جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Deep Neural Network and Partial Differential Equations

Finite Neuron Method and PINNs

December 6, 2021

Table of contents

1. Neural Network

2. RELU Network and FEM

3. Physically Informed Neural Network – PINNs

4. Conclusions

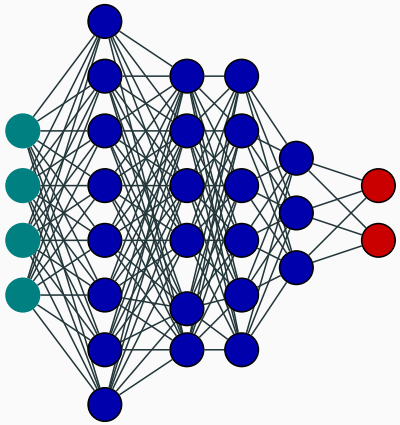
Neural Network

Neural Networks

Let us fist address the notion of what is a deep neural network and in particular what is a shallow neural network.

Let us fix an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, we extend it component wise to,

$$\sigma : \mathbb{R}^n \to \mathbb{R}^n$$
$$\mathbf{x} \mapsto (\sigma(x_1), \dots, \sigma(x_n))$$



Now we fix $\mathbf{n} \in \mathbb{N}^{(d+2)}$ and we say that a function $f : \mathbb{R}^{n_1} \to \mathbb{R}^{n_{d+2}}$ is a neural network if,

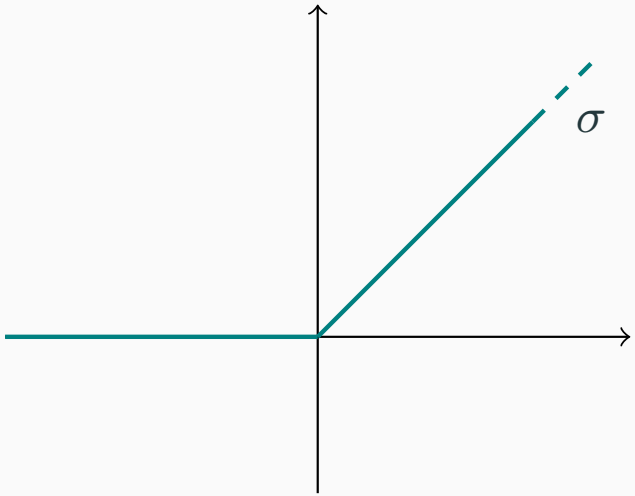
$$f(\mathbf{x}) = W_d \left(\sigma \left(W_{d-1} \dots \sigma \left(W_0 \mathbf{x} + \mathbf{b}_0\right) + \mathbf{b}_{d-1}\right) + \mathbf{b}_d\right). \tag{1}$$

We say a neural network is shallow if $d = 1$ while we say that a neural network is deep if $d > 1$. We will write DNN_1^n to denote a shallow neural network with N neurons in its hidden layer.

Rectified Linear Unit

An activation function of particular interest is the rectified linear unit (RELU), defined as,

$$\sigma : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto x^{+}.$$



RELU activation function have many interesting features and we are going to explore their connection with finite element basis in a moment.

Approximation Property DNN_1

Given a function $f : \mathbb{R}^d \to \mathbb{R}$ sufficiently smooth the following approximation estimate holds,

$$\inf_{f_N \in DNN_1^n} ||f - f_n||_{\mathcal{L}^2(\mathbb{R}^d)} \leq C_f n^{-\frac{1}{2}} \tag{2}$$

RELU Network and FEM

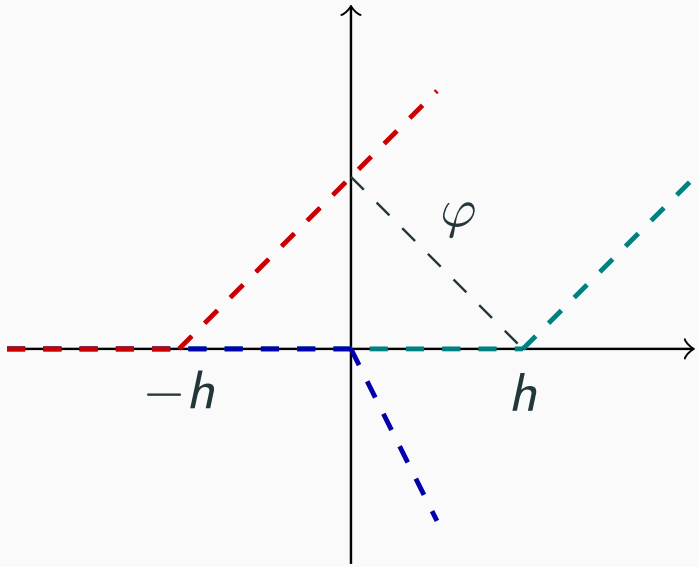
Rectified Linear Unit and FEM Shape Function

We now want to show that that one can reconstruct the linear finite elements space using RELU activation functions.

We know the finite element shape function on an uniformly spaced one dimensional mesh are defined as,

$$\varphi : [0,1] \to \mathbb{R}$$

$$x \mapsto \begin{cases} h^{-1}x & x \in [-h, 0] \\ 1 - h^{-1}x & x \in [0, h] \\ 0 & x \notin [-h, h] \end{cases}$$



now is just a matter of observing that one can rewrite the shape function φ as a linear combination of RELU functions with bias,

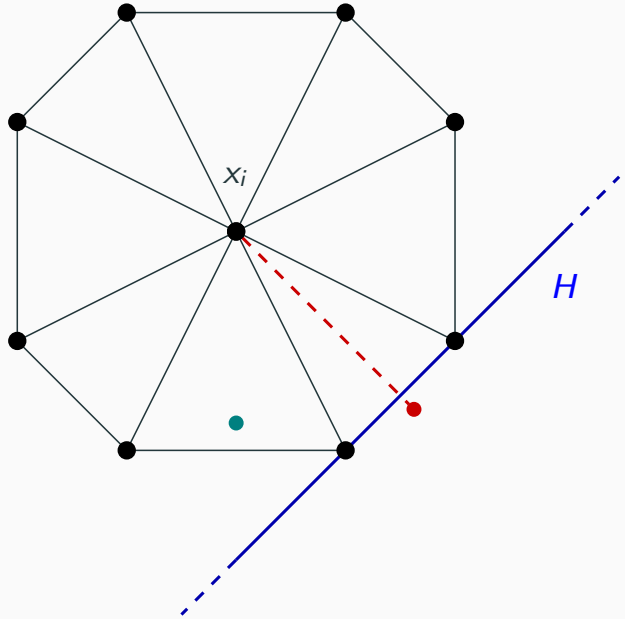
$$\varphi(x) = \boxed{h^{-1}\sigma(x+h)} - \boxed{2h^{-1}\sigma(x)} + \boxed{h^{-1}\sigma(x-h)}. \tag{3}$$

RELU and FEM – Representation of Shape Function

Now we would like to prove the same result for an n-dimensional case.

Lemma – Max Min Shape Function
Given a point x_i, we denote $G(i)$ the union of elements of the mesh having x_i ad vertex, then the tent function corresponding to x_i can be written as,

$$\varphi_i(x) = max\Big\{0, \min_{T_k \subset G(i)} g_k(x)\Big\} \qquad (4)$$



where the function g_k is the linear function coinciding with the tent function on T_k, provided $G(i)$ is convex.

To explain this we consider two different cases, the first is when x belong to $T_{k_0} \subset G(i)$. Then we consider the hyper plane H and we focus our attention on g_{k_0}. In particular for all $y \in H \cap T_{k_0}$ we notice that,

$$g_k(y) \geq g_{k_0}(y). \qquad (5)$$

RELU and FEM – Representation of Shape Function

Now $g_k(x_i) = g_{k_0}(x_i) = 1$ and therefore $g_k(x) \geq g_{k_0}(x)$.

Now let as assume that $x_i \notin G(i)$ then we consider the hyper plane connecting x and x_i, denoted in red. Now we consider the mesh element where the hyper plane lies to draw the H hyper plane. In particular since x_i and x are on different side of the hyper plane H and we know g_k is null on H then $g(x) < 0$ and therefore we need to impose the maximum with zero.

RELU and FEM – The Merge and Sort Proof

We are now ready to prove the inclusion of the FEM space with in shallow RELU neural network.

Theorem
Given a locally convex d-dimensional grid any P_1 spline on the grid characterised by N degrees of freedom can be also expressed as a RELU neural network with $log_2(k_{\mathcal{T}}) + 1$ hidden layers and $\mathcal{O}(k_{\mathcal{T}} N)$ neuron, where $k_{\mathcal{T}}$ is the maximum number of neighborhood elements in the mesh.

Proof
We notice the following to begin with,

$$min\{a, b\} = \frac{a+b}{2} - \frac{a-b}{2} = v\sigma\left(W \cdot \begin{bmatrix} a \\ b \end{bmatrix}\right), \tag{6}$$

where v and W are defined as,

$$v = \frac{1}{2}\begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \qquad W = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{7}$$

RELU and FEM – The Merge and Sort Proof

Now from the previous lemma we know we can represent each shape function as a max min problem, in particular the idea is to split each minimisation into sub minimisation until we are only comparing two elements. This operation can be represented using one RELU mapping. Therefore we need to concatenate $log_2(k_{\mathcal{T}})$ operations in this process. Now since the structure of the three we have constructed in binary we can easily see that the number of nodes needed in the network is 2^k where k is the depth of the three, this result in $\mathcal{O}(k_{\mathcal{T}})$ nodes per shape function.

RELU and FEM – Approximation Theory

Now that we have established a connection between linear finite elements and shallow RELU neural network we can use this connection to develop an interesting approximation estimate property,

Theorem
Given $\Omega = [a, b]$ and a function $f \in H^2(\Omega)$ it exists a function $f_n \in DNN_1^n$ such the following estimate works,

$$||f - f_n||_{L^2(\Omega)} \leq C_f n^{-2}. \tag{8}$$

Proof
The idea is the following, we know that for a function $f : [a, b] \to R$ that lives in $H^2(\Omega)$, it exists a linear FEM function $f_n : [a, b] \to R$ such that:

$$||f - f_n||_{L^2(\Omega)} \leq C|f|_{H^2} \left(\frac{n}{3}\right)^{-2}, \tag{9}$$

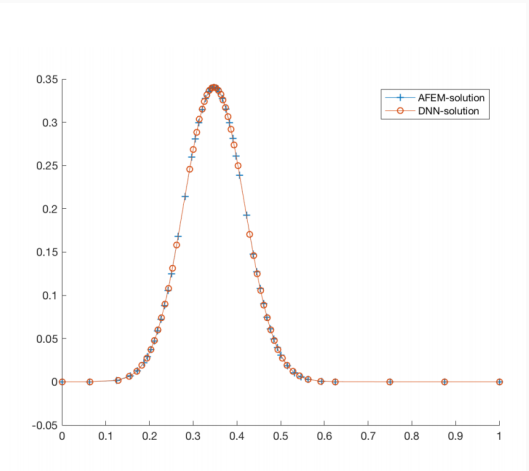
and this function can be represented as a DNN_1 with RELU activation function.

RELU and FEM – Finite Neuron Method

We now would like to present an application of RELU DNN_1 to solve partial differential equations, in particular the idea is to train the nueral network using as a loss function the well known energy asssociated with the 2nd order elliptic problem, i.e.

$$\mathcal{E}(u) = \int_\Omega |u'(x)|^2 - fu \, dx \qquad (10)$$

In particular the minimization is done in two steps, first one minimize over the neural network bias and then one minimize over the layer parameters.



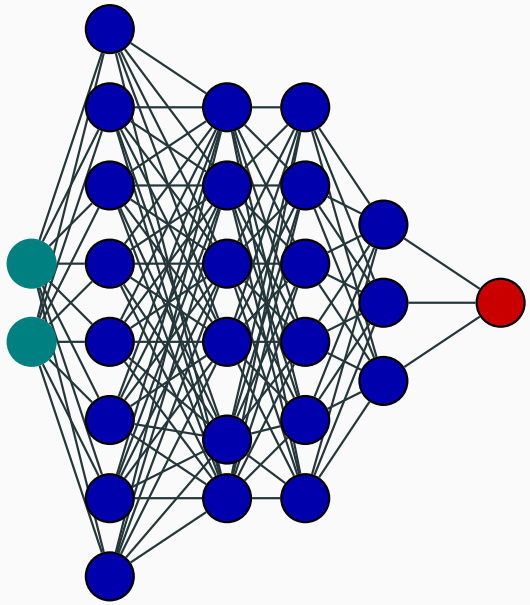
This examples show that there is "adaptivity" embedded in the scheme, and gives a special meaning to the bias layer in the network.

Physically Informed Neural Network – PINNs

Physically Informed Neural Networks – PINNs

We would like to use Neural Network to solve partial differntial equations in 2D, in particular we would like to use the physically informed neural network.

The idea behind PINN is to construct a NN with the shape on the left and minimize the averaged least square residual on collocation points that are randomly sampled.



In particular the loss functional taken into consideration is,

$$\mathcal{L}(u, f) = \frac{1}{N}\sum_{i}^{N} |\Delta u(\mathbf{x_i}) - f(x_i)|^2 + \frac{1}{N}\sum_{i}^{N} |u(\mathbf{b_i})|^2 \tag{11}$$

$$\mathbf{x_i} \sim \mathcal{U}(\Omega) \qquad \mathbf{b_i} \sim \mathcal{U}(\partial\Omega) \qquad \Omega = [0,1]^2 \tag{12}$$

PINNs– Laplacian Eigenvalue

The minimization problem for the PINN require a long training in particular the figures below are computed using 50000 iteration of ADAM.

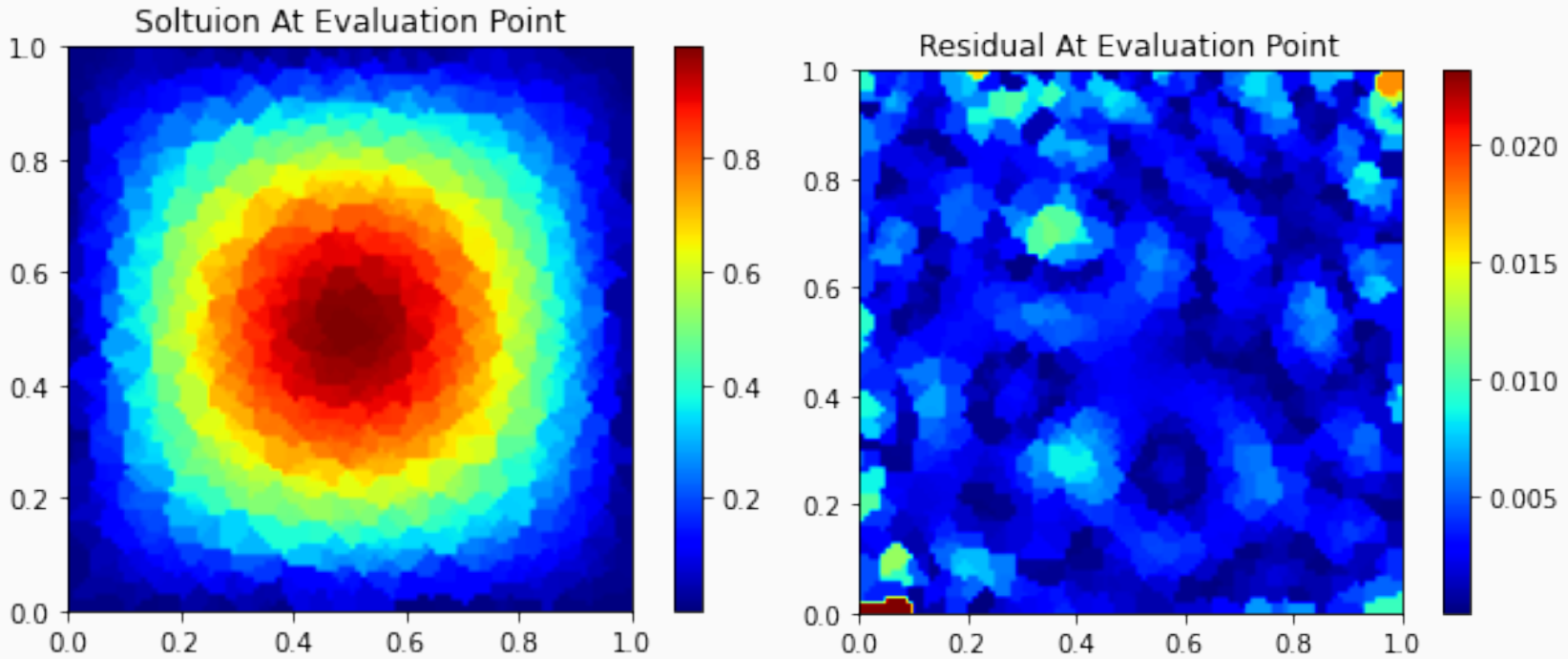


Figure 1: The figures above show the first eigenfunction of the Laplacian computed numerically and the residual both in the collocation points.

PINNs – Laplace Eigenfunctions

One strong advantage of PINNs is that they outperform all other method at inference time, by a very large factor.

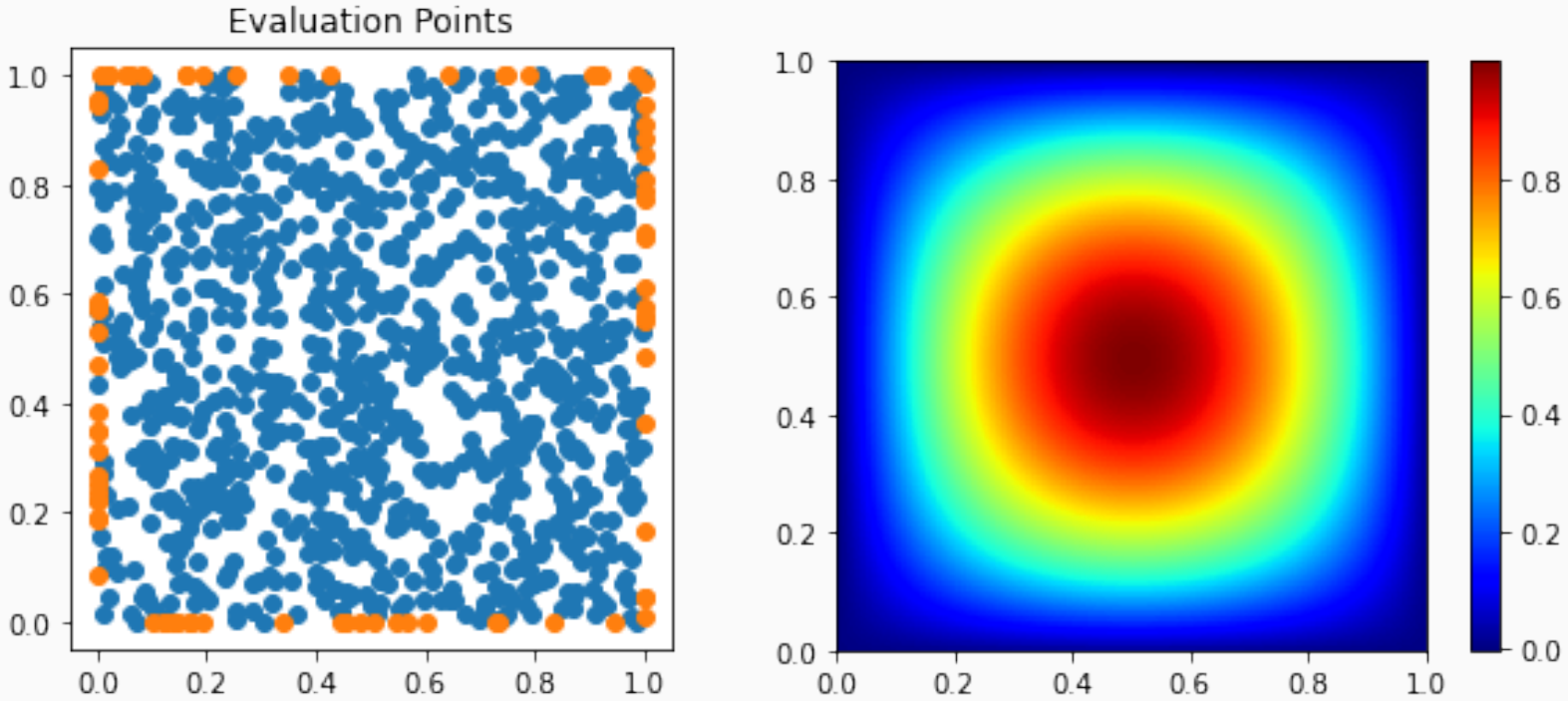


Figure 2: The figures shows on the left the points used to evaluate the collocation error, while the right one can see the solution at inference time on million degree of freedom.

Conclusions

Conclusions

Neural Network offer a very interesting and fresh approach to solving partial differential equation, which offers drastic performance improvements at inference time. Furthermore this field is reach with questions that still need to be answered,

1. **Sharp A Priori Error Estimates**, need to be developed. Some work has been done in the last year by Jinchao Xu and its collaborator.
2. **Training Technique**, faster and more efficient training techniques need to be developed for this specific problem. An interesting approach is the one the ECRC here at KAUST is taking to this problem, i.e. using second order method to solve the underlying optimization problem.
3. **Network Design**, right now the network design adopted in both PINNs and Finite Neuron is plain vanilla neural network, is there a better network layout to be used in the case of PDE ?

Thank you !