A Brief Report of Deep Ritz Method

Zeyu Jia
School of Mathematical Science
Peking University
1600010603

Dinghuai Zhang
School of Mathematical Science
Peking University
1600013525

Zhengming Zou
School of Mathematical Science
Peking University
1600011089

Abstract

For long people have met great difficulties when solving numerical partial differential equations (PDEs), especially in high-dimensional conditions. In order to solve partial differential equations effectively, Yu and E proposed the deep Ritz method [13], combining the traditional Ritz method with deep learning. With residual blocks and stochastic gradient descent, it works well under both low-dimensional and high-dimensional conditions. In this report we delve deeper into their work and propose several improvements on their models, includeing the self-adaptive regularization and actor-critic self-adaptive method. Besides, we also do numerical experiments by ourselves, and our improvements make the deep Ritz method work better. This project is held on `https://github.com/Jason-jzy/Deep-Learning-Team-Project`. All the code of our numerical experiments is included in the folder `Code`, and this report is also included in folder `Report`.

1 Introduction

Partial differential equation [3] is an important tool to characterize our world mathematically. From physics to chemistry, and from financial to engineering, partial differential equations always play an indispensable role. However, partial differential equation is always very hard to solve, and at most of the time it is impossible. So in practice, we often find some ways to simplify the original partial differential equations, and solve them. To achieve this goal, we also need to figure out some ways to solve these simplified partial differential equations, or at least, find their approximate solutions.

The simplified partial differential equation we are going to discuss in this report is the Poisson equation (1)

$$\begin{cases} \Delta u = f, & \forall x \in \Omega, \\ u = g, & \forall x \in \partial\Omega. \end{cases} \tag{1}$$

The traditional partial differential equation solving methods include finite difference method [10], finite volume method [11] and finite element method [2]. These methods are the most commonly used. And complete theories for these methods, such as consistency, stability and convergence, have been well established. However, these methods will become tough, even impossible, to complicated functions or high dimension situation. Even though the convergence still holds, the time complexity it guarantees will not permit such enormous calculations. Thus, we need to develop some methods to tackle these difficulties.

Deep learning [9] has been well developed recently. Benefited from its well approximating ability, flexibility and power to tackle the curse of dimensionality, we have achieved state of arts on so many missions, such as classification, object detection and so on. So it is natural to think about a suitable way to use deep learning to solve partial differential equations. [13] proposed a method to approximate the solution by a neural network, and use the well-known Ritz method to solve the variational problem. [5][12][1] first transfer partial differential equations into backward stochastic differential equations, and then use neural networks or other machine learning methods to approximate their solutions.

In this report, what we focus on is Deep Ritz Method [13]. We derive the theory of this algorithm, give our own interpretations and propose several novel ideas to this algorithm, including the self-adaptive sampling method and actor-critic self-adaptive sampling method. The self-adaptive sampling method, which samples points for regularization based on the approximate error, aims to improve the efficiency of the algorithm by sampling more points where functions are not fitted well and sampling less points where functions are fitted well. And the actor-critic self-adaptive sampling method is proposed to find a better function of approximation errors into sampling distribution. We also write our own numerical experiments and make some comparison between the original methods and methods we proposed.

The plan of this report is as follows: In section 2, we will give a brief introduction to Ritz method, which is the basis of deep Ritz method. The structure of deep Ritz method will be shown in section 3. In section 4, our novelty to the deep Ritz method will be exhibited and illustrated. And finally in section 5, we will present the numerical experiments we have done by ourselves and their interpretation.

2 The Ritz Method

The Ritz method [3] is a method based on the principle of least action to find the approximation to eigenvalue equations that cannot be solved easily (or at all) analytically. Mathematically, the Ritz method can be used to approximate the solution of partial differential equations in a function space. When we aim to seek a function $y(x)$ that maximize or minimize an integral $I(y(x))$. Suppose that we can approximate $y(x)$ with a linear combination of some linear independent functions:

$$y(x) = \varphi_0(x) + k_1\varphi_1(x) + ... + \varphi_N(x). \tag{2}$$

Where $\varphi_i(x), i \in \{1, 2, ..., N\}$ are a basis of the function space we are interested in. In many cases, we use the basis

$$\begin{aligned} &\cos x, \cos 2x, \cdots, \cos nx, \cdots \\ &\sin x, \sin 2x, \cdots, \sin nx, \cdots \end{aligned} \tag{3}$$

However, this basis is of infinite dimension, which means it is very tough to solve. So we always restrict our solutions space in finite dimension.

Next we take the homogeneous Dirichlet boundary value problem of the Poisson equation (4) as an example.

$$\begin{cases} \Delta u = -f(x), & x \in \Omega, \\ u = 0, & x \in \partial\Omega. \end{cases} \tag{4}$$

The weak solution of (4) corresponding to the principle of least action is

$$\begin{cases} \text{Find } \ u \in H_0(\Omega), \quad s.t. \\ I(u) = \min\limits_{v \in H_0(\Omega)} I(v), \end{cases} \tag{5}$$

where $H_0(\Omega)$ is the set of admissible functions. Using variational method, we can prove that

$$I(v) = \int_\Omega \left(\frac{1}{2} |\nabla v(x)|^2 - f(x)v(x) \right) dx. \tag{6}$$

As a matter of fact, if we assume that v is an n-dimensional function, then

$$\delta I(v) = \int_\Omega \left(\frac{\partial v}{\partial x_1} \cdot \delta \frac{\partial v}{\partial x_1} + ... + \frac{\partial v}{\partial x_n} \cdot \delta \frac{\partial v}{\partial x_n} - f(x)\delta v(x) \right) dx. \tag{7}$$

Using the integration by parts method for several integrals,

$$\int_\Omega \frac{\partial v}{\partial x_i} \cdot \delta \frac{\partial v}{\partial x_i} dx = \frac{\partial v}{\partial x_i} \cdot \delta v \Big|_{\partial\Omega} - \int_\Omega \delta v \cdot \frac{\partial^2 v}{\partial x_i^2} dx = -\int_\Omega \delta v \cdot \frac{\partial^2 v}{\partial x_i^2}. \qquad 1 \le i \le n \tag{8}$$

Substituting (8) into (7) we can obtain the following equation.

$$\delta I(v) = -\int_\Omega \delta v(\Delta v + f(x))dx = 0 \tag{9}$$

Because of the strongly convexity of the functional I in (6), the solution of (5) is unique. Hence the solution of (5) is identical to (4).

3 Deep Ritz Method with neural networks

Using the Ritz Method mentioned above, it is natural to think of solving partial differential equations with deep neural networks. Considering the partial differential equation

$$\Delta u(x) + f(x) = 0, \qquad x \in \Omega. \tag{10}$$

According to Ritz Method, what we need to do is to find the minimum of I, which is

$$\min_{u \in H} I(u), \tag{11}$$

where

$$I(u) = \int_\Omega \left(\frac{1}{2} |\nabla u(x)|^2 - f(x)u(x) \right) dx, \tag{12}$$

and H is the set of admissible function. Our main idea is to find a solution approximate to the real solution in favor of multi-layer neural networks' approximation property. And we will use the stochastic gradient descent algorithm to solve the optimization problem (11).

3.1 Building Trail Function

We use a nonlinear transformation $x \to u_\theta(x)$ defined by deep neural networks to approximate function $u(x)$. Here θ denotes network parameters in our model. Similar to ResNet structure, we use several blocks to construct our networks, where each block consists of two linear transformations, two activation functions and one residual connection. The i-th block can be expressed as

$$s_i = \phi(W_{i2} \cdot \phi(W_{i1} \cdot s_{i_1} + b_{i1}) + b_{i2}) + s_{i-1}, \tag{13}$$

where s_i is the output of the i-th layer, $W_{i1}, W_{i2} \in R^{m \times m}, b_{i1}, b_{i2} \in R^m$ and ϕ is the activation function.

Because our partial differential equation involves the Laplacian transform, we hope that the second derivative of the function $u(x)$ is not a constant. To ensure the smoothness of function u_θ, we apply the non-linear function (14) as the activation function, instead of ReLU (rectified linear unit) function.

$$\phi(x) = \max\{x^3, 0\}. \tag{14}$$

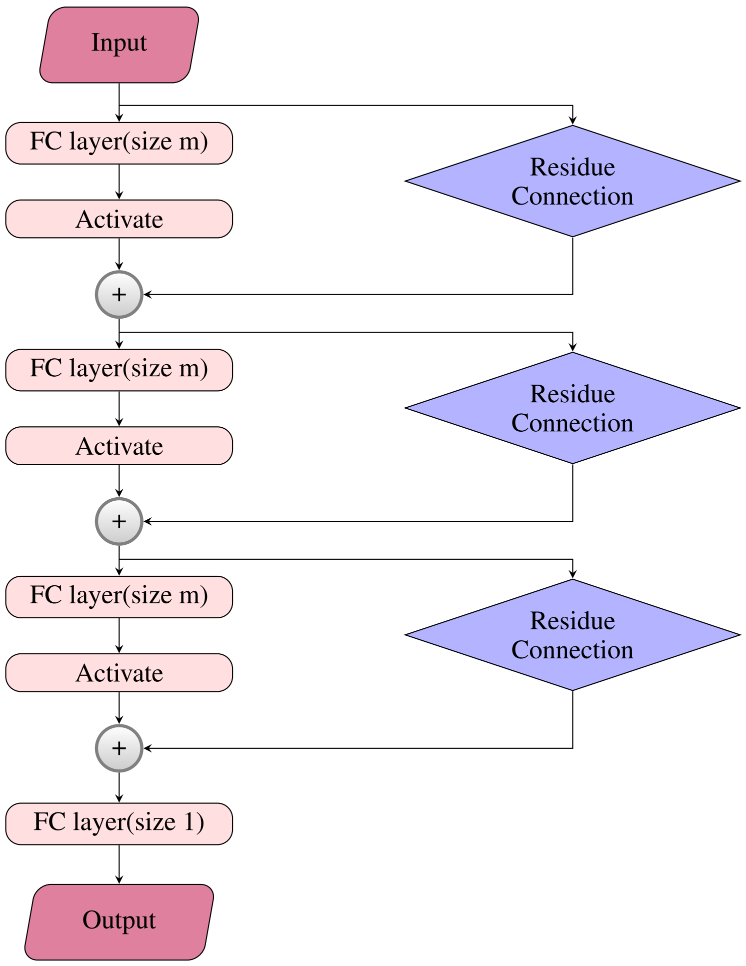
The residual connection in [6] helps to avoid gradient vanishing problems. After several blocks, we adopt a linear transform to the final result. The architecture of residual connection can be viewed in 1. And the whole network can be expressed as

$$u_\theta(x) = a \cdot f_n(x) \circ ... \circ f_1(x) + b, \tag{15}$$

where $f_i(x)$ is the i-th block and $a \in R^m, b \in R$. Note that the input vector x is not necessarily m-dimensional. In order to handle this mismatch, we can pad x with a vector of zeros. In our model, we always assume $d < m$.

After building our trail functions, the rest of our work is to minimize the $I(u)$ in (12)

Figure 1: Network Structure of Deep Ritz Method



3.2 Euler Numerical Integration Method

The first problem we need to deal with is to calculate the integral in (12) . For simplicity, we define:

$$g(x, \theta) = |\nabla u(x)|^2 - f(x)u(x), \tag{16}$$

then $I(u)$ can be expressed as

$$I(u) = \int_{\Omega} g(x, \theta) dx. \tag{17}$$

Obviously, it's impossible to calculate this integral directly. We use Euler integration method to approximate the integral.

$$I(u) = L(x, \theta) = \frac{1}{N} \sum_{i=1}^{N} g(x_i, \theta), \tag{18}$$

where x_i is taken from the uniform grid on the domain with steps 0.001.

3.3 The Stochastic Gradient Descent Algorithm

During the training process of neural networks, stochastic gradient descent (SGD) is a commonly used method for optimization. In this problem, we also choose stochastic gradient descent method to minimize $I(u)$. This optimization process can be expressed as:

$$\theta^{k+1} = \theta^k - \eta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} g(x_{i,k}, \theta^k), \tag{19}$$

where $\{x_{i,k}\}$ is the randomly selected from uniform grid points. In our settings, we use stochastic gradient descent algorithm with mini-batch and Adam [7] optimizer to optimize.

4 Our Improvements

Along with the footsteps of professor E, we have achieved very good results. Besides, to make the deep Ritz method more powerful and more efficient, we have made several improvements, including the self-adaptive sampling method and actor-critic self-adaptive sampling method. The former method has already been proved to be better in our numerical experiments. And as for the other method, we have not fully implemented, due to the limit of our computing capability.

4.1 Self-Adaptive Sampling

From our numerical experiments of the original deep Ritz method, we have observed one severe problem, that is, on the boundary, there are always some places where boundary conditions are fitted well and some places are not. If we choose a small optimization step, then no matter what the coefficient of regularization term is, places where boundary conditions are not fitted well cannot get better. And if we choose a large step, then the objective function will blow up. Hence it is necessary to use a technique which can find places where boundary conditions are not fitted well and takes more care of such places. And this gives rise to this method.

In order to find out the tougher place and enhance the regularization, we use a self-adaptive sampling method here. This idea is similar to adaptive finite element method in chapter 9 of [2], which refines the grid adaptively according to the difficulty on each grid. (this difficulty is described by the value of f given the PDE $\Delta u = f$.)

Here we adopt the self adaptive method on the regularization terms, which are the boundary constraints. Suppose the neural network function is $f(x) = \mathrm{NN}(x, \theta_k)$, then the following adaptive method will be used to sample:

- Take the Poisson equation (1) as an example. For any $x \in \partial\Omega$, we sample points according to density function $|f(x)|$ (we need to normalize this density function).

According to this, more points will be sampled at places where the neural network does not fit well, and less points will be sampled at places where the neural network fits well.

4.2 Actor-Critic Self-Adaptive Sampling

From the numerical experiments, we find out that the deep Ritz method with self-adaptive sampling method works better than the original method. However, there is still one problem involved in. That is, we actually do not know which distribution of sampling is the most suitable, and which one can achieve the best regularization effect. In the discussion of the last subsection, we choose the distribution to be proportional to the error between 0 and $f(x)$ on the boundary. This choice seems plausible, but we cannot make sure that this choice is perfect. Hence we introduce the following actor-critic way to sample.

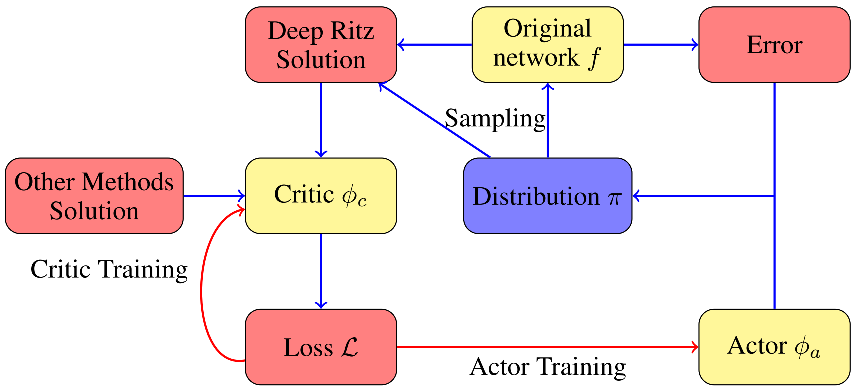
Actor-critic [8] is a famous method in reinforcement learning. The core idea is similar to generative adversarial networks (GAN) [4], both of which design two networks, an actor (generative) network and a critic (discriminator) network. The task of the actor is to generate a distribution or a policy, and the task of the critic is to analyze the difference between the generated distribution and the real distribution.

In our settings, we use the actor to generate a distribution which can be used for sampling points on boundary, as we discussed above. Some assumptions need to be clarified first. We assume the density function of the distribution is a function of the error (20) (In the last subsection, the density function is proportional to the error).

$$\pi(x) = \phi(err(x), \theta_a), \qquad \forall x \in \partial\Omega \tag{20}$$

where $err(x)$ is the error function at x, θ_a are parameters of the network, and $\pi(x)$ is the desirable sampling distribution. Besides, we also assume that the function ϕ only depends on the domain Ω and has nothing to do with the partial differential equation. Hence the actor is a network which takes

Figure 2: Architecture of Algorithm



the approximate error as input and output the sampling distribution. And in (20) it is the function ϕ with network parameters θ_a.

As for critic network in our settings, distinguished from [4] [8], we use the critic network to test the effectiveness of the sampling distribution generated by the actor. This network $\psi(u_{approx}, u_{real}, \theta_c)$ will give an effective measure of error between the approximate solution and the real solution. We assume that the network parameters only depend on Ω as well, like the actor network.

As for the training process, based on the assumption we made before, given the domain Ω, we will first choose some rather simple functions. Then we initialize solutions for their corresponding Poisson equations (do not need to be accurate) and apply deep Ritz method to solve through a few iterations. Besides, we also adopt some numerical methods, such as finite difference method, finite volume method and finite element method. And compare the solutions of these numerical methods and deep Ritz method by our critic network to obtain the loss. Then we can train the actor network to minimize the loss and train the critic network to maximize the loss. The whole process can be viewed in algorithm 1 and figure 2

Algorithm 1 Actor-critic Self-adaptive Training

```
1: **Input** Domain $\Omega$, $m, t, \theta_c, \theta_a$.
2: Generate some rather simple functions $f_1, f_2, \cdots, f_m$ defined on $\Omega$.
3: Solve these partial differential equations with finite difference method or finite volume
   method or finite element method with different grid sizes. And obtain approximate solutions
   $v_1^j, \cdots, v_m^j, 1 \leq j \leq t$.
4: For each $1 \leq i \leq m$, find $t$ initial solutions $u_i^1, \cdots, u_i^t$ randomly.
5: Use the current actor network parameter $\theta_a$ to generate sampling distributions for $u_i^1, \cdots, u_i^t$.
6: Train the original network with these distribution to obtain $u_i'^1, \cdots, u_i'^t$.
7: Put $u_i'^j$ and $v_i^j$ into the critic network with parameter $\theta_c$, and obtain the loss $\mathcal{L}_i^j$.
8: Train $\theta_a$ to minimize the sum of loss $\mathcal{L}_i^j$.
9: Train $\theta_c$ to maximize the sum of loss $\mathcal{L}_i^j$.
10: **Output:** Network parameters $\theta_a, \theta_c$.
```

5 Numerical Results

5.1 The Poisson Equation

Considering the Poisson equation:

$$\begin{cases} \Delta u = 1, & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \end{cases} \tag{21}$$

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

Table 1: Relative Loss of Different Layers with Poisson Equation

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

The exact solution to this problem is

$$u = \frac{1}{4}(x^2 + y^2 - 1). \tag{22}$$

As described above, we use three blocks (six fully connected layers) and a final linear transform with $m = 10$ to build our networks. There is a total of 671 parameters in our model. Considering the boundary condition, we need to make some modifications to our model. We have decided to use a penalty method and the modified function is:

$$I(u) = \int_{\Omega} \left(\frac{1}{2}|\nabla u(x)|^2 - u(x) \right) dx + \beta \int_{\partial\Omega} u(x)^2 dx, \tag{23}$$

where $\beta \int_{\partial\Omega} u(x)^2 dx$ is the penalty item. We choose $\beta = 500$.

After 400 iterations, the relative loss of our model is reduced to 1.3%. Our training results of Deep Ritz Method is shown in Figure 3

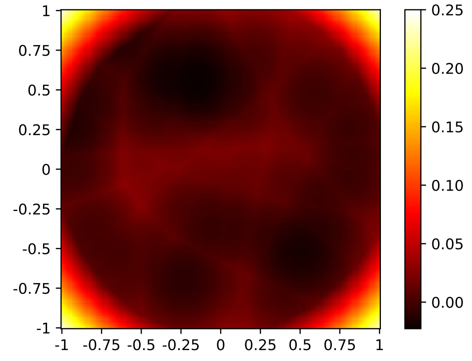


Figure 3: The results of Poisson Equation

As is shown from the figure, the dark color indicates that the absolute error between the numerical solution and the real solution is small, while the light color indicates that the absolute error between the numerical solution and the real solution is relatively large. Deep Ritz method performs well with this problem after 400 iterations.

After testing models of different layers, we find that the 3-layers model with 671 parameters performes best on this problem. Our numerical results of relative error computed by l_2-norm is shown in Table 1. Too many parameters may lead to difficulties of training, limiting the performance of the networks.

5.2 Numerical Experiments for Self-adaptive Sampling Method

We have implemented self-adaptive sampling method, which is described in Section 4.1. The results are listed in Table 2

The previous numerical experiments are carried on a computer with two Intel Core i7 CPUs (3.1 GHz). The CPU time is for experiments whose number of iterations is 10. From this results we obtain that though self-adaptive sampling method takes more time than the original method (most of the CPU time is used on calculating in the sampling process), this method provides a more accurate result.

Table 2: Comparison of Self-adaptive Sampling Method and Original Method

	Accuracy	CPU Time
Deep Ritz Method	1.3%	8.41s
Self-adaptive Sampling Method	0.39%	18.63

5.3 Other Poisson Equation

Considering the following Poisson equation:

$$\begin{cases} \Delta u = 0, & x \in \Omega, \\ u = xy, & x \in \partial\Omega, \end{cases} \tag{24}$$

where $\Omega = \{(x,y)|0 < x, y < 1\}$.

The exact solution to this problem is

$$u = xy. \tag{25}$$

Compared with the equation in (23), there are some differences in the penalty items. This time we let the regularization coefficient increase with the number of iterations, which can lead to a better training result. The changing process of regularization coefficient can be expressed as:

$$\beta_N = c^N \beta_0, \tag{26}$$

where N is the number of iteration and $c = 1.01$, $\beta_0 = 1$. After 5000 iterations, the relative loss of our model gets to 1.4%. Our training result is shown in Figure 4.

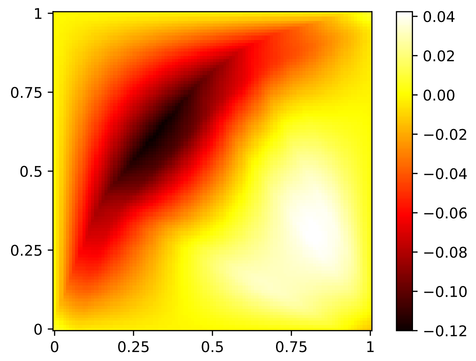


Figure 4: The Results of Poisson Equation

Same as before, the color depth in the graph reflects the absolute error from the exact value.

6 Conclusion and Discussions

In this project report, we demonstrate that deep Ritz method is simple but effective when solving a certain kind of PDEs. What's more, we propose the self-adaptive sampling method and the actor-critic self-adaptive sampling method, which lead to great improvement on the results.

While having achieved fairish results, there are still some drawbacks existing in our models, such as the deficiency of robustness in our model and too much work for tuning parameters. The ameliorations to these drawbacks may be left for future works.

Acknowledgement

The authors would like thank Bing Yu for his discussion and interpretation to his paper [13].

References

[1] Christian Beck, Arnulf Jentzen, et al. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *arXiv preprint arXiv:1709.05963*, 2017.

[2] Susanne Brenner and Ridgway Scott. *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media, 2007.

[3] L.C. Evans and American Mathematical Society. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 1998.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[5] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *arXiv preprint arXiv:1707.02568*, 2017.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[10] John C Strikwerda. *Finite difference schemes and partial differential equations*, volume 88. Siam, 2004.

[11] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[12] E Weinan, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[13] Bing Yu et al. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *arXiv preprint arXiv:1710.00211*, 2017.