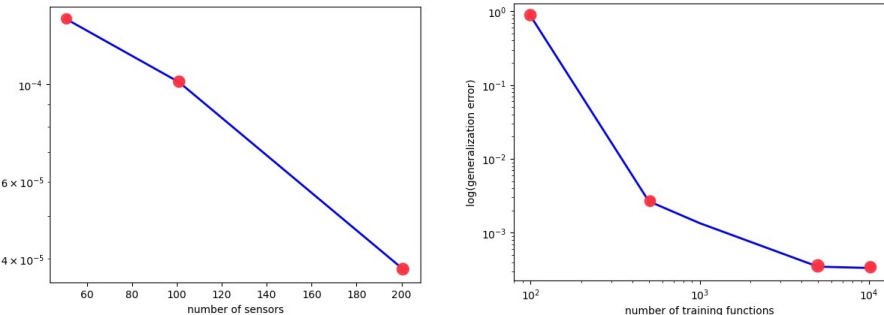


Figure: (Left) Dependence of the generalization error on the number of sensor points for $(\ell_0, \ell_1) = (0.25, 0.125)$, $n_{\text{functions}} = 10^4$. (Right) Dependence of the generalization error on the number of training functions for $(\ell_0, \ell_1) = (0.0625, 0.03125)$, $n_{\text{sensors}} = 101$.

Numerical experiments: the wave equation

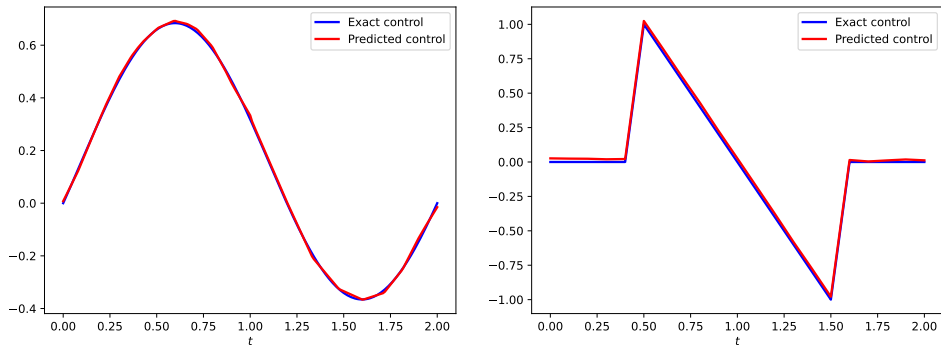


Figure: Exact versus predicted solutions. $n_{functions} = 10^4$ **(Left) Smooth initial data:** $y^0 = y^1 = \sin(\pi x)$, $(\ell_{pos}, \ell_{vel}) = (0.25, 0.125)$, $n_{sensors}=100$. Relative error $\approx 1\%$.

(Right) Non-smooth initial data: $y^0(x) = \begin{cases} 4x, & 0 \leq x \leq 0.5 \\ 0, & 0.5 < x \leq 1 \end{cases}$, $(\ell_{pos}, \ell_{vel}) = (0.0625, 0.03125)$, $n_{sensors}=10$. Relative error $\approx 4\%$.

Set-Valued Neural Networks & Set-Valued Operators



Notation

- Hausdorff distance

$$d_H(A, B) = \max \left\{ \sup_{a \in A} d(a, B), \sup_{b \in B} d(b, A) \right\}. \quad (4)$$

Notation

- Hausdorff distance

$$d_H(A, B) = \max \left\{ \sup_{a \in A} d(a, B), \sup_{b \in B} d(b, A) \right\}. \quad (4)$$

- $\mathcal{F}(Y)$ family of closed sets of a metric space Y

Notation

- Hausdorff distance

$$d_H(A, B) = \max \left\{ \sup_{a \in A} d(a, B), \sup_{b \in B} d(b, A) \right\}. \quad (4)$$

- $\mathcal{F}(Y)$ family of closed sets of a metric space Y

Theorem (Universal Approximation of Set-Valued Operators)

Let $K_1 \subset \mathbb{R}^n$, $K_2 \subset \mathbb{R}^m$ be compact sets, and let $V \subset C(K_1)$ be a compact family. Let

$$F : V \subset C(K_1) \rightrightarrows \mathcal{F}(C(K_2))$$

be a continuous set-valued function such that $F(u)$ is a convex compact set for all $u \in V$. Then, given $\epsilon > 0$, there exists a set-valued neural network

$$\mathcal{G} : V \rightrightarrows \mathcal{F}(C(K_2))$$

such that

$$d_H(F(u), \mathcal{G}(u)) < \epsilon, \quad \forall u \in V.$$



García-Cervera, C.J.; Kessler, M.; Pedregal, P.; Periago, F.: *Universal Approximation of Set-Valued Maps*. **In preparation.**

Theorem (Universal Approximation of Set-Valued Operators)

Let $K_1 \subset \mathbb{R}^n$, $K_2 \subset \mathbb{R}^m$ be compact sets, and let $V \subset C(K_1)$ be a compact family. Let $F : V \subset C(K_1) \rightrightarrows \mathcal{F}(C(K_2))$ be a continuous set-valued function such that $F(u)$ is a convex compact set for all $u \in V$. Then, given $\epsilon > 0$, there exists a set-valued neural network $\mathcal{G} : V \rightrightarrows \mathcal{F}(C(K_2))$ such that

$$d_H(F(u), \mathcal{G}(u)) < \epsilon, \quad \forall u \in V.$$

Idea of the proof.

Theorem (Universal Approximation of Set-Valued Operators)

Let $K_1 \subset \mathbb{R}^n$, $K_2 \subset \mathbb{R}^m$ be compact sets, and let $V \subset C(K_1)$ be a compact family. Let $F : V \subset C(K_1) \rightrightarrows \mathcal{F}(C(K_2))$ be a continuous set-valued function such that $F(u)$ is a convex compact set for all $u \in V$. Then, given $\epsilon > 0$, there exists a set-valued neural network $\mathcal{G} : V \rightrightarrows \mathcal{F}(C(K_2))$ such that

$$d_H(F(u), \mathcal{G}(u)) < \epsilon, \quad \forall u \in V.$$

Idea of the proof.

- 1) By compactness of V and $F(u)$,

$$V = \bigcup_{j=1}^N B_{\eta_j}(u_j), \quad \text{and} \quad \sup_{v \in F(u_j)} \|v\|_{C(K_2)} \leq M, \quad j = 1, 2, \dots, N.$$

Theorem (Universal Approximation of Set-Valued Operators)

Let $K_1 \subset \mathbb{R}^n$, $K_2 \subset \mathbb{R}^m$ be compact sets, and let $V \subset C(K_1)$ be a compact family. Let $F : V \subset C(K_1) \rightrightarrows \mathcal{F}(C(K_2))$ be a continuous set-valued function such that $F(u)$ is a convex compact set for all $u \in V$. Then, given $\epsilon > 0$, there exists a set-valued neural network $\mathcal{G} : V \rightrightarrows \mathcal{F}(C(K_2))$ such that

$$d_H(F(u), \mathcal{G}(u)) < \epsilon, \quad \forall u \in V.$$

Idea of the proof.

- 1) By compactness of V and $F(u)$,

$$V = \bigcup_{j=1}^N B_{\eta_j}(u_j), \quad \text{and} \quad \sup_{v \in F(u_j)} \|v\|_{C(K_2)} \leq M, \quad j = 1, 2, \dots, N.$$

- 2) For $u \in V$, define

$$\tilde{F}(u) = \sum_{j=1}^N \xi_j(u) F(u_j), \quad \xi_j(u) = \frac{\phi_j(u)}{\sum_{k=1}^N \phi_k(u)}, \quad \phi_j(u) = \varphi\left(\frac{\|u - u_j\|}{\eta_j}\right).$$

Theorem (Universal Approximation of Set-Valued Operators)

Let $K_1 \subset \mathbb{R}^n$, $K_2 \subset \mathbb{R}^m$ be compact sets, and let $V \subset C(K_1)$ be a compact family. Let $F : V \subset C(K_1) \rightrightarrows \mathcal{F}(C(K_2))$ be a continuous set-valued function such that $F(u)$ is a convex compact set for all $u \in V$. Then, given $\epsilon > 0$, there exists a set-valued neural network $\mathcal{G} : V \rightrightarrows \mathcal{F}(C(K_2))$ such that

$$d_H(F(u), \mathcal{G}(u)) < \epsilon, \quad \forall u \in V.$$

Idea of the proof.

- 1) By compactness of V and $F(u)$,

$$V = \bigcup_{j=1}^N B_{\eta_j}(u_j), \quad \text{and} \quad \sup_{v \in F(u_j)} \|v\|_{C(K_2)} \leq M, \quad j = 1, 2, \dots, N.$$

- 2) For $u \in V$, define

$$\tilde{F}(u) = \sum_{j=1}^N \xi_j(u) F(u_j), \quad \xi_j(u) = \frac{\phi_j(u)}{\sum_{k=1}^N \phi_k(u)}, \quad \phi_j(u) = \varphi\left(\frac{\|u - u_j\|}{\eta_j}\right).$$

- 3) It is proved that

$$d_H(F(u), \tilde{F}(u)) \lesssim \epsilon, \quad \forall u \in V.$$

- 4) Since $\xi_j : V \rightarrow \mathbb{R}$ are continuous nonlinear functionals in V , there exist

$$g_j(u) = \sum_{k=1}^{L_j} c_{j,k} \sigma \left(\sum_{r=1}^{m_j} \alpha_{j,kr} u(x_r) + \theta_{j,k} \right),$$

such that $\|\xi_j - g_j\|_{C(K_1)} \lesssim \epsilon$

- 4) Since $\xi_j : V \rightarrow \mathbb{R}$ are continuous nonlinear functionals in V , there exist

$$g_j(u) = \sum_{k=1}^{L_j} c_{j,k} \sigma \left(\sum_{r=1}^{m_j} \alpha_{j,kr} u(x_r) + \theta_{j,k} \right),$$

such that $\|\xi_j - g_j\|_{C(K_1)} \lesssim \epsilon$

- 5) Define the set- valued neural network

$$G(u) = \sum_{j=1}^N g_j(u) F(u_j),$$

which satisfies $d_H(\tilde{F}(u), G(u)) \lesssim \epsilon$.

- 4) Since $\xi_j : V \rightarrow \mathbb{R}$ are continuous nonlinear functionals in V , there exist

$$g_j(u) = \sum_{k=1}^{L_j} c_{j,k} \sigma \left(\sum_{r=1}^{m_j} \alpha_{j,kr} u(x_r) + \theta_{j,k} \right),$$

such that $\|\xi_j - g_j\|_{C(K_1)} \lesssim \epsilon$

- 5) Define the set-valued neural network

$$G(u) = \sum_{j=1}^N g_j(u) F(u_j),$$

which satisfies $d_H(\tilde{F}(u), G(u)) \lesssim \epsilon$.

- 6) By the Krein-Milman theorem,

$$F(u_j) = \overline{\text{co}}(\Gamma_j) = \overline{\text{co}}(\overline{\Gamma_j}),$$

Γ_j is the family of extremal points of $F(u_j)$

- 4) Since $\xi_j : V \rightarrow \mathbb{R}$ are continuous nonlinear functionals in V , there exist

$$g_j(u) = \sum_{k=1}^{L_j} c_{j,k} \sigma \left(\sum_{r=1}^{m_j} \alpha_{j,kr} u(x_r) + \theta_{j,k} \right),$$

such that $\|\xi_j - g_j\|_{C(K_1)} \lesssim \epsilon$

- 5) Define the set-valued neural network

$$G(u) = \sum_{j=1}^N g_j(u) F(u_j),$$

which satisfies $d_H(\tilde{F}(u), G(u)) \lesssim \epsilon$.

- 6) By the Krein-Milman theorem,

$$F(u_j) = \overline{\text{co}}(\Gamma_j) = \overline{\text{co}}(\overline{\Gamma_j}),$$

Γ_j is the family of extremal points of $F(u_j)$

- 7) Since $\overline{\Gamma_j}$ is compact, we can find $d_j \in \mathbb{N}$ and a finite set

$$A_j = \{z_{j,\ell}\}_{\ell=1}^{d_j} \subset \Gamma_j \text{ such that}$$

$$d_H(F(u_j), \overline{\text{co}}(A_j)) = d_H(\overline{\text{co}}(\Gamma_j), \overline{\text{co}}(A_j)) \lesssim \epsilon.$$

- 8) We approximate each $z_{j,\ell}$ by a neural network

$$h_{j,\ell}(\mathbf{x}) = \sum_{i=1}^{K_{j,\ell}} c_{i,j,\ell} \sigma(\omega_{i,j,\ell} \cdot \mathbf{x} + \theta_{i,j,\ell}), \quad \mathbf{x} \in K_2.$$

- 8) We approximate each $z_{j,\ell}$ by a neural network

$$h_{j,\ell}(\mathbf{x}) = \sum_{i=1}^{K_{j,\ell}} c_{i,j,\ell} \sigma(\omega_{i,j,\ell} \cdot \mathbf{x} + \theta_{i,j,\ell}), \quad \mathbf{x} \in K_2.$$

- 9) Finally, we define what we call a **Single Input Multiple Output Net (SIMONet)** as the set-valued function

$$\mathcal{G} : V \rightrightarrows \mathcal{F}(C(K_2))$$

by

$$\mathcal{G}(u) = \sum_{j=1}^N g_j(u) \operatorname{co} \left(\{h_{j,\ell_j}\}_{\ell_j=1}^{d_j} \right), \quad u \in V.$$

which approximates the original set-valued map F .



Work supported by

- Grant PID2022-141957OA-C22 funded by MCIN/AEI/10.13039/501100011033 and by ERDF A way of making Europe



- Autonomous Community of the Región of Murcia, Spain, through the programme for the development of scientific and technical research by competitive groups (21996/PI/22), included in the Regional Program for the Promotion of Scientific and Technical Research of Fundación Séneca – Agencia de Ciencia y Tecnología de la Región de Murcia..



f SéNeCa⁽⁺⁾

Agencia de Ciencia y Tecnología
Región de Murcia

Thank you for your attention !