DAS: A DEEP ADAPTIVE SAMPLING METHOD FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS
KEJUN TANG∗, XIAOLIANG WAN†, AND CHAO YANG‡

Abstract. In this work we propose a deep adaptive sampling (DAS) method for solving partial differential equations (PDEs), where deep neural networks are utilized to approximate the solutions of PDEs and deep generative models are employed to generate new collocation points that refine the training set. The overall procedure of DAS consists of two components: solving the PDEs by minimizing the residual loss on the collocation points in the training set and generating a new training set to further improve the accuracy of the current approximate solution. In particular, we treat the residual as a probability density function and approximate it with a deep generative model, called KRnet. The new samples from KRnet are consistent with the distribution induced by the residual, i.e., more samples are located in the region of large residual and less samples are located in the region of small residual. Analogous to classical adaptive methods such as the adaptive finite element, KRnet acts as an error indicator that guides the refinement of the training set. Compared to the neural network approximation obtained with uniformly distributed collocation points, the developed algorithms can significantly improve the accuracy, especially for low regularity and high-dimensional problems. We present a theoretical analysis to show that the proposed DAS method can reduce the error bound and demonstrate its effectiveness with numerical experiments.

Key words. deep learning; PDEs; adaptive sampling; deep generative models.

1. Introduction. In recent years, solving partial differential equations (PDEs) with deep learning methods is receiving increasingly more attention [14, 43, 18]. Two major types of deep learning methods have been proposed for solving PDEs, including the variational form using deep learning techniques [10, 19, 46, 20] and the physical informed neural networks (PINNs) [37, 32, 31, 18], both of which reformulate a PDE problem as an optimization problem and train a deep neural network (DNN) to approximate the solution of PDE through minimizing the corresponding loss functional. The variational form is based on the weak formulation of PDEs, while the physical informed neural networks are based on the residual loss of PDEs. Similar idea of solving PDEs via minimizing the residual loss can be traced back to the works [23, 8] in the 1990’s, where a shallow neural network is optimized on a priori fixed mesh as an approximation of the solution. Some efforts have been made to incorporate traditional computational techniques to enhance the performance of solving PDEs with deep neural networks. In [25, 26, 9, 12, 20], deep neural networks based on domain decomposition are proposed to improve the efficiency. A penalty free neural network method [35] and Phygeonet [12] are developed to deal with complex geometries and irregular domains. A weak formulation with primal and adversarial networks is proposed in [45], where the PDE problem is converted to an operator norm minimization problem induced by the weak formulation.

One critical step for all these methods is to approximate the loss functional, where the integral is usually approximated by the Monte Carlo method with collocation points randomly generated by a uniform distribution on the computational domain. Since the minimization of the discrete loss functional yields the approximate solution, the accuracy of the approximate solution is closely related to the accuracy of the discrete loss functional. In contrast to classical computational methods, where the main concern is the approximation error, one needs to balance the approximation error and the generalization error for the neural network approximation, where the approximation error mainly originates from the model capability of the neural network and the generalization error is mainly related to the data points in the training set, i.e., the samples for the discretization of the loss functional. However, in many PDE models, the uniform random sampling strategy is not efficient especially when the PDE solution has a low regularity, in other words, an integrand of low regularity may have a large variance in terms of a uniform distribution. This issue becomes worse for high-

∗Peng Cheng Laboratory, Shenzhen 518052, China (tangkj@pcl.ac.cn).
†Department of Mathematics, Center for Computation and Technology, Louisiana State University, Baton Rouge 70803 (xlwan@math.lsu.edu)
‡School of Mathematical Sciences, Peking University, Beijing 100871, China (chao.yang@pku.edu.cn)
dimensional problems due to the curse of dimensionality. In high-dimensional spaces, most of the volume of the computational domain concentrates around its surface \cite{1, 40, 44}, which means that uniform samples become less effective for training deep neural networks to approximate the solution of high-dimensional PDEs. For example, the collocation points from the uniform distribution are not suitable for solving the Fokker-Planck equation with high-dimensional spatial variables, while an adaptive strategy through sampling the current approximate solution is effective \cite{39}. In \cite{13}, a selection network is introduced to serve as a weight function to assign higher weights for samples with large point-wise residual, which yields a more accurate solution if the selection network is properly chosen. However, to obtain a valid selection network, one needs to impose additional constraints on the selection network, which is often a non-trivial task. For low-dimensional problems, it is well known that one can employ adaptive numerical schemes to deal with PDEs with low regularity solutions \cite{30, 28, 11}, which also suggests that the uniform samples are not the best choice. Therefore, adaptive sampling strategies are crucial for developing more efficient and reliable deep learning techniques for the approximation of PDEs.

In this work, we develop a deep adaptive sampling method (DAS) for the neural network approximation of PDEs based on residual minimization, where a deep generative model, called KRnet \cite{38, 42, 41}, is used to guide the sample generation for the training set. To this end, we need to construct two deep neural network models: one for approximating the solution and the other for refining the training set. The neural network approximation is achieved by the standard procedure of residual minimization. KRnet defines a transport map \cite{34} from the data distribution to a prior one (e.g. the standard Gaussian). KRnet retains two traits of flow-based generative models \cite{7, 22}: exact invertibility of the transport map and efficient computation of the Jacobian determinant, based on which one can obtain an explicit density model using the change of variables and an effective approach for generating samples through the invertible mapping. The key point in our proposed framework is that the residual is viewed as a probability density function (PDF) up to a constant and approximating this PDF can be achieved by minimizing the Kullback-Leibler (KL) divergence between the KRnet-induced density model and the residual-induced distribution. We then use KRnet to generate new collocation points, where more points are put in the region of large residual and less points are put in the region of small residual. The new collocation points can be used to further improve the accuracy of the current approximate solution since they are distributed consistently with the current residual. Simply speaking, KRnet acts as an error indicator for the refinement of the training set, which shares similarities with the classical adaptive finite element method subject to a residual-based posteriori error estimator.

The remainder of the paper is organized as follows. In the next section, we briefly describe the deep learning method used in this work for the approximation of PDEs. Our DAS approach is presented in section 3. We provide the theoretical analysis of DAS in section 4. In section 5, we demonstrate the efficiency of our adaptive sampling approach with numerical experiments. The paper is concluded in section 6.

2. Deep learning for PDEs. Let $\Omega \subset \mathbb{R}^d$ be a spatial domain, which is bounded, connected and with a polygonal boundary $\partial \Omega$, and $\mathbf{x} \in \mathbb{R}^d$ denote a spatial variable. The PDE problem is stated as: find $u(\mathbf{x}) \in F : \mathbb{R}^d \mapsto \mathbb{R}$ where $F$ is a proper function space defined on $\Omega$, such that

\[
L u(\mathbf{x}) = s(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega,
\]

\[
b u(\mathbf{x}) = g(\mathbf{x}) \quad \forall \mathbf{x} \in \partial \Omega, \tag{2.1}
\]

where $L$ is the partial differential operator, $b$ is the boundary operator, $s(\mathbf{x})$ is the source function, and $g(\mathbf{x})$ represents the boundary conditions.

Let $u(\mathbf{x}; \Theta)$ be a neural network with parameters $\Theta$. The goal is to use $u(\mathbf{x}; \Theta)$ to approximate the solution $u(\mathbf{x})$ through optimizing a loss functional defined as

\[
J(u(\mathbf{x}; \Theta)) = \| r(\mathbf{x}; \Theta) \|_{L^2, \Omega}^2 + \gamma \| b(\mathbf{x}; \Theta) \|_{L^2, \partial \Omega}^2 = J_r(u(\mathbf{x}; \Theta)) + \gamma J_b(u(\mathbf{x}; \Theta)), \tag{2.2}
\]
where $r(x; \Theta) = Lu(x; \Theta) - s(x)$, and $b(x; \Theta) = bu(x; \Theta) - g(x)$ measure how well $u(x; \Theta)$ satisfies the partial differential and boundary operators, respectively, and $\gamma > 0$ is a penalty parameter. Here, $\|u(x)\|^2_{L^2(\Omega)} = \int_{\Omega} |u(x)|^2 dx$ and $\|u(x)\|^2_{L^2(\partial\Omega)} = \int_{\partial\Omega} |u(x)|^2 dx$. The loss functional (2.2) is usually discretized numerically before the optimization with respect to $\Theta$ is addressed. In practice, one often chooses two sets of uniformly distributed collocation points $S_\Omega = \{x^{(i)}_\Omega\}_{i=1}^{N_\Omega}$ and $S_{\partial\Omega} = \{x^{(i)}_{\partial\Omega}\}_{i=1}^{N_{\partial\Omega}}$ respectively for the discretization of the two terms in the objective functional (2.2), leading to the following empirical loss

$$J_N(u(x; \Theta)) = \|r(x; \Theta)\|_{N_r, S_\Omega}^2 + \gamma \|b(x; \Theta)\|_{N_b, S_{\partial\Omega}}^2,$$  

(2.3)

where $N = (N_r, N_b)$, $\gamma > 0$, and

$$\|u(x)\|_{N_r, S_\Omega} = \left(\frac{1}{N_r} \sum_{i=1}^{N_r} u^2(x^{(i)}_\Omega)\right)^{\frac{1}{2}},$$

$$\|u(x)\|_{N_b, S_{\partial\Omega}} = \left(\frac{1}{N_b} \sum_{i=1}^{N_b} u^2(x^{(i)}_{\partial\Omega})\right)^{\frac{1}{2}}.$$  

Note that in the definition of $J_N$ we do not take into account the constants $|\Omega| = \int_{\Omega} dx$ and $|\partial\Omega| = \int_{\partial\Omega} dx$ and the ratio induced by these two constants can be dealt with by choosing $\hat{\gamma} = \frac{|\partial\Omega|}{|\Omega|}$ such that $J_N(u)$ is a Monte Carlo approximation of $J(u)$ up to a constant scaling factor $|\Omega|$. We then seek an approximate solution by minimizing the empirical loss (2.3), i.e.,

$$\min_{\hat{\Theta}} J_N(u(x; \Theta)),$$  

(2.4)

which can be solved by stochastic gradient-based methods [3, 21].

Recently, some prior error estimates of neural-network-based methods for solving PDEs are established. Combining the analysis technique of the least square finite element method [2] with the universal approximation property of neural networks [5, 16, 15, 24], Shin et. al. propose an abstract framework for the error estimation of physical informed neural networks [36]. Lu et. al. derive a prior estimate of the generalization error for the deep Ritz method with two-layer neural networks [27]. In this work, we focus on the method of residual minimization. Our approach can be extended to other cases. Suppose that $u(x; \Theta_N^*)$ is the minimizer of the empirical loss $J_N(u(x; \Theta))$ and $u(x; \Theta^*)$ is the minimizer of $J(u(x; \Theta))$, i.e.,

$$u(x; \Theta^*) = \arg \min_{\Theta} J(u(x; \Theta)),$$

$$u(x; \Theta_N^*) = \arg \min_{\Theta} J_N(u(x; \Theta)).$$

We have

$$u(x; \Theta_N^*) - u(x) = u(x; \Theta_N^*) - u(x; \Theta^*) + u(x; \Theta^*) - u(x),$$  

(2.5)

i.e.,

$$\mathbb{E} \|u(x; \Theta_N^*) - u(x)\|_\Omega \leq \mathbb{E} \|u(x; \Theta_N^*) - u(x; \Theta^*)\|_\Omega + \|u(x; \Theta^*) - u(x)\|_\Omega,$$  

(2.6)

where $\mathbb{E}$ indicates the expectation and the norm $\|\cdot\|_\Omega$ corresponds to the function space $F$ for $u(x; \Theta)$. The first term describes the statistical error from discretizing the loss functional with the Monte Carlo approximation, and the second term is the approximation error of minimizing the loss functional over the hypothesis space. The approximation error depends on the capability of neural networks, while the statistical error depends on the definition of $S_\Omega$ and $S_{\partial\Omega}$. In this work, we focus on how to reduce the statistical error. More specifically, we intend to develop an adaptive sampling approach to refine the training set and the approximate solution alternatively. For simplicity, we focus on the integration of the residual $r(x; \Theta)$ and assume that the integral on the boundary is well approximated by a prescribed $S_{\partial\Omega}$.  

3
3. Deep adaptive sampling method. Our deep adaptive sampling method, i.e., the DAS method, will be established from the viewpoint of importance sampling.

3.1. Some ideas on variance reduction. We outline our basic ideas on variance reduction in this section and more details about the algorithm will be presented later. To improve the Monte Carlo approximation of the objective functional $J(u)$, we need to reduce the variance. We here consider the importance sampling technique. For simplicity, we only consider $J_r(u(x;\Theta))$ in equation (2.2), which is the loss induced by the residual. We have

$$J_r(u(x;\Theta)) = \int_\Omega r^2(x;\Theta)dx = \int_\Omega \frac{r^2(x;\Theta)}{p(x)}p(x)dx \approx \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(x_{i\Omega};\Theta)}{p(x_{i\Omega})}, \quad (3.1)$$

where the set $\{x_{i\Omega}\}_{i=1}^{N_r}$ is generated with respect to the probability density function (PDF) $p(x)$ instead of a uniform distribution as in equation (2.3). If the variance of $r^2(X)p^{-1}(X)$ in terms of $p(x)$ is smaller than the variance of $r^2(X)$ in terms of a uniform distribution, the accuracy of the Monte Carlo approximation will be improved for a fixed sample size $N_r$. The optimal choice for $p(x)$ is

$$p(x) = \frac{r^2(x;\Theta)}{\mu}, \quad (3.2)$$

where $\mu = \int_\Omega r^2(x;\Theta)dx$. Although the optimal choice is useless in practice since $\mu$ is the quantity to be computed, it suggests that variance reduction can be achieved if $p(x)$ is close to the residual-induced distribution.

Another option for variance reduction is to relax the definition of $J_r(u)$ as:

$$J_{r,p}(u(x;\Theta)) = \int_\Omega r^2(x;\Theta)p(x)dx \approx \frac{1}{N_r} \sum_{i=1}^{N_r} r^2(x_{i\Omega};\Theta), \quad (3.3)$$

where the set $\{x_{i\Omega}\}_{i=1}^{N_r}$ is sampled from the PDF $p(x)$ and $p(x) > 0$ on $\Omega$. It is seen that the minimizer of $J_{r,p}(u(x;\Theta))$ is also the solution of problem (2.1), if we assume that the minimization is with respect to $u(x;\Theta)$ that satisfies the boundary conditions exactly. To reduce the error induced by the Monte Carlo approximation, we may adjust $p(x)$ to make the minimizer of the discretized $J_{r,p}$ correspond to a residual $r^2(x;\Theta)$ that is nearly a constant. This option is similar to the classical adaptive finite element method, where mesh refinement/coarsening is supposed to make the approximation error nearly uniform. For our case, we may again sample the PDF that is close to the residual-induced distribution and add more samples from the region of large residual into the training set. After several refinement, we end up with a training set whose underlying distribution $p(x)$ is nonuniform similar to the final nonuniform mesh of the adaptive finite element method.

To make the above ideas practical, the key issue is to generate samples efficiently from a PDF $p(x) \approx \mu^{-1}r^2(x;\Theta)$ for a fixed $\Theta$. To achieve this, we employ a recently developed deep generative model called KRnet for probability approximation and sample generation.

3.2. KRnet for PDF approximation. In this section, we briefly review KRnet which is a flow-based generative model for density estimation or approximation [39, 38]. Let $X \in \mathbb{R}^d$ be a random vector associated with a given data set, and its PDF is denoted by $p_X(x)$. The target is to estimate $p_X(x)$ using available data. Let $Z \in \mathbb{R}^d$ be a random vector associated with a PDF $p_Z(z)$, where $p_Z(z)$ is a prior distribution (e.g., Gaussian distributions). The flow-based generative modeling is to seek an invertible mapping $z = f(x)$ [7]. By the change of variables, we have the PDF of $X = f^{-1}(Z)$ as

$$p_X(x) = p_Z(f(x)) |\det \nabla_x f|, \quad (3.4)$$
Once the prior distribution \(p_Z(z)\) is specified, equation (3.4) provides an explicit density model for \(X\). The inverse of \(f(\cdot)\) provides a convenient way to sample \(X\) as \(X = f^{-1}(Z)\). The basic idea of KRnet is to define the structure of \(f(x)\) in terms of the Knothe-Rosenblatt rearrangement. Let \(\mu_Z\) and \(\mu_X\) be the probability measures of two random variables \(Z, X \in \mathbb{R}^d\) respectively. A mapping \(T: Z \mapsto X\) is called a transport map such that \(T\# \mu_Z = \mu_X\), where \(T\# \mu_Z\) is the push-forward of \(\mu_Z\) such that \(\mu_X(B) = \mu_Z(T^{-1}(B))\) for every Borel set \(B\) [4]. The transport map \(T\) given by the Knothe-Rosenblatt rearrangement has a lower-triangular structure

\[
z = T^{-1}(x) = \begin{bmatrix}
T_1(x_1) \\
T_2(x_1, x_2) \\
\vdots \\
T_d(x_1, \ldots, x_d)
\end{bmatrix}.
\]  

(3.5)

Let \(x = [x_1, \ldots, x_K]^T\) be a partition of \(x\), where \(x_i = [x_{i,1}, \ldots, x_{i,m}]^T\) with \(1 \leq K \leq d, 1 \leq m \leq d\), and \(\sum_{i=1}^{K} \dim(x_i) = d\). The KRnet takes an overall form

\[
z = f_{\text{KRnet}}(x) = \begin{bmatrix}
f_1(x_1) \\
f_2(x_1, x_2) \\
\vdots \\
f_K(x_1, \ldots, x_K)
\end{bmatrix},
\]  

(3.6)

where the number of active dimensions reduces from \(f_K(\cdot)\) to \(f_1(\cdot)\) according to the partition of \(x\). KRnet consists of one outer loop and \(K - 1\) inner loops which is illustrated in Figure 3.1. The outer loop has \(K - 1\) stages, corresponding to the \(K - 1\) mappings \(f_i\) in equation (3.6) with \(i = 2, \ldots, K\), and for each stage, an inner loop of \(L\) affine coupling layers (see equation (3.10)) is defined. More specifically, we have

\[
z = f_{\text{KRnet}}(x) = L_N \circ f_{[K-1]}^{\text{outer}} \circ \cdots \circ f_{[1]}^{\text{outer}}(x),
\]  

(3.7)

and

\[
p_{\text{KRnet}}(x) = p_Z(f_{\text{KRnet}}(x)) |\det \nabla_x f_{\text{KRnet}}|,
\]  

(3.8)

where \(f_{[i]}^{\text{outer}}\) is defined as

\[
f_{[i]}^{\text{outer}} = L_S \circ f_{[K,L]}^{\text{inner}} \circ \cdots \circ f_{[1,K]}^{\text{inner}} \circ L_R.
\]  

(3.9)

Here \(f_{[k,i]}^{\text{inner}}\) indicates a combination of one affine coupling layer and one scale and bias layer, and \(L_N, L_S\) and \(L_R\) indicate the nonlinear layer, the squeezing layer and the rotation layer, respectively, which are briefly reviewed as follows. More details can be found in [38, 39].

**Affine coupling layer** is a simple bijection construted by a shallow neural network. Let \(x = [x_1, x_2]^T\) be a partition of \(x\) with \(x_1 \in \mathbb{R}^{m}\) and \(x_2 \in \mathbb{R}^{d-m}\). The affine coupling layer is defined as follows [38]

\[
\begin{align*}
\hat{x}_1 &= x_1 \\
\hat{x}_2 &= x_2 \circ (1 + \alpha \tanh(s(x_1))) + e^\beta \circ \tanh(t(x_1)),
\end{align*}
\]  

(3.10)

where \(0 < \alpha < 1\) is a hyperparameter and the parameter \(\beta \in \mathbb{R}^{d-m}\) is trainable. This way, the Jacobian matrix is lower-triangular whose determinant can be evaluated efficiently. Furthermore, \((s, t)\) is usually modeled by a neural network \(\text{NN}\)

\[
(s, t) = \text{NN}(x_1).
\]  

(3.11)
In our numerical experiments, we set $\alpha = 0.6$. Since the affine coupling layer only updates one part of $x$, another affine coupling layer is needed for a complete update. In other words, another affine coupling layer can be used to update $\tilde{x}$ as

\[
\tilde{x}_1 = \tilde{x}_1 \odot (1 + \alpha \tanh(\tilde{s}(\tilde{x}_2))) + e^\beta \odot \tanh(\tilde{t}(\tilde{x}_2)) + e^\gamma \odot \tanh(\tilde{t}(\tilde{x}_2)) + e^\gamma \\
\tilde{x}_2 = \tilde{x}_2,
\]

where the components $\tilde{x}_1$ are updated and $\tilde{x}_2$ remains unchanged.

*Squeezing layer* $L_S$ is used to deactivate some dimensions using a mask. The squeezing layer is defined as follows

\[
q = \begin{bmatrix} 1, \ldots, 1, 0, \ldots, 0 \end{bmatrix}^T,
\]

(3.12)

where the components $q \odot x$ will keep being updated and the rest components $(1 - q) \odot x$ will be fixed from then on.

*Scale and bias layer* provides a simplification of the batch normalization [17], which is defined as

\[
\hat{x} = a \odot x + b,
\]

(3.13)

where $a$ and $b$ are trainable and initialized by the mean and standard deviation of the data. After the initialization, $a$ and $b$ will be treated as regular trainable parameters that are independent of the data.

*Rotation layer* $L_R$ defines a linear mapping of the input $x$

\[
\hat{x} = \hat{W}x.
\]
through a trainable matrix
\[
\hat{W} = \begin{bmatrix} W & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} L & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} U & 0 \\ 0 & I \end{bmatrix},
\]
where \( W \in \mathbb{R}^{n \times n}, n \) is defined in the mask \( q, I \in \mathbb{R}^{(d-n) \times (d-n)} \) is an identity matrix, and \( W = LU \) is the LU factorization of \( W \). Entries below the main diagonal of \( L \) and entries in the upper triangle of \( U \) are trainable. In practice, we simply optimize the trainable entries of \( L \) and \( U \) without enforcing the orthonormality of \( \hat{W} \) and such a simplification works well.

**Nonlinear layer** \( L_N \) provides a component-wise nonlinear transformation. For simplicity, we only consider one component \( x \) of the data. We start with a nonlinear mapping \( F(s) : [0, 1] \rightarrow [0, 1] \):
\[
F(s) = \int_0^s p(t)dt,
\]
where \( p(s) \) can be regarded as a probability density function. Let \( 0 = s_0 < s_1 < \ldots < s_{n+1} = 1 \) be a mesh of the interval \([0, 1]\) with element size \( h_i = s_{i+1} - s_i \). Define \( p(s) \) as a piece-wise linear polynomial
\[
p(s) = \frac{w_{i+1} - w_i}{h_i} (s - s_i) + w_i, \quad \forall s \in [s_i, s_{i+1}].
\]
Then \( F(s) \), corresponding to a cumulative density function, is a quadratic function
\[
F(s) = \frac{w_{i+1} - w_i}{2h_i} (s - s_i)^2 + w_i (s - s_i) + \sum_{k=0}^{i-1} \frac{w_k + w_{k+1}}{2} h_i, \quad \forall s \in [s_i, s_{i+1}],
\]
whose inverse and derivative can be explicitly computed. We decompose \((-\infty, \infty) = (-\infty, -a) \cup [-a, a] \cup (a, \infty) \) with \( a > 0 \), and define the following nonlinear mapping from \( \mathbb{R} \) to \( \mathbb{R} \)
\[
\hat{F}(x) = \begin{cases} 
\beta_s(x + a) - a, & x \in (-\infty, -a) \\
2aF\left(\frac{x + a}{2a}\right) - a, & x \in [-a, a] \\
\beta_s(x - a) + a, & x \in (a, \infty),
\end{cases}
\]
where \( \beta_s > 0 \) is a scaling factor. A nonlinear mapping is considered for the data located in \([-a, a]\) and \( \hat{F}(x) \) maps \([-a, a]\) to itself. On \((-\infty, -a) \cup (a, \infty) \), \( \hat{F}(x) \) is simply a linear mapping.

**3.3. A deep generative model for the residual-induced distribution.** Let \( p_{\text{KRnet}}(x; \Theta_f) \) be the density model induced by \( \text{KRnet} \) \( f_{\text{KRnet}}(\cdot; \Theta_f) \) (see equation (3.8)), where \( \Theta_f \) includes the model parameters. We intend to use \( p_{\text{KRnet}}(x; \Theta_f) \) to approximate the PDF induced by the residual as discussed in section 3.1. However, the support of \( p_{\text{KRnet}}(x) \) is \( \mathbb{R}^d \) while the computation domain, say \( \Omega = [-1, 1]^d \), is usually compact. We need to deal with such a discrepancy. Let \( B = (-1 + \delta, 1 + \delta)^d \) with \( 0 < \delta < \infty \) such that \( \Omega \subset B \). For each dimension of \( x \), we define the following logistic mapping
\[
y = \ell(x) = \frac{s}{2} \log \frac{x + (1 + \delta)}{1 + \delta} - x,
\]
with \( s > 0 \) being a scale parameter, which defines a one-to-one correspondence between \( x \in (-1 + \delta, 1 + \delta) \) and \( y \in (-\infty, +\infty) \). Let \( \ell(x) : B \rightarrow \mathbb{R}^d \) be a \( d \)-dimensional mapping such that
\[
\ell_i(x_i) = \ell(x_i), \quad i = 1, \ldots, d.
\]
Then the following invertible mapping
\[
z = f_{\text{KRnet}} \circ \ell(x) \tag{3.15}
\]
defines a PDF
\[ \hat{p}_{\text{KRnet}}(x; \Theta_f) = p_{\text{KRnet}}(\ell(x); \Theta_f) | \nabla_x \ell(x) |, \] (3.16)
where the support of \( \hat{p}_{\text{KRnet}}(x; \Theta_f) \) is \( B \).

We now consider a modification of \( r^2(x; \Theta) \). Define a cutoff function as
\[ h(x) = \begin{cases} 1, & x \in \Omega, \\ \prod_{i=1}^d h_\delta(x_i), & x \in B \setminus \Omega, \end{cases} \]
where \( h_\delta(x) \) is a piecewise linear function
\[ h_\delta(x) = \begin{cases} 1, & x \in [-1, 1], \\ \delta^{-1}(x + 1 + \delta), & x \in (-1 - \delta, -1), \\ \delta^{-1}(1 + \delta - x), & x \in (1, 1 + \delta), \\ 0, & x \in (-\infty, -1 - \delta] \cup [1 + \delta, \infty). \end{cases} \]

We consider a modified PDF for any \( \Theta \)
\[ \hat{r}_X(x) \propto r^2(x; \Theta) h(x). \] (3.17)
Note that both \( \hat{r}_X(x) \) and \( \hat{p}_{\text{KRnet}}(x; \Theta_f) \) have the support \( B \). We then solve the following optimization problem
\[ \Theta_f^* = \arg \min_{\Theta_f} D_{\text{KL}}(\hat{r}_X(x) || \hat{p}_{\text{KRnet}}(x; \Theta_f)), \] (3.18)
where \( D_{\text{KL}}(\cdot || \cdot) \) indicates the Kullback-Leibler (KL) divergence between two distributions. We use
\[ p_X(x) \propto \hat{p}_{\text{KRnet}, \Theta_f^*}(x) I_\Omega(x) \] (3.19)
as an approximation of the PDF induced by \( r^2(x; \Theta) \), where \( I_\Omega(x) \) is an indicator function such that \( I_\Omega(x) = 1 \) if \( x \in \Omega \); 0, otherwise. If \( \delta \ll 1 \), most the samples \( \ell^{-1} \circ f_{\text{KRnet}}^{-1}(z^{(i)}; \Theta_f^*) \) will be located in \( \Omega \). Since \( \hat{r}_X(x) \propto r^2(x; \Theta) \) on \( \Omega \), \( p_X(x) \) approximates the \( r^2(x; \Theta) \)-induced PDF well when \( \delta \) is small. In our numerical experiments, we set \( \delta = 0.01 \) and \( s = 2 \).

We now look at the approximation of \( \Theta_f^* \). The KL divergence in the optimization problem (3.18) can be written as
\[ D_{\text{KL}}(\hat{r}_X(x) || \hat{p}_{\text{KRnet}}(x; \Theta_f)) = \int_B \hat{r}_X \log \hat{r}_X dx - \int_B \hat{r}_X \log \hat{p}_{\text{KRnet}} dx. \] (3.20)
The first term on the right-hand side corresponds to the differential entropy of \( \hat{r}_X \), which does not affect the optimization with respect to \( \Theta_f \). So minimizing the KL divergence is equivalent to minimizing the cross entropy between \( \hat{r}_X \) and \( \hat{p}_{\text{KRnet}} \) [6, 33]:
\[ H(\hat{r}_X, \hat{p}_{\text{KRnet}}) = - \int_B \hat{r}_X \log \hat{p}_{\text{KRnet}} dx. \] (3.21)
Since the samples from \( \hat{r}_X \) are not available, we approximate the cross entropy using the importance sampling technique:
\[ H(\hat{r}_X, \hat{p}_{\text{KRnet}}) \approx - \frac{1}{N} \sum_{i=1}^{N} \frac{\hat{r}_X(x_B^{(i)})}{\hat{p}_{\text{KRnet}}(x_B^{(i)}; \Theta_f)} \log \hat{p}_{\text{KRnet}}(x_B^{(i)}; \Theta_f), \] (3.22)
where the samples \( \{ x_B^{(i)} \}_{i=1}^{N} \) of \( \hat{p}_{\text{KRnet}} \) can be generated efficiently by the KRnet, i.e.,
\[ x_B^{(i)} = \ell^{-1} \circ f_{\text{KRnet}}^{-1}(z^{(i)}) \] (3.23)
with $z^{(i)}$ being sampled from a standard Gaussian. We then minimize the discretized cross entropy (3.22) to obtain an approximation of $\Theta_f^*$.  

**Remark 3.1.** An alternative approach for the approximation of $\hat{r}_X(x)$ is to minimize the following KL divergence:

$$D_{KL}(\hat{p}_{\text{KRnet}}(x; \Theta_f) \| \hat{r}_X(x)) = \int_B \hat{p}_{\text{KRnet}}(x; \Theta_f) \log \hat{p}_{\text{KRnet}}(x; \Theta_f) dx - \int_B \hat{p}_{\text{KRnet}}(x; \Theta_f) \log \hat{r}_X(x) dx, \tag{3.24}$$

which can be approximated by samples from $\hat{p}_{\text{KRnet}}(x; \Theta_f)$. Note that the KL divergence is asymmetric. Minimizing the KL divergence (3.20) is not equivalent to the minimization of the KL divergence (3.24), although both minimizers will be achieved at $\hat{p}_{\text{KRnet}}(x; \Theta_f) = \hat{r}_X(x)$ if $\hat{r}_X$ can be reached exactly by a certain parameter $\Theta_f$.

### 3.4. Adaptive sampling procedure.
We are now ready to present our algorithms. In this work, we mainly focus on the adaptivity of $S_{\Theta}$ for simplicity. The key step of our adaptivity strategy is to refine the training set $S_{\Theta}$, and we provide two options to do it, corresponding to two algorithms.

I. We replace the whole current set of collocation points using new samples from the probability measure for importance sampling. This corresponds to equation (3.1).

II. We gradually add more collocation points to the current training set. This corresponds to equation (3.3), where the new samples are mainly from the region of large residual.

We first present strategy I. Let $S_{\Theta,0} = \{x^{(i)}_{\Theta,0}\}_{i=1}^{N_r}$ and $S_{\partial \Theta,0}$ be two sets of collocation points that are uniformly sampled from $\partial \Omega$ and $\partial \Omega$ respectively. Using $S_{\Theta,0}$ and $S_{\partial \Theta,0}$, we minimize the empirical loss (2.3) to obtain $u(x; \Theta_{N}^*)$. With $u(x; \Theta_{N}^*)$, we minimize the cross entropy (3.22) to get $\hat{p}_{\text{KRnet}}(x; \Theta_f^*)$. To refine the training set, a new set $S_{\Theta,1} = \{x^{(i)}_{\Theta,1}\}_{i=1}^{N_r}$ is generated by $\ell^{-1} \circ f^{-1}_{\text{KRnet}}(z^{(i)}; \Theta_f^*)$ (see equation (3.15)). Then we continue to update the approximate solution $u(x; \Theta_{N}^*)$ using $S_{\Theta,1}$ as the training set. In general, we use $S_{\Theta,k} = \{x^{(i)}_{\Theta,k}\}_{i=1}^{N_r}$ to obtain $u(x; \Theta_{N}^*)$ as

$$\Theta_{N}^{*,(k+1)} = \arg \min_{\Theta} J_N^S(u(x; \Theta)), \tag{3.25}$$

where $u(x; \Theta)$ is initialized as $u(x; \Theta_{N}^{*,(k)})$ and $J_N^S$ is defined as

$$J_N^S(u(x; \Theta)) = \frac{1}{N_r} \sum_{i=1}^{N_r} r^2(x^{(i)}_{\Theta_{N}^*,(k)}; \Theta) + \frac{1}{N_b} \sum_{i=1}^{N_b} b^2(x^{(i)}_{\partial \Theta_{N}^*,(k)}; \Theta). \tag{3.26}$$

Starting with $\hat{p}_{\text{KRnet}}(x; \Theta_f^{*,(k)})$, the density model $\hat{p}_{\text{KRnet}}(x; \Theta_f)$ is updated as

$$\Theta_f^{*,(k+1)} = \arg \min_{\Theta_f} -\frac{1}{N_r} \sum_{i=1}^{N_r} r^2(x^{(i)}_{\Theta_f^*,(k+1)}; \Theta_f^{*,(k+1)}) \log \hat{p}_{\text{KRnet}}(x^{(i)}_{\Theta_f^*,(k+1)}; \Theta_f), \tag{3.27}$$

where $x^{(i)}_{\Theta_f^*,(k+1)}$ are obtained as in equation (3.23). A new set $S_{\Theta,k+1} = \{x^{(i)}_{\Theta,k+1}\}_{i=1}^{N_r}$ of collocation points is then generated. As detailed in section 3.3, the support of data points generated by KRnet is $B = (-1 + \delta, 1 + \delta)^d$, while the computation domain is $\Omega = [-1, 1]^d$. So we need to deal with the collocation points located in $B \setminus \Omega$. Instead of neglecting these points, we project them onto $\partial \Omega$.

We define an entry-wise projection operator $P(x) : B \rightarrow \Omega$ as

$$P(x_i) = \begin{cases} -1, & \text{if } x_i < -1, \\ x_i, & \text{if } 1 \geq x_i \geq -1, \quad i = 1, \ldots, d, \\ 1, & \text{if } x_i > 1. \end{cases} \tag{3.27}$$
For a sequence of i.i.d. samples \( z^{(j)} \) generated from the standard Gaussian with \( j = 1, 2, \ldots \), we compute \( x_B^{(j)} = \mathcal{P}^{-1}(\mathcal{R}^{-1}(z^{(j)})) \). If \( x_B = \mathcal{P}(x_B) \), we assign \( x_B^{(j)} \) to \( S_{\Omega,k+1} \); otherwise, we add \( \mathcal{P}(x_B^{(j)}) \) to \( S_{\partial \Omega,k} \). The updated training set \( S_{\Omega,k+1} \) and \( S_{\partial \Omega,k+1} \) will be used for the next training stage. This procedure is repeated until the stopping criterion is satisfied (see Algorithm 1). Since the collocation points in \( S_{\Omega,k} \) will be completely replaced at the next stage, we call this type of deep adaptive sampling strategy DAS-R for short.

We now look at strategy II. Unlike DAS-R, the number of collocation points in the training set \( S_{\Omega} \) increases gradually. So we denote this type of deep adaptive sampling strategy by DAS-G for short. Starting with an initial set of collocation points \( S_{\Omega,0} = \{ x_{\Omega,0}^{(i)} \}^{n_0}_{i=1} \) (as well as \( S_{\partial \Omega,0} \)) drawn from a uniform distribution defined on \( \Omega \), we minimize the empirical loss (2.3) on the training set \( S_{\Omega} \). Our two adaptive training methods are summarized in Algorithm 1 (DAS-R) and Algorithm 2 (DAS-G), where \( N_{\text{adaptive}} \) is a given number of maximum adaptivity iterations, \( m \) is the batch size for stochastic gradient, and \( N_e \) is the number of epochs for training \( u(x; \Theta) \) and \( \hat{p}_{\text{KRnet}}(x; \Theta_f) \).

The algorithms consist of three steps: solving PDE, training KRnet and refining the training set.

---

**Algorithm 1** DAS-R for PDEs

**Input:** Initial \( \hat{p}_{\text{KRnet}}(x; \Theta_f^{(0)}) \), \( u(x; \Theta^{(0)}) \), maximum epoch number \( N_e \), batch size \( m \), initial training set \( S_{\Omega,0} = \{ x_{\Omega,0}^{(i)} \}^{N_0}_{i=1} \) and \( S_{\partial \Omega,0} = \{ x_{\partial \Omega,0}^{(i)} \}^{N_0}_{i=1} \).

1. for \( k = 0 : N_{\text{adaptive}} - 1 \) do
2. // Solve PDE
3. for \( i = 1 : N_e \) do
4. for \( j \) steps do
5. Sample \( m \) samples from \( S_{\Omega,k} \).
6. Sample \( m \) samples from \( S_{\partial \Omega,k} \).
7. Update \( u(x; \Theta) \) by descending the stochastic gradient of \( J_N^S(u(x; \Theta)) \) (see equation (3.25)).
8. end for
9. end for
10. // Train KRnet
11. for \( i = 1 : N_e \) do
12. for \( j \) steps do
13. Sample \( m \) samples from \( S_{\Omega,k} \).
14. Update \( \hat{p}_{\text{KRnet}}(x; \Theta_f) \) by descending the stochastic gradient of \( H(\hat{r}_X, \hat{p}_{\text{KRnet}}) \) (see equation (3.22)).
15. end for
16. end for
17. // Refine training set
18. Generate \( S_{\Omega,k+1} \subset \Omega \) through \( \hat{p}_{\text{KRnet}}(x; \Theta_f^{*(k+1)}) \).
19. end for

**Output:** \( u(x; \Theta_N^*) \)
Algorithm 2 DAS-G for PDEs

**Input:** Initial \( \tilde{p}_{\text{KRnet}}(x; \Theta_f^{(0)}) \), \( u(x; \Theta^{(0)}) \), maximum epoch number \( N_e \), batch size \( m \), initial training set \( S_{\Omega,0} = \{x_{\Omega,0}^{(i)}\}_{i=1}^{n_r} \) and \( S_{\partial\Omega,0} = \{x_{\partial\Omega,0}^{(i)}\}_{i=1}^{n_r} \).

1. **for** \( k = 0 : N_{\text{adaptive}} - 1 \) **do**
   2. // Solve PDE
   3. **for** \( i = 1 : N_e \) **do**
   4. **for** \( j \) steps **do**
   5. Sample \( m \) samples from \( S_{\Omega,k} \).
   6. Sample \( m \) samples from \( S_{\partial\Omega,k} \).
   7. Update \( u(x; \Theta) \) by descending the stochastic gradient of \( J_N(u(x; \Theta)) \) (see equation (2.3)).
   8. **end for**
   9. **end for**
10. // Train KRnet
11. **for** \( i = 1 : N_e \) **do**
12. **for** \( j \) steps **do**
13. Sample \( m \) samples from \( S_{\Omega,k} \).
14. Update \( \tilde{p}_{\text{KRnet}}(x; \Theta_f) \) by descending the stochastic gradient of \( H(\hat{r}_X, \tilde{p}_{\text{KRnet}}) \) (see equation (3.22)).
15. **end for**
16. **end for**
17. // Refine training set
18. Generate \( S_{\Omega,k+1}^\theta \subset \Omega \) with size \( n_r \) through \( \tilde{p}_{\text{KRnet}}(x; \Theta_f^{*(k+1)}) \).
19. \( S_{\Omega,k+1} = S_{\Omega,k} \cup S_{\Omega,k+1}^\theta \).
20. **end for**

**Output:** \( u(x; \Theta_N^*) \)

4. **Analysis of DAS.** As discussed in section 3.4, the key point of our DAS method is to achieve the variance reduction for estimating the residual loss, based on which we expect to improve the accuracy. Under certain conditions, we show that the DAS method has a variance reduction for the residual loss, and the expectation of error bound becomes smaller when the adaptive sampling strategy is employed. We introduce the following assumptions before stating our main results.

**Assumption 1.** [2] In problem (2.1), we let \( F = \mathcal{H} \) be a Hilbert space and \( \mathcal{L} \) a linear operator. Assume that the differential operator \( \mathcal{L} \) and the boundary operator \( b \) satisfy

\[
C_1 \|v\|_{2,\Omega} \leq \|\mathcal{L}v\|_{2,\Omega} + \|bv\|_{2,\partial\Omega} \leq C_2 \|v\|_{2,\Omega} \quad \forall v \in \mathcal{H}
\]

(4.1)

where \( \mathcal{H} \) is a Hilbert space defined on \( \Omega \) and the positive constants \( C_1 \) and \( C_2 \) are independent of \( v \).

The above condition is called the stability bound [2], which is essential to the existence and uniqueness of problem (2.1). Except for this assumption, the following two assumptions for the relationship between \( r(x; \Theta_f^{*(k)}) \) and \( \tilde{p}_{\text{KRnet}}(x; \Theta_f^{*(k)}) \) are given.

**Assumption 2.** Assume that \( \tilde{p}_{\text{KRnet}}(x; \Theta_f^{*(k)}) \) is the optimal candidate for the change of measure in equation (3.1)

\[
\tilde{p}_{\text{KRnet}}(x; \Theta_f^{*(k)}) = c_k r^2(x; \Theta_N^{*(k)})
\]

(4.2)
where $c_k = 1/\int_{\Omega} r^2(x; \Theta_N^{*,(k)}) dx$ is the normalization constant.

**Assumption 3.** Let
\[ R_k = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(x^{(i)}; \Theta_N^{*,(k)})}{\hat{p}_{KRnet}(x^{(i)}; \Theta_f^{*,(k-1)})} \]
be the discrete residual loss at the $k$-th stage, where each $x^{(i)}$ is drawn from $\hat{p}_{KRnet}(x; \Theta_f^{*,(k-1)})$.

Assume that $r^2(x^{(i)}; \Theta_N^{*,(k)})/\hat{p}_{KRnet}(x^{(i)}; \Theta_f^{*,(k-1)}) \in [\tau_1, \tau_2]$ almost surely for each $i = 1, 2, \ldots, N_r$.

**Theorem 4.1.** Let $u(x; \Theta_N^{*,(k)})$ be an approximate solution of (2.1). Suppose that $\hat{p}_{KRnet}(x; \Theta_f^{*,(k)})$ satisfies Assumption 2. Let $\hat{r}_{k,\hat{p}_{KRnet}}(x; \Theta_f^{*,(k)})$ denote the discrete approximation of $\left\| r(x; \Theta_N^{*,(k)}) \right\|_{\Omega}^2$ with respect to $\hat{p}_{KRnet}(x; \Theta_f^{*,(k)})$.

\[ \hat{r}_{k,\hat{p}_{KRnet}}(x; \Theta_f^{*,(k)}) = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(x^{(i)}; \Theta_N^{*,(k)})}{\hat{p}_{KRnet}(x^{(i)}; \Theta_f^{*,(k)})}, \]

where each $x^{(i)}$ is independently drawn from $\hat{p}_{KRnet}(x; \Theta_f^{*,(k)})$. Then for any PDF $q(x)$, the following inequality holds
\[ \text{Var}(\hat{r}_{k,\hat{p}_{KRnet}}(x; \Theta_f^{*,(k)})) \leq \text{Var}(\hat{r}_{k,q}(x)). \]

**Proof.** Noting that
\[ \text{Var}(\hat{r}_{k,\hat{p}_{KRnet}}(x; \Theta_f^{*,(k)})) = \frac{1}{N_r} \text{Var}(\frac{r^2(x; \Theta_N^{*,(k)})}{\hat{p}_{KRnet}(x; \Theta_f^{*,(k)})}) \]
\[ \text{Var}(\hat{r}_{k,q}(x)) = \frac{1}{N_r} \text{Var}(\frac{r^2(x; \Theta_N^{*,(k)})}{q(x)}), \]

we consider the variance of $r^2(x; \Theta_N^{*,(k)})/\hat{p}_{KRnet}(x; \Theta_f^{*,(k)})$ and $r^2(x; \Theta_N^{*,(k)})/q(x)$.

It is easy to see that the change of measure induce an unbiased estimator. So we only need to pay attention to the second-order moments. We then have
\[ \mathbb{E}(\frac{r^4(x; \Theta_N^{*,(k)})}{\hat{p}_{KRnet}(x; \Theta_f^{*,(k)})}) = \int_{\Omega} \frac{r^4(x; \Theta_N^{*,(k)})}{\hat{p}_{KRnet}(x; \Theta_f^{*,(k)})} dx \]
\[ = c_k^{-1} \int_{\Omega} r^2(x; \Theta_N^{*,(k)}) dx \]
\[ = \left[ \mathbb{E}(\frac{r^2(x; \Theta_N^{*,(k)})}{q(x)}) \right]^2 \]
\[ \leq \mathbb{E}(\frac{r^4(x; \Theta_N^{*,(k)})}{q^2(x)}) = \mathbb{E}(\frac{r^4(x; \Theta_N^{*,(k)})}{q^2(x)}), \]

where the Jensen inequality is applied. \( \Box \)

**Theorem 4.2.** Let $u(x; \Theta_N^{*,(k)}) \in F$ be a solution of (2.3) where the collocation points are independently drawn from $\hat{p}_{KRnet}(x; \Theta_f^{*,(k-1)})$. Suppose that Assumption 1 and Assumption 3 are satisfied. Given $0 < \varepsilon < 1$, the following error estimate holds
\[ \left\| u(x; \Theta_N^{*,(k)}) - u(x) \right\|_{\Omega} \leq \sqrt{2} c_1^{-1} \left( R_k + \varepsilon + \left\| h(x; \Theta_N^{*,(k)}) \right\|_{\Omega}^2 \right)^{\frac{1}{2}}. \]
with probability at least \(1 - \exp(-2N_r \varepsilon^2/\tau_2^2)\).

**Proof.** By Assumption 1, we have

\[
\|u(x; \Theta_N^{*,(k)}) - u(x)\|_{2,\Omega} \leq C_1^{-1} \left( \left\| \mathcal{L}(u(x; \Theta_N^{*,(k)}) - u(x)) \right\|_{2,\Omega} + \left\| b(u(x; \Theta_N^{*,(k)}) - u(x)) \right\|_{2,\Omega} \right) \\
\leq \sqrt{2} C_1^{-1} \left( \left\| \mathcal{L}(u(x; \Theta_N^{*,(k)}) - u(x)) \right\|_{2,\Omega}^2 + \left\| b(u(x; \Theta_N^{*,(k)}) - u(x)) \right\|_{2,\Omega}^2 \right)^{\frac{1}{2}}.
\]

Combining \(\mathcal{L}u(x) = s(x)\), \(bu(x) = C\), \(r(x; \Theta_N^{*,(k)}) = \mathcal{L}u(x; \Theta_N^{*,(k)}) - s(x)\) and \(b(x; \Theta_N^{*,(k)}) = bu(x; \Theta_N^{*,(k)}) - g(x)\) gives

\[
\|u(x; \Theta_N^{*,(k)}) - u(x)\|_{2,\Omega} \leq \sqrt{2} C_1^{-1} \left( \left\| r(x; \Theta_N^{*,(k)}) \right\|_{2,\Omega}^2 + \left\| b(x; \Theta_N^{*,(k)}) \right\|_{2,\Omega}^2 \right)^{\frac{1}{2}}.
\]  

(4.3)

Noting that \(E(R_k) = \left\| r(x; \Theta_N^{*,(k)}) \right\|_{2,\Omega}^2\), and according to the Hoeffding inequality, we have

\[
P(R_k - E(R_k) \geq -\varepsilon) \geq 1 - \exp \left( \frac{-2N_r \varepsilon^2}{\tau_2^2} \right).
\]  

(4.4)

Combining (4.3) and (4.4) gives that

\[
\left\| u(x; \Theta_N^{*,(k)}) - u(x) \right\|_{2,\Omega} \leq \sqrt{2} C_1^{-1} \left( R_k + \varepsilon + \left\| b(x; \Theta_N^{*,(k)}) \right\|_{2,\Omega}^2 \right)^{\frac{1}{2}}
\]

with probability at least \(1 - \exp(-2N_r \varepsilon^2/\tau_2^2)\).

**COROLLARY 4.3.** Under the same conditions of Theorem 4.2, suppose that Assumption 2 is satisfied and the boundary loss \(J_b(u)\) is zero, then the following inequality holds

\[
E(R_{k+1}) \leq E(R_k)
\]

**Proof.** Noting that

\[
\Theta_N^{*,(k+1)} = \arg \min_{\Theta} \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(x^{(i)}; \Theta)}{\hat{p}_{\text{Ker}(x^{(i)}; \Theta)}}
\]

Since \(\Theta_N^{*,(k+1)}\) is the optimal solution at \((k+1)\)-th stage, we have

\[
R_{k+1} = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(x^{(i)}; \Theta_N^{*,(k+1)})}{\hat{p}_{\text{Ker}(x^{(i)}; \Theta^{*,(k)})}} \leq \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(x^{(i)}; \Theta_N^{*,(k)})}{\hat{p}_{\text{Ker}(x^{(i)}; \Theta^{*,(k)})}}.
\]  

(4.5)

Plugging \(\hat{p}_{\text{Ker}(x; \Theta^{*,(k)})} = c_k r^2(x; \Theta_N^{*,(k)})\) into (4.5) gives that

\[
R_{k+1} \leq \frac{1}{c_k}.
\]

Noting that \(R_{k+1}\) is a random variable and taking its expectation, it follows that

\[
E(R_{k+1}) \leq \frac{1}{c_k} = \int_{\Omega} r^2(x; \Theta_N^{*,(k)})dx = E(R_k),
\]
which completes the proof. □

Remark 4.4. If the boundary conditions are well satisfied, and the expectation of the upper bound in Theorem 4.2 gets sharper after the adaptive sampling procedure, the DAS algorithm will improve the accuracy. However, Corollary 4.3 only shows that the expectation of $R_k$ does not get worse. This is due to the simplification of our adaptivity procedure, where refining the approximate solution and updating the training set are implemented in a sequential way. Considering that the approximation error and the statistical error coexist in the deep learning solver, updating the training set should be adapted into the learning process such that the approximation error and the statistical error can be studied in a more unified way. Research in this direction is beyond the scope of the current paper.

5. Numerical experiments. In this section, we conduct some numerical experiments to demonstrate the effectiveness of the proposed DAS method. All deep neural network models are trained by the ADAM method [21].

5.1. Low regularity test problems. In this part, two-dimensional low regularity problems are considered, where the solution of the first one has a peak and the solution of the second one has two peaks.

5.1.1. Two-dimensional peak problem. The following elliptic equation is considered
\[-\Delta u(x_1, x_2) = s(x_1, x_2) \quad \text{in } \Omega,\]
\[u(x_1, x_2) = g(x_1, x_2) \quad \text{on } \partial \Omega,\]
(5.1)
where the computation domain is $\Omega = [-1, 1]^2$. In order to quantify the error, we use the following reference solution
\[u(x_1, x_2) = \exp\left(-1000[(x_1 - r_c)^2 + (x_2 - r_c)^2]\right),\]
which has a peak at the point $(r_c, r_c)$ and decreases rapidly away from $(r_c, r_c)$. This test problem is often used to test the performance of adaptive finite element methods [29].

We choose a six-layer fully connected neural network $u(x; \Theta)$ with 32 neurons to approximate the solution, and the activation function of $u(x; \Theta)$ is set to the hyperbolic tangent function since the solution is twice differentiable. The penalty parameter in equation (2.3) is set to $\hat{\gamma} = 1$. For KRnet, we take $L = 6$ affine coupling layers, and two fully connected layers with 24 neurons for NN (see (3.11)). The activation function of KRnet is the rectified linear unit (ReLU) function since we only use the KRnet for density approximation. The number of epochs for training both $u(x; \Theta)$ and $p(x; \Theta_f)$ is set to $N_e = 3000$. The learning rate for ADAM optimizer is set to 0.0001, and the batch size is set to $m = 500$. Here, we set $(r_c, r_c) = (0.5, 0.5)$. To assess the effectiveness of our DAS methods, we generate a uniform meshgrid with size $256 \times 256$ in $[-1, 1]^2$ and compute the mean square error on these grid points.

In Figure 5.1, we plot the approximation errors given by different sampling strategies with respect to the sample size in the left plot and with respect to the number of epochs in the right plot. For each $|\mathcal{S}_\Omega|$, we take three runs with different random seeds for initialization and compute the mean error of the three runs as the final error. For DAS strategies, the numbers of adaptivity iterations are set to $N_{\text{adaptive}} = 4, 6, 8, 10$ for $|\mathcal{S}_\Omega| = 2 \times 10^3, 3 \times 10^3, 4 \times 10^3, 5 \times 10^3$ respectively, and $n_r = 500$ is set for the DAS-G strategy (see section 3.4). For the uniform sampling strategy, the number of epochs is set to be the same as the total number of epochs of each DAS method. It is clear that for this test problem the DAS methods (DAS-G and DAS-R) have a better performance than the uniform sampling strategy and DAS-R performs better than DAS-G. The errors of DAS-G and DAS-R decrease more quickly than that of the uniform sampling method in terms of both the sample size $|\mathcal{S}_\Omega|$ and the number of epochs. We also note that the errors of DAS-R and DAS-G decay in a much more smooth and consistent way than the uniform sampling method for a fixed sample size.
5.1.2. Two-dimensional test problem with two peaks. In this test problem, we consider the following equation

\[-\nabla \cdot \left[ u(x_1, x_2) \nabla (x_1^2 + x_2^2) \right] + \nabla^2 u(x_1, x_2) = s(x_1, x_2) \quad \text{in } \Omega,\]

\[u(x_1, x_2) = g(x_1, x_2) \quad \text{on } \partial \Omega,\]

where the computation domain is \(\Omega = [-1,1]^2\). The exact solution of (5.2) is chosen as

\[u(x_1, x_2) = e^{-1000[(x_1-0.5)^2+(x_2-0.5)^2]} + e^{-1000[(x_1+0.5)^2+(x_2+0.5)^2]},\]

which has two peaks at the points (0.5, 0.5) and (-0.5, -0.5). Here, the Dirichlet boundary condition on \(\partial \Omega\) is given by the exact solution.

We choose a six-layer fully connected neural network \(u(x; \Theta)\) with 64 neurons to approximate the solution of (5.2). The activation function of \(u(x; \Theta)\) is set to the hyperbolic tangent function, and the penalty parameter in equation (2.3) is set to \(\hat{\gamma} = 1\). For KRnet, we take \(L = 8\) affine coupling layers, and two fully connected layers with 48 neurons for NN (see (3.11)). The activation function of KRnet is the rectified linear unit (ReLU) function. The number of epochs for training both \(u(x; \Theta)\) and \(p(x; \Theta_f)\) is set to \(N_e = 5000\). The learning rate for ADAM optimizer is set to 0.0001, and the batch size is set to \(m = 500\). Again, we generate a uniform meshgrid with size 256 \(\times\) 256 in \([-1,1]^2\) and compute the mean square error on these grid points to assess the effectiveness of our DAS methods.
Figure 5.2: Solutions, two-dimensional peak test problem.

Figure 5.5 shows the approximation errors for this test problem, where the left one displays the errors with respect to the sample size $|S_\Omega|$ for different sampling strategies, and the right one shows the error evolution of DAS-G at different adaptivity iteration steps. For each $|S_\Omega|$, we again take three runs with different random seeds for initialization and compute the mean error of the three runs as the final error. For the DAS-G strategy, the numbers of adaptivity iterations is set to $N_{\text{adaptive}} = 5$ (also for DAS-R), and the numbers of collocation points in $S_{\Omega}^g(k = 1, 2, 3, 4)$ is set to $n_r = 500, 1 \times 10^3, 1.5 \times 10^3, 2 \times 10^3$ for $|S_\Omega| = 2.5 \times 10^3, 5 \times 10^3, 7.5 \times 10^3, 10^4$ respectively. For the uniform sampling strategy, we train the model with $2.5 \times 10^4$ epochs to match the total number of epochs of DAS methods. From Figure 5.5, it can be seen that for this test problem our DAS methods (DAS-G and DAS-R) have a better performance than the uniform sampling strategy and DAS-G performs better than DAS-R. It is seen that the approximation error decreases as the adaptivity iteration step $k$ increases.

In Figure 5.6 we compare the exact solution, the DAS solutions given by $10^4$ nonuniform samples and the approximate solution given by $10^4$ uniform samples. It is seen that DAS methods are much more effective than the uniform sampling method to capture the information around the two peaks. Figure 5.7 shows the evolution of $S_{\Omega}^g$ of DAS-G method with respect to adaptivity iterations $k = 1, 2, 3, 4$ ($|S_{\Omega}^g| = 2 \times 10^3$), where the initial training set $S_{\Omega,0}^g$ consists of uniform collocation points on $\Omega$ (see section 3.4). $S_{\Omega,1}^g$ shows that the error profile has two peaks. After the training set is augmented with $S_{\Omega,1}$, the error profile becomes more flat as shown by the distribution of $S_{\Omega,2}^g$. After the training set is augmented with $S_{\Omega,2}$, the largest error is found again around the two peaks,
and then the subsequent augmentation of the training set yields a more flat error profile. Such a pattern is repeated until no improvement can be reached.

5.2. High-dimensional test problems. Next we consider the $d$-dimensional elliptic equation

$$-\Delta u(x) = s(x), \quad x \in \Omega = [-1,1]^d,$$

with an exact solution

$$u(x) = e^{-10\|x\|^2},$$

where the Dirichlet boundary condition on $\partial\Omega$ is given by the exact solution. This test problem is also used to test the performance of adaptive finite element methods [30], where the dimensionality $d = 2$ or 3 for that setting. But we are more interested in cases with larger $d$. Note that the geometric properties of high-dimensional spaces are significantly different from our intuitions on low-dimensional ones, e.g., most of the volume of a high-dimensional cube is located around its corners [40, 1, 44]. If we use uniform samples to generate $S_\Omega$, most of the collocation points in $S_\Omega$ are near the surface of the hypercube. Since the information of the exact solution is mainly from the neighborhood of the origin, most of the samples in $S_\Omega$ may not contribute to training the neural network when $d$ is large enough.

We choose a six-layer fully connected neural network $u(x; \Theta)$ with 64 neurons to approximate the solution, and the activation function of $u(x; \Theta)$ is set to the hyperbolic tangent function. We set the penalty parameter $\hat{\gamma} = 1$ in equation (2.3). For KRnet, we set $K = 3$ and take $L = 6$ affine coupling layers, and two fully connected layers with 64 neurons for NN (see (3.11)). The activation function of KRnet is the rectified linear unit (ReLU) function. The number of epochs for training both $u(x; \Theta)$ and $p(x; \Theta_f)$ is set to $N_e = 3000$. The learning rate for ADAM optimizer is set to 0.0001, and the batch size is set to $m = 5000$. The numbers of adaptivity iterations is set to $N_{\text{adaptive}} = 5$. To measure the quality of approximation, we generate a tensor grid with $n_t$ points around the origin (in $[-0.1,0.1]^d$) where $n_t$ is the number of nodes for each dimension. We define the relative error

$$\text{Relative error} = \frac{\|u_{NN} - u\|_2}{\|u\|_2}, \quad (5.3)$$

Fig. 5.3: The errors of DAS-R at certain adaptivity iteration steps for the two-dimensional peak test problem. $|\Omega| = 5 \times 10^3$. 

and then the subsequent augmentation of the training set yields a more flat error profile. Such a pattern is repeated until no improvement can be reached.

5.2. High-dimensional test problems. Next we consider the $d$-dimensional elliptic equation

$$-\Delta u(x) = s(x), \quad x \in \Omega = [-1,1]^d,$$

with an exact solution

$$u(x) = e^{-10\|x\|^2},$$

where the Dirichlet boundary condition on $\partial\Omega$ is given by the exact solution. This test problem is also used to test the performance of adaptive finite element methods [30], where the dimensionality $d = 2$ or 3 for that setting. But we are more interested in cases with larger $d$. Note that the geometric properties of high-dimensional spaces are significantly different from our intuitions on low-dimensional ones, e.g., most of the volume of a high-dimensional cube is located around its corners [40, 1, 44]. If we use uniform samples to generate $S_\Omega$, most of the collocation points in $S_\Omega$ are near the surface of the hypercube. Since the information of the exact solution is mainly from the neighborhood of the origin, most of the samples in $S_\Omega$ may not contribute to training the neural network when $d$ is large enough.

We choose a six-layer fully connected neural network $u(x; \Theta)$ with 64 neurons to approximate the solution, and the activation function of $u(x; \Theta)$ is set to the hyperbolic tangent function. We set the penalty parameter $\hat{\gamma} = 1$ in equation (2.3). For KRnet, we set $K = 3$ and take $L = 6$ affine coupling layers, and two fully connected layers with 64 neurons for NN (see (3.11)). The activation function of KRnet is the rectified linear unit (ReLU) function. The number of epochs for training both $u(x; \Theta)$ and $p(x; \Theta_f)$ is set to $N_e = 3000$. The learning rate for ADAM optimizer is set to 0.0001, and the batch size is set to $m = 5000$. The numbers of adaptivity iterations is set to $N_{\text{adaptive}} = 5$. To measure the quality of approximation, we generate a tensor grid with $n_t$ points around the origin (in $[-0.1,0.1]^d$) where $n_t$ is the number of nodes for each dimension. We define the relative error

$$\text{Relative error} = \frac{\|u_{NN} - u\|_2}{\|u\|_2}, \quad (5.3)$$

Fig. 5.3: The errors of DAS-R at certain adaptivity iteration steps for the two-dimensional peak test problem. $|\Omega| = 5 \times 10^3$. 

and then the subsequent augmentation of the training set yields a more flat error profile. Such a pattern is repeated until no improvement can be reached.

5.2. High-dimensional test problems. Next we consider the $d$-dimensional elliptic equation

$$-\Delta u(x) = s(x), \quad x \in \Omega = [-1,1]^d,$$

with an exact solution

$$u(x) = e^{-10\|x\|^2},$$

where the Dirichlet boundary condition on $\partial\Omega$ is given by the exact solution. This test problem is also used to test the performance of adaptive finite element methods [30], where the dimensionality $d = 2$ or 3 for that setting. But we are more interested in cases with larger $d$. Note that the geometric properties of high-dimensional spaces are significantly different from our intuitions on low-dimensional ones, e.g., most of the volume of a high-dimensional cube is located around its corners [40, 1, 44]. If we use uniform samples to generate $S_\Omega$, most of the collocation points in $S_\Omega$ are near the surface of the hypercube. Since the information of the exact solution is mainly from the neighborhood of the origin, most of the samples in $S_\Omega$ may not contribute to training the neural network when $d$ is large enough.

We choose a six-layer fully connected neural network $u(x; \Theta)$ with 64 neurons to approximate the solution, and the activation function of $u(x; \Theta)$ is set to the hyperbolic tangent function. We set the penalty parameter $\hat{\gamma} = 1$ in equation (2.3). For KRnet, we set $K = 3$ and take $L = 6$ affine coupling layers, and two fully connected layers with 64 neurons for NN (see (3.11)). The activation function of KRnet is the rectified linear unit (ReLU) function. The number of epochs for training both $u(x; \Theta)$ and $p(x; \Theta_f)$ is set to $N_e = 3000$. The learning rate for ADAM optimizer is set to 0.0001, and the batch size is set to $m = 5000$. The numbers of adaptivity iterations is set to $N_{\text{adaptive}} = 5$. To measure the quality of approximation, we generate a tensor grid with $n_t$ points around the origin (in $[-0.1,0.1]^d$) where $n_t$ is the number of nodes for each dimension. We define the relative error

$$\text{Relative error} = \frac{\|u_{NN} - u\|_2}{\|u\|_2}, \quad (5.3)$$

Fig. 5.3: The errors of DAS-R at certain adaptivity iteration steps for the two-dimensional peak test problem. $|\Omega| = 5 \times 10^3$. 

and then the subsequent augmentation of the training set yields a more flat error profile. Such a pattern is repeated until no improvement can be reached.

5.2. High-dimensional test problems. Next we consider the $d$-dimensional elliptic equation

$$-\Delta u(x) = s(x), \quad x \in \Omega = [-1,1]^d,$$

with an exact solution

$$u(x) = e^{-10\|x\|^2},$$

where the Dirichlet boundary condition on $\partial\Omega$ is given by the exact solution. This test problem is also used to test the performance of adaptive finite element methods [30], where the dimensionality $d = 2$ or 3 for that setting. But we are more interested in cases with larger $d$. Note that the geometric properties of high-dimensional spaces are significantly different from our intuitions on low-dimensional ones, e.g., most of the volume of a high-dimensional cube is located around its corners [40, 1, 44]. If we use uniform samples to generate $S_\Omega$, most of the collocation points in $S_\Omega$ are near the surface of the hypercube. Since the information of the exact solution is mainly from the neighborhood of the origin, most of the samples in $S_\Omega$ may not contribute to training the neural network when $d$ is large enough.

We choose a six-layer fully connected neural network $u(x; \Theta)$ with 64 neurons to approximate the solution, and the activation function of $u(x; \Theta)$ is set to the hyperbolic tangent function. We set the penalty parameter $\hat{\gamma} = 1$ in equation (2.3). For KRnet, we set $K = 3$ and take $L = 6$ affine coupling layers, and two fully connected layers with 64 neurons for NN (see (3.11)). The activation function of KRnet is the rectified linear unit (ReLU) function. The number of epochs for training both $u(x; \Theta)$ and $p(x; \Theta_f)$ is set to $N_e = 3000$. The learning rate for ADAM optimizer is set to 0.0001, and the batch size is set to $m = 5000$. The numbers of adaptivity iterations is set to $N_{\text{adaptive}} = 5$. To measure the quality of approximation, we generate a tensor grid with $n_t$ points around the origin (in $[-0.1,0.1]^d$) where $n_t$ is the number of nodes for each dimension. We define the relative error

$$\text{Relative error} = \frac{\|u_{NN} - u\|_2}{\|u\|_2}, \quad (5.3)$$
where $\bm{u}_{NN}$ and $\bm{u}$ denote two vectors whose elements are the function values of $u(x; \Theta)$ and $u(x)$ at the tensor grid respectively.

We first investigate the relation between the error and the dimensionality $d$ when the uniform sampling strategy is employed. Figure 5.8(a) shows the relative errors in terms of a varying $d$ for a sample size $|\Omega| = 2 \times 10^5$. To roughly match the number of grid points for different $d$, we set $n_r = 16, 6, 4, 3$ for $d = 4, 6, 8, 10$ respectively. It is seen that the relative error grows quickly to $O(1)$ as $d$ increases. However, as shown in Figure 5.8(b), all training losses are finally close to zero for $d = 4, 6, 8, 10$. This phenomenon demonstrates that the uniform sampling method become less effective as $d$ increases and the convergence of the approximate solution is highly dependent on the
(a) The exact solution.  (b) DAS-R approximation.  
(c) DAS-G approximation.  (d) Approximation using the uniform sampling strategy.

Fig. 5.6: Solutions, two-dimensional test problem with two peaks.

choice of $S_\Omega$ for a large $d$.

Figure 5.9 shows the relative errors for the uniform sampling strategy, DAS-R and DAS-G, where different numbers of samples $|S_\Omega|$ are considered. For each $|S_\Omega|$, we again take three runs with different random seeds for initialization and compute the mean error of the three runs as the final error. For the DAS-G strategy, the numbers of collocation points in $S_{\Omega,k} (k = 1, 2, 3, 4)$ is set to $n_c = 10^4, 2 \times 10^4, 3 \times 10^4, 4 \times 10^4$ for $|S_\Omega| = 5 \times 10^4, 10^5, 1.5 \times 10^5, 2 \times 10^5$ respectively. For the uniform sampling strategy, we train the model with $1.5 \times 10^4$ epochs to match the total number of epochs of DAS methods. Both DAS-G and DAS-R improve the accuracy significantly compared to the uniform sampling strategy. In addition, the error of DAS-G decreases slightly faster than that of DAS-R for this test problem. In Figure 5.10 we compare the error evolution of different sampling strategies. From the left plot of Figure 5.10, as the number of epochs increases, the errors of DAS-G and DAS-R decrease quickly, while the error of the uniform sampling strategy does not decrease. This result suggests that for high-dimensional problems DAS methods are able to achieve a good approximation with a relatively small number of nonuniform samples while much more uniform samples are needed for the same accuracy. The right plot of Figure 5.10 shows the error of DAS-G at each adaptivity iteration step $k$. It is seen that the error drops dramatically after we refine the solution using $S_{\Omega,1}$. Figure 5.11 and 5.12 show 3000 samples from the training sets ($|S_\Omega| = 2 \times 10^5$) DAS-R and DAS-G for the first four adaptivity iterations, where the components $x_6$ and $x_7$ are used for visualization. We have also checked the other components, and no significantly different results were found. For DAS-R, 3000 samples are randomly chosen from $S_{\Omega,k} (k = 1, 2, 3, 4)$. For DAS-G,
3000 samples for visualization are randomly selected from $S_{\Omega,k}^g$ ($k = 1, 2, 3, 4$). It is seen that the profile of $S_{\Omega,k}$ is gradually flattened as $k$ increases, meaning the nonuniform samples are able to smooth the error profile which has a peak around the origin. As for DAS-G, the improvement takes a slightly different path. $S_{\Omega,1}^g$ shows that the error profile has a peak around the origin. After the training set is augmented with $S_{\Omega,1}$, the error profile becomes quite flat as shown by the distribution of $S_{\Omega,2}^g$. This is as expected since more collocation points are added to the neighborhood of the origin which should reduce the approximation error over there. After the training set is augmented with $S_{\Omega,2}$, the largest error is found again around the origin, and the subsequent augmentation of the training set yields a quite flat error profile. Such a pattern is similar to what we have observed in Figure 5.7.

6. Conclusion. In this paper we have developed a deep adaptive sampling (DAS) method to improve the neural network approximation of PDEs iteratively. The key idea of DAS is to employ a deep generative model to generate collocation points that are consistent with the distribution induced by an appropriate error indicator function. In this way, the training set is refined according to the regularity of the PDE solution, which follows the similar principle of adaptive mesh refinement of classical numerical methods. Numerical experiments have shown that the DAS method is able to significantly improve the accuracy for the approximation of low regularity problems especially when the dimensionality is relatively large. The proposed DAS method provides a very general and flexible framework for an adaptive learning strategy. There are several possible ways to further improve it. First, DAS consists of two DNN-based models: one model serves as an approximator for the PDE solution and the other one serves as an error indicator for the selection of collocation points. Both models can be chosen in terms of a certain criterion. In this work, we use a regular DNN for PDE approximation and KRnet for density approximation and sample generation. Second, the underlying distribution for the training set can be problem dependent. In this work, we choose the residual-induced distribution. We may also use the gradient of the approximation solution to define
Fig. 5.8: The convergence behavior of high-dimensional PDEs with uniform sampling method. Loss is close to zero, but the error is still large for the ten-dimensional test problem.

Fig. 5.9: The error w.r.t sample size $|S_\Omega|$, ten-dimensional test problem.

an indicator distribution. In [39], we employ KRnet to approximate the Fokker-Planck equation, where the collocation points are sampled from the approximate solution. Third, the DAS method is not limited to steady-state PDE problems. We may employ the DAS method on the space-time domain to refine the training set for the approximation of time-dependent problems. Last but not least, the current training process can also be improved. Although the current DAS methods work well enough to demonstrate the effectiveness of the algorithm, many questions remain open, e.g., what is the optimal way for the two deep models to communicate and what is the optimal sample size for $S_{\Omega, k}$. Research on these issues will be reported in forthcoming papers.

Acknowledgments: X. Wan has been supported by NSF grant DMS-1913163. C. Yang has been supported by NSFC grant 12131002.
Fig. 5.10: The error evolution of different sampling strategies with $|S_\Omega| = 2 \times 10^5$ and $d = 10$. Left: A comparison of DAS-G, DAS-R and the uniform sampling method; Right: The error evolution of DAS-G at different adaptivity iteration steps.

Fig. 5.11: The evolution of $S_{\Omega,k}$ in DAS-R, ten-dimensional test problem.

REFERENCES

[1] A. Blum, J. Hopcroft, and R. Kannan, Foundations of data science, Cambridge University Press, 2020.
[2] P. Bochev and M. Gunzburger, Least-squares methods for hyperbolic problems, in Handbook of Numerical Analysis, vol. 17, Elsevier, 2016, pp. 289–317.
[3] L. Bottou, F. E. Curtis, and J. Nocedal, Optimization methods for large-scale machine learning, SIAM Review, 60 (2018), pp. 223–311.
[4] G. Carlier, A. Galichon, and F. Santambrogio, From Knothe’s transport to Brenier’s map and a continu-
Fig. 5.12: The evolution of $S_{\Omega,k}^g$ in DAS-G, ten-dimensional test problem.

[5] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals and Systems, 2 (1989), pp. 303–314.
[6] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, A tutorial on the cross-entropy method, Annals of Operations Research, 134 (2005), pp. 19–67.
[7] L. Dinh, J. Sohl-Dickstein, and S. Bengio, Density estimation using real NVP, arXiv preprint arXiv:1605.08803, (2016).
[8] M. DiNepiry and N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, Communications in Numerical Methods in Engineering, 10 (1994), pp. 195–201.
[9] S. Dong and Z. Li, Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations, Computer Methods in Applied Mechanics and Engineering, 387 (2021), p. 114129.
[10] W. E and B. Yu, The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, Communications in Mathematics and Statistics, 6 (2018), pp. 1–12.
[11] H. C. Elman, D. J. Silvester, and A. J. Wathen, Finite elements and fast iterative solvers: With applications in incompressible fluid dynamics, Oxford University Press, USA, 2014.
[12] H. Gao, L. Sun, and J.-X. Wang, Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, Journal of Computational Physics, 428 (2021), p. 110079.
[13] Y. Gu, H. Yang, and C. Zhou, Selectnet: Self-paced learning for high-dimensional partial differential equations, Journal of Computational Physics, 441 (2021), p. 110444.
[14] J. Han, A. Jentzen, and E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proceedings of the National Academy of Sciences, 115 (2018), pp. 8505–8510.
[15] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks, 4 (1991), pp. 251–257.
[16] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Networks, 2 (1989), pp. 359–366.
[17] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[18] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Physics-informed machine learning, Nature Reviews Physics, 3 (2021), pp. 422–440.
[19] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, Variational physics-informed neural networks for solving
partial differential equations, arXiv preprint arXiv:1912.00873, (2019).
[20] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, Computer Methods in Applied Mechanics and Engineering, 374 (2021), p. 113547.
[21] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, (2017).
[22] D. P. Kingma and P. Dhariwal, Glow: Generative flow with invertible 1x1 convolutions, in Advances in Neural Information Processing Systems, 2018, pp. 10215–10224.
[23] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, 9 (1998), pp. 987–1000.
[24] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Networks, 6 (1993), pp. 861–867.
[25] K. Li, K. Tang, T. Wu, and Q. Liao, D3M: A deep domain decomposition method for partial differential equations, IEEE Access, 8 (2020), pp. 5283–5294.
[26] W. Li, X. Xiang, and Y. Xu, Deep domain decomposition method: Elliptic problems, in Proceedings of The First Mathematical and Scientific Machine Learning Conference, J. Lu and R. Ward, eds., vol. 107 of Proceedings of Machine Learning Research, Princeton University, Princeton, NJ, USA, 20–24 Jul 2020, PMLR, pp. 269–286.
[27] J. Lu, Y. Lu, and M. Wang, A priori generalization analysis of the deep ritz method for solving high dimensional elliptic equations, arXiv preprint arXiv:2101.01708, (2021).
[28] K. Mekchay and R. H. Nochetto, Convergence of adaptive finite element methods for general second order linear elliptic pdes, SIAM Journal on Numerical Analysis, 43 (2005), pp. 1803–1827.
[29] W. F. Mitchell, A collection of 2D elliptic problems for testing adaptive grid refinement algorithms, Applied Mathematics and Computation, 220 (2013), pp. 350–364.
[30] P. Morin, R. H. Nochetto, and K. G. Siebert, Convergence of adaptive finite element methods, SIAM Review, 44 (2002), pp. 631–658.
[31] G. Pang, L. Lu, and G. E. Karniadakis, fPINNs: Fractional physics-informed neural networks, SIAM Journal on Scientific Computing, 41 (2019), pp. A2603–A2626.
[32] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics, 378 (2019), pp. 686–707.
[33] R. Y. Rubinstein and D. P. Kroese, The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning, Springer Science & Business Media, 2013.
[34] F. Santambrogio, Optimal transport for applied mathematicians, Birkäuer, NY, 55 (2015), pp. 58–63.
[35] H. Sheng and C. Yang, PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries, Journal of Computational Physics, (2020), p. 110085.
[36] Y. Shin, Z. Zhang, and G. E. Karniadakis, Error estimates of residual minimization using neural networks for linear PDEs, arXiv preprint arXiv:2010.08019, (2020).
[37] J. Shigano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, Journal of Computational Physics, 375 (2018), pp. 1339–1364.
[38] K. Tang, X. Wan, and Q. Liao, Deep density estimation via invertible block-triangular mapping, Theoretical & Applied Mechanics Letters, 10 (2020), p. 143.
[39] ———, Adaptive deep density approximation for Fokker-Planck equations, arXiv preprint arXiv:2103.11181, (2021).
[40] R. Vershynin, High-dimensional probability: An introduction with applications in data science, vol. 47, Cambridge University Press, 2018.
[41] X. Wan and K. Tang, Augmented KRnet for density estimation and approximation, arXiv preprint arXiv:2105.12866, (2021).
[42] X. Wan and S. Wei, VAE-KRnet and its applications to variational Bayes, arXiv preprint arXiv:2006.16431, (2020).
[43] E. Weinan, The dawning of a new era in applied mathematics, Notices of the American Mathematical Society, 68 (2021), pp. 565–571.
[44] J. Wright and Y. Ma, High-dimensional data analysis with low-dimensional models: Principles, computation, and applications, Cambridge University Press, 2021.
[45] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, Journal of Computational Physics, 394 (2019), pp. 56–81.