Learning Green's Function Efficiently Using Low-Rank Approximations

Kishan Wimalawarne [1] **Taiji Suzuki** [1 2] **Sophie Langer** [3]

Abstract

Learning the Green's function using deep learning models enables to solve different classes of partial differential equations. A practical limitation of using deep learning for the Green's function is the repeated computationally expensive Monte-Carlo integral approximations. We propose to learn the Green's function by low-rank decomposition, which results in a novel architecture to remove redundant computations by separate learning with domain data for evaluation and Monte-Carlo samples for integral approximation. Using experiments we show that the proposed method improves computational time compared to MOD-Net while achieving comparable accuracy compared to both PINNs and MOD-Net.

1. Introduction

The Green's function is a well-known method to solve and analyze Partial Differential Equations (PDE) (Evans, 2010; Bebendorf & Hackbusch, 2003). Recently, deep learning models have been applied to learn the Green's function enabling to obtain solutions for both linear and nonlinear PDEs (Luo et al., 2022) as well as to learn PDEs in irregular domains (Teng et al., 2022). Luo et al. (2022) further demonstrated that deep learning models can parameterize a class of PDEs compared to learning a specific PDE.

The MOD-Net (Luo et al., 2022) uses the Green's function approximation by using a neural network and Monte-Carlo integration to parameterize solutions for PDEs. A limitation of the MOD-Net is that the approximation of the Green's function by Monte-Carlo integration leads to high computational costs due to their repeated evaluations for each domain element. In this research, we propose to extend MOD-Net with low-rank decomposition of the Green's function. The proposed model results in a two network architecture, where one network learns on the domain elements to be evaluated and other network learns over all the Monte-Carlo samples, hence, avoid redundant repeated computations of Green's function at each domain element. Using experiments with the Poisson 2D equation and the linear reaction-diffusion equation we show that our proposed method is computationally feasible compared to MOD-Net, while achieving comparable accuracy compared to PINNs (Raissi et al., 2017) and MOD-Net. Additionally, we show that our proposed method has the ability to interpolate within the solution space as a neural operator similar to MOD-Net.

2. Learning Green's Function by MOD-Net

Let us consider a domain $\Omega \subset \mathbb{R}^d$, a differential operator $\mathcal{L}$, a source function $g(\cdot)$ and a boundary condition $\phi(\cdot)$, then a partial differential equation can be represented as,

$$\begin{aligned} \mathcal{L}[u](x) &= g(x), \qquad x \in \Omega \\ u(x) &= \phi(x), \qquad x \in \partial\Omega. \end{aligned}$$

For a linear PDE with the Dirichlet boundary condition $\phi(\cdot) = 0$, we can find a Green's function $G : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ for a fixed $x' \in \Omega$ as follows:

$$\begin{aligned} \mathcal{L}[G](x) &= \delta(x - x'), \qquad x \in \Omega \\ G(x, x') &= 0, \qquad x \in \partial\Omega, \end{aligned}$$

which leads to a solution function $u(\cdot)$ as

$$u(x) = \int_\Omega G(x, x') g(x') dx'. \tag{1}$$

Recently developed MOD-Net (Luo et al., 2022) proposes to learn the Green's function by using a neural network. It uses a neural network $G_{\theta_1}(x, x')$ with parameters denoted by θ_1 to replace the operator $G(x, x')$ of (1). Given a set S_Ω consisting of random samples from Ω, the Monte-Carlo approximation of the Green's function integral (1) results in the following:

$$u_{\theta_1}(x; g) = \frac{|\Omega|}{|S_\Omega|} \sum_{x' \in S_\Omega} G_{\theta_1}(x, x') g(x'). \tag{2}$$

[*]Equal contribution [1]Department of Mathematical Informatics, The University of Tokyo, Tokyo, Japan [2]Center for Advanced Intelligence Project (AIP), RIKEN, Tokyo, Japan [3]Faculty of Electrical Engineering, Mathematics, and Computer Science, University of Twente, Enschede, The Netherlands. Correspondence to: Kishan Wimalawarne <kishanwn@gmail.com>.

Accepted after peer-review at the 1st workshop on Synergy of Scientific and Machine Learning Modeling, SynS & ML ICML, Honolulu, Hawaii, USA. July, 2023. Copyright 2023 by the author(s).

MOD-Net (Luo et al., 2022) also proposes of a nonlinear extension of the above as

$$u_{\theta_1,\theta_2}(x;g) = F_{\theta_2}\left(\frac{|\Omega|}{|S_\Omega|}\sum_{x'\in S_\Omega} G_{\theta_1}(x,x')g(x')\right), \quad (3)$$

where F_{θ_2} is a neural network (θ_2 representing parameters).

It has been shown (Luo et al., 2022) that Green's function can learn as an operator for varying $g(\cdot)$ to parameterize a class of PDE. Following (Luo et al., 2022), let us consider K different parametarizations of a PDE class by specifying $g^k(\cdot)$ for $k = 1, \cdots, K$. Let $S^{\Omega,k}$ and $S^{\partial\Omega,k}$ are domain elements from the interior and boundary, respectively, for each $k = 1, \ldots, K$. Then the objective function for MOD-Net for the formulation in (2) is given as

$$R_S = \frac{1}{K}\sum_{k\in[K]}\left(\lambda_1\frac{1}{|S^{\Omega,k}|}\sum_{x\in S^{\Omega,k}}\|\mathcal{L}[u_{\theta_1}(x;g^k)](x) - g^k(x)\|_2^2 + \lambda_2\frac{1}{|S^{\partial\Omega,k}|}\sum_{x\in S^{\partial\Omega,k}}\|u_{\theta_1}(x;g^k)\|_2^2\right), \quad (4)$$

where λ_1 and λ_2 are regularization parameters. Notice that when $K = 1$ the above objective function learns a specific instance of a PDE.

A major limitation with MOD-Net is the computational bottleneck due to the Monte-Carlo approximation of the integrals (2) and (3). The repeated summation by elements of S_Ω with respect to each domain element in (4) can become both computationally costly and redundant.

3. Proposed Method

We propose to extend MOD-Net with low-rank presentation of the Green's function to remove redundant computations and improve computational feasibility.

First, we put forward the following approximation for low-rank decomposition of the Green's function from Bebendorf & Hackbusch (2003). For any $0 < \epsilon < 1$ sufficiently small and $R \geq c^d\lceil\log(\epsilon^{-1})\rceil^d + \lceil\log(\epsilon^{-1})\rceil$, and $D_1, D_2 \subset \Omega$, Bebendorf & Hackbusch (2003) have given decomposition with functions $v_i(\cdot)$ and $w_i(\cdot)$, $i = 1, \ldots, R$ as

$$G_R(x,y) = \sum_{i=1}^{R} v_i(x)w_i(y), \quad x \in D_1,\ y \in D_2, \quad (5)$$

such that

$$\|G(x,\cdot) - G_R(x,\cdot)\|_{L^2(D_1)} \leq \epsilon\|G(x,\cdot)\|_{L^2(\hat{D}_1)}, \quad (6)$$

where $\hat{D}_1 \subset \Omega$ a set slightly large than D_1.

We now apply the above low-rank decomposition (5) of the Green's function to (2). We propose to learn functions $u_i(x)$ and $v_i(x)$, $i = 1, \ldots, R$ using two neural networks $F_{\gamma_1} : \mathbb{R}^d \to \mathbb{R}^R$ and $H_{\gamma_2} : \mathbb{R}^d \to \mathbb{R}^R$ where γ_1 and γ_2 represent parameters. In practice, R can be problem and data dependent, hence, may require to be considered as a hyperparameter. Further, we assume that $D_1 = D_2 = \hat{D}_1 = \Omega$ for (5) and (6) and there exist some neural network that can learn a low-rank representation.

Next, we apply the learning of low-rank decomposition to (2) which leads to the following expansion

$$\begin{aligned} u_{\gamma_1,\gamma_2}(x;g) &= \frac{|\Omega|}{|S_\Omega|}\sum_{y\in S_\Omega} G(x,y)g(y) \\ &\approx \frac{|\Omega|}{|S_\Omega|}\sum_{y\in S_\Omega}\sum_{i=1}^{R} F_{\gamma_1}(x)_i H_{\gamma_2}(y)_i g(y) \\ &= \frac{|\Omega|}{|S_\Omega|}\sum_{i=1}^{R} F_{\gamma_1}(x)_i \sum_{y\in S_\Omega} H_{\gamma_2}(y)_i g(y) \\ &= \frac{|\Omega|}{|S_\Omega|} F_{\gamma_1}(x)^\top\left[\sum_{y\in S_\Omega} H_{\gamma_2}(y)g(y)\right]. \end{aligned} \quad (7)$$

The last step of (7) shows the separation of the learning with Monte-Carlo samples from the set S_Ω by $H_{\gamma_2}(\cdot)$ and learning with each x by $F_{\gamma_1}(\cdot)$. This separated learning helps to avoid the computationally costly repeated evaluations in (2) and (3) since we only need to compute the network $H_{\gamma_2}(\cdot)$ once for all input x at each learning iteration.

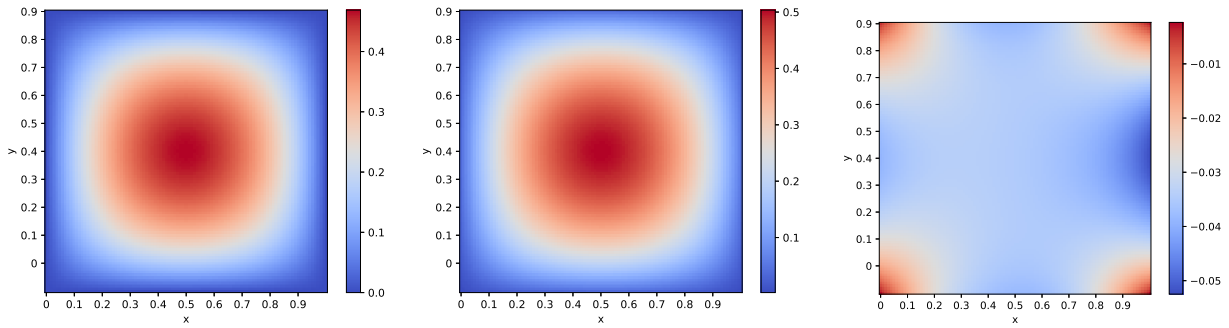
Using the construction in (7) we propose two neural network architectures to improve over (2) and (3). For simplicity, we omit the factor $|\Omega|/|S_\Omega|$ due to its redundancy in the learning process. Further, we assume that elements S_Ω are sampled once and fixed during the learning and prediction processes.

We put forward *DecGreenNet* as a direct construction from (7) with $|S_\Omega| = P$ as

$$u_{\gamma_1,\gamma_2}(x;g) = F_{\gamma_1}(x)^\top\sum_{i=1}^{P} H_{\gamma_2}(y_i)g(y_i). \quad (8)$$

Next, we construct a nonlinear extension of MOD-Net (3) with the low-rank decomposition following (7). Here we remove the summation on the right and arrange its elements as a concatenation. By introducing an additional neural network $O_{\gamma_3} : \mathbb{R}^P \to \mathbb{R}$, we propose *DecGreenNet-NL* as

$$u_{\gamma_1,\gamma_2,\gamma_3}(x;g) = O_{\gamma_3}\Big(F_{\gamma_1}(x)^\top \mathrm{concat}\big[H_{\gamma_2}(y_1)g(y_1), H_{\gamma_2}(y_2)g(y_2), \ldots, H_{\gamma_2}(y_P)g(y_P)\big]\Big), \quad (9)$$



(a) Exact solution (b) Predicted solution (c) Error

Figure 1: Interpolation results for Poisson 2D problem by DecGreenNet-NL for the parameter $a = 15$ (a) Exact solution, (b) Predicted solution by DecGreenNet-NL and (c) Error

Method	Network structure	P	Loss	Time(sec)
PINNs	[2, 64, 64, 1]	-	3.5e-4	80.14
DecGreenNet	F_{γ_1} = [2,512, 512, 512, 512, 50],H_{γ_2} = [2, 16, 16, 16, 1]	100	2.86e-4	117.95
DecGreenNet-NL	F_{γ_1} = [2,64,64,64,64,64,50],H_{γ_2} = [2,64,64,64,50],O_{γ_3} = [100,1]	100	1.5e-3	550.45
MOD-Net	[4, 128, 128, 128, 128, 1]	10	1.05e-3	721330

Table 1: Network structures, number of random samples, test loss, and computational times for a single instance learning of Poisson 2D equation for $a = 15$ using PINNs, DecGreenNet, DecGreenNet-NL, and MOD-Net

where $\mathrm{concat}(\cdots)$ is an operation to concatenation of input vectors. Note that when O_{γ_3} is replaces by a vector of ones ($O_{\gamma_3} = [1, 1, \ldots, 1] \in \mathbb{R}^P$), (9) becomes equivalent to (8).

Optimal neural architectures (layers and hidden units) of F_{γ_1}, H_{γ_1}, and O_{γ_3} need to be discovered by hyperparameter tuning. Additionally, R and P should be considered as hyperparameters. Furthermore, the activation functions for neural networks require higher-order differential capacity in relation to the differential operators in the PDE. In general, we use the activation function $\mathrm{ReLUK}(x) := \max\{0, x\}^K$ where $K \in \mathbb{N}_+$.

4. Experiments

We experimented with PDEs used in (Luo et al., 2022) and (Teng et al., 2022) to evaluate our proposed models.

4.1. Experimental Setup

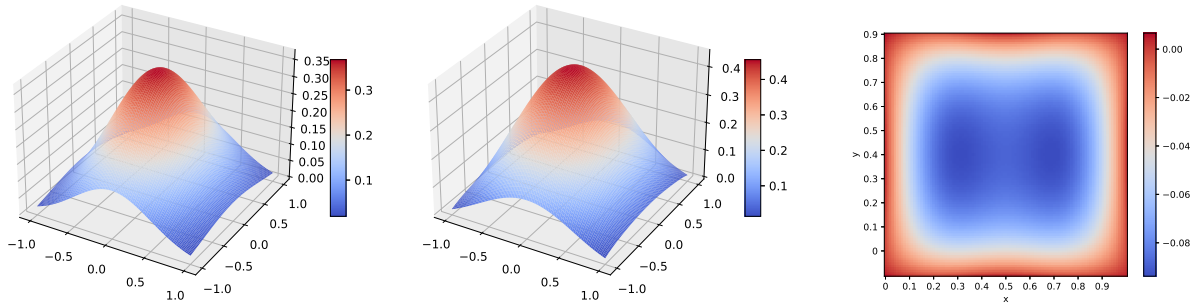
In all our experiments we set $\lambda_1 = \lambda_2 = 1$ in the objective function (4) for all models. For both DecGreenNet and DecGreenNet-NL selected the layers and hidden units of each neural network by hyperparameter tuning. We selected layers from $1, \ldots, 6$ and hidden units from 2^h $h = 3, \ldots, 6$ for both $F_{\gamma_1}(\cdot)$ and $H_{\gamma_2}(\cdot)$ of (8) and (9). For $O_{\gamma_3}(\cdot)$ we selected from hidden layers $0, 1, 2$ and hidden units $4, 8, 16$. We represent the network structure of network of models by notation $[in, h, \ldots, h, out]$, in, out, and h represent dimensions of input, output, and hidden layers, respectively. We experimented with $R \in \{5, 50, 100\}$. We used PINNs as a baseline method by performing hyperparameter tuning for layers and hidden units varying from $1, \ldots, 5$ and 2^h $h = 3, \ldots, 6$, respectively. We also used MOD-Net as a baseline method, however, we only used 10 random samples to approximate the Green's function due to high computational cost. We used the activation function $\mathrm{ReLU3}(\cdot)$ for all models. We used the Pytorch environment with Adam optimization method with learning rate of 0.001 and weight decay set to zero. All experiments were conducted on NVIDIA A100 GPUs with CUDA 11.6 on a Linux 5.4.0 environment. We provide the code at `https://github.com/kishanwn/DecGreenNet`.

4.2. Poisson 2D Equation

As our first experiment, we used the Poisson 2D problem with multiple paramterizations under the same setting used in (Luo et al., 2022). The Poisson 2D problem for the the domain of $\Omega = [0, 1]^2$ is specified as

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

where $g(x, y) = -a(x^2 - x + y^2 - y)$. The analytical solution of the above problem is $u(x, y) = \frac{a}{2}x(x-1)y(y-1)$. In (Luo et al., 2022), multiple parameterizations of the Poisson equation is specified by setting different values for a. Following a similar setting as in (Luo et al., 2022), we



(a) Exact solution (b) Predicted solution (c) Error

Figure 2: Comparison between the exact solution and predicted solution by DecGreenNet of the reaction-diffusion equation (a) Exact solution, (b) Predicted solution by DecGreenNet and (c) Error.

Method	Network structure	P	Loss	Time (sec)
PINNs	[2,128, 128, 128, 128, 128,1]	-	2.6e-4	99.48
DecGreenNet	F_{γ_1} = [2,512, 512, 512, 512, 512,50], H_{γ_2} = [2,32,32,32,50]	300	7.76e-5	355.25
DecGreenNet-NL	F_{γ_1} = [2, 256, 256, 256, 256, 256, 50], H_{γ_2} = [2, 64, 64, 64, 64, 50] O_{γ_3} = [300,1]	300	2.70e-4	1075.24
MOD-Net	[4,128,128,128,1]	10	1.26e-4	23696

Table 2: Network structures, number of random samples, test loss, and computational times for the linear reaction-diffusion equation using PINNs, DecGreenNet, DecGreenNet-NL, and MOD-Net

used $a := a_k = 10k$ where $k = 1, \cdots, 10$ which lead to $g(x, y) := g^k(x, y) = -a_k(x^2 - x + y^2 - y),\ k = 1 \ldots, 10$.

We found that DecGreenNet-NL with $F_{\gamma_1} = [2, 256, 256, 256, 256, 256, 50]$, $H_{\gamma_2} = [2, 64, 64, 64, 64, 50]$, and $O_{\gamma_3} = [100, 1]$ provided the best solution. From the learned model, we interpolate the solution for the Poisson 2D equation with $a = 15$. The low error of the interpolated solution in Figure 1 shows the operator learning capability of our method, hence, the ability to learn parameterization for a class of PDE.

As our second experiment with Poisson 2D, we conducted experiments to evaluate solutions for a single instance of the Poisson 2D equation with $a = 15$ ($K = 1$ in (4)). Table 1 shows the details on learning with PINNs, MOD-Net, and single instance learning by DecGreenNet and DecGreenNet-NL. Both our proposed methods obtained lower values for test loss compared to PINNs and MOD-Net, in addition to the significantly small time compared to MOD-Net.

4.3. Reaction-Diffusion Equation

We experimented with the linear reaction-diffusion equation (Teng et al., 2022) in the domain $\Omega \in [-1, 1]^2$ specified as

$$\mathcal{L}(u) = -\nabla \cdot ((1 + 2x^2)\nabla) + (1 + y^2)u \qquad (10)$$

The exact solution is $u(x, y) = e^{-(x^2+2y^2+1)}$. From both Table 2 and Figure 2, we observe that DecGreenNet obtains a good accuracy for learning the linear reaction-diffusion function with moderate computational time.

5. Broader Impact

We believe that the computational advantages and operator learning capability of our proposed method would be conducive to solving important problems in science. A limitation of our method is the considerable amount of hyperparameters that needed to be tuned to find the optimal neural architectures. We do not know any negative impacts from our proposed methods.

6. Acknowledgement

TS was partially supported by JSPS KAKENHI (20H00576) and JST CREST. KW was partially supported by JST CREST.

7. Conclusion and Future Work

We provide a computationally feasible low-rank model to learning PDEs with Green's function. Theoretical analysis such as convergence bounds for our model is open for future work. Further extensions of our model to solve high-dimensional PDEs is another future direction.

References

Bebendorf, M. and Hackbusch, W. Existence of $\mathcal{H}$-matrix approximants to the inverse fe-matrix of elliptic operators with L_∞-coefficients. *Numer. Math.*, 2003.

Evans, L. C. *Partial differential equations*. American Mathematical Society, 2010.

Luo, Z. L., Yaoyu, T. Z., Weinan, E., Xu, J., Zhi-Qin, and Zheng, M. Mod-net: A machine learning approach via model-operator-data network for solving pdes. *CiCP*, 2022.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017.

Teng, Y., Zhang, X., Wang, Z., and Ju, L. Learning green's functions of linear reaction-diffusion equations with application to fast numerical solver. In *MSML*, 2022.