Appl. Math. Mech. -Engl. Ed., **44**(7), 1151–1174 (2023)

APPLIED MATHEMATICS AND MECHANICS (ENGLISH EDITION)

https://doi.org/10.1007/s10483-023-2992-6

Deep convolutional Ritz method: parametric PDE surrogates without labeled data*

J. N. FUHG[1], A. KARMARKAR[1], T. KADEETHUM[2], H. YOON[2], N. BOUKLAS[1,†]

1. Sibley School of Mechanical and Aerospace Engineering, Cornell University, New York 14853, U. S. A.;

2. Geomechanics Department, Sandia National Laboratories, Albuquerque 87123, U. S. A.

(Received Dec. 6, 2022 / Revised Jan. 14, 2023)

Abstract The parametric surrogate models for partial differential equations (PDEs) are a necessary component for many applications in computational sciences, and the convolutional neural networks (CNNs) have proven to be an excellent tool to generate these surrogates when parametric fields are present. CNNs are commonly trained on labeled data based on one-to-one sets of parameter-input and PDE-output fields. Recently, residual-based deep convolutional physics-informed neural network (DCPINN) solvers for parametric PDEs have been proposed to build surrogates without the need for labeled data. These allow for the generation of surrogates without an expensive offline-phase. In this work, we present an alternative formulation termed deep convolutional Ritz method (DCRM) as a parametric PDE solver. The approach is based on the minimization of energy functionals, which lowers the order of the differential operators compared to residual-based methods. Based on studies involving the Poisson equation with a spatially parameterized source term and boundary conditions, we find that CNNs trained on labeled data outperform DCPINNs in convergence speed and generalization abilities. The surrogates generated from the DCRM, however, converge significantly faster than their DCPINN counterparts, and prove to generalize faster and better than the surrogates obtained from both CNNs trained on labeled data and DCPINNs. This hints that the DCRM could make PDE solution surrogates trained without labeled data possibly.

Key words physics-informed constraint, physics-informed neural network (PINN), deep energy network, convolutional neural network (CNN)

Chinese Library Classification O175.29
2010 Mathematics Subject Classification 35-02, 92B20

1 Introduction

Partial differential equations (PDEs) are omnipresent in engineering science, and play a significant role in describing physical problems. In this context, many computational tasks

∗ Citation: FUHG, J. N., KARMARKAR, A., KADEETHUM, T., YOON, H., and BOUKLAS, N. Deep convolutional Ritz method: parametric PDE surrogates without labeled data. *Applied Mathematics and Mechanics* (*English Edition*), **44**(7), 1151–1174 (2023) https://doi.org/10.1007/s10483-023-2992-6

† Corresponding author, E-mail: nb589@cornell.edu

involve repeated evaluations of PDEs (given statistically similar parameterized inputs). These parametric solutions to PDEs are of interest in different application areas such as topology optimization[1–3], uncertainty quantification[4–6], PDE-constrained optimization problems[7–9], and multiscale problems[10–12]. Due to the excessive computational costs of high-fidelity numerical solvers such as the finite-element[13] or finite-volume method[14], surrogate modeling (also known as metamodeling or reduced order modeling (ROM)) approaches have been adopted to obtain a quick-to-evaluate model that (ideally) allows one to obtain a prediction of the PDE solution (given an input) in a fraction of the time that high-fidelity numerical solvers take. Prominently, these surrogates are based on the offline-online paradigm, a methodology using labeled data, where in the offline phase, datasets are generated from high-fidelity simulations that are utilized for training the model and building the basis for the online prediction phase. Furthermore, intrusive and non-intrusive models can be distinguished. Intrusive models typically use projections of the full order model (i.e., from the high-fidelity solver) onto a reduced space where the access to the discretized PDE operators is required. In this context, the most popular intrusive ROM method uses a proper orthogonal decomposition (POD) with a Galerkin projection (POD-Galerkin)[15–16]. Reduced order models that do not rely on discretized PDE operators are referred to as non-intrusive models. These models directly learn the input-output mapping from the high-fidelity simulation data. Different machine learning approaches have been used to generate non-intrusive surrogates for parametric PDEs, e.g., Gaussian process regression[17–18], feedforward neural networks (FNNs)[6,19], convolutional neural networks (CNNs)[20–22], generative adversarial networks[23–24], and pure data-driven operator-based techniques[25–26]. Since the numerical models are often computationally expensive, adaptive (or active) sampling methods have been developed that iteratively aim to find the set of input values which, when evaluated and concatenated to the training dataset, best enhance the accuracy of the surrogate model[27–28]. These are commonly applied to scalar-valued parameter inputs[29] as well as input fields for which known parametrizations exist[30].

In the last few years, physics-informed machine learning (PIML) techniques have had a significant impact on computational science and engineering. In PIML, the training space of machine learning models is constrained by the physical knowledge of the underlying system and more specifically the PDEs at hand. This can mainly take two forms. (i) Traditional data-driven reduced order models are enhanced by combining the information of the labeled data (e.g., loss between model prediction and ground truth PDE solution) with physical constraints (e.g., enforcement of conservation laws and fulfillment of boundary conditions)[31–32]. (ii) Machine learning techniques, e.g., neural networks, are used as full PDE-solvers using automatic differentiation[33], and can be used with or without labeled data. A realization of the latter is physics-informed neural networks (PINNs), which use the point-wise residual information of the PDE to tune the trainable parameters of an FNN such that the outputs of the network at independent parameter inputs approximate the solution to an initial-boundary value problem[34–37]. These were initially developed for forward and inverse problems involving non-parametric PDEs, i.e., the forward solver could only obtain the solution for a fixed spatial distribution of PDE coefficients (which could for example correspond to spatially varying material parameters and body forces) similar to what a finite element solver would be able to achieve. The distinguishing feature of PINNs was the ability to incorporate limited labeled data towards the solution of inverse problems. Later, different machine learning-based techniques were developed to use PDE residuals as means towards building surrogates for parametric PDEs. In this context, deep operator networks (DeepONets) with labeled data (e.g., from numerical simulations)[38–39] and without labeled data[40] can be seen as an extension to the PINN architecture. Parametric PDEs have also been solved by using CNN-parametrizations of input fields. In that setting, the PDE residual is obtained by using a finite-difference layer at the output of the network, avoiding the automatic differentiation for the approximation of the differential operators of the PDE. Since residual information is necessary to constrain the trainable parameters, we term these models deep convolutional physics-informed neural networks (DCPINNs). DCPINNs have been successfully deployed for stochastic PDEs[41], irregular domains[42], and spatiotemporal forward-solvers[43]. The advantage of these approaches is their ease of implementation and input field parametrization compared to other existing approaches.

In contrast to these residual-based approaches, PDE solvers based on energy minimization principles have been studied. With FNNs, the deep Ritz method (DRM)[44] and extensions[44–46] are used for the numerical solution of variational problems. A similar approach termed deep energy method[47–48] has been employed in the field of computational mechanics. Energy-based approaches have a significant advantage in comparison to PDE residual architectures because they require lower order differential operators and therefore show faster convergence behavior and have fewer issues with vanishing or exploding gradients than PINNs[49–50]. However, none of the presented energy-based approaches are designed for generating surrogates for parametric PDEs but instead act as simple forward solvers.

Hence, in this work, we develop an energy-based solver for parametric PDEs with CNNs. We term this approach deep convolutional Ritz method (DCRM). The proposed technique is studied on the two-dimensional Poisson equation with a spatially parameterized source term and boundary conditions for proof of concept. We compare the method to surrogate models obtained from pure CNNs with labeled data and DCPINNs without labeled data, and find that the generalization error of DCRM models converges significantly faster than both CNNs and DCPINNs, and shows improved generalization capabilities on unseen input data.

The paper is structured as follows. Section 2 provides an overview of surrogate modeling of parametric PDEs, including residual and energy formulations, and the basic concepts of CNNs (for ROM of PDEs) and DCPINNs are summarized and explained. In Section 3, we propose the DCRM and explain all its components. In Section 4, the treatment of boundary conditions for CNN-like structures in the context of PDEs is explained. In Section 5, the method is applied to modeling the parameterized Poisson equation, and the results are compared to those obtained by CNN and DCPINN surrogate modeling of PDEs. Lastly, Section 6 discusses the results and concludes the work.

2 Surrogate modeling for parametric PDEs

Consider a time-independent PDE given by

$$\mathcal{N}(u(\boldsymbol{x}), \cdots) = 0, \quad \boldsymbol{x} \in \Omega, \tag{1}$$

where $\mathcal{N}$ is a differential operator . Here, u must satisfy both Dirichlet and Neumann boundary conditions. This is known as the strong form of the PDE, and is the functional form that is typically utilized by PINN formulations.

If the Euler-Lagrange equation of an energy functional $\mathcal{E}$ coincides with the PDE of Eq. (1) (c.f. Anderson and Duchamp[51]), the solution to the PDEs can alternatively be found by utilizing the minimization of an energy as follows:

$$\min_u E(u(\boldsymbol{x}), \cdots) = \int \mathcal{E}(u(\boldsymbol{x}), \cdots). \tag{2}$$

Here, u is constrained by the same set of boundary conditions as Eq. (1). In the context of using numerical techniques (such as neural networks) to find solutions to PDEs, the direct minimization of the energy function of Eq. (2) has a significant advantage in comparison to finding solutions based on the strong form (Eq. (1)) since lower order differential operators are needed (since obtaining the strong form from Eq. (2) involves differentiation of the functional $\mathcal{I}$).

Remark 1 The variational forms of the energy equation (2) are the starting point for numerical techniques like the finite-element method[52] and virtual element method[53]. For systems based on conservation laws, e.g., energy, mass, and momentum, the energy functional can typically be derived from physical understanding. Alternatively, the problem of finding an energy functional is known as the "inverse problem of the calculus of variations", and has received a lot of attention over the years[54–56]. For an overview of different energy functionals, the calculus of variations, and some physical examples, we refer to Weinstock[57]. It needs to be pointed out that not all PDEs admit an energy functional.

In this work, we focus on the Poisson equation

$$\begin{cases} -\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}) \quad \text{in} \quad \Omega = [0,1]^2, \\ u = g_{\mathrm{D}} \quad \text{on} \quad \partial\Omega_{\mathrm{D}}, \\ \dfrac{\partial u}{\partial \boldsymbol{n}} = g_{\mathrm{N}} \quad \text{on} \quad \partial\Omega_{\mathrm{N}}, \end{cases} \tag{3}$$

where $\partial\Omega_{\mathrm{D}} \cup \partial\Omega_{\mathrm{N}} = \partial\Omega$, $\partial\Omega_{\mathrm{D}}$ refers to the part of the boundary where Dirichlet boundary conditions are applied, and $\partial\Omega_{\mathrm{N}}$ refers to the part of the boundary where Neumann boundary conditions are applied. We are interested in obtaining a deterministic surrogate model for Eq. (3) for a parameterized source term $f(\boldsymbol{x})$ and parameterized boundary conditions g_{D}, where both can vary spatially. It can be shown (see Evans[58]) that the energy functional that yields the same solution as Eq. (3) is given by

$$E[u] = \int_{\Omega} \frac{1}{2}\|\nabla u\|^2 - \int_{\Omega} uf - \int_{\partial\Omega_{\mathrm{N}}} ug_{\mathrm{N}}. \tag{4}$$

As highlighted before, the variational energy formulation involves only first-order differential operators, which is one order lower than the strong form (Eq. (3)).

A pure data-driven approach would utilize the following dataset consisting of N samples to generate a surrogate model for this application:

$$\mathcal{D} = \{\underbrace{(\boldsymbol{F}_i, \boldsymbol{G}_i^1, \boldsymbol{G}_i^2, \cdots)}_{\text{inputs}}, \underbrace{\boldsymbol{U}_i}_{\text{output}}\}_{i=1}^{N}, \tag{5}$$

where $\boldsymbol{F}$ is a source term field, $\boldsymbol{G}_i^k$ is the kth descriptor of the boundary condition of the ith sample, and $\boldsymbol{U}$ is the output field. The set of the source term and boundary condition descriptors define the input to the reduced order model, while the output is given by the primary output field $\boldsymbol{U}$.

When working with CNNs (c.f. Subsection 2.1), the dataset has to be converted into an image-like format. Standard numerical PDE solvers already provide and necessitate discretized versions of the source and output fields, which (if necessary) can be interpolated into uniform grids of height $N_{\mathrm{DOF},x}$ and width $N_{\mathrm{DOF},y}$. The parametrization of the boundary condition descriptors $(\boldsymbol{G}_i^1, \boldsymbol{G}_i^2, \cdots)$ into images, which can be processed by a CNN to learn the spatial correlation of changing boundary conditions, is more challenging. One possible option, which can also provide spatial context information, is masking functions[59] of the form

$$\mathbb{I}_g(\boldsymbol{x}) = \begin{cases} g(\boldsymbol{x}) \quad \text{if} \quad \boldsymbol{x} \in \partial\Omega, \\ 0.0, \quad \text{otherwise}, \end{cases} \tag{6}$$

which generates images that project the boundary condition onto its correct spatial position at the border of the image while zeroing all pixels that are not part of the boundary. The distinctions between the Neumann and Dirichlet boundary conditions can be made by using an additional masking function as follows:

$$\mathbb{I}_t(\boldsymbol{x}) = \begin{cases} 1.0 \quad \text{if} \quad \boldsymbol{x} \in \partial\Omega_{\mathrm{D}}, \\ 2.0 \quad \text{if} \quad \boldsymbol{x} \in \partial\Omega_{\mathrm{N}}, \end{cases} \tag{7}$$

which indicates the spatial positions of the boundary condition types at the image borders. Other common ways of imposing boundary conditions in CNN-based models are discussed in Alguacil et al.[59]. Without loss of generality, we limit ourselves in the following to the Dirichlet boundary conditions, which can be described by using a single masked boundary image. Hence, in the case with given labeled data from PDE solvers, the input images can be structured into a four-dimensional (4D) tensor of dimension $[N,\ N_{\mathrm{C,in}}=2,\ N_{\mathrm{DOF},x},\ N_{\mathrm{DOF},y}]$. Similarly, the 4D output tensor is of size $[N,\ N_{\mathrm{C,out}}=1,\ N_{\mathrm{DOF},x},\ N_{\mathrm{DOF},y}]$. Here, $N_{\mathrm{C,in}}$ and $N_{\mathrm{C,out}}$, respectively, denote the numbers of the input and output channels.

We can define the training dataset

$$\mathcal{D}_I = \{\underbrace{(\boldsymbol{F}_i, \boldsymbol{G}_i)}_{\text{inputs}},\ \underbrace{\boldsymbol{U}_i}_{\text{output}}\}_{i=1}^{N}, \tag{8}$$

where $\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{U} \in \mathbb{R}^{N\times 1\times N_{\mathrm{DOF},x}\times N_{\mathrm{DOF},y}}$. A CNN can now be trained to obtain a surrogate model for the parameterized Poisson equation. In cases where no labeled data (i.e., only the input data) is available, finite-difference (FD) stencils can be appended to the CNN to discretize and solve the PDE and therefore learn the structure of the PDE. In order to simplify notation, consider $N_{\mathrm{DOF}} = N_{\mathrm{DOF},x} = N_{\mathrm{DOF},y}$ in the following.

Recently, Zhu et al.[41] and Gao et al.[42] used CNNs with FD on the strong form of the PDE equations (1) and (3) to obtain the surrogate (in this work, we refer to this approach as DCPINN). As an alternative, we propose the DCRM (c.f. Section 3), which instead utilizes the energy formulation of Eqs. (2) and (4) as a starting point to derive surrogates for parametric PDEs without labeled data. In the following, we shortly summarize the components of a CNN, define the specific architecture used in this work, and elaborate on the DCPINN approach.

2.1 Data-driven approach with CNNs and labeled data

CNNs have been widely applied in image-based learning applications due to their ability to capture long-range spatial correlation and to efficiently share parameters (especially in comparison to FNNs). They offer a convenient parametrization of PDE input and output fields, and hence allow one to coherently obtain surrogate models for parametrized PDEs. One negative aspect of CNNs in the context of solving PDEs is that they (generally) require Cartesian uniform grids and rectangular domains as input and output fields. However, efforts have been made to extend CNNs to non-Euclidean and non-uniform grids[60–62].

In their basic form, CNNs are multilayer networks built around the convolutional operation[63]. The first and last layers of the network are the input and output layers, respectively. The network value at the lth layer in the kth channel at the node location (i, j) is obtainable by

$$z_{k,i,j}^{l} = \boldsymbol{w}_k^{l,\mathrm{T}} \boldsymbol{x}_{i,j}^{l} + b_k^{l}, \tag{9}$$

where $\boldsymbol{w}_k^l$ and b_k^l are the shared weights (also known as kernel or filter) and biases of the kth channel, respectively, and $\boldsymbol{x}_{i,j}^l$ is the input patch around the pixel location (i, j). The trainable parameters can be summarized by the set $\boldsymbol{\theta} = \{\cup_{\forall k,l}(\boldsymbol{w}_k^l, b_k^l)\}$.

One such operation with a single channel is schematized in Fig. 1, where the bias term is not considered. The mapping can be enhanced by introducing nonlinearities to CNNs by defining an activation function $\sigma(\cdot)$ which takes $z_{k,i,j}^l$ as an input and returns the activation value $a_{k,i,j}^l$.

$$a_{k,i,j}^{l} = \sigma(z_{k,i,j}^{l}). \tag{10}$$

Two typical activation functions are the ReLU function expressed as

$$a_{k,i,j} = \max(z_{k,i,j}, 0) \tag{11}$$

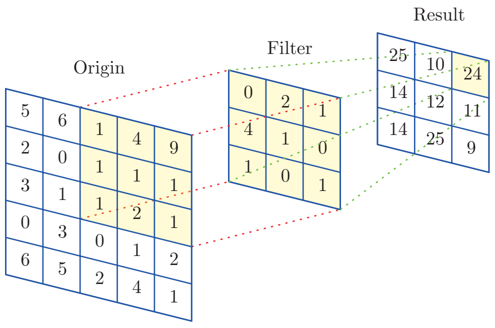


Fig. 1 Convolution operation with a 3×3 filter and no bias term (color online)

and the function expressed as

$$a_{k,i,j} = \max(z_{k,i,j}, 0) + \lambda \min(z_{k,i,j}, 0), \tag{12}$$

where $\lambda \in (0, 1)$.

Over the years, the basic CNN architecture of combining Eqs. (9) and (10) has been gradually improved by the addition of other features that increase the robustness of CNNs. Pooling layers have been added to lower the computational burden of CNN forward and backward propagations and to aid generalization[64]. One commonly applied pooling layer is known as max-pooling, which reads

$$y_{k,i,j} = \max\Big(\sum_{(m,n)\in\mathcal{R}_{i,j}} a_{k,m,n} \Big), \tag{13}$$

where $\mathcal{R}_{i,j}$ is a local neighborhood around (i, j). The opposite operation of pooling is known as upsampling. Other features of CNNs include the dropout operation introduced by Hinton et al.[65] and Srivastava et al.[66], which takes an input tensor and sets some of its element values to zero with probability p based on Bernoulli distributed samples. It has been shown that the operation can be used to effectively reduce overfitting[65]. Similarly, batch normalization re-centers and re-scales the inputs of each individual layer, which has shown to be advantageous[67].

In this work, we utilize a well-established CNN architecture known as UNet[68]. The shape we here adopt is from Kadeethum et al.[24], where it was shown to proficiently work as a data-driven ROM approach for parametrized PDEs. The structure employs an encoder-decoder framework with skip connections[69], which allows to project the main features linking the inputs and outputs into a latent space similar to autoencoders[70–71]. This enables the network to ignore insignificant data (i.e., the zeros of the boundary mask layers) and generalize more easily[72]. Since, in contrast to autoencoders, the input data of the network differs from its outputs, the network can be better described as a heteroencoder[73–74]. Two basic components build this network, i.e., (i) a contracting block primarily composed of two convolutional layers followed by a maxpooling operation and (ii) an expanding block that consists of an upsampling layer and two consecutive convolutional layers. The outputs of the convolutional layers are subjected to a Leaky ReLU activation function with $\lambda = 0.2$, while the upsampling and consecutive convolutional layers are transformed by an ReLU activation. The encoder-decoder framework is then defined by an input convolutional layer followed by six encoding blocks, six decoding blocks, and an output convolution. We choose a kernel size of 3×3 with padding, and set the stride to 1. The architecture of the network is detailed in Table 1, which also provides extra information about each layer, such as input and output sizes, activation functions, and where Dropout and Batch normalizations are used. All of the hyperparameters are chosen according to the ones used in Ref. [24].

Table 1 Detailed description of the UNet used in this study, where the input and output sizes are represented by $[N,\ N_{\mathrm{C,in}},\ N_{\mathrm{DOF},x},\ N_{\mathrm{DOF},y}]$ and $[N,\ N_{\mathrm{C,out}},\ N_{\mathrm{DOF},x},\ N_{\mathrm{DOF},y}]$, respectively, $N_{\mathrm{C,in}} = 2$, and $N_{\mathrm{C,out}} = 1$

Block	Input size	Output size	Dropout	BN	Activation
1st convolutional layer	$[N,\ N_{\mathrm{C,in}},\ 128,\ 128]$	$[N,\ 32,\ 128,\ 128]$	–	–	Linear
1st contracting block	$[N,\ 32,\ 128,\ 128]$	$[N,\ 64,\ 64,\ 64]$	✓	✓	Leaky ReLU
2nd contracting block	$[N,\ 64,\ 64,\ 64]$	$[N,\ 128,\ 32,\ 32]$	✓	✓	Leaky ReLU
3rd contracting block	$[N,\ 128,\ 32,\ 32]$	$[N,\ 256,\ 16,\ 16]$	✓	✓	Leaky ReLU
4th contracting block	$[N,\ 256,\ 16,\ 16]$	$[N,\ 512,\ 8,\ 8]$	–	✓	Leaky ReLU
5th contracting block	$[N,\ 512,\ 8,\ 8]$	$[N,\ 1\,024,\ 4,\ 4]$	–	✓	Leaky ReLU
6th contracting block	$[N,\ 1\,024,\ 4,\ 4]$	$[N,\ 2\,048,\ 2,\ 2]$	–	✓	Leaky ReLU
1st expanding block	$[N,\ 2\,048,\ 2,\ 2]$	$[N,\ 1\,024,\ 4,\ 4]$	–	✓	ReLU
2nd expanding block	$[N,\ 1\,024,\ 4,\ 4]$	$[N,\ 512,\ 8,\ 8]$	–	✓	ReLU
3rd expanding block	$[N,\ 512,\ 8,\ 8]$	$[N,\ 256,\ 16,\ 16]$	–	✓	ReLU
4th expanding block	$[N,\ 256,\ 16,\ 16]$	$[N,\ 128,\ 32,\ 32]$	–	✓	ReLU
5th expanding block	$[N,\ 128,\ 32,\ 32]$	$[N,\ 64,\ 64,\ 64]$	–	✓	ReLU
6th expanding block	$[N,\ 64,\ 64,\ 64]$	$[N,\ 32,\ 128,\ 128]$	–	✓	ReLU
2nd convolutional layer	$[N,\ 32,\ 128,\ 128]$	$[N,\ N_{\mathrm{C,out}},\ 128,\ 128]$	–	–	Linear

Using this network and given the input and output datasets of the parameterized Poisson equation of Eq. (5), we can formulate a loss function used for training the surrogate. Since we are faced with a regression problem, we define a mean-squared error (MSE) of the form

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_{\mathrm{DOF}}^2 N} \sum_{i=1}^{N} \sum_{j=1}^{N_{\mathrm{DOF}}} \sum_{k=1}^{N_{\mathrm{DOF}}} \left(\widehat{U}_{i,1,j,k}(\boldsymbol{\theta}) - U_{i,1,j,k}\right)^2 + \underbrace{\text{boundary conditions}}_{\text{if not implicit}}, \tag{14}$$

where $\widehat{\boldsymbol{U}}$ is the output prediction of the network, and $\boldsymbol{\theta}$ are the trainable parameters. The loss consists of the MSE part and a second (optional) part that is used to enforce the correct boundary conditions on the output. In this work, we implicitly enforce the boundary conditions by using padding, which will be described in Section 4. We can see that this is a residual error, i.e., the more accurate the surrogate model (with regard to the training data), the closer $\mathcal{L}$ gets to zero. The optimized set of parameters of the CNN can then be obtained by solving the optimization problem

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}), \tag{15}$$

which can be optimized by using a stochastic gradient optimizer such as Adam[75]. A flow chart for training the CNN (given labeled data of the Poisson equation) is depicted in Fig. 2.

2.2 DCPINN without labeled data

Residual errors might be an essential component of linear and nonlinear regression analyses[76–77]. Hence, it is only natural that machine learning solvers of PDEs mainly employ residual formulations, see Tripathy and Bilionis[6] and Zhu et al.[41]. Given the output $\widehat{\boldsymbol{U}}$ of a CNN and a discretized source field $\boldsymbol{F}$, the residual formulation for the Poisson equation (Eq. (3)) reads

$$\mathcal{R} = \Delta\widehat{\boldsymbol{U}} + \boldsymbol{F}. \tag{16}$$

If the boundary conditions are fulfilled and mean$(\mathcal{R}) \to 0$, the output of the CNN, i.e., $\widehat{\boldsymbol{U}}$, will be a good approximation of the true solution field $\boldsymbol{U}$. Hence, in contrast to the labeled data

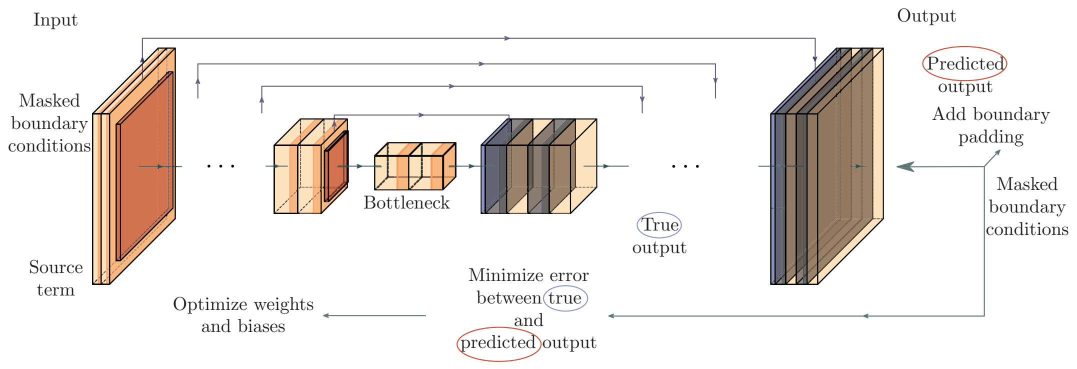

Input
Masked boundary conditions
Source term
Bottleneck
True output
Output
Predicted output
Add boundary padding
Masked boundary conditions
Minimize error between true and predicted output
Optimize weights and biases

Fig. 2 Schematic flow of training a CNN architecture with labeled data to build a surrogate for the Poisson equation with source term and boundary condition parametrizations. Orange and red components indicate the contracting blocks (before bottleneck), and blue and yellow components specify the expanding blocks (after bottleneck) (color online)

case of Subsection 2.1, where the difference between the network output and the true (known) output is minimized to train the network, in DCPINNs, the minimization of the residual $\mathcal{R}$ is the goal of the training process. The requirement constrains the solution space so that the boundary conditions need to be fulfilled.

The minimization of the left-hand side of Eq. (16) requires the discretization of the Laplacian differential operator Δ. In PINN formulations involving FNNs where the dependent parameters are the network's inputs, the differential operator can be resolved by using automatic differentiation of the network outputs concerning the inputs. This is not an option in the current CNN formulation since the network inputs are the parameter fields of the PDEs (in the case of the Poisson equation $\boldsymbol{F}$).

We can make use of the fact that CNN filters (c.f. Fig. 1) are structurally equivalent to finite difference stencils[32]. We can see this in the following example. Consider the Laplacian operator discretized by central difference schemes as follows:

$$\begin{aligned}\Delta u(x,y) &= u_{xx}(x,y) + u_{yy}(x,y)\\ &\approx \frac{u(x-h,y)+u(x+h,y)-4u(x,y)+u(x,y-h)+u(x,y+h)}{h^2}\\ &=: \Delta_h u(x,y),\end{aligned} \tag{17}$$

where h $(=1/N_{\mathrm{DOF}})$ represents the constant spacing variable. Equation (17) is equivalent to the commonly applied 5-point stencil[78]

$$\Delta_h = \frac{1}{h^2}\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}. \tag{18}$$

Hence, if we replace the convolutional filter (or kernel) of Fig. 1 with the stencil of Eq. (18), we can obtain a simple way to discretize the Laplacian operator spatially. Therefore, by adding a fixed, non-trainable convolutional operation at the end of a CNN, we can approximate the residual formulation of Eq. (16).

Ultimately, the DCPINN formulation can be trained by minimizing the following loss function:

$$\mathcal{L} = \frac{1}{N_{\mathrm{DOF}}^2 N} \sum_{i=1}^{N} \sum_{j=1}^{N_{\mathrm{DOF}}} \sum_{k=1}^{N_{\mathrm{DOF}}} \left(\widehat{\mathcal{R}}_{i,1,j,k}\right)^2 + \underbrace{\text{enforcing boundary conditions}}_{\text{if not implicitly}}, \tag{19}$$

which consists of a term for the mean squared of the approximated residual and a term that constraints the solution space by enforcing the boundary conditions. Since we opt to implicitly enforce the boundary condition (see Section 4), the latter term is automatically zero. The DCPINN approach is depicted in Fig. 3.

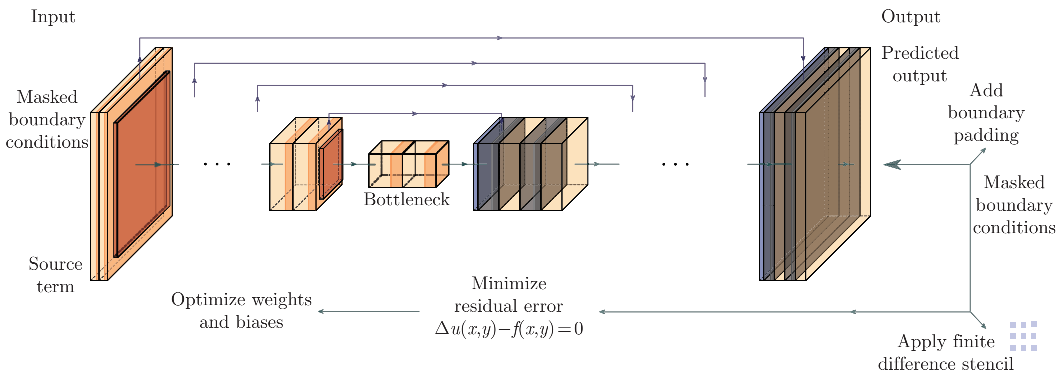

Input
Output
Masked boundary conditions
Predicted output
Add boundary padding
Bottleneck
Masked boundary conditions
Source term
Optimize weights and biases
Minimize residual error $\Delta u(x,y)-f(x,y)=0$
Apply finite difference stencil

Fig. 3 Schematic flow of training a CNN architecture by using DCPINNs without labeled data to build a surrogate for the Poisson equation with source term and boundary condition parametrizations. Orange and red components indicate the contracting blocks (before bottleneck), and blue and yellow components specify the expanding blocks (after bottleneck) (color online)

3 DCRM without labeled data

So far, we have reviewed two different ways of generating surrogates for parametric PDEs, i.e., (i) the classical approach, which uses a CNN trained on labeled data, and (ii) DCPINN, which does not need labeled data and instead solves the PDE based on the PDE residual error. Here, we propose an additional framework called the DCRM. It works essentially similar to the DCPINN, but instead of finding a solution by minimizing the residual of the PDE, akin to the DCPINN, the DCRM aims to minimize the energy functional (c.f. Eq. (2)) which is a non-residual value. This idea has recently been explored in the context of solving non-parametric PDEs[44,47–48,79].

Energy-based solvers for PDEs were generally first proposed over 100 years ago by Walter Ritz[80]1. From this point on, different solution techniques for PDEs (the finite-element method being the most prominent) have been developed, incorporating these initial ideas. Instead of requiring that the PDE holds at every material point, like in the strong form, functionals state that the conditions must only be met in an average sense. This leads to lower order differential operators in the energy functional compared to their counterparts in the strong form, but requires the evaluation of integrals that span lines, area, or volume. Hence, in general, we seek

1There is an ongoing debate whether Ritz or Rayleigh was the first to conceive the idea[44]. We do not intend the name “deep convolutional Ritz method” to be controversial. Our approach is named in the style of an earlier paper[81] termed “deep Ritz method.”

to find a solution to a variational problem of the form

$$\min_u E(u(\boldsymbol{x}), \cdots), \tag{20}$$

where the trial function u is restricted to a set of admissible functions. The DCRM finds a solution to this problem based on the following ideas.

(I) A CNN is used as an approximation of the trial function u. This output is then constrained based on the boundary conditions.

(II) The parametric fields of the variational problem (and therefore PDE) act as an input to the CNN.

(III) The differential operators are approximated by using finite difference stencils conforming with the grid of the CNN itself. This alleviates the need for additional backpropagation operations for the calculation of differential operators.

(IV) Numerical integration schemes are used to approximate the integrals that manifest in the energy formulation, which is necessary for the minimization of the energy informing the loss function of the CNN.

(V) The optimization problem is solved by using the stochastic gradient descent algorithm.

In the following, these components are discussed for the particular case of the Poisson equation. However, it can similarly be applied to other energy functionals to build a solver for PDEs.

Given the CNN output $\widehat{\boldsymbol{U}}$ and the source field $\boldsymbol{F}$, we can rewrite Eq. (4) in a tensorial form that reads

$$E[u] = \int_0^1 \int_0^1 \underbrace{\left(\frac{1}{2}\|\nabla\widehat{\boldsymbol{U}}\|^2 - \widehat{\boldsymbol{U}}\boldsymbol{F}\right)}_{I} - \int_{\partial\Omega_{\mathrm{N}}} \underbrace{\widehat{\boldsymbol{U}}|_{\partial\Omega_{\mathrm{N}}}\boldsymbol{g}_{\mathrm{N}}}_{e}, \tag{21}$$

where $\widehat{\boldsymbol{U}}|_{\partial\Omega_{\mathrm{N}}}$ are the values of $\widehat{\boldsymbol{U}}$ at the spatial positions where the Neumann boundary conditions are prescribed. In order to minimize Eq. (21), an efficient way to differentiate and integrate in this framework must be established. Similarly to the DCPINN, convolutional filters can be used to discretize the differential operations. When $h = 1/N_{\mathrm{DOF}}$, both components of ∇ can individually be discretized by using central differences which read and result in

$$\begin{cases} \dfrac{\partial u}{\partial x} \approx \dfrac{u_{i+1,j} - u_{i-1,j}}{2h} \quad \text{with the filter} \quad \dfrac{1}{2h}\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \\[2ex] \dfrac{\partial u}{\partial y} \approx \dfrac{u_{i,j+1} - u_{i,j-1}}{2h} \quad \text{with the filter} \quad \dfrac{1}{2h}\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \end{cases} \tag{22}$$

Hence, after the CNN output is obtained $\widehat{\boldsymbol{U}}$, two fixed non-trainable convolutional operations can independently be adopted to approximate the gradient operator. Since the CNN output is given in the form of a regular grid, the integrals of Eq. (21) can simply be approximated by using classical numerical integration techniques, yielding

$$\widehat{E} = \sum_{i=1}^{N} \sum_{j=1}^{N_{\mathrm{DOF}}} \sum_{k=1}^{N_{\mathrm{DOF}}} W_{j,k} I_{i,j,k} + \sum_{i=1}^{N} \sum_{j=1}^{n_{\mathrm{BCN}}} w_j e_{i,j}. \tag{23}$$

Here, n_{BCN} denotes the degrees of freedom affected by the Neumann boundary, W and w are integration weights, and I and e are defined in Eq. (21). An overview of existing numerical integration techniques is given in Davis and Rabinowitz[82]. Without loss of generality, we employ Simpson's rule in this work. The integration weights for this method are given in Appendix A.

After obtaining an approximation $\widehat{E}$ of the energy E inside the system, the trainable parameters of the DCRM framework for the Poisson equation can be obtained by minimizing the loss function

$$\mathcal{L} = \widehat{E} + \underbrace{\text{enforcing boundary conditions}}_{\text{here enforced implicitly}}, \tag{24}$$

where the combination of $\widehat{E}$ and a boundary term must be minimized. One major issue with the DCRM in comparison to the DCPINN is the fulfillment of the boundary conditions. Whereas in purely residual-based approaches, as in CNNs with labeled data or DCPINNs, the boundary conditions could potentially be enforced in a weak sense by adding a penalty term to the loss function. This is not immediately possible with the DCRM. This is because non-residual values ($\widehat{E}$) and residual-based values (penalization of boundary conditions) are mixed in one loss function. This is problematic, and would seriously dampen the potential of the DCRM because the penalty parameter would have to be chosen very carefully such that the penalty term is always dominant compared to $\widehat{E}$, which would lead to long training time and hyperparameter searches. To circumvent these problems, the boundary conditions will be hard encoded into the output of the CNN by using a padding approach that will be explained in Section 4. A schematic overview over the DCRM is provided in Fig. 4. Table 2 summarizes the advantages and disadvantages of the DCRM compared to the CNN and DCPINN.

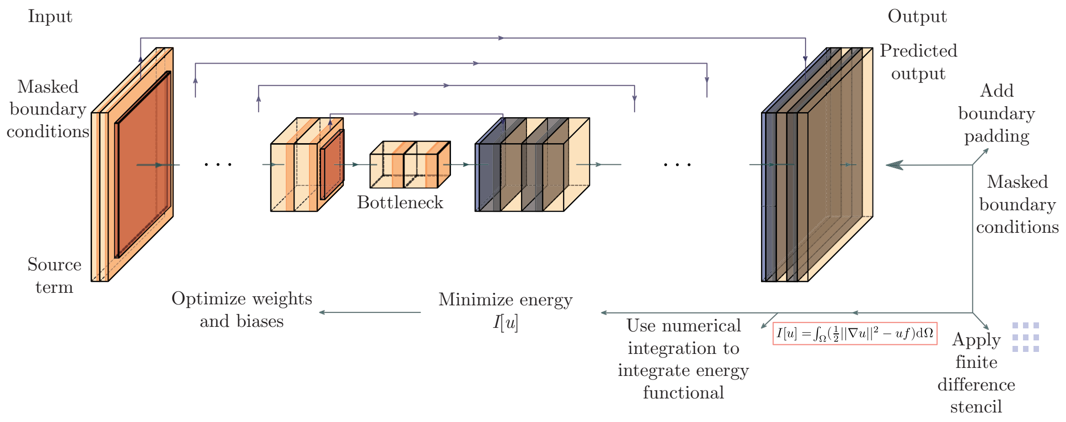

Input
Output
Masked boundary conditions
Predicted output
Add boundary padding
Bottleneck
Masked boundary conditions
Source term
Optimize weights and biases
Minimize energy $I[u]$
Use numerical integration to integrate energy functional
$I[u] = \int_\Omega (\frac{1}{2}\|\nabla u\|^2 - uf)\mathrm{d}\Omega$
Apply finite difference stencil

Fig. 4 Schematic flow of training a Unet by using the DCRM without labeled data to build a surrogate for the Poisson equation with source term and boundary condition parametrizations. Orange and red components indicate the contracting blocks (before bottleneck), and blue and yellow components specify the expanding blocks (after bottleneck) (color online)

4 Hard-enforcement of boundary conditions

In all of the discussed schemes throughout this work, the loss function (Eqs. (15), (19), and (24)) used to train the models involves constraining the output of the CNN to adhere to the boundary conditions imposed on the PDE. In this section, we describe and discuss a way to enforce different boundary conditions with an alternate approach, i.e., strictly enforcing them. This contrasts with the weak imposition of the conditions with penalty approaches added to the loss. As mentioned in the last section, this is especially important for the DCRM since

Table 2 Advantages and disadvantages of the CNN and DCPINN compared to the proposed DCRM framework

Technique	Advantage	Disadvantage
CNN	Easy implementation	Labeled data required
DCPINN	No labeled data needed; soft-enforcement of boundary conditions	Full differential operator of PDE; approximation of differential operator
DCRM (proposed here)	Lower order differential operators than the DCPINN; no labeled data needed	Numerical integration needed; approximation of differential operator; hard-enforcement of boundary conditions; energy functionals not available for all PDEs

non-residual and residual-based components would be combined in the loss function, which could potentially lead to problems for the optimization. Even though the enforcement of the boundary condition is not a requirement in the case where labeled data is available (Subsection 2.1), it is still beneficial in terms of the output accuracy when specific parts of the output do not need to be learned but are implicitly enforced on the output. Different ways of imposing boundary conditions in CNN-based surrogate modeling for PDEs are discussed in Alguacil et al.[59]. In this work, we briefly discuss how to enforce Dirichlet, Neumann, and periodic boundary conditions having 3×3 convolutional filters in mind.

4.1 Dirichlet boundary condition

The exact imposition of Dirichlet boundary conditions can be achieved by adding a layer around the output of the image containing the boundary conditions that have to be enforced. Thereby, enlarge the image from $N_{\mathrm{DOF}} \times N_{\mathrm{DOF}}$ to $N_{\mathrm{DOF}} + 2 \times N_{\mathrm{DOF}} + 2$. In the machine learning community, adding additional pixels to the edge of the image is known as padding. An example of the padding approach can be seen in Fig. 5. Here, the image on the left side represents the output of our CNN. Then, to enforce g_{D} everywhere on the edge of the output, this value is padded to the initial image, creating the image on the right.

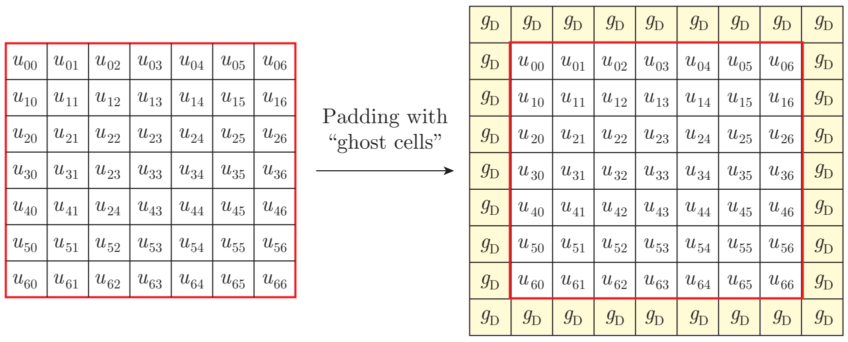


Fig. 5 Padding under constant Dirichlet boundary conditions (color online)

4.2 Neumann boundary condition

The hard-imposition of Neumann boundary conditions is more complex than that of Dirichlet boundary conditions, and is dependent on the CNN filter used to discretize the differential operator. Under the consideration of a central difference scheme, the boundary condition can be approximated by

$$\frac{\partial u}{\partial \boldsymbol{n}} = \nabla u \cdot \boldsymbol{n} = \begin{bmatrix} u_{,1} \\ u_{,2} \end{bmatrix} \cdot \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} = \begin{bmatrix} \dfrac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \\ \dfrac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \end{bmatrix} \cdot \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} = g_{\mathrm{N}}. \tag{25}$$

This equation can be used to define the values of extra layers (padding) that are added to the outer edges of the CNN output to enforce the boundary condition which is similar to "ghost" nodes that are typically applied in the finite difference method. If we, for example, consider the right-hand edge of the image with $\boldsymbol{n} = [1, 0]^{\mathrm{T}}$, the condition reads

$$\begin{cases} \begin{bmatrix} u_{,1} \\ u_{,2} \end{bmatrix} \cdot \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} = \dfrac{u_{i+1,j} - u_{i-1,j}}{2h} = g_{\mathrm{N}}, \\ u_{i+1,j} = 2hg_{\mathrm{N}} + u_{i-1,j}. \end{cases} \tag{26}$$

Figure 6 shows an example of this operation when $g_{\mathrm{N}} = 0$ everywhere on the boundary. We can see that the corner points of the padded later are not defined by this operation. In traditional finite difference schemes, specific equations would be added to the equation system to allow to seamlessly solve for the required corner values that enforce the conditions. However, anticipating some numerical error (in case that g_{N} is different in the neighboring positions of the corner), we can set the corner values as the averages of neighboring boundary points defined by the operation described above.

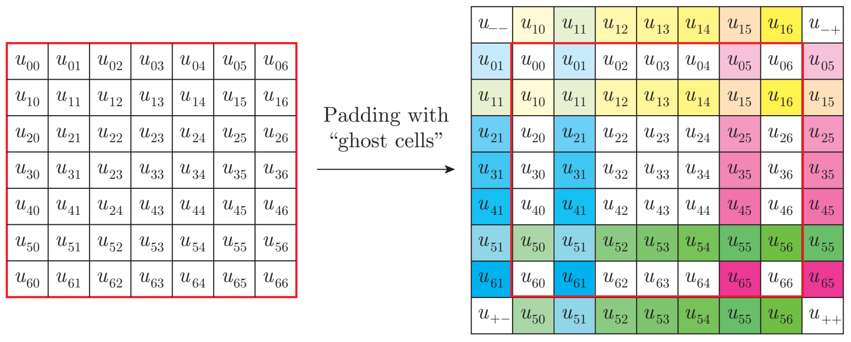


Fig. 6 Padding under constant Neumann boundary $\frac{\partial u}{\partial \boldsymbol{n}} = \nabla u \cdot \boldsymbol{n} = 0.0$ conditions for second-order schemes. The corner values can be obtained as the average of neighboring pixel values (color online)

4.3 Periodic boundary condition

Periodic boundary conditions defined as

$$u_{N+1,j} = u_{0,j}, \quad u_{i,M+1} = u_{i,0} \tag{27}$$

can be hard-encoded by adding padding layers to two sides of the image with the values of the opposing sides. This operation was described by Mohan et al.[32]. This approach is visually explained in Fig. 7 for 3×3 filters.

5 Numerical studies

In this section, the DCRM will be employed and be compared to the DCPINN and labeled-data-CNN for three different test cases involving the Poisson equation (c.f. Eq. (3)) of increasing complexity to highlight its performance. All network architectures were implemented

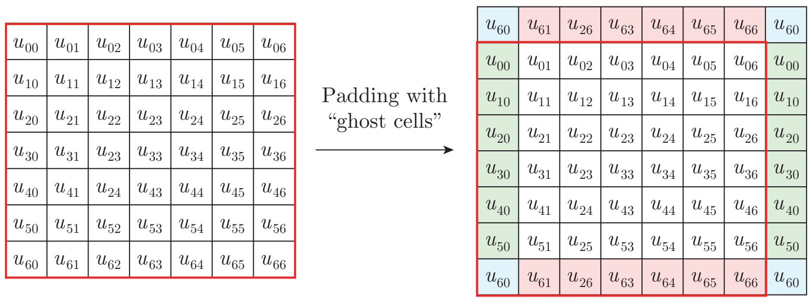

u_{00} u_{01} u_{02} u_{03} u_{04} u_{05} u_{06}
u_{10} u_{11} u_{12} u_{13} u_{14} u_{15} u_{16}
u_{20} u_{21} u_{22} u_{23} u_{24} u_{25} u_{26}
u_{30} u_{31} u_{23} u_{33} u_{34} u_{35} u_{36}
u_{40} u_{41} u_{24} u_{43} u_{44} u_{45} u_{46}
u_{50} u_{51} u_{52} u_{53} u_{54} u_{55} u_{56}
u_{60} u_{61} u_{62} u_{63} u_{64} u_{65} u_{66}
Padding with "ghost cells"
u_{60} u_{61} u_{26} u_{63} u_{64} u_{65} u_{66} u_{60}
u_{00} u_{01} u_{02} u_{03} u_{04} u_{05} u_{06} u_{00}
u_{10} u_{11} u_{12} u_{13} u_{14} u_{15} u_{16} u_{10}
u_{20} u_{21} u_{22} u_{23} u_{24} u_{25} u_{26} u_{20}
u_{30} u_{31} u_{23} u_{33} u_{34} u_{35} u_{36} u_{30}
u_{40} u_{41} u_{24} u_{43} u_{44} u_{45} u_{46} u_{40}
u_{50} u_{51} u_{25} u_{53} u_{54} u_{55} u_{56} u_{50}
u_{60} u_{61} u_{26} u_{63} u_{64} u_{65} u_{66} u_{60}

Fig. 7 Padding under periodic boundary conditions for second-order schemes (color online)

in Pytorch[83]. In the following, we employ the Adam optimizer[75] with a learning rate of 1×10^{-4} as suggested and employed by Kadeethum et al.[24]. The labeled data for the benchmark CNN is obtained by using a finite difference solver, whose solutions will also be seen as ground truths in the following. In order to allow a performance comparison of the different techniques, the total absolute output error defined by

$$\mathcal{E}_{\text{abs}} = \sum_{i=1}^{N} \sum_{j=1}^{N_{\text{DOF}}} \sum_{k=1}^{N_{\text{DOF}}} (\widehat{\mathcal{R}}_{i,1,j,k})^2 \tag{28}$$

is used. The input fields were normalized before training. To allow for consistent comparisons, when the output data is used for training (CNN trained on labeled data), the output fields are normalized as well. The trainable parameters of all the architectures are initialized with the same values by using Pytorch's default initialization.

5.1 Case 1: non-parametric forward solver

The first case compares the DCRM and DCPINN for solving a single forward problem of the Poisson equation without any parametrization. Consider an inhomogeneous boundary value problem of the form

$$\begin{cases} \Delta u(x,y) = 20\pi^2(x^2+y^2)\sin\left(\pi\left(x+\dfrac{\pi}{4}\right)\left(y+\dfrac{\pi}{4}\right)\right) \quad \text{in} \quad \Omega, \\ u = g \quad \text{on} \quad \partial\Omega, \end{cases} \tag{29}$$

where the Dirichlet boundary condition takes the form

$$\begin{cases} g = 1 \quad \text{on} \quad \{(x,0)\cup(x,1)\cup(0,y)\subset\partial\Omega\}, \\ g = \cos(2\pi y) \quad \text{on} \quad \{(1,y)\subset\partial\Omega\}. \end{cases} \tag{30}$$

The ground truth solution to this problem is shown in Fig. 8(a). Figure 8(b) compares the absolute error evolution of the DCRM and DCPINN over the training process. It can be seen that the DCRM initially converges to its minimum after less than 5 000 epochs. The absolute error of the DCPINN eventually gets lower than the DCRM but only after around 45 000 epochs, making the DCRM converge around 9 times faster. To put this into context, Figs. 9(a) and 9(b) show the CNN output of the DCRM after 3 500 training epochs and the absolute error over the domain, respectively. We can see that even after only 3 500 epochs, the output of the DCRM already closely resembles the ground truth of Fig. 8(a). The output and absolute error of the DCPINN after 3 500 epochs are depicted in Figs. 9(c) and 9(d) and highlight the significant difference between the convergence speeds of the DCRM and DCPINN. However, the DCRM

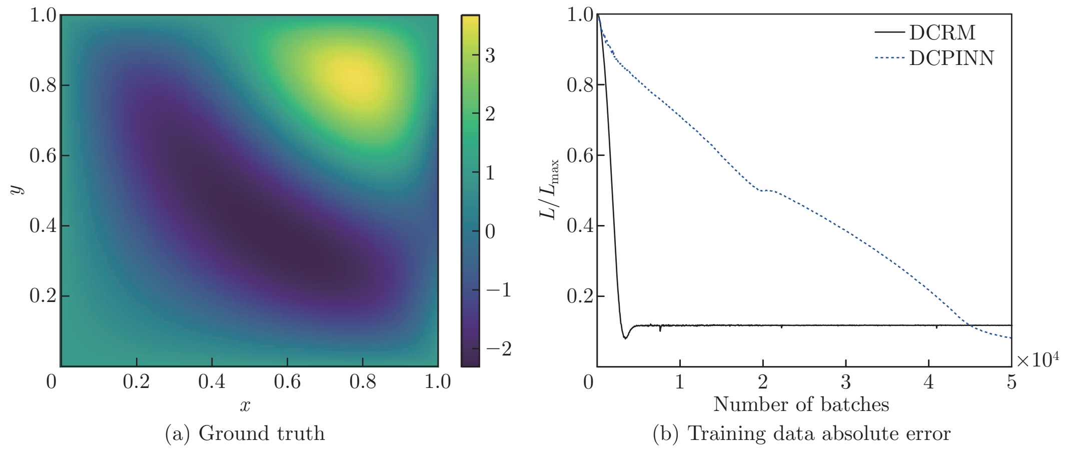


Fig. 8 Case 1: forward solver. (a) Ground truth solution of the forward problem. (b) Normalized absolute error between the ground truth solution and the DCRM and DCPINN predictions over the training process (color online)

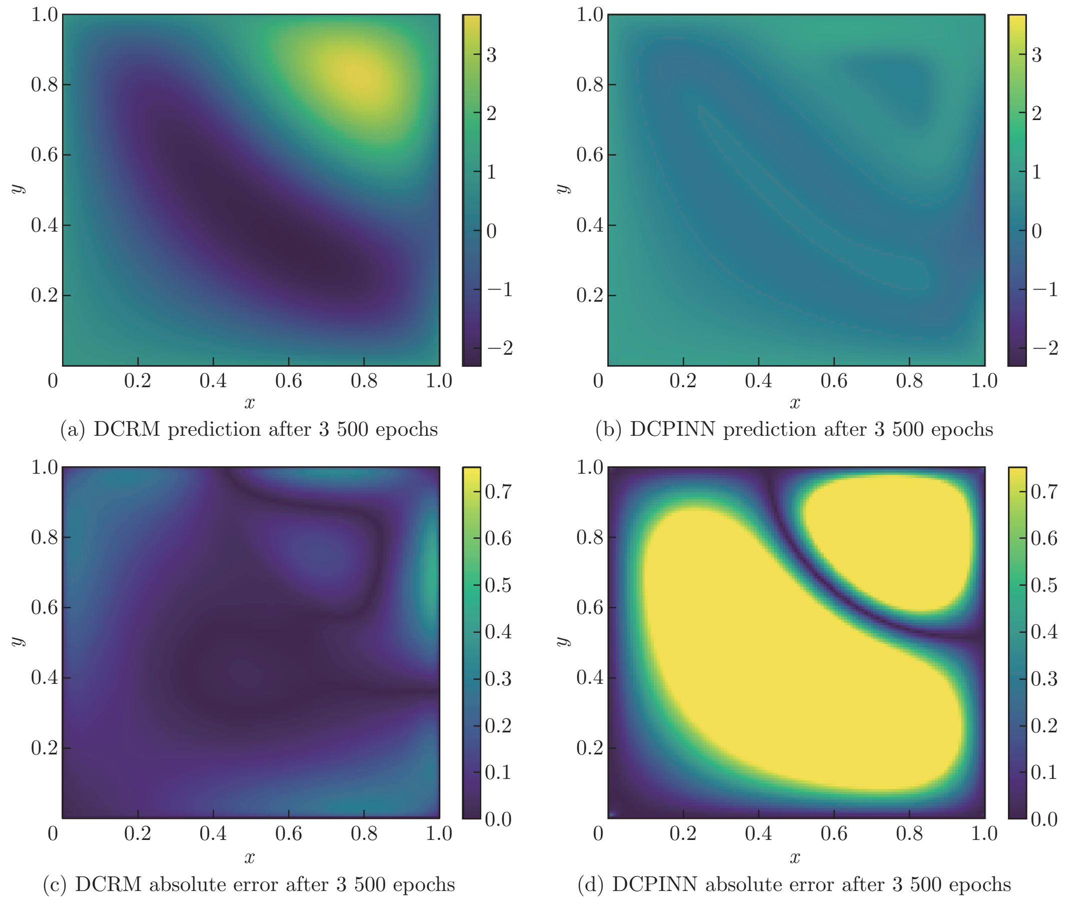


Fig. 9 Case 1: predicted fields after 3 500 training epochs. (a), (c) DCRM predicted field and absolute error field. (b), (d) DCPINN predicted field and absolute error field (color online)

appears to not be able to reduce the residual error as far as the DCPINN as training continues. We believe that this is due to the fact that the energy-based DCRM solver appears to get stuck in a local minimum more easily. This might be due to the global nature of the energy functional approximation. Similar observations have been made in other works[48,84]. Furthermore, the error of the numerical integration scheme might further emphasize the differences between the predicted and true solutions which might appear due to the different approximations of the differential operators. However, as we will see in the following two examples, these problems appear to not be as significant when using the DCRM as a surrogate for parametric PDEs.

5.2 Case 2: forward solver with parametrized source term

The second case aims to generate a surrogate solution to the PDE considering a parametrized source term of the Poisson equation. We compare the performances of the DCRM to those of the DCPINN and a data-driven CNN (trained on labeled data from an FD solver). Consider the following boundary value problem:

$$\begin{cases} \Delta u(x,y) = f(x,y) \quad \text{in} \quad \Omega, \\ u = g \quad \text{on} \quad \partial\Omega, \end{cases}$$

where the boundary conditions are specified as

$$\begin{cases} g = \cos(2\pi y) \quad \text{on} \quad \{(1,y) \subset \partial\Omega\}, \\ g = 1 \quad \text{on} \quad \{\partial\Omega \setminus (1,y)\}. \end{cases} \tag{31}$$

We obtain different source terms as samples of the following function:

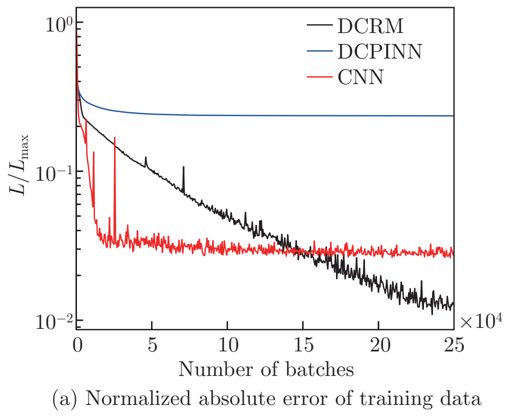
$$f(x,y) = \alpha\pi^2(x^2+y^2)\sin(\pi(x+\beta)(y+\gamma)),$$

where we consider

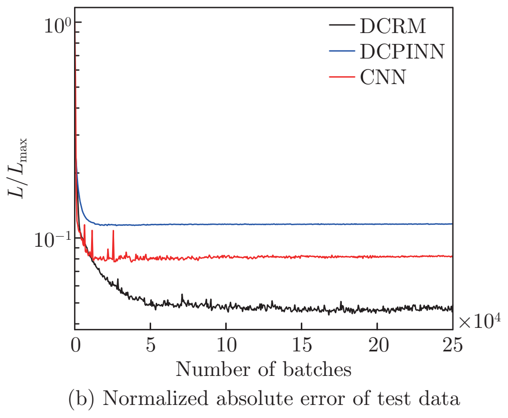
$$\alpha \in [1,10], \quad \beta \in \left[0, \frac{\pi}{2}\right], \quad \gamma \in \left[0, \frac{\pi}{2}\right],$$

which means that we cover a wide spectrum of complex source terms with a significant variation in the magnitude and spatial variability, making the process of obtaining an accurate and generalizable surrogate challenging. We consider 100 different samples of the source term for training purposes, which are obtained by using Latin hypercube sampling (LHS). The surrogate models are then tested on 1 000 different source terms sampled by LHS. The models are trained by using a batch size of 2. Figure 10(a) depicts the normalized absolute error on the training data over the training process.

It can be seen that even though all three different approaches quickly lower the error initially, the DCPINN performs significantly worse than the other two methods. The CNN with labeled data initially converges most quickly, but the DCRM has the lowest final error after training. The error over the test cases highlights a different performance for generalization as highlighted in Fig. 10(b). Here, even though trained without data, the DCRM consistently yields the lowest error on unseen data over (essentially) the whole training process. This is a surprising result since the CNN trained on labeled data initially has a lower error on the test data. However, it seems that the physical constraints on the DCRM allow it to generalize more effectively. The same cannot be seen with the DCPINN. Hence, at least in the studied application case, the comparison of the results of the DCPINN and the CNN with labeled data would suggest that the availability of the (at least some) data of the input-output mapping is necessary for obtaining the best generalizing surrogate model for parametric PDEs. However, by employing the DCRM to build the surrogate model, labeled data can be made entirely obsolete with regard to the generalization quality.



(a) Normalized absolute error of training data



(b) Normalized absolute error of test data

Fig. 10 Case 2: transfer learning through variations in the source term. Training and test errors over the training process. There are 100 parametric data fields for training and 1 000 parametric data fields for testing (color online)

5.3 Case 3: forward solver with parametrized source term and boundary conditions

The last case aims to consider both a parameterized source term and the boundary conditions. We compare the DCRM to both the DCPINN and a CNN trained on labeled data. The PDE in this case is given by

$$\begin{cases} \Delta u(x,y) = f(x,y) \quad \text{in} \quad \Omega, \\ g \quad \text{on} \quad \{\partial\Omega\}, \end{cases}$$

where the Dirichlet boundary values g are sampled from the following one-dimensional (1D) function:

$$g(z) = \eta \cos(z), \quad z \in [0, 2\pi], \quad \eta \in [-1, 1],$$

which is projected onto the boundary of the domain (see Fig. 11). Similarly to the previous case, the source term is a realization of the function defined by Eq. (5) with the parameter values defined in the ranges of Eq. (2). The training data is generated by sampling both the boundary and source term functions 250 times with LHS, while 1 000 test inputs are generated the same way. The models are trained by using a batch-size of 2.

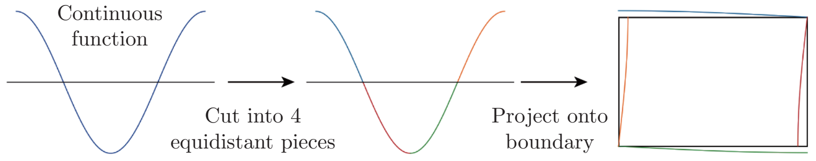


Fig. 11 Parametrization of boundary condition. Projection from 1D function (color online)

Figure 12(a) depicts the normalized error over the training data plotted over the training process. The error evolution of the three frameworks shows a similar behavior to that observed in Subsection 5.2. The errors of the DCPINN and CNN converge quickly, with the DCPINN performing worse. The DCRM, on the other hand, continues to decrease until reaching a lower error level than the CNN at the end of training. The most important criterion for the quality of the surrogate models is the generalization error. The normalized error of the test data is plotted over the training process in Fig. 12(b). We can draw the same conclusions as discussed for the previous case (Subsection 5.2). The DCPINN performs worse than the CNN trained with labeled data. The DCRM, however, consistently has the best generalization quality over the whole training process, meaning that it performs the best on unseen data. This highlights again that (for the presented problem) the DCRM can entirely cancel the need for labeled data, whereas the DCPINN cannot.

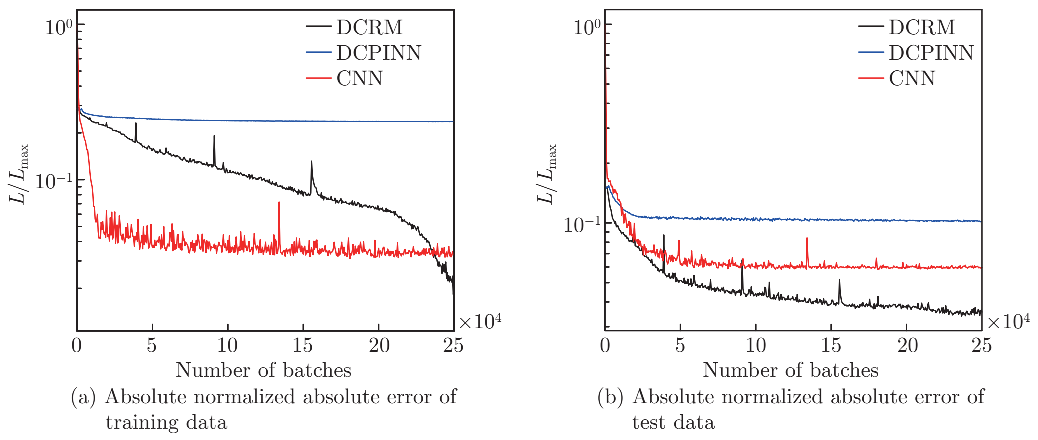


Fig. 12 Case 3: transfer learning through variations in the source term and boundary conditions. Training and test errors over the training process. There are 250 parametric data fields for training and 1 000 for testing (color online)

6 Discussion, conclusions, and outlook

In this work, we propose the DCRM as a new way of generating surrogate models for parametric PDEs entirely without labeled data. The method is based on the abilities of CNNs to parametrize inputs and output fields.

We summarize how CNNs can be used to obtain surrogate models for parametric PDEs with given labeled data. Additionally, DCPINNs are reviewed, in which a surrogate model is generated without labeled data by minimizing the residual of the PDE. In contrast to both of these methods, which rely on residual error formulations, DCRMs are trained to minimize the energy functional of the PDEs. Here, the integrals of the functionals are approximated by using numerical integration techniques, and the differential operators are resolved by adding a non-trainable, fixed convolutional filter at the output of the CNN, which is equivalent to the finite-difference operation.

The technique has a significant advantage compared to DCPINNs since the degree of the differential operator is lower, making the training and accuracy of the method quicker. Furthermore, we discuss ways to enforce different types of boundary conditions in a strict implicit way, which is critical for the DCRM since, here, non-residual and residual values are combined within the loss function. The surrogate models generated by training CNNs on labeled data, CNNs, and DCRM are tested on a Poisson equation problem with a parametric source term and boundary conditions. The presented results indicate that DCPINNs alone do not show a better generalization ability than plain CNNs trained on labeled data as discussed in Zhu et al.[41]. However, DCRMs generalize better than both DCPINNs and CNNs (with labeled data) without requiring any output data. Hence, using a DCRM to generate a surrogate model for the parametric Poisson equation makes access to one-to-one data pairs of inputs and outputs seemingly unnecessary.

This is also emphasized by the fact that the duration of the extra steps during the training process, i.e., numerical integration and differentiation, is (time-wise) insignificant compared to the time it takes to complete a backward pass through the presented network, which is equivalent for all three compared frameworks. Hence, training a DCRM takes the same amount of time as a CNN where the labeled data is already available. However, even for this simple problem where the finite-difference solver takes seconds to evaluate, the time it takes to generate the labeled data cannot be neglected, making the DCRM the most efficient and best-performing way to generate a surrogate model for the parametric Poisson problem. For boundary value problems where the numerical solver needs significant computational time, this effect might be even more pronounced. This will be the subject of future studies where surrogate models for parametric elasticity and phase-field problems will be studied by using the DCRM approach. Next steps could also include problems that involve irregular domains, which have already been solved by using regular DCPINNs[42]. Additionally, the method can be extended to spatiotemporal problems. However, these are particularly challenging because the efficient learning of temporal evolutions is still an open and active area of research. In order to solve these problems, we intend to look into ways of extending the CNN architecture of the DCRM into network types that can accumulate hidden state information such as convolutional long short-term memory networks[85]. This architecture has already been successfully used by Ren et al.[43] in the context of DCPINNs.

Furthermore, DCRMs need to be tested on their ability to deal with a larger number of input parameters/fields.

Conflict of interest The authors declare no conflict of interest.

Data availability statement Codes related to this work including the implementation of DCRM will be published on Github: https://github.com/FuhgJan/DCRM, after the work has been accepted for publication.

Acknowledgements This work was supported by the Laboratory Directed Research and Development Program at Sandia National Laboratories (No. 218328). Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC (NTESS), a wholly owned subsidiary of Honeywell International Inc., for the U. S. Department of Energy's National Nuclear Security Administration (DOE/NNSA) under contract DE-NA0003525. This written work is co-authored by an employee of NTESS. The employee, not NTESS, owns the right, title, and interest in and to the written work and is responsible for its contents. Any subjective views or opinions that might be expressed in the written work do not necessarily represent the views of the U. S. Government. The publisher acknowledges that the U. S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this written work or allow others to do so, for U. S. Government purposes. The DOE will provide public access to results of federally sponsored research in accordance with the DOE Public Access Plan.

References

[1] GOGU, C. Improving the efficiency of large scale topology optimization through on-the-fly reduced order model construction. *International Journal for Numerical Methods in Engineering*, **101**(4),

281–304 (2015)

[2] XIA, L. and BREITKOPF, P. A reduced multiscale model for nonlinear structural topology optimization. *Computer Methods in Applied Mechanics and Engineering*, **280**, 117–134 (2014)

[3] KESHAVARZZADEH, V., KIRBY, R. M., and NARAYAN, A. Robust topology optimization with low rank approximation using artificial neural networks. *Computational Mechanics*, **68**(6), 1297–1323 (2021)

[4] ROACHE, P. J. Quantification of uncertainty in computational fluid dynamics. *Annual Review of Fluid Mechanics*, **29**(1), 123–160 (1997)

[5] CHEN, P., QUARTERONI, A., and ROZZA, G. Reduced basis methods for uncertainty quantification. *SIAM/ASA Journal on Uncertainty Quantification*, **5**(1), 813–869 (2017)

[6] TRIPATHY, R. K. and BILIONIS, I. Deep UQ: learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics*, **375**, 565–588 (2018)

[7] BIEGLER, L. T., GHATTAS, O., HEINKENSCHLOSS, M., and WAAMDERS, B. V. B. Large-scale PDE-constrained optimization: an introduction. *Real-Time PDE-Constrained Optimization*, Springer, Berlin/Heidelberg (2003)

[8] FAHL, M. and SACHS, E. W. Reduced order modelling approaches to PDE-constrained optimization based on proper orthogonal decomposition. *Large-Scale PDE-Constrained Optimization*, Springer, Berlin/Heidelberg (2003)

[9] ZAHR, M. J. and FARHAT, C. Progressive construction of a parametric reduced-order model for PDE-constrained optimization. *International Journal for Numerical Methods in Engineering*, **102**(5), 1111–1135 (2015)

[10] FUHG, J. N., BOEHM, C., BOUKLAS, N., FAU, A., WRIGGERS, P., and MARINO, M. Model-data-driven constitutive responses: application to a multiscale computational framework. *International Journal of Engineering Science*, **167**, 103522 (2021)

[11] FUHG, J. N., MARINO, M., and BOUKLAS, N. Local approximate Gaussian process regression for data-driven constitutive models: development and comparison with neural networks. *Computer Methods in Applied Mechanics and Engineering*, **388**, 114217 (2022)

[12] FUHG, J. N. and BOUKLAS, N. On physics-informed data-driven isotropic and anisotropic constitutive models through probabilistic machine learning and space-filling sampling. *Computer Methods in Applied Mechanics and Engineering*, **394**, 114915 (2022)

[13] WRIGGERS, P. *Nonlinear Finite Element Methods*, Springer, Berlin/Heidelberg (2008)

[14] MOUKALLED, F., MANGANI, L., and DARWISH, M. The finite volume method. *The Finite Volume Method in Computational Fluid Dynamics*, Springer, Berlin/Heidelberg (2016)

[15] BERKOOZ, G., HOLMES, A. P., and LUMLEY, J. L. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, **25**(1), 539–575 (1993)

[16] COUPLET, M., BASDEVANT, C., and SAGAUT, P. Calibrated reduced-order POD-Galerkin system for fluid flow modelling. *Journal of Computational Physics*, **207**(1), 192–220 (2005)

[17] GUO, M. W. and HESTHAVEN, J. S. Reduced order modeling for nonlinear structural analysis using Gaussian process regression. *Computer Methods in Applied Mechanics and Engineering*, **341**, 807–826 (2018)

[18] ORTALI, G., DEMO, N., and ROZZA, G. Gaussian process approach within a data-driven POD framework for fluid dynamics engineering problems. *arXiv Preprint*, arXiv: 2012.01989 (2020) https://doi.org/10.48550/arXiv.2012.01989

[19] BHATTACHARYA, K., HOSSEINI, B., KOVACHKI, N. B., and STUART, A. M. Model reduction and neural networks for parametric PDEs. *arXiv Preprint*, arXiv: 2005.03180 (2020) https://doi.org/10.48550/arXiv.2005.03180

[20] ZHU, Y. H. and ZABARAS, N. Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, **366**, 415–447 (2018)

[21] KUTYNIOK, G., PETERSEN, P., RASLAN, M., and SCHNEIDER, R. A theoretical analysis of deep neural networks and parametric PDEs. *Constructive Approximation*, **55**(1), 73–125 (2022)

[22] KHOO, Y. H., LU, J. F., and YING, L. X. Solving parametric PDE problems with artificial neural networks. *European Journal of Applied Mathematics*, **32**(3), 421–435 (2021)

[23] SUN, A. Y. Discovering state-parameter mappings in subsurface models using generative adversarial networks. *Geophysical Research Letters*, **45**(20), 11137–11146 (2018)

[24] KADEETHUM, T., O'MALLEY, D., FUHG, J. K., CHOI, Y., LEE, J., VISWANATHAN, H. S., and BOUKLAS, N. A framework for data-driven solution and parameter estimation of PDEs using conditional generative adversarial networks. *Nature Computational Science*, **1**(12), 819–829 (2021)

[25] LI, Z. Y., KOVACHKI, N., AZIZZADENESHELI, K., LIU, B., BHATTACHARYA, K., STUART, A., and ANANDKUMAR, A. Neural operator: graph kernel network for partial differential equations. *arXiv Preprint*, arXiv: 2003.03485 (2020) https://doi.org/10.48550/arXiv.2003.03485

[26] LI, Z. Y., KOVACHKI, N. B., AZIZZADENESHELI, K., LIU, B., BHATTACHARYA, K., STUART, A., and ANANDKUMAR, A. Fourier neural operator for parametric partial differential equations. *arXiv Preprint*, arXiv: 2010.08895 (2020) https://arxiv.org/abs/2010.08895

[27] LIU, H. T., ONG, Y. S., and CAI, J. F. A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. *Structural and Multidisciplinary Optimization*, **57**(1), 393–416 (2018)

[28] FUHG, J. N., FAU, A., and NACKENHORST, U. State-of-the-art and comparative review of adaptive sampling methods for kriging. *Archives of Computational Methods in Engineering*, **28**(4), 2689–2747 (2021)

[29] FUHG, J. N. and FAU, A. A classification-pursuing adaptive approach for Gaussian process regression on unlabeled data. *Mechanical Systems and Signal Processing*, **162**, 107976 (2022)

[30] SCHOBI, R., SUDRET, B., and WIART, J. Polynomial-chaos-based kriging. *arXiv Preprint*, arXiv:1502.03939 (2015) https://doi.org/10.48550/arXiv.1502.03939

[31] WANG, R., KASHINATH, K., MUSTAFA, M., ALBERT, A., and YU, R. Towards physics-informed deep learning for turbulent flow prediction. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Virtual Event, Association for Computing Machinery, 1457–1466 (2020)

[32] MOHAN, A. T., LUBBERS, N., LIVESCU, D., and CHERTKOV, M. Embedding hard physical constraints in neural network coarse-graining of 3D turbulence. *arXiv Preprint*, arXiv:2002.00021 (2020) https://doi.org/10.48550/arXiv.2002.00021

[33] GRIEWANK, A. On automatic differentiation. *Mathematical Programming: Recent Developments and Applications*, Kluwer Academic Publishers, The Netherlands (1989)

[34] LAGARIS, I. E., LIKAS, A., and FOTIADIS, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, **9**(5), 987–1000 (1998)

[35] RAISSI, M., PERDIKARIS, P., and KARNIADAKIS, G. E. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, **378**, 686–707 (2019)

[36] WESSELS, H., WEIßENFELS, C., and WRIGGERS, P. The neural particle method — an updated lagrangian physics informed neural network for computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, **368**, 113127 (2020)

[37] FUHG, J. N., KALOGERIS, I., FAU, A., and BOUKLAS, N. Interval and fuzzy physics-informed neural networks for uncertain fields. *Probabilistic Engineering Mechanics*, **68**, 103240 (2022)

[38] LU, L., JIN, P. Z., and KARNIADAKIS, G. E. Deeponet: learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv Preprint*, arXiv:1910.03193 (2019) https://doi.org/10.48550/arXiv.1910.03193

[39] LU, L., JIN, P. Z., PANG, G. F., ZHANG, Z. Q., and KARNIADAKIS, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, **3**(3), 218–229 (2021)

[40] WANG, S. F., WANG, H. W., and PERDIKARIS, P. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, **7**(40), eabi8605 (2021)

[41] ZHU, Y. H., ZABARAS, N., KOUTSOURELAKIS, P. S., and PERDIKARIS, P. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, **394**, 56–81 (2019)

[42] GAO, H., SUN, L. N., and WANG, J. X. PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *Journal of Computational Physics*, **428**, 110079 (2021)

[43] REN, P., RAO, C. Q., LIU, Y., WANG, J. X., and SUN, H. PhyCRNet: physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. *Computer Methods in Applied Mechanics and Engineering*, **389**, 114399 (2022)

[44] WEINAN, E. and YU, B. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, **6**(1), 1–12 (2018)

[45] LIAO, Y. L. and MING, P. B. Deep Nitsche method: deep Ritz method with essential boundary conditions. *Communications in Computational Physics*, **29**, 1365–1384 (2021)

[46] DUAN, C. G., JIAO, Y. L., LAI, Y. M., LU, X. L., and YANG, Z. J. Convergence rate analysis for deep Ritz method. *Communications in Computational Physics*, **31**(4), 1020–1048 (2022)

[47] SAMANIEGO, E., ANITESCU, C., GOSWAMI, S., NGUYEN-THANH, V. M., GUO, H. W., HAMDIA, K., ZHUANG, X., and RABCZUK, T. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, **362**, 112790 (2020)

[48] FUHG, J. N. and BOUKLAS, N. The mixed deep energy method for resolving concentration features in finite strain hyperelasticity. *Journal of Computational Physics*, **451**, 110839 (2022)

[49] KRISHNAPRIYAN, A., GHOLAMI, A., ZHE, S. D., KIRBY, R., and MAHONEY, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, **34**, 26548–26560 (2021)

[50] WANG, S. F., YU, X. L., and PERDIKARIS, P. When and why pinns fail to train: a neural tangent kernel perspective. *Journal of Computational Physics*, **449**, 110768 (2022)

[51] ANDERSON, I. and DUCHAMP, T. On the existence of global variational principles. *American Journal of Mathematics*, **102**(5), 781–868 (1980)

[52] REDDY, J. N. *Energy Principles and Variational Methods in Applied Mechanics*, John Wiley & Sons, New York (2017)

[53] BEIRO, D. A., VEIGA, L., BREZZI, F., MARINI, L. D., and RUSSO, A. The hitchhiker's guide to the virtual element method. *Mathematical Models and Methods in Applied Sciences*, **24**(8), 1541–1573 (2014)

[54] DOUGLAS, J. Solution of the inverse problem of the calculus of variations. *Proceedings of the National Academy of Sciences*, **25**(12), 631–637 (1940)

[55] TAKENS, F. A global version of the inverse problem of the calculus of variations. *Journal of Differential Geometry*, **14**(4), 543–562 (1979)

[56] ZENKOV, D. V. *The Inverse Problem of the Calculus of Variations: Local and Global Theory and Applications*, Atlantis Press, North Carolina (2015)

[57] WEINSTOCK, R. *Calculus of Variations: with Applications to Physics and Engineering*, Dover Publications, New York (1974)

[58] EVANS, L. C. *Partial Differential Equations*, American Mathematical Society, Washington, D. C. (1998)

[59] ALGUACIL, A., PINTO,W. G., BAUERHEIM, M., JACOB, M. C., and MOREAU, S. Effects of boundary conditions in fully convolutional networks for learning spatiotemporal dynamics. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, The Netherlands, 102–117 (2021)

[60] MASCI, J., BOSCAINI, D., BRONSTEIN, M. M., and VANDERGHEYNST, P. Geodesic convolutional neural networks on Riemannian manifolds. *IEEE International Conference on Computer Vision Workshops*, IEEE, 37–45 (2015)

[61] QI, C. R., LI, Y., HAO, S., and GUIBAS, L. J. Pointnet++: deep hierarchical feature learning on point sets in a metric space. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Perth, 5105–5114 (2017)

[62] JIANG, C. Y., WANG, D. Q., HUANG, J. W., MARCUS, P., and NIEßNER, M. Convolutional neural networks on non-uniform geometrical signals using Euclidean spectral transformation. *International Conference on Learning Representations*-2019, International Society for Machine Learning, New Orleans (2019)

[63] GU, J. X., WANG, Z. H., KUEN, J., MA, L. Y., SHAHROUDY, A., SHUAI, B., LIU, T., WANG, X. X., WANG, L., WANG, G., CAI, J. F., and CHEN, T. Recent advances in convolutional neural networks. *Pattern Recognition*, **77**, 354–377 (2018)

[64] ESTRACH, J. B., SZLAM, A., and LECUN, Y. Signal recovery from pooling representations. *Proceedings of Machine Learning Research*, **32**(2), 307–315 (2014)

[65] HINTON, G. E., SRIVASTAVA, N., KRIZHEVSKY, A., SUTSKEVER, I., and SALAKHUTDINOV, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv Preprint*, arXiv: 1207.0580 (2012) https://doi.org/10.48550/arXiv.1207.0580

[66] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., and SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, **15**(1), 1929–1958 (2014)

[67] IOFFE, S. and SZEGEDY, C. Batch normalization: accelerating deep network training by reducing internal covariate shift. *arXiv Preprint*, arXiv:1502.03167 (2015) https://doi.org/10.48550/arXiv.1502.03167

[68] RONNEBERGER, O., FISCHER, P., and BROX, T. U-net: convolutional networks for biomedical image segmentation. *arXiv Preprint*, arXiv:1505.04597 (2015) https://doi.org/10.48550/arXiv.1505.04597

[69] MAO, X. J., SHEN, C. H., and YANG, Y. B. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *30th Conference on Neural Information Processing Systems*, the Neural Information Processing Systems (NIPS) Foundation, Barcelona (2016)

[70] WANG, W., HUANG, Y., WANG, Y. Z., and WANG, L. Generalized autoencoder: a neural network framework for dimensionality reduction. *27th IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Columbus (2014)

[71] BADRINARAYANAN, V., KENDALL, A., and CIPOLLA, R. SegNet: a deep convolutional encoderdecoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**(12), 2481–2495 (2017)

[72] HINTON, G. E. and SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *Science*, **313**(5786), 504–507 (2006)

[73] ROWEIS, S. and BRODY, C. *Linear Heteroencoders*, Technical report, Gatsby Computational Neuroscience Unit, Alexandra House, London (1999)

[74] BRIDGMAN, W., ZHANG, X., TEICHERT, G., KHALIL, M., GARIKIPATI, K., and JONES, R. A heteroencoder architecture for prediction of failure locations in porous metals using variational inference. *Computer Methods in Applied Mechanics and Engineering*, **398**, 115236 (2022)

[75] KINGMA, D. and BA, J. Adam: a method for stochastic optimization. *arXiv Preprint*, arXiv:1412.6980 (2014) https://doi.org/10.48550/arXiv.1412.6980

[76] MONTGOMERY, D. C., PECK, E. A., and VINING, G. G. *Introduction to Linear Regression Analysis*, John Wiley & Sons, New York (2021)

[77] SEBER, G. A. F. and WILD, C. J. *Nonlinear Regression*, John Wiley & Sons, New York (1990)

[78] LEVEQUE, R. J. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, Society for Industrial and Applied Mathematics, Philadelphia (2007)

[79] HAMEL, C. M., LONG, K. N., and KRAMER, S. L. Calibrating constitutive models with full-field data via physics informed neural networks. *arXiv Preprint*, arXiv: 2203.16577 (2022) https://doi.org/10.48550/arXiv.2203.16577

[80] RITZ, W. *Über eine neue Methode zur Lösung Gewisser Variationsprobleme der Mathematischen Physik*, Walter de Gruyter, New York (1909)

[81] LEISSA, A. W. The historical bases of the Rayleigh and Ritz methods. *Journal of Sound and Vibration*, **287**(4-5), 961–978 (2005)

[82] DAVIS, P. J. and RABINOWITZ, P. *Methods of Numerical Integration*, Dover Publications, New York (2007)

[83] PASZKE, A., GROSS, S., MASSA, F., ADAM LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., DESMAISON, A., KÖPF, A., YANG, E., DEVITO, Z., RAISON, M., TEJANI, A., CHILAMKURTHY, S., STEINER, B., FANG, L., BAI, J., and CHINTALA, S. Pytorch: an imperative style, high-performance deep learning library. *arXiv Preprint*, arXiv:1912.01703 (2019) https://doi.org/10.48550/arXiv.1912.01703

[84] BAI, J. S., RABCZUK, T., GUPTA, A., ALZUBAIDI, L., and GU, Y. T. A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics. *Computational Mechanics*, **71**, 543–562 (2023)

[85] SHI, X. J., CHEN, Z. R., WANG, H., YEUNG, D. Y., WONG, W. K., and WOO, W. C. Convolutional LSTM network: a machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, **28**, 802–810 (2015)

Appendix A

The weights of the simple Simpson's rule are given by

$$\boldsymbol{W} = \frac{1}{3(n-1)} \begin{bmatrix} 1 & 4 & 2 & 4 & \cdots & 4 & 2 & 4 & 1 \end{bmatrix}, \tag{A1}$$

where n is the number of the interpolation points along the boundary.

The weights of the double Simpson's rule read

$$\boldsymbol{W} = \frac{1}{9(N_{\mathrm{DOF}}-1)(N_{\mathrm{DOF}}-1)} \begin{bmatrix} 1 & 4 & 2 & 4 & \cdots & 2 & 4 & 1 \\ 4 & 16 & 8 & 16 & \cdots & 8 & 16 & 4 \\ 2 & 8 & 4 & 8 & \cdots & 4 & 8 & 2 \\ 4 & 16 & 8 & 16 & \cdots & 8 & 16 & 4 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 2 & 8 & 4 & 8 & \cdots & 4 & 8 & 2 \\ 4 & 16 & 8 & 16 & \cdots & 8 & 16 & 4 \\ 1 & 4 & 2 & 4 & \cdots & 2 & 4 & 1 \end{bmatrix}, \tag{A2}$$

where N_{DOF} is the number of degrees of freedom in the x- and y-directions.