Paper 6: Deep Ritz Method

By: Khaled Kerouch, Jinwei Zhang

Authors of Paper:

Weinan E, Bing Yu

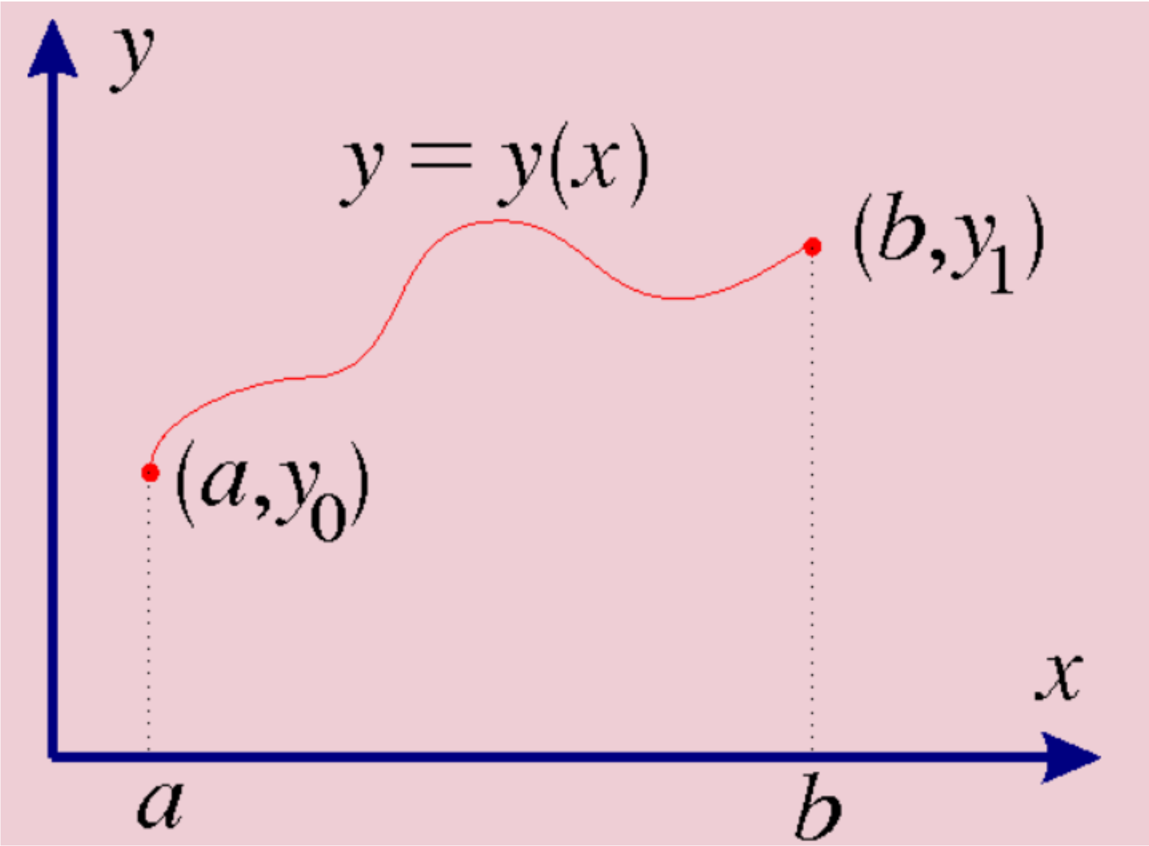
Today

- Introduction
- Methodology
- Poisson Equation
- Key Results
- Discussion
- Conclusion
- Q&A

Deep Ritz Method

- Variational Problems
- Nonlinear, adaptive, potential to work in rather high dimensions
- Neural network representation of functions
- Ritz Method

Variational Problems



- Finding minima and maxima of functionals.
- Functional: Maps functions to real numbers.

Example Problem

$$\min_{u \in H} I(u)$$

Where,

$$I(u) = \int_{\Omega} \left(\frac{1}{2} |\nabla u(x)|^2 - f(x)u(x) \right) dx$$

Mathematical Solution (1D)

- Assume no boundary conditions. In our case $\Omega = [0,1]$ and u(a) = u(b) = 0.
- $I(u) = \int_\Omega L\big(x, u(x), u'(x)\big)dx$
- $\delta I(u) = -\int_\Omega \big(f(x) + u''(x)\big)\delta u \, dx$
- $\delta I(u) = 0$
- $-u'' - f(x) = 0$
- $\Rightarrow u'' = -f(x)$

$$L(x, u(x), u'(x)) = \frac{1}{2}u'(x)^2 - f(x)u(x)$$

$$\delta I = \int_\Omega \left(\frac{\partial L}{\partial u} - \frac{d}{dx}\frac{\partial L}{\partial u'}\delta u(x)dx \right) + \frac{\partial L}{\partial u'}(b)\delta u(b) - \frac{\partial L}{\partial u'}(a)\delta u(a)$$

$$\int_a^b f(x)h(x)\,\mathrm{d}x = 0 \quad \Rightarrow \quad f(x) = 0$$

Multivariate Case

$$F\big(x, u(x), \nabla u(x)\big) = \frac{1}{2}|\nabla u(x)|^2 - f(x)u(x)$$

$$\delta I(u) = -\int_\Omega \delta u\big(\Delta u + f(x)\big)dx$$

$$\delta I(u) = 0$$

$$\Rightarrow \Delta u = -f(x)$$

$$J[\rho] = \int F(r, \rho(r), \nabla\rho(r))dr$$

$$\delta J = \int \left(\frac{\partial F}{\partial \rho} - \nabla \cdot \frac{\partial F}{\partial \nabla \rho}\right)\phi(r)dr$$

Methodology

3 core ideas:

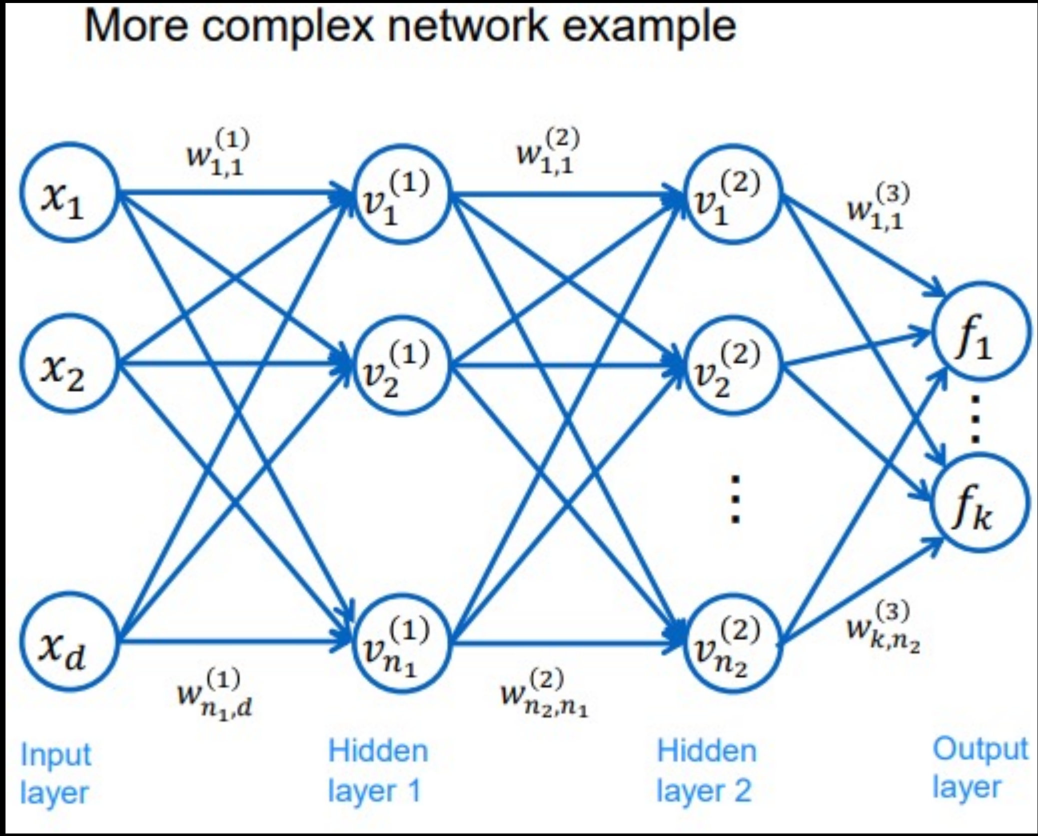
- Deep neural network-based approximation of trial function
- Algorithm for Solving final optimization (Mini-Batch SGD)
- Numerical quadrature rule for the functional

Fully Connected Neural Network

- $z^{(1)} = W_1 x$
- $v^{(1)} = \phi(z^{(1)})$
- $z^{(2)} = W_2 v^{(1)}$
- $v^{(2)} = \phi(z^{(2)})$

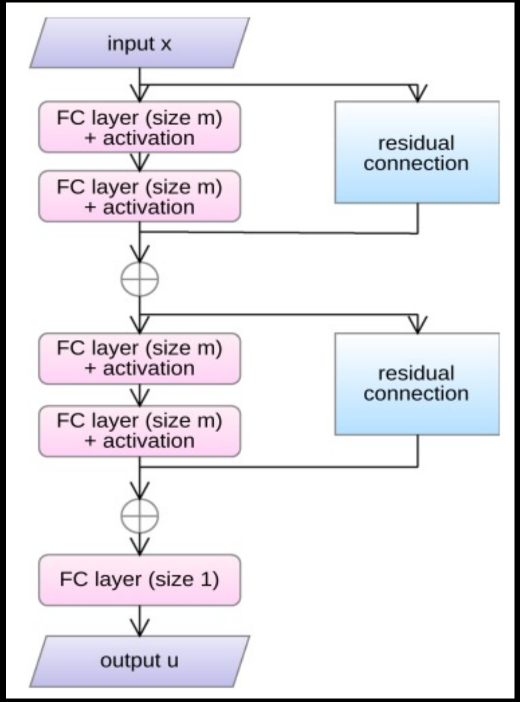
$, i \in [1, m]$

- $f = W_3 v^{(2)} = W_3 \cdot \phi(z^{(2)}) = W_3 \cdot \phi(W_2 \cdot \phi(W_1 x))$



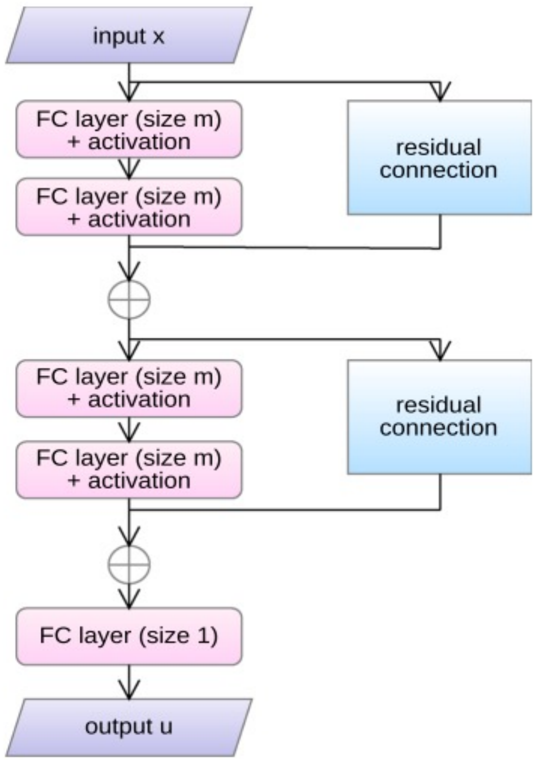
Idea 1: Neural Network based Approximation of trial function

- Nonlinear transformation: $x \rightarrow z_{\theta}(x) \in \mathbb{R}^m$, where θ set of parameters
- Network constructed by stacking blocks: Each block consists of 2 linear transformation, two activation functions + residual connection
  - ➢ Input s, output t. $s, t \in \mathbb{R}^m$



Architecture

- i-th block:
  - $t = B_i(s) = \phi\left(W_{i,2} \cdot \phi\left(W_{i,1}s + b_{i,1}\right) + b_{i,2}\right) + s$
- where $W_{i,1}, W_{i,2} \in \mathbb{R}^{m \times m}$ and $b_{i,1}, b_{i,2} \in \mathbb{R}^m$ are parameter of said block
- ϕ is activation function, in this case $\phi(x) = \max\{x^3, 0\}$.



Idea 1: Continued

Full n-layer network expressed as:

- $z_\theta(x) = B_n \circ \cdots \circ B_1(x)$

We obtain u by

- $u(x;\theta) = a \cdot z_\theta(x) + b.$

θ new parameter set {θ, a, b}

Substituting this into I + defining g:

- $g(x{:}\,\theta) = \frac{1}{2}|\nabla_x u(x;\theta)|^2 \; - f(x)u(x;\theta)$

$$\min_{u \in H} I(u),$$

$$I(u) = \int_\Omega \left(\frac{1}{2}|\nabla u(x)|^2 - f(x)u(x) \right) \mathrm{d}x$$

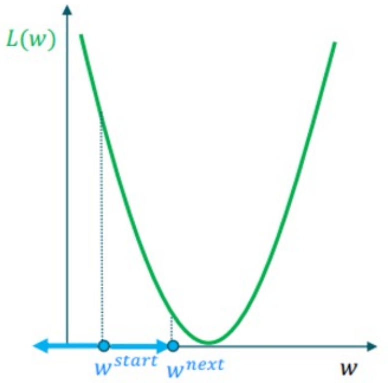
$$\Rightarrow \min_{\theta} \mathrm{L}(\theta), \qquad L(\theta) = \int_\Omega g(x;\theta)dx$$

Gradient Descent

- Optimization Problem:

$$\min_{w \in R^d} L(w) := \frac{1}{N}\sum_{i=1}^{N} L_i(w)$$

Where L(w) is avg loss and $L_i(w)$ loss for i-th data point.



Setting now: Minimize $L(w)$ with scalar w

Gradient descent algorithm to minimize $L(w)$

Start at initial w^0 — for simplicity, fixed for all t

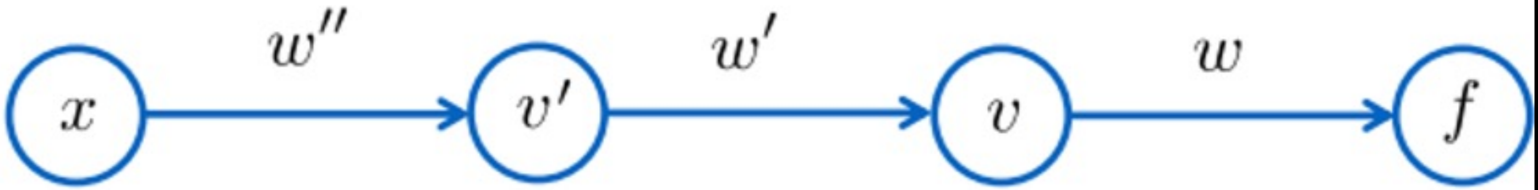
At each step t, $w^{t+1} = w^t - \eta \nabla_w L(w^t)$

Stop e.g. when $|L(w^{t+1}) - L(w^t)| \leq \epsilon$

Output $\hat{w} = \cancel{w^{final}}$ w^T

Backpropagation

ANN with 1 output, **2** hidden and 1 input unit: $f(x;W) = f(x;[w,w',w'']) = w\varphi(w'\varphi(w''x))$



$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial w} = l'(f)v = \delta v$$

$$\frac{\partial l}{\partial w'} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial v}\frac{\partial v}{\partial z}\frac{\partial z}{\partial w'} = \delta w\dot{\varphi}(z)v'$$

$$\frac{\partial l}{\partial w''} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial v}\frac{\partial v}{\partial z}\frac{\partial z}{\partial v'}\frac{\partial v'}{\partial z'}\frac{\partial z'}{\partial w''} = \delta w\dot{\varphi}(z)w'\dot{\varphi}(z')x$$

Stochastic Gradient Descent

Remember training loss definition for parameterized functions $L(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_w(x_i))$

Gradient descent algorithm

Start at initial w^0

Until $w^{t+1} - w^t$ small, repeat:

Update $w^{t+1} \leftarrow w^t - \eta \nabla_w L(w^t)$

Output $\hat{w} = w^T$

compute gradients at all points

$\frac{1}{n}\sum_{i=1}^{n} \nabla_w \ell(y_i, f_w(x_i))$

Costly! memory: require $O(nd)$ values to compute
computation: $O(n \times \text{cost to compute } \nabla l)$ per update.

S(tochastic) G(radient) D(escent) algorithm

Start at initial w^0

Until $w^{t+1} - w^t$ small, repeat:

Update $w^{t+1} \leftarrow w^t - \eta \nabla_w L_S(w^t)$

Output $\hat{w} = w^T$

random subset of points $S \subset [1, \ldots, n]$ (minibatch SGD)

$\nabla_w L_S(w) = \frac{1}{|S|} \sum_{i \in S} \nabla_w \ell(y_i, f_w(x_i))$

when S is just one random point it's called SGD

Idea 2: SGD

In our case:

$$\theta^{k+1} = \theta^k - \eta \nabla L_{\gamma^k}(\theta^k).$$

Here $\{\gamma^k\}$ i.i.d random variables, uniformly distributed over {1,2,...,N}. N := #Data Points

Data Points?

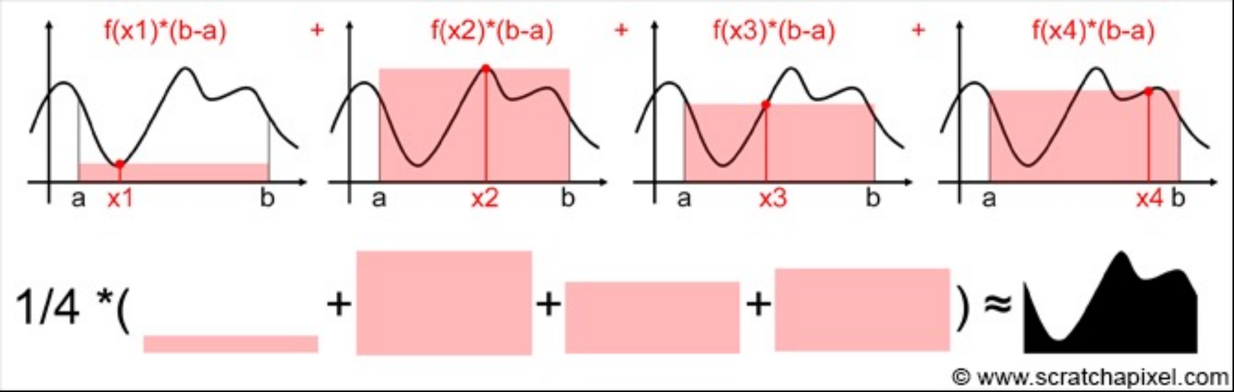
View integral in I as continuous sum

⇒ Each point in Ω then becomes a data point

$$I(u) = \int_\Omega \left(\frac{1}{2} |\nabla u(x)|^2 - f(x)u(x) \right) \mathrm{d}x$$

Idea 3: Monte Carlo Integration

$$\min_{\theta} L(\theta), \quad L(\theta) = \int_{\Omega} g(x;\theta)\mathrm{d}x. \quad I(u) = L(x,\theta) = \frac{1}{N}\sum_{i=1}^{N} g(x_i,\theta), \qquad I \approx Q_N \equiv V\frac{1}{N}\sum_{i=1}^{N} f(\overline{\mathbf{x}}_i)$$



Hence

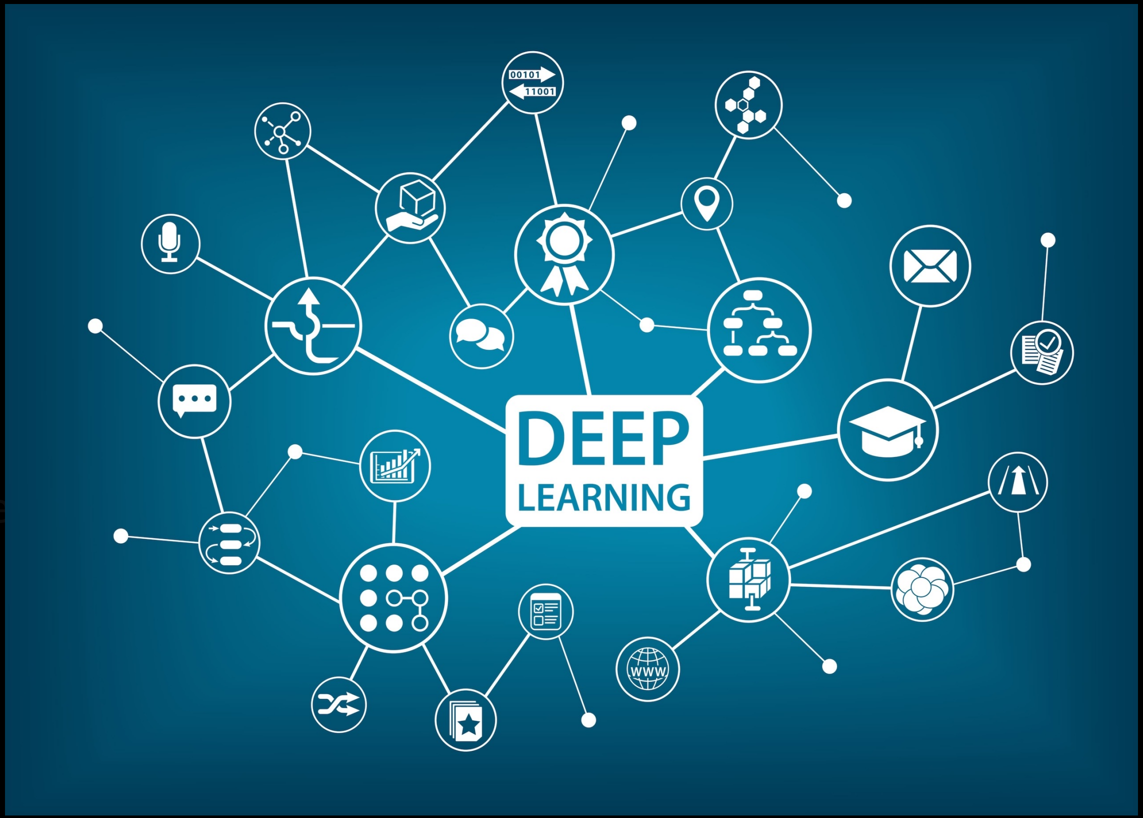
$$\theta^{k+1} = \theta^k - \eta\nabla_\theta \frac{1}{M}\sum_{j=1}^{M} g\left(x_{j,k};\theta^k\right)$$

where for each k, $\{x_j, k\}$ set of points on Ω uniformly sampled

Example Applications

- Poisson Equations
- Transfer Learning
- Eigenvalue Problems

Eigenvalue proble



Poisson Equation

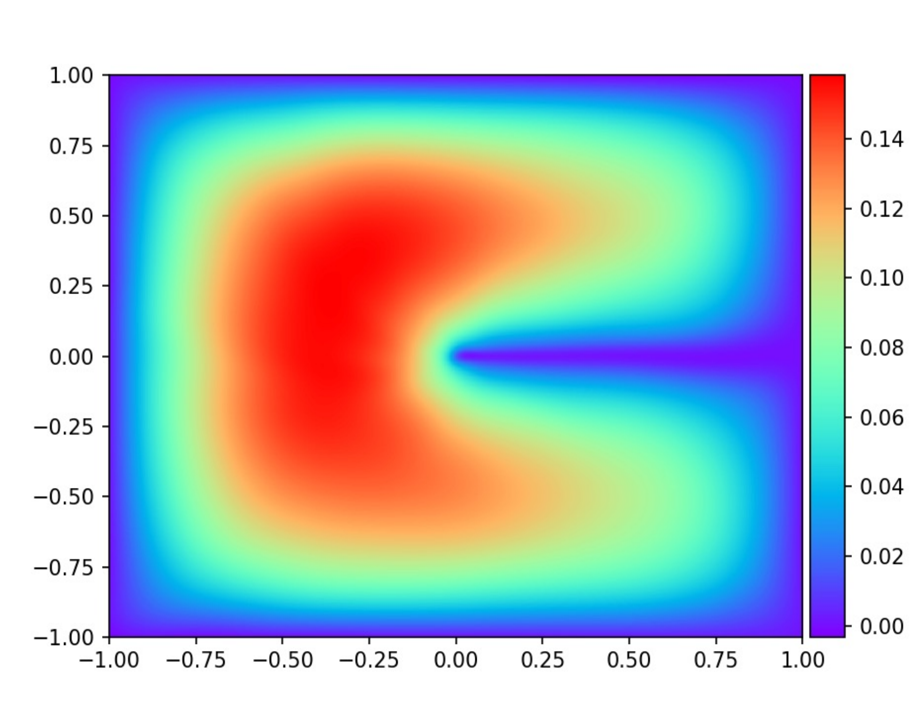
- $-\nabla u=f$
- Applications: Electrostatics, Gravitation, Heat Transfer, Fluid Dynamics
- Formulation as a Variational Problems: $I(u)= \int (\frac{1}{2} |\nabla u|^2 - fu)dx$
- Training the Network: Optimization and Loss Function

Consider the Poisson equation:

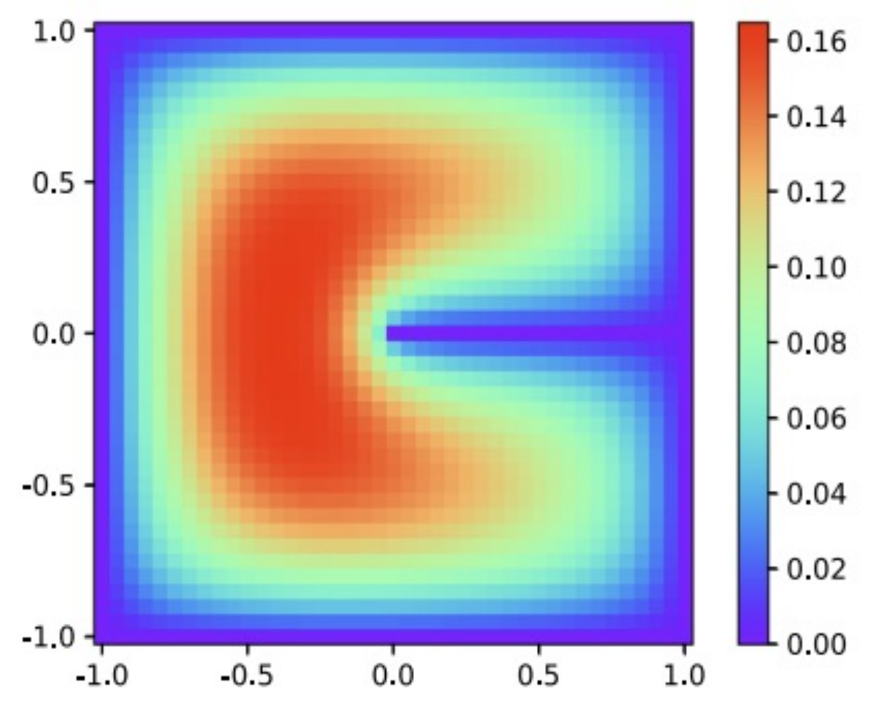
$$I(u) = \int_{\Omega} \left(\frac{1}{2} |\nabla_x u(x)|^2 - f(x)u(x) \right) \mathrm{d}x + \beta \int_{\partial\Omega} u(x)^2 \mathrm{d}s.$$

$$\begin{aligned} -\Delta u(x) &= 1, \quad x \in \Omega \\ u(x) &= 0, \quad x \in \partial\Omega, \end{aligned}$$

where $\Omega = (-1, 1) \times (-1, 1) \backslash [0, 1) \times \{0\}$

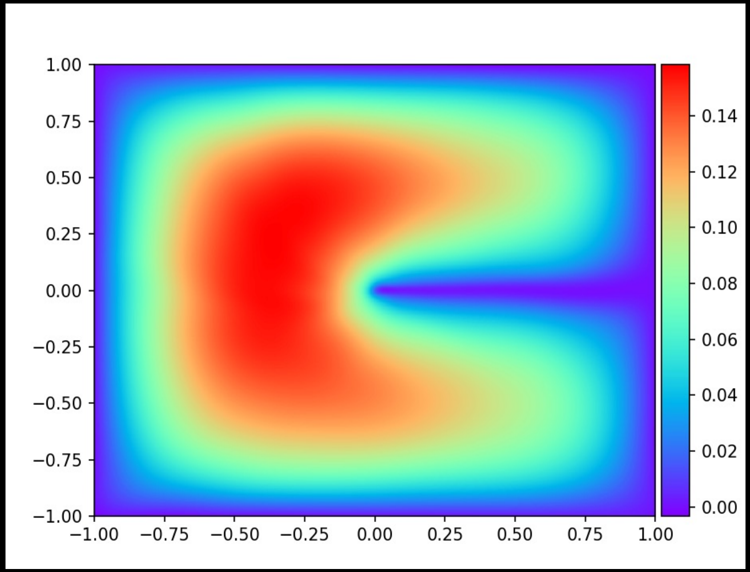


Solution of Deep Ritz Method,
811 parameters

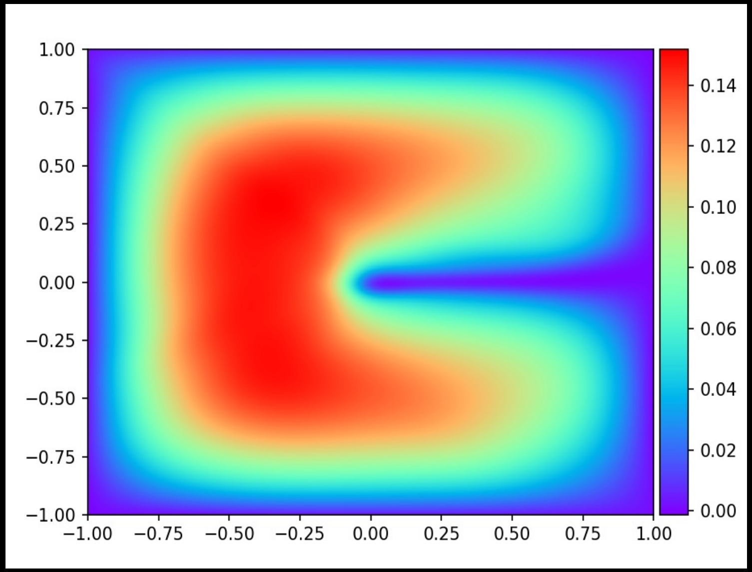


Solution of finite difference method, 1681 *parameters*

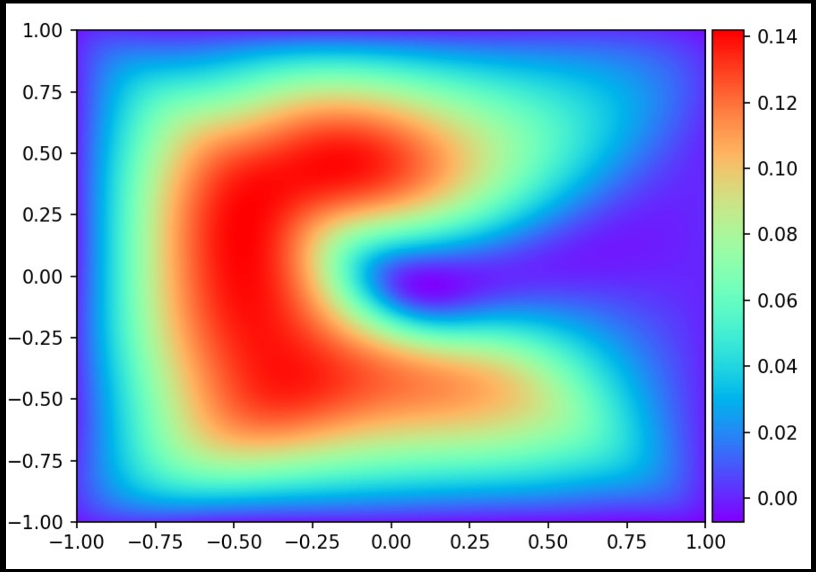
Performance comparison



Original Model



Activation function $\phi(x) = \max\{x, 0\}$



1 block instead of 4

To analyze the error more quantitatively, we consider the following problem

$$\Delta u(x) = 0, \qquad x \in \Omega$$
$$u(x) = u(r, \theta) = r^{\frac{1}{2}} \sin \frac{\theta}{2}, \qquad x \in \partial\Omega,$$

where $\Omega = (-1, 1) \times (-1, 1) \backslash [0, 1) \times \{0\}$.

Method	Blocks No.	Parameters	Relative L_2 error
DRM	3	591	0.0079
	4	811	0.0072
	5	1031	0.00647
	6	1251	0.0057
FDM		625	0.0125
		2401	0.0063

Error of Deep Ritz Method (DRM) and finite difference method (FDM)

Considering the Poisson equation:

$$\begin{cases} \Delta u = 1, & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \end{cases}$$

where $\Omega = \{(x, y) | x^2 + y^2 < 1\}$.

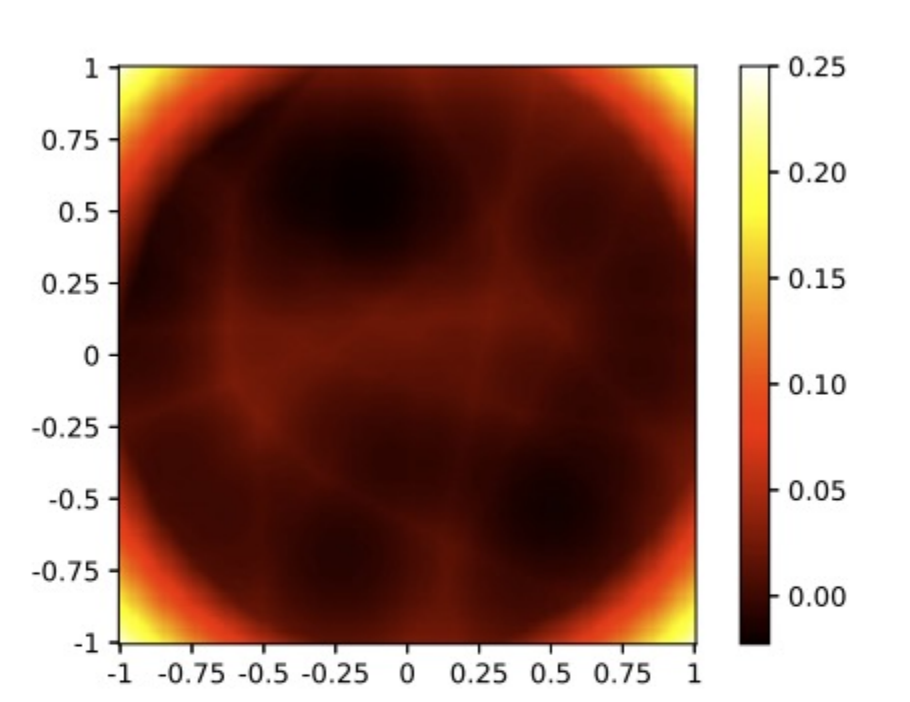
The exact solution to this problem is

$$u = \frac{1}{4}(x^2 + y^2 - 1).$$

$$I(u) = \int_{\Omega} \left(\frac{1}{2} |\nabla u(x)|^2 - u(x) \right) dx + \beta \int_{\partial\Omega} u(x)^2 dx,$$

Blocks Num	Parameters	Relative Loss (%)
1	231	3.2
2	451	2.0
3	671	1.3
4	891	1.5

Relative Loss of Different Layers with Poisson Equation



The results of Poisson Equation

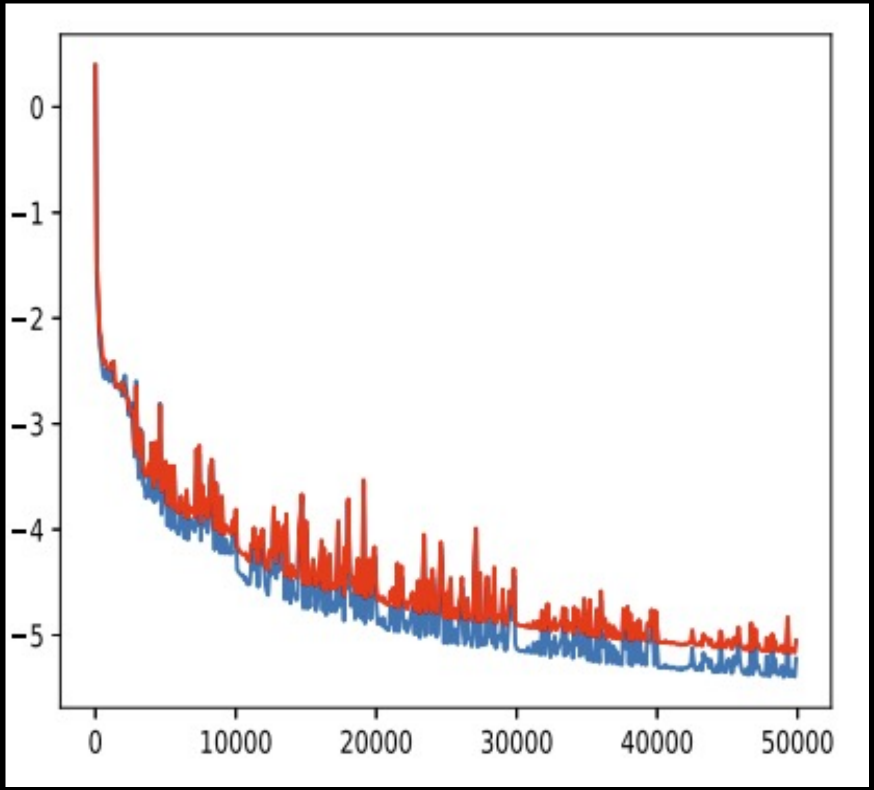
Poisson Equations in higher dimension

Consider ($d = 10$):

$$-\Delta u = 0, \qquad x \in (0,1)^{10}$$

$$u(x) = \sum_{k=1}^{5} x_{2k-1}x_{2k}, \qquad x \in \partial(0,1)^{10}.$$

$u(x) = \sum_{k=1}^{5} x_{2k-1}x_{2k},$

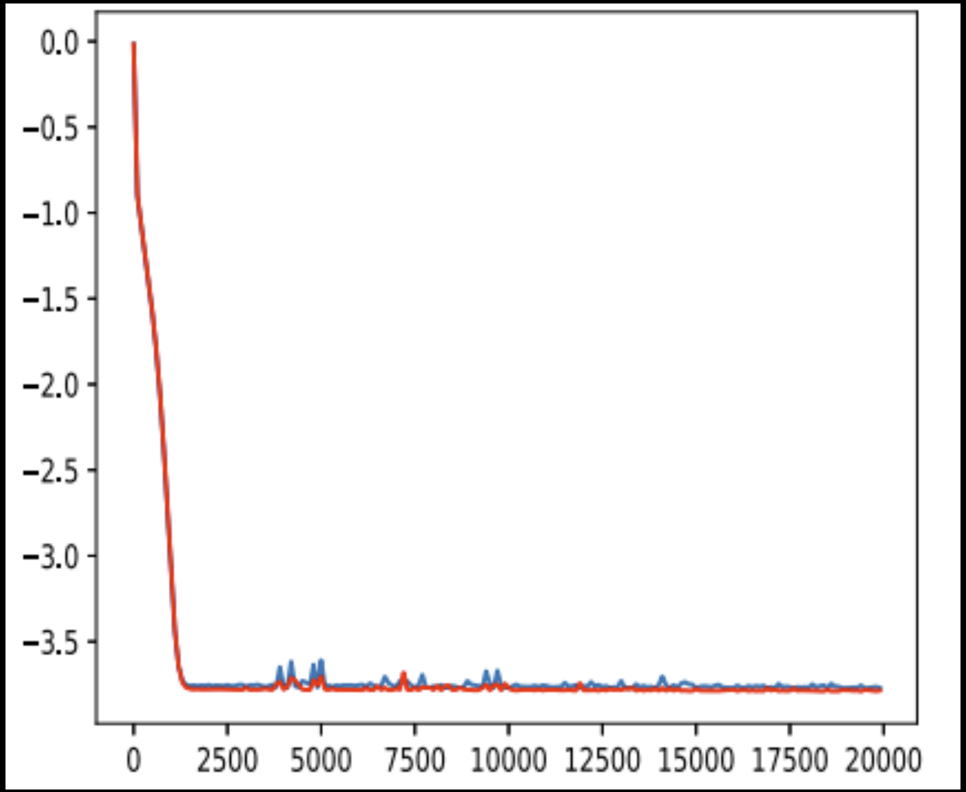


Blue curves: relative error of u
red curves: relative error on the boundary

Poisson Equations in higher dimension

$$
\begin{aligned}
-\Delta u &= -200, && x \in (0,1)^d \\
u(x) &= \sum_k x_k^2, && x \in \partial(0,1)^d, \\
& && d = 100
\end{aligned}
$$

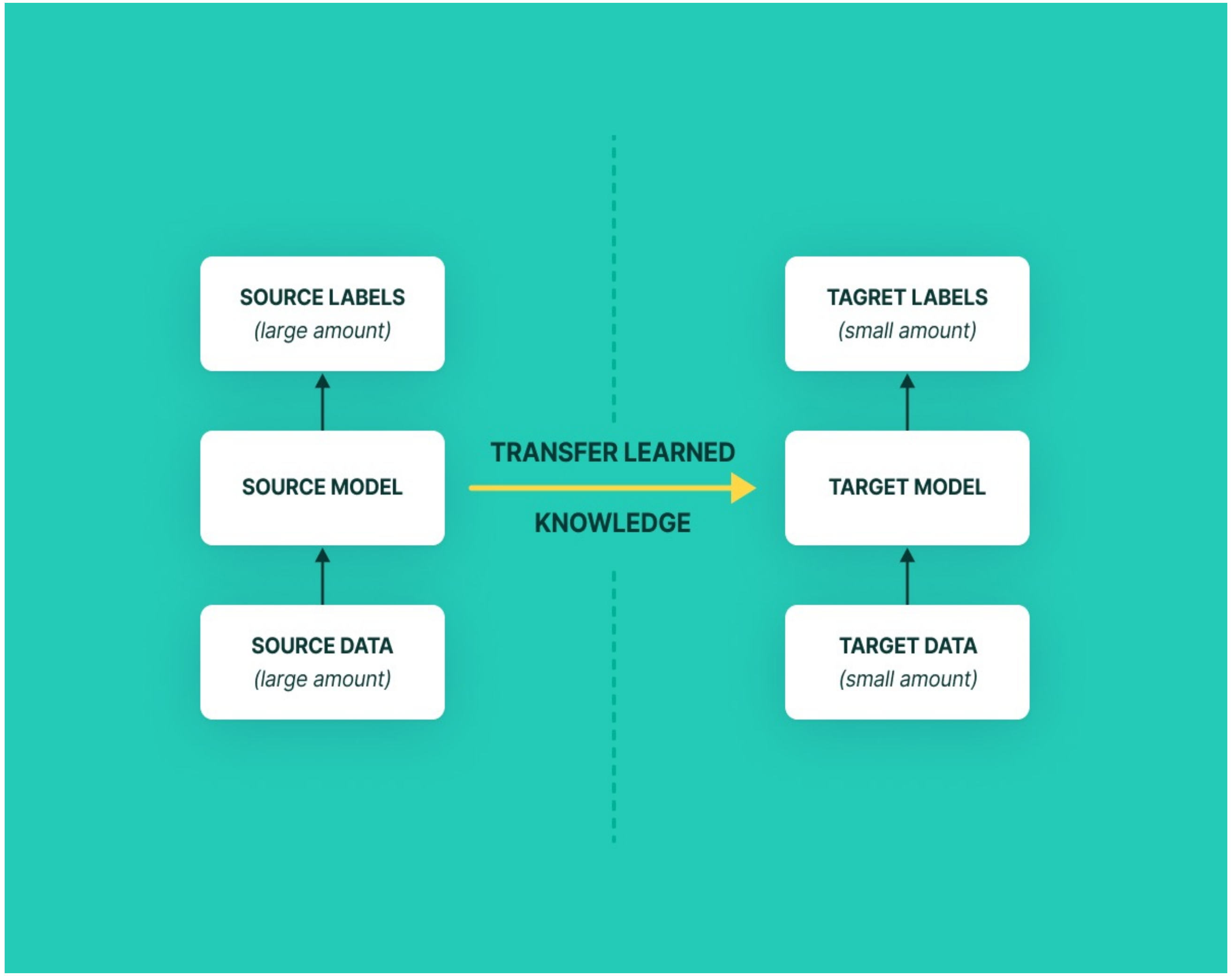
$u(x) = \sum_k x_k^2$



Blue curves: relative error of u
Red curves: relative error on the boundary

Transfer Learning

- enhance the training process by leveraging pre-trained network weights
- Benefits and Insights
- Challenges and Observations



Consider the problem:

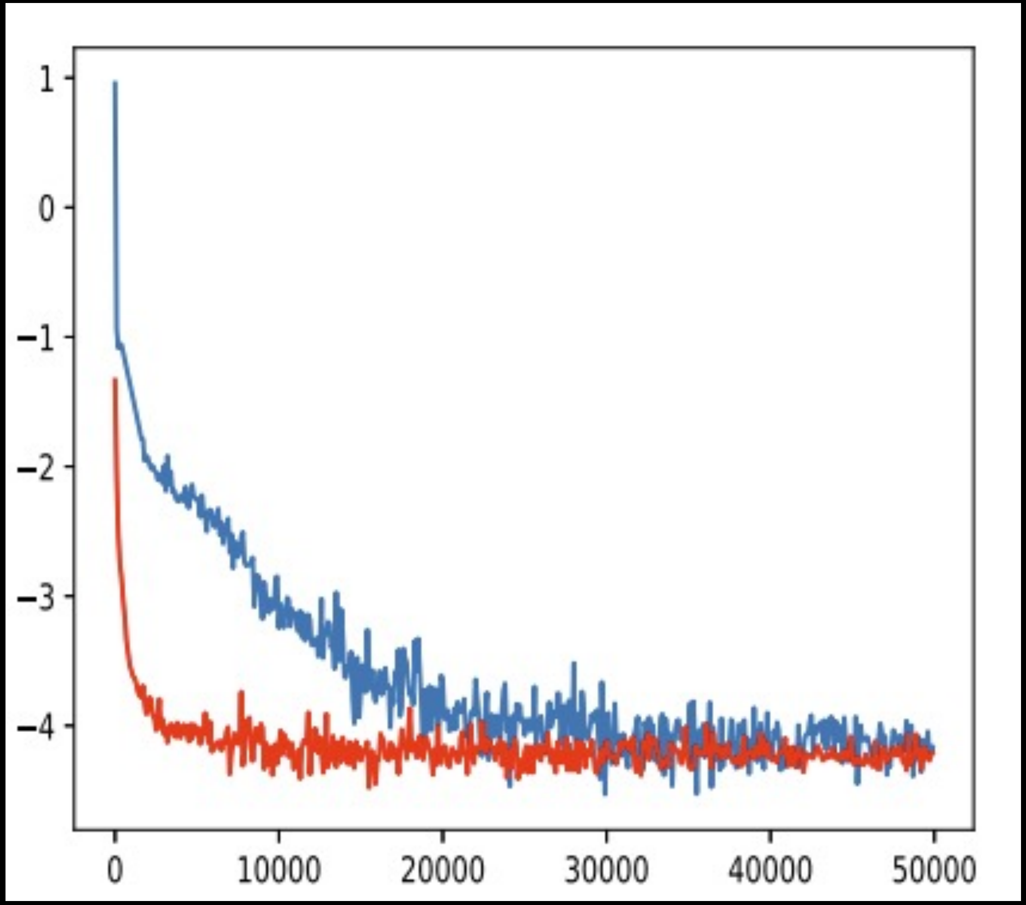
$$\begin{aligned} -\Delta u(x) &= 6(1+x_1)(1-x_1)x_2 + 2(1+x_2)(1-x_2)x_2, && x \in \Omega \\ u(x) &= r^{\frac{1}{2}} \sin\frac{\theta}{2} + (1+x_1)(1-x_1)(1+x_2)(1-x_2)x_2, && x \in \partial\Omega, \end{aligned}$$

where $\Omega = (-1,1) \times (-1,1) \backslash [0,1) \times \{0\}$

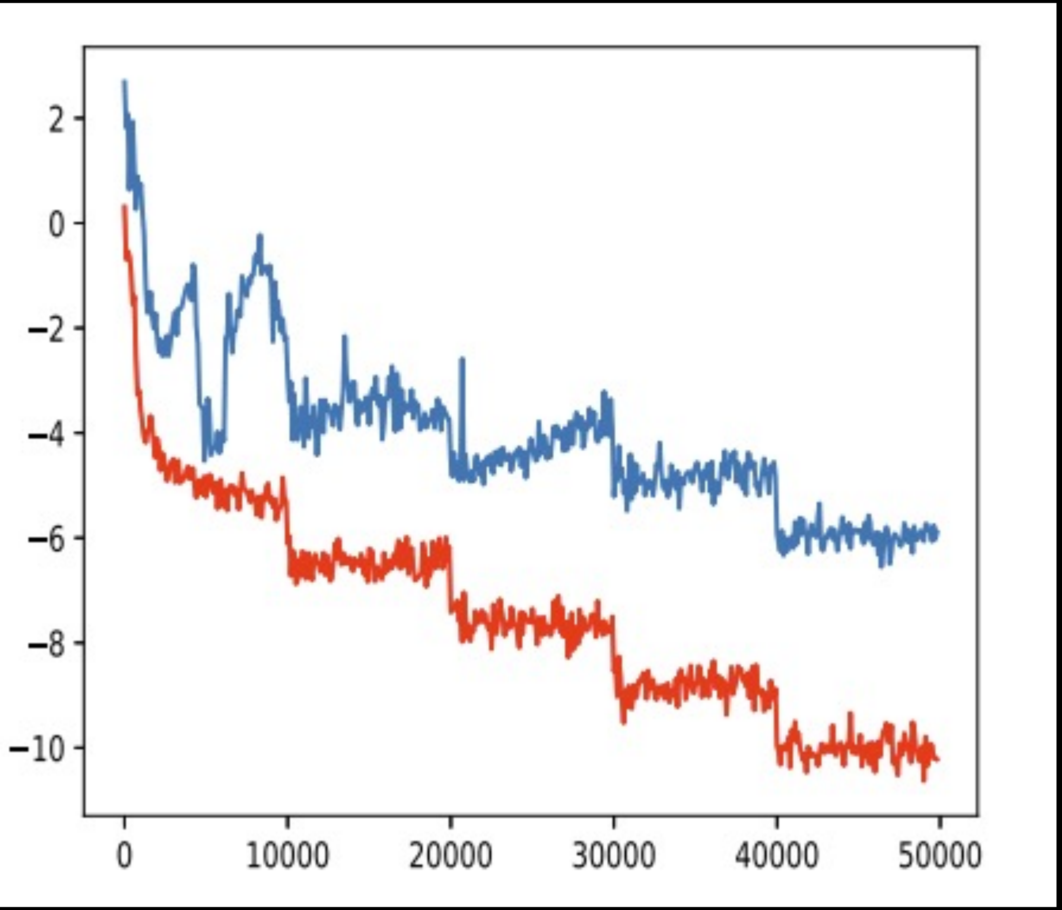
We also transfer the weights from the problem:

$$\begin{aligned} -\Delta u(x) &= 0, && x \in \Omega \\ u(x) &= r^{\frac{1}{2}} \sin\frac{\theta}{2}, && x \in \partial\Omega, \end{aligned}$$

where $\Omega = (-1,1) \times (-1,1) \backslash [0,1) \times \{0\}$.



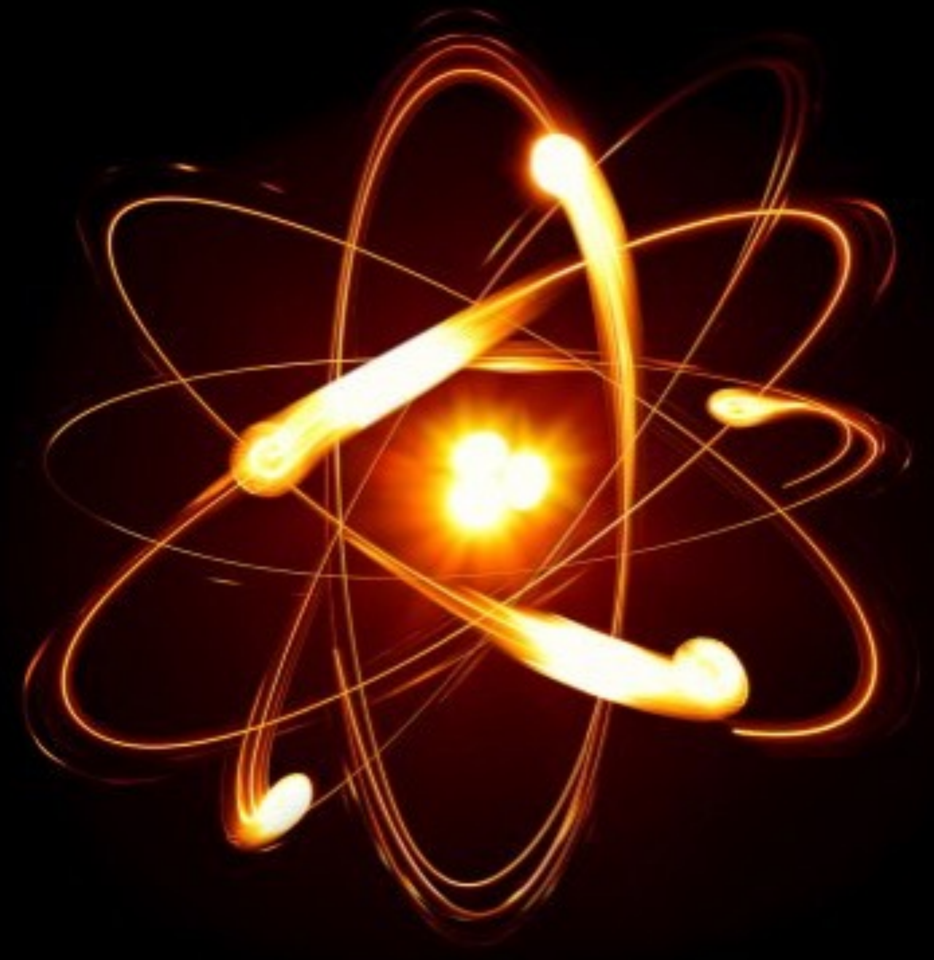
Red curve: training process with weight transfer
Blue curve: training process with random initialization
Graph: how the natural logarithm of the error changes during training



how the natural logarithm of $||W||_2^2$ changes during training
ΔW: change in W after 100 training steps
W: weight matrix

Eigenvalue problems

- $A\varphi=\lambda\varphi$
- Applications in engineering and physics
- Formulation as a variational problem: minimize $R(u)=\int|\nabla u|^2dx \,/\, \int u^2dx$
- Implementation using Deep Learning
-



Consider the following problem:

$$-\Delta u + v \cdot u = \lambda u, \quad x \in \Omega$$
$$u|_{\partial\Omega} = 0.$$

In practice, we use

$$L(u(x;\theta)) = \frac{\int_\Omega |\nabla u|^2 \mathrm{d}x + \int_\Omega v u^2 \mathrm{d}x}{\int_\Omega u^2 \mathrm{d}x} + \beta \int_{\partial\Omega} u(x)^2 \mathrm{d}x + \gamma \left(\int_\Omega u^2 \mathrm{d}x - 1 \right)^2 .$$

Dimension d	Exact λ_0	Approximate	Error (%)
1	9.87	9.85	0.20
5	49.35	49.29	0.11
10	98.70	92.35	6.43

Error of Deep Ritz Method

Consider the potential function

$$v(x) = \begin{cases} 0, & x \in [0,1]^d \\ \infty, & x \notin [0,1]^d \end{cases}$$

The problem is then equivalent to solving:

$$\begin{aligned} -\Delta u &= Eu, & x &\in [0,1]^d \\ u(x) &= 0, & x &\in \partial[0,1]^d. \end{aligned}$$

The smallest eigenvalue is $\lambda_0 = d\pi^2$.

Dimension d	Exact λ_0	Approximate	Error (%)
1	1	1.0016	0.16
5	5	5.0814	1.6
10	10	11.26	12.6

Error of Deep Ritz Method

Discussion

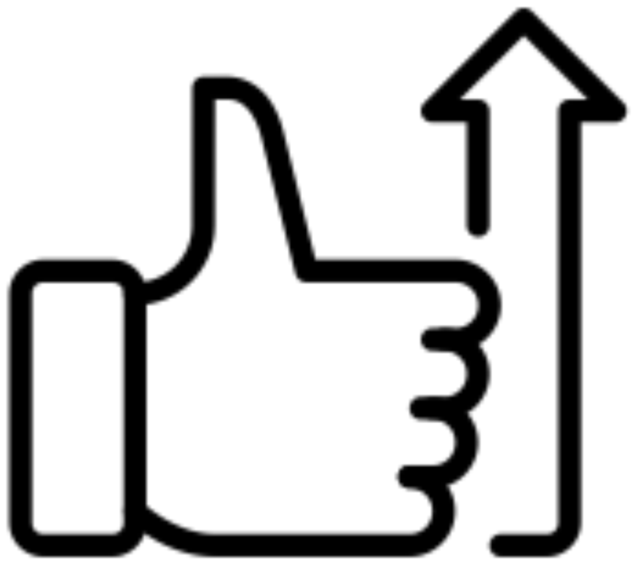
Strength of the Methods

Limitations and Challenges

Future Directions

Advantages of Deep Ritz Method

- Adaptability to various types of variational problems
- Capability to handle high-dimensional spaces effectively
- Integration with modern computational frameworks



Limitations and Challenges

- Non-convex optimization
- Formulation as variational problems
- Treatment of boundary conditions

Future Research Opportunities

- Explore different network architectures and activation functions
- Application to other variational problems
- Enhancements in the training process to improve efficiency and reduce computational costs

Conclusions

- Potential in high dimensions
- Not suitable for all PDEs
- Profiting from development of neural networks

Thank You for Your Attention! Questions?