Size Lowerbounds for Deep Operator Networks

Anirbit

University of Manchester

Reference

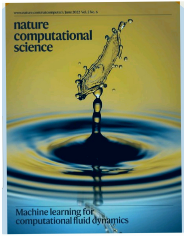
This talk is based on work published in Transactions on Machine Learning Research (TMLR) in Feb, 2024.

This paper is co-authored with,

Amartya Roy, PhD student in IIT-Delhi
(then working @ Bosch, India)

Table of Contents

A Resurgence of the A.I. Driven Ways to Solve Differential Equations



Massachusetts Institute of Technology

Education Research Innovation Admissions + Aid Campus Life

MIT News

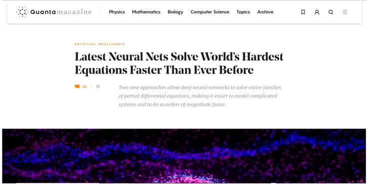
ON CAMPUS AND AROUND THE WORLD

SUBSCRIBE

Technique could efficiently solve partial differential equations for numerous applications

MIT researchers propose “PEDS” method for developing models of complex physical systems in mechanics, optics, thermal transport, fluid dynamics, physical chemistry, climate, and more.

Sandi Miller | Department of Mathematics
January 8, 2024



Quanta magazine

Physics Mathematics Biology Computer Science Topics Archive

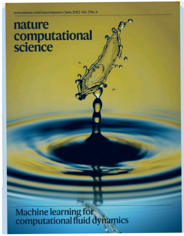
ARTIFICIAL INTELLIGENCE

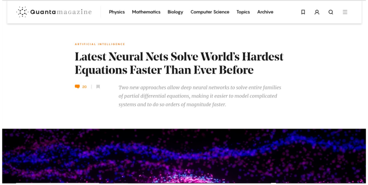
Latest Neural Nets Solve World’s Hardest Equations Faster Than Ever Before

20

Two new approaches allow deep neural networks to solve entire families of partial differential equations, making it easier to model complicated systems and to do so orders of magnitude faster.

A Resurgence of the A.I. Driven Ways to Solve Differential Equations





The ability of "Machine Learning" methods to solve differential equations heralds a new era in a plethora of classical fields. And in this talk, we shall study *possibly* one of the most powerful neural methods to solve differential equations - A **DeepONet**

Decoding DeepONets: A Schematic Overview.

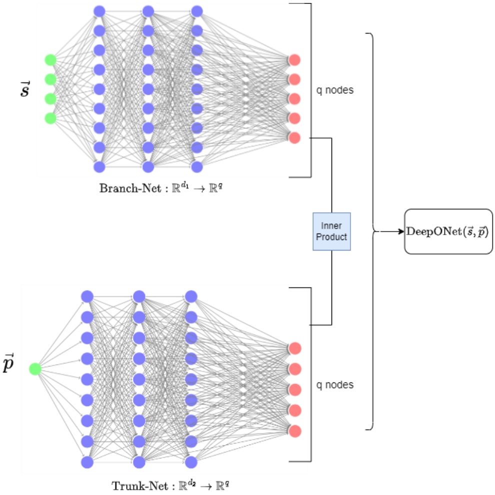


Figure 1: A Sketch of the DeepONet Architecture

How to Use a DeepONet?

As a concrete example of approximating maps between infinite dimensions, consider the task of solving pendulum O.D.Es,

$$\mathbb{R}^2 \ni \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} y \\ v \end{bmatrix} = \begin{bmatrix} v \\ -k \cdot \sin(y) + f(t) \end{bmatrix} \in \mathbb{R}^2$$

How to Use a DeepONet?

As a concrete example of approximating maps between infinite dimensions, consider the task of solving pendulum O.D.Es,

$$\mathbb{R}^2 \ni \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} y \\ v \end{bmatrix} = \begin{bmatrix} v \\ -k \cdot \sin(y) + f(t) \end{bmatrix} \in \mathbb{R}^2$$

For a fixed initial condition, here the training/test data sets would be 3−tuples of the form,

$$(\boldsymbol{x}_B(f), x_T, y)$$

$y \in \mathbb{R}$ is the angular position of the pendulum at time $t = x_T$ for the forcing function f discretized to $\boldsymbol{x}_B(f)$. Typically y is a standard O.D.E. solver's approximate solution. Given m such training data samples, the ℓ_2 empirical loss of a DeepONet, $\tilde{\mathcal{G}}$ would be,

How to Use a DeepONet?

As a concrete example of approximating maps between infinite dimensions, consider the task of solving pendulum O.D.Es,

$$\mathbb{R}^2 \ni \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} y \\ v \end{bmatrix} = \begin{bmatrix} v \\ -k \cdot \sin(y) + f(t) \end{bmatrix} \in \mathbb{R}^2$$

For a fixed initial condition, here the training/test data sets would be 3−tuples of the form,

$$(\boldsymbol{x}_B(f), x_T, y)$$

$y \in \mathbb{R}$ is the angular position of the pendulum at time $t = x_T$ for the forcing function f discretized to $\boldsymbol{x}_B(f)$. Typically y is a standard O.D.E. solver's approximate solution. Given m such training data samples, the ℓ_2 empirical loss of a DeepONet, $\tilde{\mathcal{G}}$ would be,

$$\hat{\mathcal{L}}(\tilde{\mathcal{G}}) := \frac{1}{2m}\sum_{i=1}^{m}\left(y_i - \tilde{\mathcal{G}}(\boldsymbol{x}_B(f_i), x_{T,i})\right)^2$$

No Need To Know The PDE!

NOTE : This loss function to be minimized **only needs a sufficient number of valid input-output pairs to be given and not a knowledge of the underlying PDE!** And this situation has real-world motivations where the underlying dynamical system might not be known,

– but a DeepONet can still find the right map!

No Need To Know The PDE!

NOTE : This loss function to be minimized **only needs a sufficient number of valid input-output pairs to be given and not a knowledge of the underlying PDE!** And this situation has real-world motivations where the underlying dynamical system might not be known,

– but a DeepONet can still find the right map!

This is a distinguishing feature of "Operator Learning".

How to Use a DeepONet?

Suppose $U \subset \mathbb{R}^n$ is the compact domain on which we seek the PDE solutions and $D \subset \mathbb{R}^d$ is the domain of the forcing functions. Then we can imagine 2 associated operators,

- A DeepONet, $\mathcal{N} : C(D) \to L^2(U)$ i.e $\mathcal{N}(f)(\boldsymbol{x}_T) := \mathrm{DON}(\boldsymbol{x}_B(f), \boldsymbol{x}_T)$.
- A solution operator $\mathcal{G} : C(D) \to L^2(U)$ which maps forcing functions to PDE solutions.

How to Use a DeepONet?

Suppose $U \subset \mathbb{R}^n$ is the compact domain on which we seek the PDE solutions and $D \subset \mathbb{R}^d$ is the domain of the forcing functions. Then we can imagine 2 associated operators,

- A DeepONet, $\mathcal{N} : C(D) \to L^2(U)$ i.e $\mathcal{N}(f)(\boldsymbol{x}_T) := \mathrm{DON}(\boldsymbol{x}_B(f), \boldsymbol{x}_T)$.
- A solution operator $\mathcal{G} : C(D) \to L^2(U)$ which maps forcing functions to PDE solutions.

Then it has been shown that, if μ is a measure s.t $\mathcal{G} \in L^2(\mu)$, then for every $\epsilon > 0$, there exists an operator network $\mathcal{N} : C(D) \to L^2(U)$, such that,

$$\|\mathcal{G} - \mathcal{N}\|_{L^2(\mu)} = \left(\int_{C(D)} \|\mathcal{G}(f) - \mathcal{N}(f)\|^2_{L^2(U)} d\mu(f) \right)^{1/2} < \epsilon.$$

(S. Lanthaler, S. Mishra, G. E. Karniadakis, doi.org/10.1093/imatrm/tnac001 (2022))

How to Use a DeepONet?

Suppose $U \subset \mathbb{R}^n$ is the compact domain on which we seek the PDE solutions and $D \subset \mathbb{R}^d$ is the domain of the forcing functions. Then we can imagine 2 associated operators,

- A DeepONet, $\mathcal{N} : C(D) \to L^2(U)$ i.e $\mathcal{N}(f)(\boldsymbol{x}_T) := \mathrm{DON}(\boldsymbol{x}_B(f), \boldsymbol{x}_T)$.
- A solution operator $\mathcal{G} : C(D) \to L^2(U)$ which maps forcing functions to PDE solutions.

Then it has been shown that, if μ is a measure s.t $\mathcal{G} \in L^2(\mu)$, then for every $\epsilon > 0$, there exists an operator network $\mathcal{N} : C(D) \to L^2(U)$, such that,

$$\|\mathcal{G} - \mathcal{N}\|_{L^2(\mu)} = \left(\int_{C(D)} \|\mathcal{G}(f) - \mathcal{N}(f)\|^2_{L^2(U)} d\mu(f) \right)^{1/2} < \epsilon.$$

(S. Lanthaler, S. Mishra, G. E. Karniadakis, doi.org/10.1093/imatrm/tnac001 (2022))

The above gives an **“universal approximation theorem for operators”**

How DeepONets Train : An Intuition

Intuition about how a DeepONet works...

If $(\beta_i(\boldsymbol{x}_B), \tau_i(\boldsymbol{x}_T)), i = 1, \ldots, q$ are the output coordinates of the the branch and the trunk net then, the quantity "$\text{DeepONet}(\boldsymbol{x}_B, \boldsymbol{x}_T) = \sum_{i=1}^{q} \beta_i(\boldsymbol{x}_B) \cdot \tau_i(\boldsymbol{x}_T)$" can be read as :

the branch network output gives the coefficients with which to linearly combine the trunk network's coordinate functions - which are vectors/functions in the space where we are seeking the PDE solutions.

How DeepONets Train : An Intuition

Intuition about how a DeepONet works...

If $(\beta_i(\boldsymbol{x}_B), \tau_i(\boldsymbol{x}_T)), i = 1, \ldots, q$ are the output coordinates of the the branch and the trunk net then, the quantity "$\text{DeepONet}(\boldsymbol{x}_B, \boldsymbol{x}_T) = \sum_{i=1}^{q} \beta_i(\boldsymbol{x}_B) \cdot \tau_i(\boldsymbol{x}_T)$" can be read as :

the branch network output gives the coefficients with which to linearly combine the trunk network's coordinate functions - which are vectors/functions in the space where we are seeking the PDE solutions.

Hence heuristically, this is joint optimization over 2 objectives:
(a) Finding q good functions in the solution space
(Learning the trunk)
& (b) Finding a good vector in the span of the former.
(Learning the branch).

Table of Contents

A Size Requirement for Deep Operator Nets to be "Good"

Theorem (Informal Statement of Our Main Theorem)

With high probability over sampling a $n-$ noisy training data, if a class of fixed size bounded output DeepONets has to have a predictor that can achieve empirical training error below a label noise dependent threshold, then necessarily q — the common output dimension of the branch and the trunk — must be lower bounded as $q = \Omega\left(\sqrt[4]{n}\right)$

Speciality of our Theorem

- **This stands as one of the very rare instances where a proof of architectural constraint for training performance for any neural network structure has been established.**

Speciality of our Theorem

- **This stands as one of the very rare instances where a proof of architectural constraint for training performance for any neural network structure has been established.**

- **This lower-bound is "universal" i.e it's independent of the specific Partial Differential Equation that a DeepONet aims to solve.**

The Formal Setup for DeepONet Theory

Definition (Training Dataset Property)

$(y_i, (\boldsymbol{s}_i, \boldsymbol{p}_i))$ be i.i.d. input-output pair, And $y_i \in [-B, B]$ and we define the conditional random variable $g(\boldsymbol{s}_i, \boldsymbol{p}_i) = \mathbb{E}[y \mid (\boldsymbol{s}_i, \boldsymbol{p}_i)]$

Definition (Branch Functions & Trunk Functions)

$$\mathcal{B} := \left\{ B_{\boldsymbol{w}} \mid B_{\boldsymbol{w}} : \mathbb{R}^{d_1} \to \mathbb{R}^q, \mathrm{Lip}\left(B_{\boldsymbol{w}}\right) \leq L_B \& \|\boldsymbol{w}\|_2 \leq W_B \& \left\|B_{\boldsymbol{w}}\right\|_\infty \leq \mathcal{C} \right\}$$

$$\mathcal{T} := \left\{ T_{\boldsymbol{w}} \mid T_{\boldsymbol{w}} : \mathbb{R}^{d_2} \to \mathbb{R}^q, \mathrm{Lip}\left(T_{\boldsymbol{w}}\right) \leq L_T \& \|\boldsymbol{w}\|_2 \leq W_T \& \left\|T_{\boldsymbol{w}}\right\|_\infty \leq \mathcal{C} \right\}$$

The bound of $\mathcal{C}$ in the above definitions abstracts out the model of the branch and the trunk functions being nets having a layer of bounded activation functions in their output layer.

The Formal Setup for DeepONet Theory

Definition (**The DeepONet(DON) Class**)

$$
\begin{aligned}
\mathcal{H} := \{ h_{\boldsymbol{w}_b, \boldsymbol{w}_t} = h_{(\boldsymbol{w}_b, \boldsymbol{w}_t)} \mid \\
& \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \ni (\boldsymbol{s}, \boldsymbol{p}) \mapsto h(\mathbf{s}, \boldsymbol{p}) := \langle B_{\boldsymbol{w}_b}(s), T_{\boldsymbol{w}_t}(\boldsymbol{p}) \rangle \in \mathbb{R}, \\
& B_{\boldsymbol{w}_b} \in \mathcal{B} \ \& \ T_{\boldsymbol{w}_t} \in \mathcal{T} \}
\end{aligned}
$$

The Formal Setup for DeepONet Theory

Definition (The DeepONet(DON) Class)

$$\mathcal{H} := \{h_{\boldsymbol{w}_b,\boldsymbol{w}_t} = h_{(\boldsymbol{w}_b,\boldsymbol{w}_t)} \mid \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \ni (\boldsymbol{s},\boldsymbol{p}) \mapsto h(\mathbf{s},\boldsymbol{p}) := \langle B_{\boldsymbol{w}_b}(s), T_{\boldsymbol{w}_t}(\boldsymbol{p})\rangle \in \mathbb{R}, B_{\boldsymbol{w}_b} \in \mathcal{B} \ \& \ T_{\boldsymbol{w}_t} \in \mathcal{T}\}$$

Note that $\forall \theta > 0$, $\exists$ a "θ-cover" of this function space $\mathcal{H}_\theta$ such that, $\forall h_{\boldsymbol{w}_b,\boldsymbol{w}_t} \in \mathcal{H}$, $\exists h_{(\boldsymbol{w}_{b,\frac{\theta}{2}},\boldsymbol{w}_{t,\frac{\theta}{2}})} \in \mathcal{H}_\theta$, s.t $\left\|\boldsymbol{w}_b - \boldsymbol{w}_{b,\frac{\theta}{2}}\right\| \leq \frac{\theta}{2}$ and $\left\|\boldsymbol{w}_t - \boldsymbol{w}_{t,\frac{\theta}{2}}\right\| \leq \frac{\theta}{2}$ and $\boldsymbol{w}_{b,\frac{\theta}{2}}$ and $\boldsymbol{w}_{t,\frac{\theta}{2}}$ being elements of the $\frac{\theta}{2}$ covering space of the set of branch and trunk weights respectively.

The Formal Setup for DeepONet Theory

Definition (**Defining J/Lipschitzness in Weights of the Nets)**

Given any two valid weight vectors w_1 and w_2 for a "branch function" $\boldsymbol{B}$ we assume to have the following inequality for some fixed $J > 0$,

$$\sup_{\boldsymbol{s}} \|B_{\boldsymbol{w}_1}(\boldsymbol{s}) - B_{\boldsymbol{w}_2}(\boldsymbol{s})\|_\infty \leq J \cdot \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|$$

And similarly for the trunk functions.

The Formal Setup for DeepONet Theory

Definition (Defining J/Lipschitzness in Weights of the Nets)

Given any two valid weight vectors w_1 and w_2 for a "branch function" $\boldsymbol{B}$ we assume to have the following inequality for some fixed $J > 0$,

$$\sup_{\boldsymbol{s}} \|B_{\boldsymbol{w}_1}(\boldsymbol{s}) - B_{\boldsymbol{w}_2}(\boldsymbol{s})\|_\infty \leq J \cdot \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|$$

And similarly for the trunk functions.

That such a Lipschitzness property holds for nets has been proven many times, including in iconic papers like, A Universal Law of Robustness via Isoperimetry by Sebastien Bubeck and Mark Sellke

— the work which inspires our analysis most closely.

4 Lemmas of The Story

Lemma (1. A Standard Result About Covering Numbers)

For any space X with Euclidean metric, we denote as $N(\theta, X)$ the covering number of it at scale θ. Further recall that d_B and d_T are the total number of parameters in any branch and trunk function in the chosen sets $\mathcal{B}$ and $\mathcal{T}$, respectively – and let their weight spaces be $\mathcal{W}_{\mathcal{B}}$ and $\mathcal{W}_{\mathcal{T}}$.

Let $\mathcal{W}_{\mathcal{H}} = \mathcal{W}_{\mathcal{B}} \times \mathcal{W}_{\mathcal{T}}$ denote the sets of allowed weights of $\mathcal{H}$ — the corresponding DeepONet class. Then the following three bounds hold for any $\theta > 0$,

$$N(\theta, \mathcal{W}_{\mathcal{B}}) \leq \left(\frac{2W_B\sqrt{d_B}}{\theta}\right)^{d_B} \qquad N(\theta, \mathcal{W}_{\mathcal{T}}) \leq \left(\frac{2W_T\sqrt{d_T}}{\theta}\right)^{d_T}$$

$$N(\theta, \mathcal{W}_{\mathcal{H}}) \leq N(\theta/2, \mathcal{W}_{\mathcal{B}}) \cdot N(\theta/2, \mathcal{W}_{\mathcal{T}})$$

4 Lemmas of The Story

Lemma (2. Cover Shifting Lemma)

We recall the definition of $\mathcal{H}$, the DON class, B the label bound & J from weight Lipschitzness of the nets. For any $h \in \mathcal{H}$ and any training data we denote the corresponding empirical risk as,
$\hat{R}(h) := \frac{1}{n}\sum_{i=1}^{n}(y_i - h(\boldsymbol{s}_i, \boldsymbol{p}_i))^2$. *Then,* $\forall \theta > 0$ *we have,*

$$\hat{\mathcal{R}}\left(h_{(\boldsymbol{w}_{b,\frac{\theta}{2}}, \boldsymbol{w}_{t,\frac{\theta}{2}})}\right) \leq \hat{\mathcal{R}}\left(h_{(\boldsymbol{w}_b, \boldsymbol{w}_t)}\right) + q\mathcal{C}J\theta \cdot \left(B + 2q\mathcal{C}^2\right)$$

and $\boldsymbol{w}_{b,\frac{\theta}{2}}$ *and* $\boldsymbol{w}_{t,\frac{\theta}{2}}$ *be s.t.* $\left\|\boldsymbol{w}_b - \boldsymbol{w}_{b,\frac{\theta}{2}}\right\| \leq \frac{\theta}{2}$ *and* $\left\|\boldsymbol{w}_t - \boldsymbol{w}_{t,\frac{\theta}{2}}\right\| \leq \frac{\theta}{2}$.

4 Lemmas of The Story

Lemma (2. Cover Shifting Lemma)

We recall the definition of $\mathcal{H}$, the DON class, B the label bound & J from weight Lipschitzness of the nets. For any $h \in \mathcal{H}$ and any training data we denote the corresponding empirical risk as,
$\hat{R}(h) := \frac{1}{n}\sum_{i=1}^{n}(y_i - h(\boldsymbol{s}_i, \boldsymbol{p}_i))^2$. *Then, $\forall \theta > 0$ we have,*

$$\hat{\mathcal{R}}(h_{(\boldsymbol{w}_{b,\frac{\theta}{2}}, \boldsymbol{w}_{t,\frac{\theta}{2}})}) \leq \hat{\mathcal{R}}(h_{(\boldsymbol{w}_b, \boldsymbol{w}_t)}) + q\mathcal{C}J\theta \cdot \left(B + 2q\mathcal{C}^2\right)$$

and $\boldsymbol{w}_{b,\frac{\theta}{2}}$ and $\boldsymbol{w}_{t,\frac{\theta}{2}}$ be s.t. $\left\|\boldsymbol{w}_b - \boldsymbol{w}_{b,\frac{\theta}{2}}\right\| \leq \frac{\theta}{2}$ and $\left\|\boldsymbol{w}_t - \boldsymbol{w}_{t,\frac{\theta}{2}}\right\| \leq \frac{\theta}{2}$.

Thus we see that it is quantifiable as to how much is the increment in the empirical risk, when for a given training data a DeepONet is replaced by another with weights within a distance of θ from the original - and that this increment is parametric in θ

4 Lemmas of The Story

Lemma (3. Correlation Bounding Lemma)

$\forall\theta > 0$, and for $z_i := y_i - g(\boldsymbol{s}_i, \boldsymbol{p}_i) = y_i - \mathbb{E}[y \mid (\boldsymbol{s}_i, \boldsymbol{p}_i)]$,

$$\mathbb{P}\left(\exists\ h_{(\boldsymbol{w}_{b,\frac{\theta}{2}}, \boldsymbol{w}_{t,\frac{\theta}{2}})} \in \mathcal{H}_\theta \mid \frac{1}{n}\sum_{i=1}^{n} h_{(\boldsymbol{w}_{b,\frac{\theta}{2}}, \boldsymbol{w}_{t,\frac{\theta}{2}})}(\boldsymbol{s}_i, \boldsymbol{p}_i)\, z_i \geq \frac{\theta}{4}\right)$$

$$\leq \frac{2^{2(d_B+d_T)+1}}{\theta^{(d_B+d_T)}} \cdot \left(W_B\sqrt{d_B}\right)^{d_B} \cdot \left(W_T\sqrt{d_T}\right)^{d_T} \cdot \exp\left(-\frac{2n\theta^2}{8^4 \cdot (B \cdot q\mathcal{C}^2)^2}\right)$$
$$+ 2\exp\left(-\frac{n\theta^2}{8^3 \cdot B^2 \cdot q^2 \cdot \mathcal{C}^4}\right)$$

4 Lemmas of The Story

Lemma (3. Correlation Bounding Lemma)

$\forall \theta > 0$, and for $z_i := y_i - g(\boldsymbol{s}_i, \boldsymbol{p}_i) = y_i - \mathbb{E}[y \mid (\boldsymbol{s}_i, \boldsymbol{p}_i)]$,

$$\mathbb{P}\left(\exists\ h_{(\boldsymbol{w}_{b,\frac{\theta}{2}}, \boldsymbol{w}_{t,\frac{\theta}{2}})} \in \mathcal{H}_\theta \mid \frac{1}{n}\sum_{i=1}^{n} h_{(\boldsymbol{w}_{b,\frac{\theta}{2}}, \boldsymbol{w}_{t,\frac{\theta}{2}})}(\boldsymbol{s}_i, \boldsymbol{p}_i)\, z_i \geq \frac{\theta}{4}\right)$$

$$\leq \frac{2^{2(d_B+d_T)+1}}{\theta^{(d_B+d_T)}} \cdot \left(W_B\sqrt{d_B}\right)^{d_B} \cdot \left(W_T\sqrt{d_T}\right)^{d_T} \cdot \exp\left(-\frac{2n\theta^2}{8^4 \cdot (B \cdot q\mathcal{C}^2)^2}\right)$$
$$+ 2\exp\left(-\frac{n\theta^2}{8^3 \cdot B^2 \cdot q^2 \cdot \mathcal{C}^4}\right)$$

The RHS above shows that the possibility of a predictor in the cover space with high average correlation with label noise, falls exponentially with the #training data.

Proof Sketch for Lemma [3]

- For each data i and each predictor $h_{(\mathbf{w}_{b,\frac{\theta}{2}},\mathbf{w}_{t,\frac{\theta}{2}})}$, we further define the random variable,

$$Y_{\theta,i} := \left(\left(h_{(\mathbf{w}_{b,\frac{\theta}{2}},\mathbf{w}_{t,\frac{\theta}{2}})}(\mathbf{s}_i, \mathbf{p}_i) - \mathbb{E}[h_{(\mathbf{w}_{b,\frac{\theta}{2}},\mathbf{w}_{t,\frac{\theta}{2}})}]\right) \underbrace{(y_i - g(\mathbf{s}_i, \mathbf{p}_i))}_{z_i} \right)$$

- One can show $Y_{\theta,i}$ is a centered and a bounded r.v and then one can prove concentration bounds for the data average of it.
- The lemma follows by union bound on the finite class of functions $\mathcal{H}_\theta$

4 Lemmas of The Story

Lemma (4. Low Correlation To Noise Implies Low Error DeepONet)

We continue in the same setup as in the previous lemma and further recall the definition of $\sigma^2 := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\left(y_i - g\left(\boldsymbol{s}_i, \boldsymbol{p}_i\right)\right)^2\right]$ *and* $g(\boldsymbol{s}, \boldsymbol{p}) = \mathbb{E}\left[y \mid (\boldsymbol{s}, \boldsymbol{p})\right]$ *and* $\forall \theta > 0$

$$\mathbb{P}\left(\exists\ h_{\boldsymbol{w}_b, \boldsymbol{w}_t} \in \mathcal{H} \mid \frac{1}{n}\sum_{i=1}^{n} \left(y_i - h_{\boldsymbol{w}_b, \boldsymbol{w}_t}(\boldsymbol{s}_i, \boldsymbol{p}_i)\right)^2 \leq \sigma^2 - \theta\right)$$
$$\leq 2\exp\left(-\frac{n\theta^2}{288 B^2}\right) + \mathbb{P}\left(\exists\ h_{\boldsymbol{w}_b, \boldsymbol{w}_t} \in \mathcal{H} \mid \frac{1}{n}\sum_{i=1}^{n} h\left(\boldsymbol{s}_i, \boldsymbol{p}_i\right) z_i \geqslant \frac{\theta}{4}\right)$$

4 Lemmas of The Story

Lemma (4. Low Correlation To Noise Implies Low Error DeepONet)

We continue in the same setup as in the previous lemma and further recall the definition of $\sigma^2 := \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\left(y_i - g\left(\boldsymbol{s}_i, \boldsymbol{p}_i\right)\right)^2\right]$ *and* $g(\boldsymbol{s}, \boldsymbol{p}) = \mathbb{E}\left[y \mid (\boldsymbol{s}, \boldsymbol{p})\right]$ *and* $\forall \theta > 0$

$$\mathbb{P}\left(\exists\ h_{\boldsymbol{w}_b, \boldsymbol{w}_t} \in \mathcal{H} \mid \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_{\boldsymbol{w}_b, \boldsymbol{w}_t}(\boldsymbol{s}_i, \boldsymbol{p}_i)\right)^2 \leq \sigma^2 - \theta\right)$$
$$\leq 2\exp\left(-\frac{n\theta^2}{288B^2}\right) + \mathbb{P}\left(\exists\ h_{\boldsymbol{w}_b, \boldsymbol{w}_t} \in \mathcal{H} \mid \frac{1}{n}\sum_{i=1}^{n} h\left(\boldsymbol{s}_i, \boldsymbol{p}_i\right) z_i \geqslant \frac{\theta}{4}\right)$$

The above lemma reveals an intimate connection between the empirical error of DeepONets and the correlation of its output with label noise.

Proof Sketch : Step 1

What We Want to Ensure,

$$\forall \delta \in (0,1),\ (1-\delta) \leq \mathbb{P}(\text{low}-\text{error}-\text{DeepONet}-\text{exists})$$

Proof Sketch : Step 1

What We Want to Ensure,

$$\forall \delta \in (0,1),\ (1-\delta) \leq \mathbb{P}(\mathrm{low{-}error{-}DeepONet{-}exists})$$

To Find the Necessary Conditions for the Above, the Strategy is to Find an *Unconditional* Upperbound on this Probability of the Good Event that Concerns Us.

Proof Sketch : Step 1

First, we invoke on $\mathcal{H}_\theta$ the Last Lemma "Low Correlation To Noise Implies Low Error DeepONet"

$$\mathbb{P}\left(\exists h_{(\boldsymbol{w}_{b,\frac{\theta}{2}},\boldsymbol{w}_{t,\frac{\theta}{2}})} \in \mathcal{H}_\theta \mid \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_{(\boldsymbol{w}_{b,\frac{\theta}{2}},\boldsymbol{w}_{t,\frac{\theta}{2}})}(\boldsymbol{s}_i,\boldsymbol{p}_i)\right)^2 \leq \sigma^2 - \theta\right)$$

$$\leq 2\exp\left(-\frac{n\theta^2}{288 \cdot B^2}\right)$$

$$+\mathbb{P}\left(\exists h_{(\boldsymbol{w}_{b,\frac{\theta}{2}},\boldsymbol{w}_{t,\frac{\theta}{2}})} \in \mathcal{H}_\theta \mid \frac{1}{n}\sum_{i=1}^{n} h_{(\boldsymbol{w}_{b,\frac{\theta}{2}},\boldsymbol{w}_{t,\frac{\theta}{2}})}(\boldsymbol{s}_i,\boldsymbol{p}_i)\, z_i \geqslant \frac{\theta}{4}\right)$$

Step-2

Next, we invoke on the RHS above the "Correlation Bounding Lemma",

$$\mathbb{P}\left(\exists h_{(\boldsymbol{w}_{b,\frac{\theta}{2}},\boldsymbol{w}_{t,\frac{\theta}{2}})} \in \mathcal{H}_\theta \mid \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_{(\boldsymbol{w}_{b,\frac{\theta}{2}},\boldsymbol{w}_{t,\frac{\theta}{2}})}(\boldsymbol{s}_i,\boldsymbol{p}_i)\right)^2 \leq \sigma^2 - \theta\right)$$
$$\leq 2\exp\left(-\frac{n\theta^2}{288 \cdot B^2}\right) + \frac{2^{2(d_B+d_T)+1}}{\theta^{(d_B+d_T)}} \cdot \left(W_B\sqrt{d_B}\right)^{d_B} \cdot \left(W_T\sqrt{d_T}\right)^{d_T}$$
$$\cdot \exp\left(-\frac{2n\theta^2}{8^4 \cdot (B \cdot q\mathcal{C}^2)^2}\right) + 2\exp\left(-\frac{n\theta^2}{8^3 \cdot (B \cdot q \cdot \mathcal{C}^2)^2}\right)$$

Step-3

From the "Cover Shifting Lemma" we know how the error of a predictor changes between itself and its closest cover point.

Using that we can derive an upperbound on the probability of a good DeepONet to exist in the function class.

Step-3

From the "Cover Shifting Lemma" we know how the error of a predictor changes between itself and its closest cover point.

Using that we can derive an upperbound on the probability of a good DeepONet to exist in the function class.

The required necessary condition now follows from demanding that this upperbound be above $1 - \delta$, (after a bunch of messy algebra!).

Step-3...Continues

Main Theorem : Lowerbounds for DeepONets Whose Branch and Trunk End in Sigmoid Gates

Let W and s be bounds on the norms of the weights of the branch and the trunk and the total number of trainable parameters respectively.

Then $\forall \delta \in (0,1)$, and any arbitrary positive constant $\epsilon > 0$ if with probability at least $1-\delta$ with respect to the sampling of the data $\{(y_i, (\boldsymbol{s}_i, \boldsymbol{p}_i)) \mid i = 1, \ldots, n\}$, $\exists\ h_{\boldsymbol{w}_b, \boldsymbol{w}_t} \in \mathcal{H}$ s.t,

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - h_{\boldsymbol{w}_b, \boldsymbol{w}_t}(\boldsymbol{s}_i, \boldsymbol{p}_i))^2 \leq \sigma^2 - \epsilon\,(1 + J \cdot (B+2))$$

Step-3...Continues

Main Theorem : Lowerbounds for DeepONets Whose Branch and Trunk End in Sigmoid Gates

then,

$$q \geq n^{\frac{1}{4}}$$

$$\cdot \left(\frac{\epsilon^2}{288 \cdot B^2} \cdot \frac{1}{\ln\left(2 + e^{-s \cdot \alpha'} \cdot \left(\frac{4 \cdot \min\{d_B, d_T\}^2}{\epsilon} \cdot W\sqrt{s}\right)^s\right) + \ln\left(\frac{2}{1-\delta}\right)} \right)^{\frac{1}{4}}$$

where $\sigma^2 := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\left(y_i - g\left(\boldsymbol{s}_i, \boldsymbol{p}_i\right)\right)^2\right]$ and $g(\boldsymbol{s}, \boldsymbol{p}) = \mathbb{E}\left[y \mid (\boldsymbol{s}, \boldsymbol{p})\right]$ and if the branch net has $\alpha-$fraction of the training parameters then $\alpha' = \frac{\alpha}{2}\ln\frac{1}{\alpha} + \frac{1-\alpha}{2}\ln\frac{1}{1-\alpha}$.

Experimental Set-up

In this experiment, we aim to show that at a fixed number of parameters, increasing the output dimension q and the training data as q^2 results in a monotonic decrease in DeepONet training error. We use the advection-diffusion-reaction (ADR) PDE as a test case, given by:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + ku^2 + f(x), \quad x \in [0,1], t \in [0,1]$$
$$(at\ D = 0.01\ and\ k = 0.01)$$

The DeepONet empirical loss being attempted to be minimized is $\hat{\mathcal{L}}_{\text{DeepONet}} := \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \mathcal{G}_\theta(f_i)(p_i)\right)^2$, where p_is are points in the (x,t) space and y_is are the value of the PDE solution at p_i when the inhomogeneous term is f_i.

Experiments

We trained 10 DeepONet models with varying widths, keeping total size equal. Performance increased monotonically when $\frac{q}{\sqrt{n}}$ was constant but not when $\frac{q}{n^{2/3}}$ was constant, indicating insufficient data scaling.

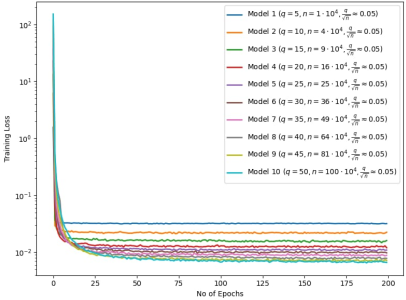


Figure 2: Training Loss vs Epoch in fixed $\frac{q}{\sqrt{n}}$ setting

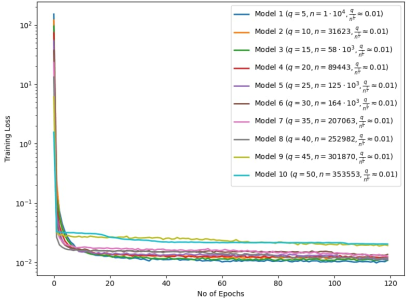


Figure 3: Training Loss vs Epoch in fixed $\frac{q}{n^{\frac{2}{3}}}$ setting

Putting It All Together

Our theorem shows that a certain data size dependent largeness of q (the common output dimension) is needed if there has to exist a bounded weight DeepONet at that q which can have their empirical error below the label noise threshold.

From our experiments, we have shown that there is some non-trivial range of q along which empirical risk improves with q for a fixed model size
— if *the amount of training data is scaled quadratically with* q.

We envisage that trying to prove this “scaling law” can be a very interesting direction for future exploration in theory.

Future Work

This proof technique is modular (in terms of what are the main 4 lemmas required) and hence possibly provides a path to get similar lowerbounds for various ML setups. Lets try this on more models!

Future Work

This proof technique is modular (in terms of what are the main 4 lemmas required) and hence possibly provides a path to get similar lowerbounds for various ML setups. Lets try this on more models!

Physics Informed Neural Nets (PINNs) are an increasingly popular neural net setup which seems to solve various PDEs – but one PDE at a time, unlike DeepONets as we have been seeing.

We have run this proof technique on the PINN setup for solving $d-$variable Hamilton-Jacobi-Bellman PDEs and derived lowerbounds on the size of depth-2 nets required to lower the training error. *As opposed to the proof presented here — in the PINN setup we can derive lowerbounds on the total number of trainable parameters of the net.*

Future Work

This proof technique is modular (in terms of what are the main 4 lemmas required) and hence possibly provides a path to get similar lowerbounds for various ML setups. Lets try this on more models!

Physics Informed Neural Nets (PINNs) are an increasingly popular neural net setup which seems to solve various PDEs – but one PDE at a time, unlike DeepONets as we have been seeing.

We have run this proof technique on the PINN setup for solving $d-$variable Hamilton-Jacobi-Bellman PDEs and derived lowerbounds on the size of depth-2 nets required to lower the training error. *As opposed to the proof presented here — in the PINN setup we can derive lowerbounds on the total number of trainable parameters of the net.*

This is a upcoming work with my student Sebastien Andre-Sloan and Prof. Matthew Colbrook @ DAMTP, Cambridge.

Future Work

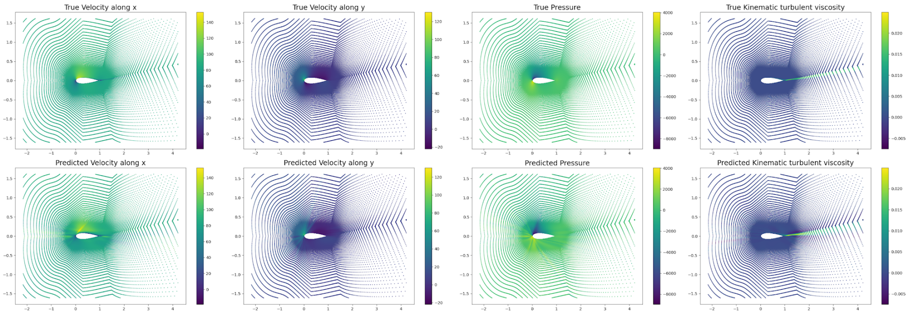


Figure 4: [work with my studennts, Sebastien Andre Sloan and Dibyakanti Kumar] In this example we can see an example of a multi-output DeepONet predicting the wind velocity, pressure and viscosity around (varying) toy plane wing shapes. Such setups are still far outside the grasp of theory.

Questions?