# Literature Search of the Deep Ritz Method

Christopher Glasgow, Kristine Martinez, and Emalina Huerta

May 2023

Chris: Deeper analysis of the Deep Ritz Method and how the sources we chose connected to and compared with the main paper.

Kristine: Reading Densely Connected Convolutional Networks to figure out the pros and cons that are linked to the main paper.

Emalina: Reading Solving High-Dimensional Partial Differential Equations Using Deep Learning to figure out the pros and cons that are linked to the main paper.

# 1 Overview

We will be conducting a literature search on two of the references from *The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems*. Our literature search will start with the reference paper, *Solving High-Dimensional Partial Differential Equations Using Deep Learning* We will begin with a summary of this paper, and then discuss the strength and shortcomings as well. We will then look into how this reference paper was used inside of our main paper, and compare to see if any advancements were made from the reference paper when used in the main paper. After stating all our observations for that reference paper, we will proceed to our second paper, *Densely Connected Convolutional Networks*, and perform the same analysis. After both reference papers have been discussed and compared to the main paper, we will go into benchmark tests of the methods and algorithms in each paper.

# 2 Solving High-Dimensional Partial Differential Equations Using Deep Learning

## 2.1 Summary

The paper introduces deep learning approach for solving semi-linear parabolic Partial Differential Equations (PDE). The approach to satisfy the Backward Scholes Differential Equations(BSDE) to deep Neural Networks (NN). The proposed algorithm uses Black Scholes Equation, Hamiltonian -Jacobian -Bellman(HJB) and Allen Cahn Equations. "BSDE is essential methodology introduces deep BSDE method". Deep learning is successful in Artificial Intelligence (AI) and adapts to the problem of solving higher dimensional PDE. This reference provides a general class of PDE that can be solved using the proposed approach and discuss previous methods for solving higher dimensional problems.

## 2.2 Strengths

The paper introduces high dimensional PDEs using backward stochastic differential equations. It takes the gradient of the Hessian function of u with respect to x. The solution is to satisfy BSDE NN. Stochastic gradient descent-type (SGD) is used to approximate the parameters. This has created a huge advancement in the mathematical physics field. Methodology of the paper extends the power of deep learning in NN to high dimensional PDEs including Black- Scholes equations and the HJB. The paper uses finance, physics and engineering where high dimensional PDEs are ambiguous. Since this paper there have been new advancements in high dimensional PDEs.

## 2.3 Shortcomings

In economics we consider many different interacting components. This article is looking at multiple different components at the same time instead of using one model as a representative. Relying on ad hoc assumptions can also come with some short comings because it may only

be used for one focus at a time. Ad hoc is used for their approximations in their process. The methodology is fairly general in the process and incapable of dealing with quantum many body problem due to the difficulty of dealing with Pauli exclusion principle.

## 2.4 Use in Deep Ritz paper

This reference paper is used in the Deep Ritz paper when discussing using deep learning-based models. There is an assumption made, that when performing computer vision or other artificial intelligence tasks, using deep learning would be a powerful tool for assessing these issues when it comes to higher dimensions. This assumption is confirmed by the results of the Deep BSDE method used in *Solving High-Dimensional Partial Differential Equations Using Deep Learning*. Since the Deep Ritz Method is a deep-learning based method, it can then be claimed that it too would be a powerful tool for solving such high dimensional problems.

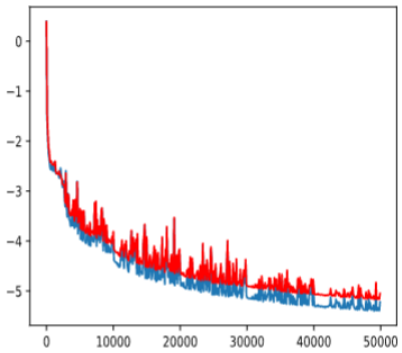
## 2.5 Advancements made from this paper

The Deep Ritz Method was then tested in higher dimensions using the Poisson equation:

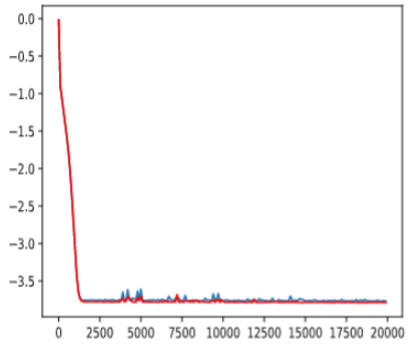
$$-\Delta u = 0, \qquad x \in (0,1)^d$$

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

They compared the dimensions 10 and 100 by setting $d = 10$ and $d = 100$ in the above formula, and then trained each model up to see the relative error for for $u$ and $\partial u$. The figure below shows their results using logarithm scale. Both versions were trained for 50,000 iterations and then the relative $L_2$ error was measured again. In the 10-dimensional case, the error was reduced to 0.4%. In the 100-dimensional case, the error was reduced to 2.2%.




(a) $d = 10$




(b) $d = 100$

Figure 1: Total error and error at the boundary during the training process. The x-axis represents the iteration steps. The blue curves show the relative error of $u$. The red curves show the relative error on the boundary. **(a)** $\ln e$ and $\ln loss_{boundary}$, $d = 10$. **(b)** $\ln e$ and $\ln loss_{boundary}$, $d = 100$.

# 3 Densely Connected Convolutional Networks

## 3.1 Summary

CNN is a type of neural network that is often applied to image processing problem which this paper tackles. The Dense Convolutional Network (DenseNet) introduced is the idea in which the network connects each layer to every other layer in a feed-forward procedure.

Traditional CNN with L layers have L connections while DenseNet has $\frac{L(L+1)}{2}$ direct connections (Huang, 2018, p.2). For each layer the feature maps of all preceding layers are used as inputs and its own feature maps are used as input into all successive layers. A problem that usually happens with Convolutional Networks is that the information about the input tends to vanish as it passes through layers and DenseNet allieviates that problem with its feature-map resuses. As proposed in the paper, DenseNet connects all layers using the same sizes of feature maps directly with each other and preserves the feed-forward procedure as each layer gathers its inputs from all its layers beforehand passing forwards its own feature maps to the next layers (Huang, 2018, p.2). This is shown in Figure 1.

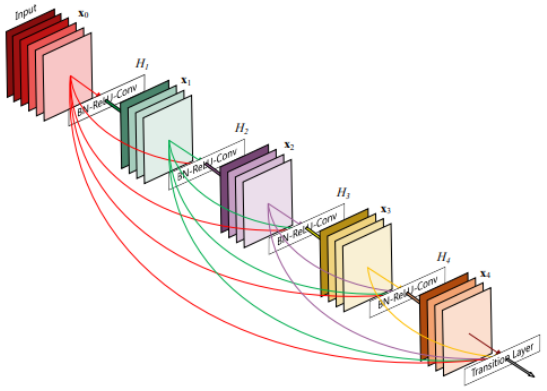



**Figure 1:** A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Features are combined by concatenating them which introduces the $\frac{L(L+1)}{2}$ connections in an L-layer network instead of just L as used in traditional architectures. By concatenating feature maps the variation in the input to the following layers increases and leads to improved efficiency.

As stated in the paper, given a single image $x_0$ passes through a CNN composed of L layers, each of those layers implement a non-linear transformation $H_{l(.)}$ where $l$ indexes the layer. $H_{l(.)}$ is the composite function of operations that include Batch Normalization (BN), ReLU, Pooling, and Convolution (Conv) and the output of the $lth$ layer is denoted as $x_l$. This $lth$ layer receives the feature maps of all preceding layers, $x_0, ..., x_{l-1}$ as the input.

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]) \tag{1}$$

where $[x_0, x_1, ..., x_{l-1}]$ is the concatenation of the feature maps produced by layers $0, .., l-1$. This dense connectivity is why this procedure is called "Dense Convolutional Network" (Huang, 2018, p.3)

The composite function defined $H_{l(.)}$ is comprised of three consecutive operations: BN, ReLU, and a 3x3 Conv. ReLU acts as an activation function ensuring nonlinearity as the data moves through each layer in the network. The pooling layers facilitate down-sampling of a particular feature map making processing much faster by reducing the number of parameters the network needs to process. In figure 2 of the paper, the network is divided into these "dense blocks" and between them consists of the transition layers where convolution and pooling calculations are performed (Huang, 2018, p.3).

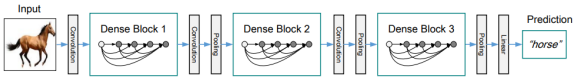



**Figure 2:** A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

The hyper-parameter, $k$, that is used in the paper is the growth rate of the network for which it regulates how much new information each layer contributes to the whole network. This information can be accessed from anywhere within the network and eliminates the need to replicate the information at each layer.

Each layer only produces a hyper-parameter $k$ output feature maps however it usually contains more inputs. A 1x1 conv is introduced as a bottleneck layer before each 3x3 conv to reduce the amount of input feature maps which allows improved computation efficiency (Huang, 2018, p.4).

This paper mentions the use of the following data sets: CIFAR - consists of colored natural images, SVHN - Street View House Numbers consisting of 32x32 colored digit images, and ImageNet - consists of 1.2 million images for training, $50,000$ for validation (Huang, 2018, p.5).

This network is trained using stochastic gradient descent (SGD). The classification results on the data sets CIFAR and SVHN show that the error rates are much lower than that of the error rates of the ResNet architecture that was tested. The results concluded that without bottleneck layers there's a trend in which DenseNets perform better as the parameters increase. Furthermore, DenseNet utilizes parameters more efficiently than ResNet with bottleneck and dimension reduction at the transition layers (conv, pooling). This means that DenseNet architecture is less prone to over-fitting.

The classification results on ImageNet presented in figure 3 of the paper show that DenseNets perform at the same standard with ResNets while using fewer parameters (Huang, 2018, p.6). This paper utilizes ResNets architecture to benchmark results with DenseNets.

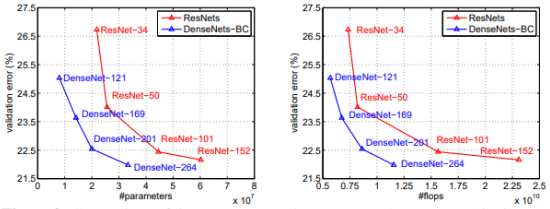



**Figure 3:** Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

DenseNets is similar to ResNets however the inputs to $H_{l(.)}$ are concatenated instead of

summed in ResNets.

## 3.2 Strengths

The advantages of implementing DenseNets are that it tackles the vanishing-gradient problem which refers to the information input disappearing towards the end or beginning of the network, usage of feature map reuses and minimization of parameter usage. DenseNets results conclude improvements in accuracy with increasing number of parameters along with no signs of performance issues or over-fitting (Huang, 2018, p.8). Additionally it requires less computation to achieve excellent performance compared to other network architectures.

## 3.3 Shortcomings

As mentioned in the paper, a "naive implementation of DenseNet may contain memory inefficiencies" (Huang, 2018, p.5). The authors reference their own technical report that helps reduce memory consumption on GPUs. DenseNet could also potentially gain more accuracy by fine-tuning its hyper-parameters since it was optimized in ResNet architecture.

## 3.4 Use in Deep Ritz paper

This reference paper was used in the Deep Ritz paper when trying to build a deep neural network to help solve eigenvalue problems. They used the DenseNet deep neural network used in *Densely Connected Convolutional Networks* as a reference when building the Deep Ritz neural network. The structure of this network can be seen in the figure below.

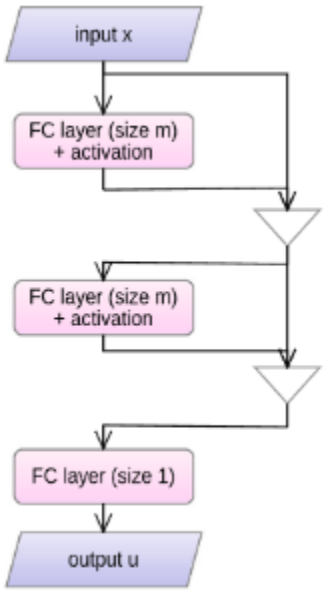



Figure 2: Network structure used for the eigenvalue problem. There are skip connections between every pairwise layer. The triangles denote concatenation operations.

### 3.5 Advancements made from this paper

This deep neural network was built in consideration to the following quantum mechanics eigenvalue problem:

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

with $v$ being the potential function. The principle for smallest eigenvalue is:

$$min \quad \frac{\int_\Omega |\nabla u|^2 \, dx + \int_\Omega v u^2 \, dx}{\int_\Omega u^2 \, dx}$$
$$s.t. \quad u|_{\partial\Omega} = 0,$$

which is called the Rayleigh quotient. To avoid getting the trivial solution of $u = 0$, the following condition is also introduced:

$$\int_\Omega |\nabla u|^2 \, dx = 1.$$

There was two tests done of the eigenvalue problem, using both the infinite potential well, and the harmonic oscillator. When using the infinite potential well, the potential function is

$$v(x) = 0 \qquad if \qquad x \in [0,1]^d$$
$$v(x) = \infty \qquad if \qquad x \notin [0,1]^d$$

which gives us the eigenvalue problem:

$$-\Delta u = Eu, \qquad x \in [0,1]^d$$
$$u(x) = 0, \qquad x \in \partial[0,1]^d.$$

The smallest eigenvalue in the infinite potential well problem is $\lambda_0 = d\pi^2$, and the following table contains the results from the Deep Ritz Method with dimensions of 1, 5, and 10. When

| Dimension $d$ | Exact $\lambda_0$ | Approximate | Error (%) |
|---|---|---|---|
| 1 | 9.87 | 9.85 | 0.20 |
| 5 | 49.35 | 49.29 | 0.11 |
| 10 | 98.70 | 92.35 | 6.43 |

Figure 3: Error of Deep Ritz Method for infinite potential well

using the harmonic oscillator, the potential function in $\mathbb{R}^d$ is:

$$v(x) = |x|^2,$$

but was truncated to being in $[-3,3]^d$ for simplicity. The smallest eigenvalue in the infinite potential well problem is $\lambda_0 = d$, and the following table contains the results from the Deep Ritz Method with dimensions of 1, 5, and 10. We can see that for both of the tests, the error increases substantially as the dimensions increase, meaning that there is still la lot of work to be done there.

| Dimension $d$ | Exact $\lambda_0$ | Approximate | Error (%) |
|---|---|---|---|
| 1 | 1 | 1.0016 | 0.16 |
| 5 | 5 | 5.0814 | 1.6 |
| 10 | 10 | 11.26 | 12.6 |

Figure 4: Error of Deep Ritz Method for the harmonic oscillator

# 4 Benchmark Test Comparisons

As we discussed earlier, in the 100 dimensional case for solving high dimension equations, the Deep Ritz Method had an error rate of 2.2% when working on the Poisson equation. The Deep BSDE Method was not testing on the Poisson equation, but was tested with the 100 dimensional cases for the Black-Scholes Equation, the Hamilton-Jacobi-Bellman Equation, and the Allen-Cahn Equation. The table below shows the results for the relative error of using the Deep BSDE Method on these equations. We can see that, although the different

| Equation | Relative Error |
|---|---|
| Black-Scholes | 0.46% |
| Hamilton-Jacobi-Bellman | 0.17% |
| Allen-Cahn | 0.30% |

Table 1: The following relative error results from using the Deep BSDE Method in the 100-Dimension cases of the Black-Scholes Equation, the Hamilton-Jacobi-Bellman Equation, and the Allen-Cahn Equation.

methods were tested on different equations, the Deep BSDE Method appears to be much more accurate when looking into the 100-dimensional cases. When looking at the Deep Ritz deep-learning base model that was formulated similarly to the Densenet deep-learning model, we remember that the errors in the 10-dimensional eigenvalue problems were 6.43% and 12.6% for both the previously mentioned methods. The Densenet model was not tested on eigenvalue problems, so we again cannot test both methods at doing the same problem, but the Densenet method was tested on CIFAR and SVHN data sets, with error rates of 3.46% and 17.18% respectively. We can see that Densenet has more variance in their results, and averages to a similar error as the 10 dimensions eigenvalue case for the Deep Ritz Method. However, in lower dimensions, the Deep Ritz method does seem to heavily outperform the Densenet Method.

# 5 References

[1] E, Weinan, and Bing Yu. "The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems." Communications in Mathematics and Statistics, vol. 6, no. 1, 2018, pp. 1–12, https://doi.org/10.1007/s40304-018-0127-z.

[2] Han, Jiequn, et al. "Solving High-Dimensional Partial Differential Equations Using Deep Learning." Proceedings of the National Academy of Sciences, vol. 115, no. 34, 2018, pp. 8505–8510, https://doi.org/10.1073/pnas.1718942115.

[3] Huang, Gao, et al. "Densely Connected Convolutional Networks." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, https://doi.org/10.1109/cvpr.2017.243.