Deep Neural Networks and Smooth Approximation of PDEs

Kamil Doległo[1], Maciej Paszyński[1][0000−0001−7766−6052], and Leszek Demkowicz[2][0000−0001−7839−8037]

[1] AGH University of Science and Technology, Kraków, Poland
maciej.paszynski@agh.edu.pl
[2] Oden Institute, The University of Texas in Austin, USA
leszek@oden.utexas.edu

Abstract. We focus on Isogeometric Analysis (IGA) approximations of Partial Differential Equations (PDEs) solutions. We consider linear combinations of high-order and continuity base functions utilized by IGA. Instead of using the Deep Neural Network (DNN), which is the concatenation of linear operators and activation functions, to approximate the solutions of PDEs, we employ the linear combination of higher-order and continuity base functions, as employed by IGA. In this paper, we compare two methods. The first method trains different DNN for each coefficient of the linear computations. The second method trains one DNN for all coefficients of the linear combination. We show on model L-shape domain problem that training several small DNNs learning how to span B-splines coefficients is more efficient.

Keywords: Partial Differential Equations, Isogeometric Analysis, Physics Informed Neural Networks, Stochastic Gradient Descent

1 Introduction

Isogeometric Analysis (IGA) [1] employs smooth high-order and continuity base functions for approximation of solutions of Partial Differential Equations (PDEs). Physics Informed Neural Networks (PINN) [2] approximate the solution of a given PDE with Deep Neural Network (DNN) being the concatenation of several linear operators and non-linear activation functions. The Stochastic Gradient Descent (SGD) [3] is used to find the coefficients of the DNN approximating a given PDEs. In this presentation, we consider how smooth linear combinations of higher-order and continuity base functions used by IGA can be employed by DNNs and SGD method to approximate solutions of PDEs. In [4], we described how IGA could be used to approximate the coefficients of a linear combination of B-splines, employed for the solution of a family of PDEs depending on the right-hand side and boundary condition functions. In this presentation, we compare two methods. The first method, following [4] is to set up and train different DNNs for different coefficients of linear combinations. The second method is to

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_41

set up and train one DNNs for all coefficients of linear combination. We consider two problems. First, the simple family of one-dimensional problem

$$u''(x) = f(x) = n^2\pi^2 sin(n\pi x), u(0) = 0, u'(1) = g(n) = n\pi cos(n\pi) \quad (1)$$

for $x \in (0,1)$, used to explain our strategy. The family of solutions $u_n(x) = sin(n\pi x)$ depends on the parameter n defining the forcing and boundary condition. Second, we focus on model two-dimensional L-shape domain problem. The DNNs proposed in this paper learn how to span the smooth IGA combinations of B-splines approximating the solutions of PDEs.

Several Deep Neural Network approximating coefficients of IGA base functions. First, we assume that we approximate the solution of PDE by a linear combination of B-splines $u(x;n) \approx u_n(x) = \sum_i u_i(n) B^x_{i,p}$. Following [4] we approximate coefficients of linear combinations by several DNNs, one $u_i(n) \approx DNN_i(n)$ for each i-th coefficient

$$u(x;n) \approx u_n(x) = \sum_i u_i(n) B_{i,p}(x) \approx \sum_i DNN_i(n) B_{i,p}(x) \quad (2)$$

$$DNN_i(n) = c_i \sigma\left(a_i n + b_i\right) + d_i = \frac{c_i}{1 + exp(-a_i n - b_i)} + d_i \quad (3)$$

We define the error function

$$error_i(n) = 0.5 \left(DNN_i(n) - u_i(n)\right)^2 \quad (4)$$

with $u_i(n)$ computed using IGA method. The training procedure is employed independently for each $DNN_i(n)$ approximating different coefficients of the linear combination as a function of forcing and boundary conditions depending on n parameter. These DNNs learn an operator of different forcing and boundary conditions (parameterized by n) into coefficients of B-spline base functions approximating solutions of PDE (depending on forcing and boundary conditions).

One Deep Neural Network approximating all coefficients of IGA base functions. The second method consists in approximating of coefficients of linear combinations by one DNN, namely $DNN(n) = (u_1, ..., u_N)$ computing all the coefficients

$$u(x;n) = \sum_i u_i(n) B_{i,p}(x) = \sum_i \left(DNN(n)\right)_i B_{i,p}(x) \quad (5)$$

$$DNN(n) = C\sigma\left(An + B\right) + D = \frac{C}{1 + exp(An - B)} + D \quad (6)$$

where A, B, C, D are vectors of size $N \times 1$. We define

$$error(x) = \frac{\left(\sum_i \left(DNN(n)\right)_i - u_i(n)\right)^2}{2} \quad (7)$$

with $u_i(n)$ computed using IGA method. Here we have a single DNN learning an operator from forcing and boundary conditions (parameterized by n) into coefficients of B-spline basis functions approximating the solution of PDE (depending on forcing and boundary conditions parameterized by n).

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_41

Physics Informed Neural Network. Finally, we compare to PINN [2] approach, which approximates a single solution of (1) for a fixed n, with concatenation of linear operators and non-linear activation functions

$$PINN(x) = c\sigma\left(ax+b\right)+d = \frac{c}{1+exp(-ax-b)}+d \tag{8}$$

where $\sigma(x) = \frac{1}{1+exp(-x)}$ is e.g. the sigmoid activation function. We define the error functions for the approximation of PDE and b.c.

$$\begin{gathered} error_1(x) = 0.5\left(PINN''(x)\right)^2, error_2 = 0.5\left(PINN'(1)-g(n)\right)^2, \\ error_3 = 0.5\left(PINN(0)\right)^2. \end{gathered} \tag{9}$$

Here, DNN learns how to approximate a single solution of the PDE directly.

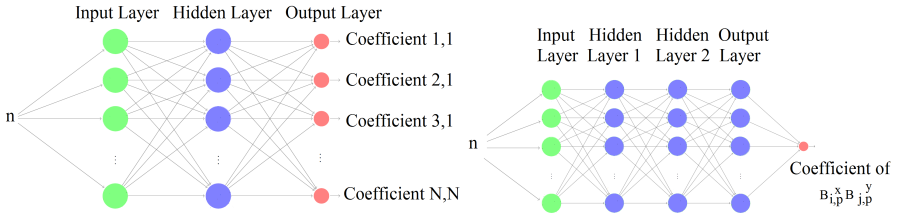

Fig. 1: Visualization of the first and the second DNN architectures.

Experimental verification. We verify the methods on model 2D L-shape domain problem $\Delta u = 0$ with zero Dirichlet b.c., and the Neumann b.c. $\frac{\partial u}{\partial n} = g(x,y)$. We embed L-shape in a square and set $1/4$ of the domain to 0. We assume the exact solution of this PDE of the form $u_{exact}(x,y) = sin(2\pi n\cdot x)\cdot sin(2\pi n\cdot y)$. using the manufactured solution with $g = \frac{\partial u_{exact}}{\partial n}$. We discretize with B-spline basis functions, seeking $u_h(x,y) = \sum_{i=1,\ldots,N;j=1,\ldots n} u_{ij}B^x_{i,p}B^y_{j,p}$. In our experiment, we have 42×42 two-dimensional cubic B-splines of C^2 continuity, with $N = 42+3 = 45$, the total of 2025 coefficients. In general the DNN is given by

$$\begin{gathered} l^{(0)} = x, l^{(1)} = \sigma^{(1)}\left(W^{(1)}l^{(0)}+b^{(1)}\right),\cdots \\ \cdots, l^{(n)} = \sigma^{(n)}\left(W^{(n)}l^{(n-1)}+b^{(n)}\right), l^{(out)} = W^{(out)}l^{(n)}+b^{(out)}. \end{gathered} \tag{10}$$

We employ two methods, see Figure 1. The first uses one DNN for each coefficient of the linear combination. Each DNN has input with n parameter, 600 neurons in the input layer, 600 neurons in two hidden layers, and 600 neurons in the output layers, sigmoid activation function, and one output value of the B-spline coefficient. So we have 2025 DNNs approximating 2025 coefficients of B-splines. The second one uses one DNN to compute all the linear combination coefficients. This DNN has input with n parameter, 1000 neurons in the input

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_41

layer, 1000 neurons in the hidden layer, and 1000 neurons in the output layer, ReLU activation function, and $N \times N$ output B-spline coefficients (2025 output values in our case). Figure 2 shows that training with 100 samples is cheaper and more accurate if we train several small DNNs, learning how to span B-splines.

Conclusions. We show that DNN can be used to approximate coefficients of linear combinations of higher-order and continuity base functions employed by IGA to approximate solutions of PDEs. The DNN learnt how to span the linear combinations of smooth B-splines. We obtained the approximation of the solution of a family of PDEs that is of higher-order and continuity, smooth and easily differentiate. In the future work we may employ the idea proposed by the BSDE method [5], approximating the gradients of the solution of PDE.

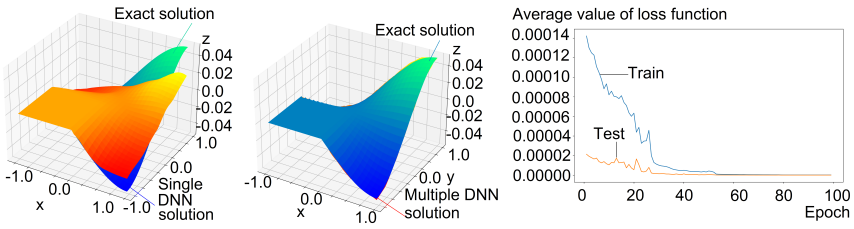

Fig. 2: Several DNNs, one for each coefficients work well as opposed to one DNN for all coefficients. Exact solution (blue). **Left panel:** Solution from one DNN for all coefficients (red), MSE=1.41e-4. **Middle panel:** Solution from several DNNs one for each coefficients. MSE=8.42e-7. **Right panel:** Training and testing with 100 samples of one DNN approximing a single B-spline coefficient.

Acknowledgement. The European Union's Horizon 2020 Research and Innovation Program of the Marie Skłodowska-Curie grant agreement No. 777778.

References

1. John Austin Cottrell, Thomas J. R. Hughes, Yuri Bazilevs, *Isogeometric Analysis: Towards Unification of Computer Aided Design and Finite Element Analysis* John Wiley and Sons, (2009)
2. Raissi Maziar, Paris Perdikaris, George E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, **378**, 686–707 (2019)
3. Bottou Léon, *Online Algorithms and Stochastic Approximations*, Online Learning and Neural Networks (1998) Cambridge University Press.
4. Kamil Dołęgło, Anna Paszyńska, Maciej Paszyński, Leszek Demkowicz, *Deep neural networks for smooth approximation of physics with higher order and continuity B-spline base functions*, arXiv:2201.00904, 1-44 (2022)
5. Yutian Wang, Yuan-Hua Ni, *Deep BSDE-ML Learning and Its Application to Model-Free Optimal Control*, arxiv:2201.01318, 1-20 (2022)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_41