MACHINE LEARNING AND DOMAIN DECOMPOSITION METHODS - A SURVEY

AXEL KLAWONN*†, MARTIN LANSER*†, AND JANINE WEBER*†

Abstract. Hybrid algorithms, which combine black-box machine learning methods with experience from traditional numerical methods and domain expertise from diverse application areas, are progressively gaining importance in scientific machine learning and various industrial domains, especially in computational science and engineering. In the present survey, several promising avenues of research will be examined which focus on the combination of machine learning (ML) and domain decomposition methods (DDMs). The aim of this survey is to provide an overview of existing work within this field and to structure it into domain decomposition for machine learning and machine learning-enhanced domain decomposition, including: domain decomposition for classical machine learning, domain decomposition to accelerate the training of physics-aware neural networks, machine learning to enhance the convergence properties or computational efficiency of DDMs, and machine learning as a discretization method in a DDM for the solution of PDEs. In each of these fields, we summarize existing work and key advances within a common framework and, finally, disuss ongoing challenges and opportunities for future research.

Key words. scientific machine learning, domain decomposition, survey, hybrid modelling, physics-aware neural networks

AMS subject classifications. 65F10, 65N22, 65N55, 68T05, 68T07

1. Introduction. Domain decomposition methods (DDMs) are divide-and-conquer strategies which decompose a given problem into a number of smaller subproblems. This strategy often leads to easier parallelizable algorithms where the subproblems can be solved on different processors (cpu or gpu). Sometimes, an additional problem is needed for parallel scalability, the so-called global problem, or to gather and connect local information obtained from the solution of the local subproblems. DDMs can be useful in different cases, for example, if the original problem is too large to be solved in the available memory, the computing time is too long on a single processor, or the decomposed problem has other preferable properties, for example, being better conditioned, yielding more accurate results, or boosting generalization properties and mitigating the spectral bias, respectively, of machine learning models. DDMs have a long and successful history in constructing parallel scalable preconditioners for linear and nonlinear systems obtained from discretized partial differential equations. Recently, there has been an increased interest in using DDMs in combination with machine learning algorithms, especially in the field of scientific machine learning (SciML).

In recent years, the rapidly growing research area of SciML has drawn increasing interest and attention in various fields of applications, as, for example, in computational fluid mechanics [20, 26], optics and electromagnetic applications [8, 55], and earthquake detections [83]. Originally, the term SciML was introduced and made publicly known by the report [1] prepared for the U.S. Department of Energy. The core idea of SciML is to combine algorithms from supervised or unsupervised machine learning with other domain-specific methods or knowledge to develop new, hybrid methods. In particular, the authors of [1] have identified six priority research di-

*Department of Mathematics and Computer Science, University of Cologne, Weyertal 86-90, 50931 Köln, Germany, {axel.klawonn, martin.lanser, janine.weber}@uni-koeln.de, url: http://www.numerik.uni-koeln.de

†Center for Data and Simulation Science, University of Cologne, Germany, url: http://www.cds.uni-koeln.de
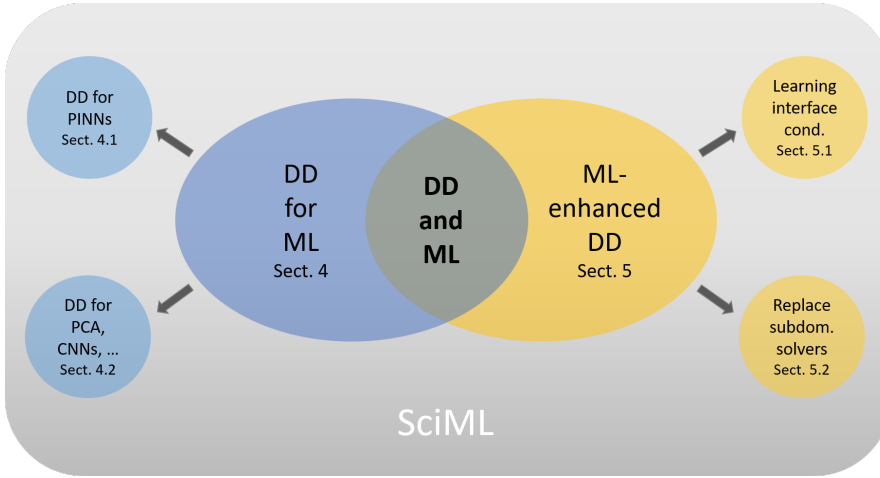
FIGURE 1.   *Schematic representation of the structure of this review paper. All reviewed work on the combination of machine learning and domain decomposition methods are clustered into the two main groups domain decomposition (DD) for machine learning (ML), section 4, and machine learning-enhanced domain decomposition, section 5.*

rections in SciML which are, among others, domain-awareness, interpretability, and robustness of the considered method. With respect to practical applications, this means that a key factor of SciML methods is to connect "black-box" approaches such as deep learning with well-understood and theoretical approaches from science and expert knowledge.

One framework that fits very well to the priority research directions of SciML are physics-informed neural networks (PINNs) [75]. PINNs are a specific machine learning technique usually used to solve forward or inverse problems involving Partial Differential Equations (PDEs). Similar to other deep learning models, PINNs approximate PDE solutions by training a neural network such that a given loss function is minimized. However, the characteristic feature of PINNs is that knowledge with respect to the underlying physics behind the data is integrated into the loss function as well as, possibly, initial or boundary conditions of the computational domain's boundary. For a detailed overview of existing research and applications around PINNs as well as remaining challenges and open research questions, we refer to [11].

One drawback of PINNs and also other neural network architectures, in general, is the large computational complexity of the related optimization problem for large problem domains and multi-scale problems as well as the resulting long training times. Hence, a wide range of attempts have been made to distribute and parallelize the training of different network architectures; see [5, 72, 91] for a comprehensive overview. In general, the wide range of techniques used in parallel and distributed deep learning can roughly be categorized into *data parallelism*, that is, partitioning the input samples, *model parallelism*, that is, partitioning the network structure, and *pipelining*, that is, partitioning by layer; see also [5, Sect. 6]. From an abstract perspective, model parallelism can be and often is interpreted as a form of DDMs [73, 90].

In this paper, our aim is to review existing work on combining DDMs with different machine learning models. Although data parallelism can be also interpreted as a form of domain decomposition such that the global data space is decomposed into smaller subspaces operating only on parts of the data, in the following, we focus
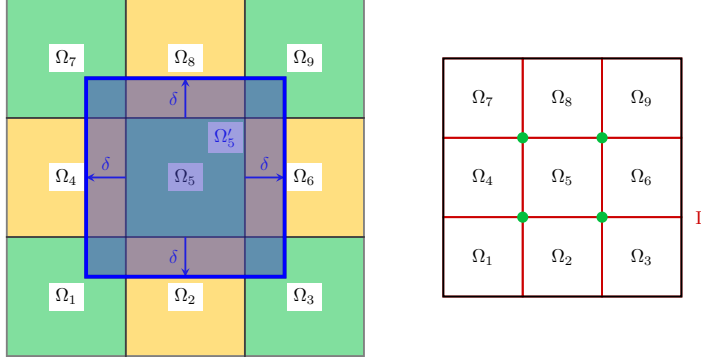
FIGURE 2. *Decomposition of a domain $\Omega \subset \mathbb{R}^2$ into nine nonoverlapping subdomains Ω_i, $i = 1, ..., 9$.* **Left:** *The overlapping domain decomposition is obtained by extending the nonoverlapping subdomains by the width δ. The resulting overlapping subdomain Ω_5' is marked in blue.* **Right:** *The interface Γ is marked in red.*

on model parallelism and pipelining when using domain decomposition in machine learning algorithms. The remainder of the paper is organized as follows. In section 2, we provide a general description of DDMs for the solution of PDEs and introduce some necessary notation. Subsequently, in section 3, we briefly summarize the main idea and central algorithmic formulae of multilayer perceptrons, PINNs, and the Deep Ritz method. Note that both, section 2 and section 3, can be skipped by the experienced reader who is already familiar with DDMs and PINNs and who can directly proceed with section 4. In the present paper, we categorize all cited work into two main classes:

  i) Methods using domain decomposition within machine learning models (section 4)
  ii) Machine learning-enhanced DDMs (section 5).

Additionally, we classify all methods in group i) into two subgroups. First, we consider all approaches using domain decomposition for the acceleration and parallel training of PINNs; see subsection 4.1. Second, in subsection 4.2, we consider approaches using domain decomposition in machine learning algorithms other than PINNs, in particular, classical supervised and unsupervised algorithms. With respect to work within class ii), one can also subdivide the approaches into two subclasses; see also [38]. In this case, the first class consists of methods where machine learning is used to improve the convergence properties or the computational efficiency within classical DDMs. This is typically done by learning or approximating optimal interface conditions or by learning other optimal parameters; see subsection 5.1. In the second class, different types of neural networks are used as discretization methods and replace classical local subdomain or coarse solvers based on finite elements or finite differences; see subsection 5.2. For a schematic overview of the described categorization and the structure of this paper, see Figure 1.

Note that for some approaches which combine domain decomposition with machine learning the categorization is not uniquely defined. For example, D3M [58] and DeepDDM (Deep Domain Decomposition Method) [60] can be interpreted as replacing subdomain solvers within a DDM by neural networks but, at the same time, also as a DDM within neural network training. Hence, we have made the following classifications to the best of our knowledge and with regard to the subject focus of the concrete considered work.

2. Domain decomposition methods for the solution of PDEs. In this section, we briefly introduce the main principles of overlapping and nonoverlapping DDMs as well as some necessary notation.

DDMs [73, 90] are highly scalable, iterative methods for the solution of sparse, linear systems of equations as, for example, arising from the discretization of PDEs. DDMs have been shown to be numerically stable for many practically relevant problems and are designed for the application on parallel computers. Generally speaking, they rely on a divide-and-conquer strategy and decompose the original global problem into a number of smaller subproblems. Mathematically, this corresponds to a subdivision of the domain into a number of smaller subdomains, where the local problems can be solved in parallel on different processors of a parallel computer. Additionally, a small global problem is often needed for numerical and parallel scalability. Certain mathematical conditions at the interface of neighboring subdomains have to be satisfied such that the solution of the original problem is recovered.

For a generic description of DDMs, we consider a boundary value problem of the general form

$$(2.1) \qquad \begin{aligned} \mathcal{L}(u) &= f \quad \text{in} \quad \Omega \\ \mathcal{B}(u) &= g \quad \text{on} \quad \partial\Omega \end{aligned}$$

on the domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, where $\mathcal{L}$ is a linear, second-order, elliptic differential operator and $\mathcal{B}$ represents the boundary conditions.

In order to compute a numerical solution of boundary value problem (2.1), we discretize the variational formulation of the given linear PDE with an appropriate numerical method, for example, conforming finite elements. This results in a, usually sparse, linear system of equations

$$(2.2) \qquad K_g u_g = f_g.$$

The basic idea of DDMs is then, instead of directly solving the completely assembled system of equations (2.2), to decompose the problem into smaller subproblems and thus, solving smaller systems of equations which are assembled on local parts of the domain exclusively. In the following, we denote these local parts of the domain as subdomains and we assume to have a decomposition of Ω into $N \in \mathbb{N}$ nonoverlapping subdomains Ω_i, $i = 1, ..., N$, which fulfills $\overline{\Omega} = \bigcup_{i=1}^{N} \overline{\Omega}_i$; see also Figure 2. Moreover, we define $\Gamma := \left(\bigcup_{i=1}^{N} \partial\Omega_i\right) \setminus \partial\Omega$ as the interface between neighboring subdomains, that is, degrees of freedom shared by at least two subdomains. In order to define overlapping DDMs, we additionally introduce overlapping subdomains Ω_i', $i = 1, ..., N$, where the overlap δ is a measure for the amount of local information shared between neighboring subdomains; see also Figure 2 (left) for a visualization.

However, simply solving decoupled local subdomain systems independently from each other might result in discontinuities along the interface Γ and will not result in the correct solution of the original and global system. Hence, in order to obtain the correct global solution which is continuous across the different subdomains, also communication between the different subproblems is necessary, that is, a global coupling between the different subdomains has to be ensured. With respect on how the global exchange of information across the original domain Ω is ensured, we distinguish between one- and two-level DDMs. One-level DDMs, where information is only exchanged between neighboring subdomains, result in a rate of convergence which deteriorates with a growing number of subdomains when used for the approximate
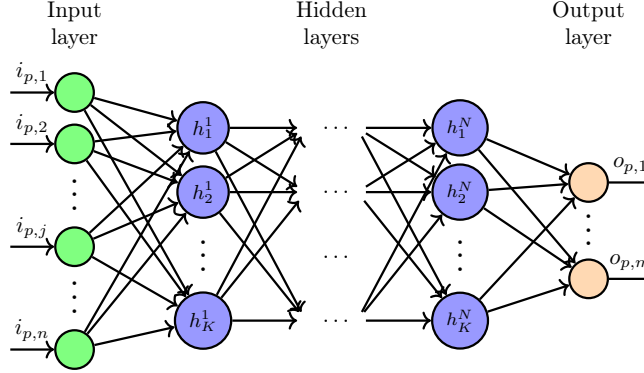
Input
layer

Hidden
layers

Output
layer

FIGURE 3. *Structure of a dense feedforward neural network with n input nodes (marked in green), m output nodes (marked in orange), and N hidden layers with K neurons per layer (marked in blue). Figure taken from [93, Fig. 7.1].*

solution of elliptic systems of PDEs; see, for example, [90]. A remedy is obtained using two-level DDMs by the setup and the solution of an additional, small global problem which needs to be solved in each iteration of the iterative solver. In particular, in two-level DDMs, the additional global problem ensures the fast global transport of information between the different subdomains as well as the scalability and robustness of the iterative method. In the context of two-level DDMs, we refer to this global problem as *coarse problem* and to the related solution space as *coarse space*. Let us note that the idea of a coarse problem is also transferred to domain decomposition approaches for, e.g., training PINNs; cf. subsection 4.1. In classic DDMs, a coarse space is often defined by certain weighted averages or translations and rotations along the interface between neighboring subdomains. Additionally, also more sophisticated coarse spaces exist which are, for example, obtained by solving certain eigenvalue problems and which are problem-dependent or, in other words, adaptive. However, a detailed overview of existing classic and adaptive coarse spaces in DDMs would be beyond the scope of this review article.

3. Multilayer perceptrons, PINNs, and Deep Ritz. In this section, we briefly provide some preliminaries with respect to multilayer perceptrons, including a short outlook on convolutional neural networks and residual neural networks, see subsection 3.1, as well as PINNs and the Deep Ritz method; see subsection 3.2. For more details on deep learning and more specialized network architectures, we refer to, for example, [9,25,33]. Let us note that for experienced readers, who are already familiar with multilayer perceptrons, PINNs, and the Deep Ritz method, it might be more convenient to skip the following section and proceed directly with section 4.

3.1. Feedforward multilayer perceptrons. In general, neural networks or multilayer perceptrons, respectively, are used within supervised learning to approximate a functional relation between given input and output data. Usually, the overall aim is to minimize a loss function which measures the difference between the ground truth of the given data and the output, that is, estimated classification or regression values, of the neural network. The following description of multilayer perceptrons is loosely based on [36, 78, 93]. Generally, a feedforward neural network can be interpreted as a directed, weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$, and a weight function $w : \mathcal{E} \to \mathbb{R}$; see, for example, [78, Chapt. 20.1]. An

exemplary visualization of the graph of a dense feedforward neural network is presented in Figure 3. A neural network is assumed to be organized in layers, that is, the set of nodes $\mathcal{V}$ can be represented as the union of nonempty, disjoint subsets $\mathcal{V}_i \subset \mathcal{V}, i = 0, \ldots, N+1$. For feedforward neural networks, these sets are defined such that for each edge $e \in \mathcal{E}$ there exists an $i \in \{0, \ldots, N\}$ with e being an edge between a node in $\mathcal{V}_i$ and one in $\mathcal{V}_{i+1}$; see [78, Chapt. 20.1]. In general, different layers can perform different transformations on their input. A neural network usually starts with an input layer (marked in green in Figure 3), where the features $\{i_p\}_{p=1}^P$ of the external data are used as input data, and concludes with an output layer (marked in orange in Figure 3). We denote the layers in between the input and the output layer as hidden layers and the nodes in a neural network as neurons.

Mathematically, the relation between two consecutive layers is the conjunction of a linear mapping defined by the weight function w and a nonlinear activation function α. In particular, the nonlinearity of the activation function enables a neural network to approximate highly complex relations between the input and output data.

Generally, for a generic activation function α, the output of the k-th layer of the neural network can be written as

$$y = \alpha^k(x, W^k, b^k),$$

where $W^k = (w_{ij}^k)_{i,j}$ and b^k are the weight matrix and the bias vector, respectively. Note that an entry w_{ij}^k of W^k corresponds to the value of the weight function w associated with the corresponding edge between layer $\mathcal{V}_{k-1}$ and $\mathcal{V}_k$. Then, the application of a complete neural network with N hidden layers to an input vector $i \in I$ is given by

$$
\begin{aligned}
h^1 &= \alpha^1(i, W^1, b^1), \\
h^{k+1} &= \alpha^{k+1}(h^k, W^{k+1}, b^{k+1}), \quad 1 \leq k < N, \\
o &= (W^{N+1})^T h^N + b^{N+1}
\end{aligned}
$$

(3.1)

where h^k is the output of the k-th hidden layer and $o \in O$ is the (final) output vector. The computation of the output vector o is usually performed without an additional application of the activation function. Note that for specific neural network architectures, the consecutive relation between two layers can be more complicated than written in (3.1). In subsection 4.2, we consider also DDMs for the training of convolutional neural networks (CNNs) and residual neural networks (ResNets). In CNNs, convolutions, that is, kernel functions, and pooling functions are gradually applied to input data with a grid-like structure such that the information within the data is aggregated and condensed throughout the network layers. For more mathematical details on CNNs, we refer to, for example, [56], [25, Chapt. 9], and [9, Chapt. 5]. For ResNets, skip connections that perform identity mappings are added between distant blocks of hidden layers of a multilayer perceptron or a CNN. Hence, the connected block of hidden layers approximates the residual function with respect to the input data which can help to mitigate the degradation problem of very deep neural networks; see [34] for further details.

3.2. PINNs and Deep Ritz. The key concept of PINNs is to integrate domain specific knowledge in form of physical laws or domain expertise modeled by ordinary or partial differential equations into neural networks. In particular, this is done by enhancing the loss function with the residual error of a certain differential equation;
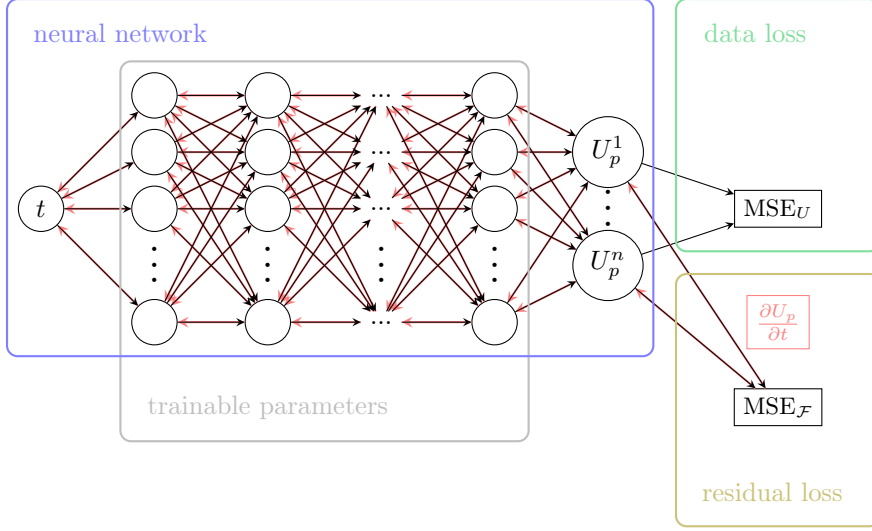
FIGURE 4. *Graphical representation of a PINN. The derivatives needed for the evaluation of the residual are computed using automatic differentiation in a backward propagation (red). Figure in modified form in [27, Fig. 3].*

see [75] for more details. Hence, the objective is to obtain solutions which do not only fit some given training data but also satisfy a given ordinary or partial differential equation in a least square sense. The following descriptions are loosely based on [27, Sect. 4].

For the mathematical derivation of a physics-enhanced loss function, we consider again the general boundary value problem in (2.1) on the domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$ with a linear, second-order, elliptic differential operator $\mathcal{L}$ and boundary conditions $\mathcal{B}$. We now aim to define a PINN which solves (2.1), that is, we aim for a feedforward neural network $\mathcal{N}(\cdot, W, b)$ that fulfills (2.1) in a least square sense with weights W and biases b. As input data of the neural network, we use a set of collocation points located inside the domain Ω as well as a set on the boundary $\partial\Omega$. In order to obtain a neural network $\mathcal{N}$ solving the boundary value problem (2.1), we enhance the classic loss function by a point-wise error of the residual of the PDE. Thus, the physics-informed loss function is defined as

$$
\begin{aligned}
\mathcal{M}(W, b) \quad &:= \mathcal{M}_\Omega(W, b) + \mathcal{M}_{\partial\Omega}(W, b) \\
\mathcal{M}_\Omega(W, b) \quad &:= \frac{1}{N_f} \sum_{i=1}^{N_f} |\mathcal{L}(\mathcal{N}(x_f^i, W, b)) - f(x_f^i)|^2 \\
\mathcal{M}_{\partial\Omega}(W, b) \quad &:= \frac{1}{N_g} \sum_{i=1}^{N_g} |\mathcal{B}(\mathcal{N}(x_g^i, W, b)) - g(x_g^i)|^2,
\end{aligned}
$$
(3.2)

with collocation points x_f^i, $i = 1, ..., N_f$, located in the domain Ω and collocation points x_g^i, $i = 1, ..., N_g$, located on the boundary $\partial\Omega$. During the training of the neural network, the derivatives occurring in the operators $\mathcal{L}$ and $\mathcal{B}$ are evaluated using the backward propagation algorithm and automatic differentiation [4]. Taking advantage of this, the training of the neural network then consists of solving the optimization problem

$$
\{W^*, b^*\} := \arg \min_{\{W, b\}} \mathcal{M}(W, b)
$$
(3.3)

using a stochastic gradient approach based on mini-batches built from all collocation points. An exemplary PINN is shown in Figure 4. Note that the loss term $\mathcal{M}_\Omega(W, b)$ enforces the PINN to satisfy the condition $\mathcal{L}(u) = f$ in a least square sense, while $\mathcal{M}_{\partial\Omega}(W, b)$ enforces the boundary condition.

An approach that is very closely related to PINNs is the Deep Ritz method [19] which uses dense neural networks (DNNs) for solving PDEs in variational form. The Deep Ritz method is based on the Ritz approach to formulate a given PDE as in (2.1) as an equivalent minimization problem which is then discretized and solved by a DNN in combination with a numerical integration method. For a formal description of the Deep Ritz method, we assume that (2.1) is self-adjoint in the following. Then, solving the PDE is equivalent to the solution of the minimization problem

$$(3.4) \qquad \min_u \mathcal{E}(u) \quad \text{s.t.} \quad \mathcal{B}(u) = g \quad \text{on} \quad \partial\Omega$$

where $\mathcal{E}(u)$ is an appropriate energy functional. The basic idea of the Deep Ritz method is now to approximate the function u in (3.4) by a DNN and to use a numerical quadrature rule where each integration point in Ω is used as a collocation point to approximate the minimizing functional $\mathcal{E}(u)$. This yields the discrete loss function for the Deep Ritz method

$$\widetilde{\mathcal{M}}(W, b) := \widetilde{\mathcal{M}}_\Omega(W, b) + \widetilde{\mathcal{M}}_{\partial\Omega}(W, b) \quad \text{with}$$

$$(3.5) \qquad \widetilde{\mathcal{M}}_\Omega(W, b) = \frac{1}{\widetilde{N}_f} \sum_{i=1}^{\widetilde{N}_f} h(\widetilde{x}_f^i, W, b)$$

$$\widetilde{\mathcal{M}}_{\partial\Omega}(W, b) = q \, \frac{1}{\widetilde{N}_g} \sum_{i=1}^{\widetilde{N}_g} |\mathcal{B}(\widetilde{\mathcal{N}}(\widetilde{x}_g^i, W, b)) - g(\widetilde{x}_g^i)|^2,$$

where $h(\cdot)$ is the inner function of the energy functional $\mathcal{E}(u)$ approximated by the DNN $\widetilde{\mathcal{N}}$ and q is a Lagrange multiplier. The derivatives of the neural network $\widetilde{\mathcal{N}}$ occuring in (3.5) are then evaluated using the backward propagation algorithm and automatic differentiation [4]. For a detailed derivation of the loss function (3.5) for the exemplary case of $\mathcal{L}(u) = \Delta u$, we refer to [38, Sect. 3.2].

4. Domain decomposition for machine learning. In this section, we review recent works where ideas from domain decomposition are used to accelerate and parallelize the training of different machine learning models, and, in particular, deep learning models. As described in section 1, the wide range of techniques used in parallel and distributed deep learning can roughly be categorized into *data parallelism*, *model parallelism*, and *pipelining*; see also [5, Sect. 6]. Let us note again that, within this paper, we focus on model parallel training methods and pipelining rather than on data parallelism; see also section 1. In particular, the interpretation of model parallelism as a DDM is, from our point of view, more straightforward and also allows for analogous concepts in distributed ML corresponding to a coarse space in DDMs.

In the following, we will distinguish the different methods using DDMs in ML into two classes. First, we consider all approaches using domain decomposition for the acceleration and parallel training of very specific neural networks, that is, physics-informed neural networks, in subsection 4.1. For details on PINNs, we refer to subsection 3.2. Second, in subsection 4.2, we provide an overview of approaches using domain decomposition in other ML algorithms than PINNs, in particular, classical supervised and unsupervised algorithms.

DD for PINNs	DD	PINN formulation	Interface conditions
cPINN [44]	nonoverlapping	conservation law	conservative quantity
XPINN [40, 43]	nonoverlapping	residuals	averaging
GatedPINN [86]	adaptive	residuals	weighted average
APINN [41]	adaptive	residuals	weighted average
DPINN [17]	nonoverlapping	residuals	flux interface loss
PECANN [3]	nonoverlapping	residuals	generalized Robin-type
Mosaic Flow [23, 92]	overlapping	residuals	alterating Schwarz-like
PINN-PINN, PINN-FOM [85]	overlapping	residuals	loss term/strong enforc.
Iterative algorithms [97]	overlapping	residuals	weighted average
hp-VPINNs [46]	nonoverlapping	var. residuals	via variational loss
TgNN-wf [46]	nonoverlapping	var. residuals	via variational loss
PFNN-2 [81]	overlapping	var. residuals	via boundary network
FBPINN [14, 70]	overlapping	residuals	Schwarz-like
Nonlinear preconditioning [54]	nonoverlapping	residuals	additive/multiplicative

TABLE 1

Summary of the current state of domain decomposition methods to accelerate the training of PINNs as reviewed in subsection 4.1. *We categorize all papers according to the defined domain decomposition, the considered PINN formulation, and the concrete method of how to integrate interface constraints between neighboring subdomains. We mark the papers according to the following color code: dark green: based on a nonoverlapping domain decomposition, blue: based on an overlapping domain decomposition, light green: an adaptive domain decomposition automatically trained by one or several neural networks.*

4.1. Domain decomposition to accelerate the training of physics-aware neural networks. In general, integrating ideas from domain decomposition into PINNs usually aims at accelerating and parallelizing the training of PINNs, mainly by reducing the complexity of the PINN optimization problem for large problem domains and multi-scale problems. Roughly speaking, this means that instead of solving one large optimization problem, the idea is to use a divide-and-conquer strategy by solving, that is, training smaller local problems in parallel. This helps to parallelize the training and, potentially, can also help to reduce the complexity of the global optimization problem. Analogously to DDMs, the resulting related key question within these parallel training approaches is how to ensure that the different local problems communicate with each other such that the local solutions match across the interface. In the following, we review different parallel training methods for different types of PINNs. We will observe that some approaches deal with the communication between different local problems by integrating additional interface terms in the PINN loss function, whereas other approaches define overlapping subdomains and directly exchange information among the overlapping subdomain areas. In Table 1, a structured overview of all papers reviewed in the following can be found.

In [44], conservative PINNs (cPINNs) are introduced that make use of a nonoverlapping domain decomposition approach for conservation laws. Here, the spatial computational domain is decomposed into nonoverlapping subdomains and for each subdomain, a separate local PINN, which also enforces certain interface conditions with its neighboring subdomains with respect to the solution and the local PDE residuals, is trained. In [44], the authors present numerical results for different scalar nonlinear and systems of conservation laws.

The idea of cPINNs is further extended in [43] to generic PDEs and arbitrary space-time domains resulting in extended PINNs (XPINNs). Similar as in [44], local sub-PINNs are trained for local subdomains of a domain independently from each other and in parallel. In particular, in XPINNs, the domain can be decomposed in

both, space and time and no explicit assumptions of a specific PDE type are made. The approximated solutions of the local sub-PINNs are combined by enforcing continuity across the interface by averaging local solutions across neighboring subdomain interfaces as well as by enforcing a residual continuity condition. In [12, Sect. 3.2], the authors propose to additionally enforce continuity of first-order derivatives between different sub-PINNs to further reduce the training error close to the interface.

First steps regarding a direct comparison of PINNs and XPINNs with respect to generalization properties are presented in [40]. Specifically, the authors derive an a priori generalization bound based on the complexity of the considered PDE problem as well as an a posteriori generalization bound based on the matrix norm of the neural network after optimization and use this theory to derive observations with respect to the generalization properties of XPINNs. On the one hand, the decomposition of the global problem into smaller subproblems in XPINNs decreases the complexity of each subproblem and hence can boost generalization. On the other hand, each subproblem is trained with a smaller amount of training data which tends to increase overfitting of the local models. In [40], different experimental results are presented which validate that the two factors in the derived generalization bounds lead to a tradeoff when comparing the performance of XPINNs to PINNs.

In [82], a unified parallel algorithm is presented for cPINNs and XPINNs which employs domain decomposition in space and space-time, respectively. The distributed framework in [82] for the specific PINN extensions from [44] and [43] is constructed by a hybrid programming model using MPI [28, 84]. Within the unified framework, the authors compare the parallel performance of cPINNs and XPINNs for different forward problems. The experimental results indicate that cPINNs are more effective regarding communication cost whereas XPINNs provide greater flexibility such that they can also handle space-time domain decompositions for any differential equations as well as deal with arbitrarily shaped subdomains.

The authors of [86] propose to automatically learn a domain decomposition of a PINN by incorporating conditional computing into the PINN framework in order to learn an arbitrary decomposition of the neural network, which is adaptively tuned during the training process. Here, conditional computing means that only certain neurons or units of a neural network are activated depending on the network input; see also [6, 79] for more details. Based on the idea of conditional computing the authors introduce GatedPINNs which rely on a number of *experts* which decompose the neural network and are each modeled by a simple neural network themselves. The outputs of the different experts are then aggregated by a linear or nonlinear *gate network*. The authors provide comparative experimental results for the GatedPINN and standard PINN approach and are able to reduce the training time significantly.

A related approach was recently introduced in [41], where also a *gate network* is trained to find an augmented domain decomposition and flexibel parameter sharing. Concretely, the gate network does not need any specific interface losses and as output data, provides a weight average of several local subnets. Additionally, it uses a flexible partial parameter sharing of the subnets and initializes the subnets with the hard decomposition obtained from XPINN, resulting in an augmented PINN (APINN). The authors of [41] provide results for different PDEs comparing APINN to XPINN and classic PINNs showing that APINNs result in an improved performance and better generalization properties.

In [17], the authors use ideas from finite volume methods where the computational domain is partitioned into multiple cells to define a distributed PINN (DPINN) for the data-efficient solution of partial differential equations. Here, the authors decompose

the computational domain into nonoverlapping cells, which can also be interpreted as the subdomains in a DDM, and train a separate, local PINN for each cell with a loss function based exclusively on collocation points from the interior of the cell. Each of these local PINNs is relatively simple and, for the results presented in [17], only two layers deep. In addition to the PINN losses for each cell, a loss term for the interface conditions is introduced which is associated with collocation points located on the boundary of the cells. This interface loss is inspired by the flux conditions of the finite volume method. Then, the DPINN approach is based on training all local PINNs together by minimizing the sum of all local losses plus the interface loss. In particular, the described approach requires the exchange of interface information in each step of the optimization approach used in the training of the network.

A related approach which mostly differs by the treatment of the boundary conditions between neighboring subdomains is presented in [3]. There, the authors use an optimized Schwarz-type nonoverlapping domain decomposition for solving forward and inverse PDE problems using PINNs. Within each subdomain, physics and equality constrained artificial neural networks (PECANNs) [2] are trained in parallel as solvers for each subdomain. Moreover, generalized Robin-type interface transmission conditions are considered as an additional constraint on the solution of each subdomain to ensure continuity. Hence, local information are exchanged at the end of each local training within each outer iteration of the described method. The authors of [3] show that by incorporating the Robin-type interface constraints into the iterative learning process, the effectiveness and reliability of the method can be enhanced.

In [23], an end-to-end parallelization of the physics-informed neural PDE solver Mosaic Flow [92] based on domain decomposition is presented. Roughly speaking, Mosaic Flow is an iterative algorithm inspired by the alternating Schwarz method designed for solving PDEs with arbitrary boundary conditions on diverse domains. It basically consists of two steps. First, local physics-informed PDE solvers (SDNets) are pretrained on small domains with arbitrary boundary conditions, using the discretized boundary function as input data; see also [92] for more mathematical details. Then, in a second step, the Mosaic Flow predictor is trained which decomposes a large domain into overlapping subdomains and uses the pretrained SDNets as subdomain solvers. In [23], the authors combine data parallel training and domain parallelism for inference to develop a distributed domain decomposition algorithm for Mosaic Flow to achieve strong scalability for its parallel training on GPUs. Numerical results for the parallel solution of the Laplace equation on large domains, distributed on 32 GPUs, are presented.

A related approach which is also inspired by the alternating Schwarz method is presented in [85]. In [85], the authors use the alternating Schwarz method and an overlapping domain decomposition to couple local PINNs and full order models, that is, local solutions obtained via the finite element, finite difference, or finite volume method with each other to accelerate the training of PINNs. Furthermore, three different approaches to enforce Dirichlet boundary conditions within the local subdomain PINNs are empirically compared. Concretely, the three different approaches are the following: A weak boundary condition enforcement through the loss function, a strong enforcement through a solution transformation, and a combination of both such that the boundary conditions are enforced strongly on the computational domain's boundary and weakly on the subdomains' boundaries. Numerical examples are provided for the one-dimensional steady state advection-diffusion equation with focus on the advection-dominated regime (high Péclet number). The presented results show that the convergence of the Schwarz method is strongly linked to the chosen boundary

condition implementation within the local, coupled PINNs.

A different iterative approach to parallelize the training of PINNs while also reducing the communication cost is presented in [97]. In [97], the authors propose an iteration method which is based on the classical additive Schwarz method. Hence, similarly as in [58, 60], an outer training loop for the global network is defined such that, in each iteration of the outer training loop, decoupled smaller local problems are solved which are then stitched back together by a weighted average. In each outer iteration, the local problems obtain their boundary values from the current global iteration and an overlapping decomposition of the collocation points into subdomains is used. To compute the next iterate of the global problem, a weighted sum of the previous iterate and the partitioned neural network solution using a relaxation parameter is computed. Additionally, to further improve the convergence properties of the described approach, the authors of [97] propose a two-level extension by solving an additional coarse problem with specific structure in each outer iteration. Note that related preliminary work has also been published in [49, 96] by the authors. In [49], first experimental results for the proposed additive Schwarz algorithm have been presented whereas [96] contains first results regarding the convergence analysis of the approach.

In [46], the authors introduce hp-variational PINNs (hp-VPINNs) which are based on the variational formulation of the residuals of the considered PDE, similar to the approach considered in [58]. In particular, the authors use piecewise polynomial test functions within the variational formulation of the considered PDE such that each test function v_j is always a polynomial of a chosen order over the subdomain j and zero otherwise. Let us note that with respect to the implementation of this approach, the authors employ a single DNN to approximate the solution over the whole computational domain despite virtually decomposing the domain into several subdomains; see also [46, Remark 2.1]. Due to the use of a single DNN no subdomain interface conditions have to be considered in the loss function. However, this makes it more difficult to parallelize hp-VPINNs. Different numerical examples of function approximation for continuous and discontinuous functions as well as for solving the Poisson equation are presented.

An approach which is closely related to [46] and which is also based on the weak, that is, the variational form of a given system of PDEs is presented in [95]. The idea of the presented neural networks is to integrate the weak formulation of the PDE in the loss function as well as data constraints and initial or boundary conditions. Due to the theory guided structure of the networks based on the weak formulation of the considered PDE the authors refer to their approach by the acronym TgNN-wf. In particular, similar to [46], the authors use locally defined test functions to perform a domain decomposition of the computational domain to accelerate the training process. As a further novelty, the authors formulate the original loss minimization problem into a Lagrangian duality problem in order to optimize the penalty term parameters of the different components of the loss function within the training process. The authors provide comparative results for the strong form TgNN and TgNN-wf for an unsteady-state 2D single-phase flow problem and a 1D two-phase flow problem in terms of accuracy, training time, and robustness with respect to noise in the training data.

In [81], the authors use an overlapping domain decomposition approach for the training process of penalty-free neural networks (PFNN), originally introduced in [80]. Applying the ideas introduced in [80] for PFNN, the resulting new PFNN-2 adopts two neural networks, with one network learning the initial condition and essential boundary conditions of a given PDE problem, and a second network approximating

the solution of the PDE in its variational form on the remaining part of the domain. To speedup the learning process, the authors introduce an overlapping decomposition of the domain into subdomains and train two local networks, that is, one for the initial and boundary conditions and a second one for the local interior solution, for each subdomain. This approach requires direct exchange of information between neighboring subdomains to enforce continuity via the locally trained networks which learn the boundary conditions of each subdomain.

In [70], the authors transfer ideas from classical finite element methods where the solution of a PDE or an ODE is expressed as a sum of a finite set of basis functions with a compact support to PINNs. The resulting method uses local window functions to form a global function, that is, the global solution on the entire domain, and is referred to as Finite Basis Physics-Informed Neural Network (FBPINN). The global function is hence expressed as a sum of the localized window functions which are learned by local neural networks. In particular, the local window or basis functions are defined over small, overlapping subdomains such that, in FBPINNs, no additional interface condition is necessary. In [70], the authors present numerical results for both small and large, multi-scale problems.

The idea of FBPINNs is further developed in [14] and [15]. In [14], additive, multiplicative, and hybrid iteration methods based on a Schwarz-like domain decomposition method for the training of FBPINNs are introduced. In [15], these ideas are extended even further by adding multiple levels of domain decompositions to the solution ansatz, similar as in classical multilevel Schwarz methods. This results in improved convergence properties and increased accuracies in the considered PDE solutions and, at the same time, mitigates convergence problems related to the spectral bias of neural networks.

The performance and convergence properties of L-BFGS [62] optimizers used for the training of PINNs are enhanced in [54] by using nonlinear preconditioning strategies based on Schwarz-like domain decomposition approaches. More specifically, nonlinear additive and multiplicative preconditioners are constructed by decomposing the network model, that is, the corresponding parameters in a layer-wise manner. This is, in principle, different from the aforementioned approaches where ideas from DDMs are used to decompose the computational domain. The authors present results for both additive and multiplicative nonlinear preconditioners for different benchmark problems, indicating that the proposed layer-wise decomposition of the network can significantly accelerate the convergence of the L-BFGS optimizer.

4.2. Domain decomposition for classical supervised and unsupervised machine learning algorithms. Apart from the wide application area of DDMs for PINNs, the idea of domain decomposition has also been transferred to other machine learning algorithms of which we try to give a broad overview in this section. Note that the idea of DDMs is, on the one hand, applied to the training of different supervised machine learning models, for example, convolutional or residual neural networks, and, on the other hand, also to unsupervised algorithms, as, for example, principle component analysis (PCA); see Figure 5 for a schematic overview.

In [29, 30], the idea of DDMs is transferred to the training of CNNs for different image recognition problems. The authors propose to decompose a global CNN along its width into a finite number of smaller, local subnetworks which can be trained independently from each other. In particular, the local CNNs are trained in parallel without any communication between the different processes. The obtained weights of the local subnetworks are then used as an initialization for the subsequently trained
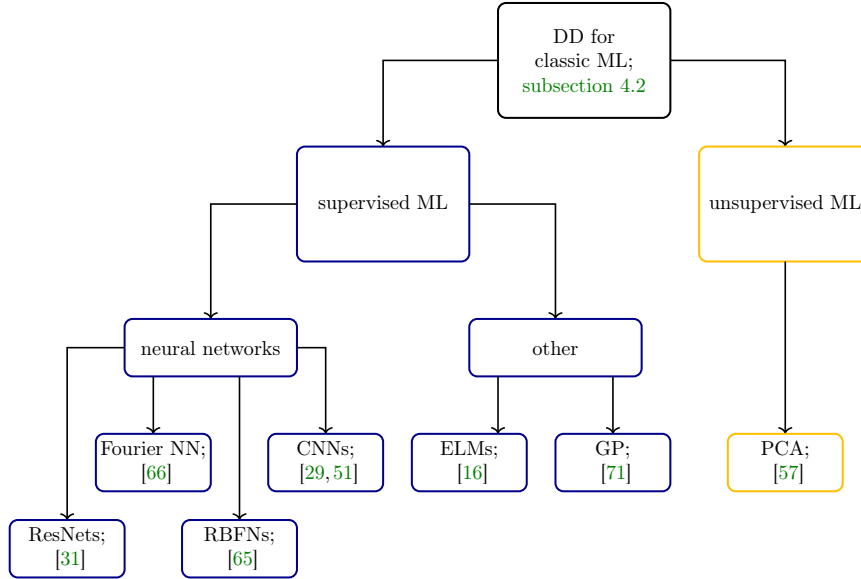
FIGURE 5. *Schematic representation of the articles reviewed in subsection 4.2 with respect to domain decomposition methods in machine learning.*

global network by using a transfer learning strategy. The approach can be interpreted as an analogy to nonlinear preconditioning [63] and the decomposition of the inputs of the global CNN along its width corresponds to a decomposition of the input images into smaller partial images. The authors present results for different two-dimensional image data indicating that the suggested idea can significantly accelerate the training of the global network on a GPU cluster.

A slightly similar but different parallel training approach also for CNNs in image recognition has been proposed in [51]. In [51], the authors decompose two- or three-dimensional image data into nonoverlapping or overlapping subsets and train smaller CNNs operating exclusively on parts of the decomposed input data. Based on the decomposed image data, all local CNNs provide a probability distribution with respect to the different classes of the underlying image classification problem. Then, in a second step, a global coarse DNN is trained which evaluates the local decisions into a final, global decision. In particular, this approach does not require the training of a global, large CNN operating on the entire images and the smaller, local CNNs can be trained in parallel on different GPU clusters. The results provided for two- and three-dimensional image data show a significant acceleration of the training and additionally, provide at least the same classification accuracies compared to the training of a single, global CNN for almost all tested datasets.

Another model parallel training approach based on domain decomposition, that is, a layer-parallel training approach has been suggested in [31] for ResNets [34]. In this work, the authors make use of the fact that the forward propagation through a ResNet can be interpreted as a forward Euler discretization of a time-dependent ordinary differential equation (ODE) [18] where the time-dependent control variables represent the weights of the neural network. Hence, the time domain in the time-dependent ODE corresponds to the layer domain of a ResNet. Based on this interpretation, the authors employ a multigrid reduction in time approach [22] to replace the classic

forward and backward propagation in the training progress by a parallel nonlinear multigrid iteration applied to the layer domain such that chunks of layers can be processed in parallel. In [31], scalability and speedup results for this method are presented for three different image recognition problems.

In [66], domain decomposition techniques are used to define a parallel training process of trigonometric neural networks which use trigonometric activation functions. Based on a nonoverlapping domain decomposition, three local neural networks are trained for each subdomain. The first neural network, denoted as primary neural network is completely local, the second one, denoted as boundary neural network, learns for each subdomain the interface conditions with its neighbors, and the third one, denoted as modifier neural network, then learns the update for the primary network including the interface conditions. Finally, the local solution is defined as the sum of the primary and modifier network. This process is repeated for each subdomain until convergence.

Another work on the combination of DD and a special type of neural networks is presented in [65]. The authors combine different types of radial basis function networks (RBFNs) [77], which typically use only one hidden layer, with the concept of domain decomposition to approximate nonlinear functions or to solve Poisson's equation on a rectangular domain. The computational domain is decomposed into nonoverlapping subdomains and each of the subdomains is discretized by a shallow RBFN. Based on this decomposition, the authors propose an iterative solution algorithm where, in each iteration, all local subproblems are solved using the RBFNs. Then, in each iteration of the outer training loop, the interface condition between the subdomains is estimated and updated using boundary integral equations.

In [16], the idea of extreme learning machines (ELMs) [42] is combined with domain decomposition and local neural networks. Basically, in order to compute the solution for a given PDE, the authors of [16] divide the domain into subdomains and represent the solution on each of these subdomains by a local feedforward neural network. To obtain a global solution, continuity constraints are imposed along the subdomain boundaries. Originating from the idea of ELMs, each local neural network consists of a small number of hidden layers, while its last hidden layer is typically wide. As in ELMs, the weights and bias coefficients in all hidden layers of the local neural networks are pre-set to random values and, during training, only the weight coefficients of the output layers of the local neural networks are training parameters. Moreover, also in analogy to ELMs, the overall neural network is trained by a linear or nonlinear least squares computation instead of using the backpropagation algorithm. The authors compare the presented method with the deep Galerkin method (DGM), the PINN method, and classical FEM.

In [71], the idea of domain decomposition is not transferred to a special type of neural network but, instead, to Gaussian Process (GP) regression [94]. Given that the computational complexity $O(N^3)$, where N is the number of training samples, is often a major limitation in GP regression, the main idea of the authors is to decompose the optimization problem related to the GP regression into smaller local optimization problems that provide local predictions. This is in analogy of decomposing a domain into smaller subdomains. To deal with the possible mismatch problem of the local predictions along the boundaries of neighboring subdomains, the authors of [71] impose continuity constraints, analogously to interface conditions in DDMs.

In [57], the authors propose a local PCA approach which also relies on the idea of DDMs. The PCA is applied to different images which are decomposed into nonoverlapping smaller subimages. Here, instead of computing the principal components for
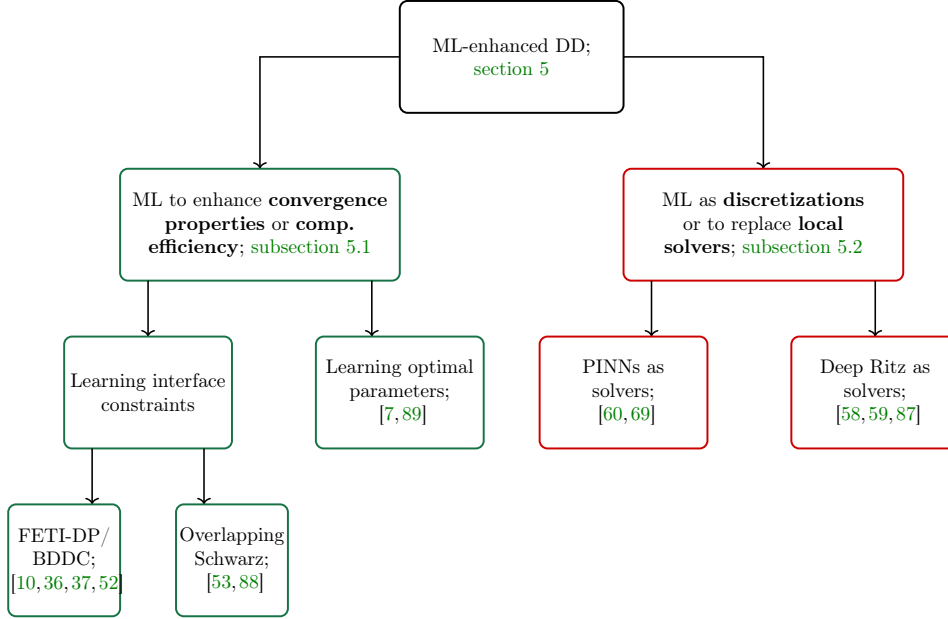
FIGURE 6.   *Schematic representation of the articles reviewed in section 5 with respect to machine learning-enhanced domain decomposition methods.*

the entire images, only principal components for the different subimages are determined. Then, in a second step, the local principal components are used to compute low-dimensional projections of the different subimages which are composed again to a lower dimensional representation of the original global image. The authors show empirically that the decomposed PCA requires a smaller number of principal components to provide an accurate low-dimensional representation of the original image compared to the global PCA. Moreover, the authors provide a mathematical proof that the local PCA reduces the summation pollution effect of the global PCA and thus, results in a lower or equal reconstruction error compared to the global PCA. In other words, the local PCA is never worse than the global PCA; see [57] for more details.

5. Machine learning-enhanced domain decomposition methods. In this section, we review recent work which uses different, mainly supervised, machine learning algorithms to enhance different DDMs. In general, one can divide the approaches using ML to enhance DDMs into two classes; see also [38]. The first class consists of methods where ML is used to improve the convergence properties or the computational efficiency within classical DDMs. Typically, this is done by learning or approximating optimal interface conditions or by learning other optimal parameters. Existing approaches of this class are considered in subsection 5.1. In the second class, different types of neural networks are used as discretization methods and replace classical local subdomain or coarse solvers based on finite elements or finite differences. Existing methods of this class are summarized in subsection 5.2. For a schematic overview of the existing literature, see Figure 6.

5.1. Machine learning to enhance convergence properties or computational efficiency of domain decomposition methods. As explained in section 2,

introducing appropriate global communication and interface conditions between subdomains in a DDM is a crucial task to ensure the correct and globally continuous solution as well as a satisfactory convergence behavior. Hence, a number of different attempts have been made to automatically construct such interface conditions, for example, by using regression neural networks.

In [10], the authors propose an approach to learn an adaptive coarse space for BDDC (Balancing Domain Decomposition by Constraints) algorithms for stochastic elliptic PDEs. The authors focus on a specific adaptive coarse space [47, 48] where local generalized eigenvalue problems have to be solved for parts of the interface between neighboring subdomains. In particular, these eigenvalue problems depend on the (stochastic) coefficients of the PDE. Hence, the authors train a dense feedforward neural network to approximate the relation between the stochastic PDE coefficients and the adaptive coarse space. As an input for the neural networks, a truncation of the Karhunen-Loève expansion [98] is used. Numerical results are presented for different oscillatory and high contrast coefficients, a fixed discretization of the domain $\Omega \subset \mathbb{R}^2$, and a neural network with a single hidden layer. Note that, so far, this method is not independent of the discretization and thus has to be retrained for different mesh resolutions.

As a preparatory step to automatically construct an adaptive coarse space for DDMs, in [36, 37], the authors present a supervised classification approach to identify parts of the interface where adaptive coarse constraints are necessary. Then, adaptive constraints resulting from the local generalized eigenvalue problem in [67, 68] are integrated into a FETI-DP (Finite Element Tearing and Interconnecting - Dual Primal) coarse space exclusively for edges or faces, respectively, which are categorized as critical by a pretrained neural network. In particular, the decision of the neural network is based on a sampling procedure of the coefficient function of the underlying elliptic PDE and is independent of the finite element discretization. Hence, the trained neural network can be evaluated for different finite element meshes as used in the training procedure and can be applied to both, two- and three-dimensional problems. Additionally, in [39], the described approach is also analogously applied to the AGDSW (Adaptive Generalized Dryja-Smith-Widlund) method [35] in two spatial dimensions.

In [52], the work described above has been extended to directly learn the adaptive constraints to setup a robust FETI-DP coarse space in two spatial dimensions. Here, the authors train a fixed number of regression neural networks to directly predict an approximation of adaptive edge constraints. Again, the input data used for the neural networks rely on a mesh-independent sampling strategy such that the trained neural networks can be applied to different finite element mesh resolutions. Numerical results are provided for different elliptic PDE problems and realistic coefficient distributions orginating from a microsection of a dual-phase steel. In [50], these results are further extended to irregular decompositions obtained by the graph partitioning software METIS [45].

In order to provide a robust and efficient iterative solution method for the time-harmonic Maxwell's equations, in [53], the authors use a feedforward neural network to approximate the interface operator in an optimized Schwarz method [13]. The concrete formulation of the optimized Schwarz method for the time-harmonic Maxwell's equations is based on [21]. In [53], preliminary numerical results for a two-dimensional domain decomposed into two subdomains are provided. Additionally, comparative results for different network architectures and different activation functions are presented and compared with classical DDMs.

A related approach to learn optimal interface conditions also for an optimized

Schwarz method [24] using Graph Convolutional Neural Networks (GNNs) is suggested in [88]. There, the authors aim to learn the subdomain interface matrices for both, structured grids where the optimal matrix values can be predicted by Fourier analysis and for unstructured grids where the optimal Schwarz parameters are generally not known. Hence, the GNNs are trained using as input the degrees of freedom of a discretized problem, its decomposition, and sparsity constraints on the subdomain interface matrices. As output, the networks predict approximated optimized values for these matrices. Additionally, the authors consider an improved loss function which is a relaxation of the ideal Frobenius norm minimization.

In [89], a new multigrid GNN (MG-GNN) architecture to learn optimized parameters in two-level DDMs is introduced. In particular, the new MG-GNN model is used to learn the Robin-type subdomain boundary conditions in a two-level optimized Schwarz method and the overall coarse-to-fine interpolator. In a way, the work in [89] can thus be seen as an extension of the method in [88], where also GNNs are used to learn optimal interface conditions in a one-level optimized Schwarz method.

A slightly different approach, where not adaptive constraints themselves but interface conditions in form of varying overlap between neighboring subdomains are learned is presented in [7]. More precisely, a method is suggested to optimize the width δ of the overlap in Schwarz methods such that the number of expected floating point operations until convergence is minimized. The authors compare four different regression algorithms for two-dimensional diffusion problems with jumps in the coefficient function. As input for the different regression algorithms, the authors use certain features collected for each subdomain as well as its neighborhood, as, for example, the maximal or minimal coefficient within certain sets of rows and columns of degrees of freedom in the surrounding of the boundary of the overlapping subdomain.

5.2. Replacing the subdomain solvers in domain decomposition methods by neural networks. In addition to learning optimal interface conditions to improve the robustness of DDMs, a different approach that has attracted much attention very recently is to replace the subdomain solvers by machine learning algorithms as, for example, different types of neural networks. As a starting point for this section, we consider again boundary value problem (2.1) and, in the following, review some works which aim to solve the local subdomain problems by training a neural network $\mathcal{N}$ with certain collocation points and the loss function given in (3.2) or (3.5), respectively.

In [60] and [58], the subdomain solvers in a parallel overlapping Schwarz method are replaced by PINNs and the Deep Ritz method, respectively, such that each of these local networks solves a single subdomain problem. The latter method is named D3M in [58] and the former method DeepDDM in [60]. As explained above, in any DDM, it is crucial to exchange information between the local subdomains in order to obtain a continuous, global solution. In DeepDDM [60] as well as in D3M [58], which are both based on a parallel overlapping Schwarz fixed point iteration, the exchange of information is enforced by additional boundary conditions, which change in each fixed-point iteration until convergence; see [61] or [90] for further details on the parallel overlapping Schwarz method. These additional boundary conditions result in a third loss term of the local subdomain networks with additional collocation points for the exchange of information between neighboring subdomains.

In [69], the authors extend the DeepDDM algorithm [60] described above by a coarse space correction, similarly to what is done in traditional two-level DDMs [90]. The aim is to improve the convergence properties of the resulting algorithm for an

increasing number of subdomains due to the enhanced exchange of global information. To define a coarse problem, the authors of [69] introduce an additional coarse sampling of the complete domain defined by a set of coarse collocation points. These coarse collocation points are then used as input data for an additional fully connected network that is trained subsequently after the local networks in each outer training iteration. The authors present numerical results for different elliptic boundary value problems indicating the advanced numerical scalability for an increasing number of subdomains compared to the original DeepDDM approach.

A different extension of the D3M approach [58] is presented in [59]. Here, the aim is to mitigate the spectral bias [64, 74, 76] observed with the local networks in the original D3M approach by introducing new multi Fourier feature networks (MFFNet) in each local subdomain. This results in a new Fourier feature based deep DDM (F-D3M) which is more successful in approximating the high frequency modes of a PDE solution. In particular, in order to obtain satisfactory approximations for both, low and high frequency modes, the MFFNet consists of different subnetworks with different random frequencies of the Fourier features and whose outputs are concatenated with a linear layer to compute the final output of the MFFNet.

In [87], the subdomain solvers within a DDM are also replaced by a Deep Ritz approach but, in contrast to [58], the computational domain is decomposed into nonverlapping subdomains. In contrast to replacing the subdomain solvers by neural networks within an overlapping Schwarz method as done in D3M or DeepDDM, in its nonoverlapping counterpart a direct flux exchange across subdomain interfaces is required. In particular, simply replacing the subdomain solvers in nonoverlapping subdomains by the classic Deep Ritz approach often results in a local minimizer where the given Dirichlet boundary conditions are satisfied satisfactory but with inaccurate Neumann boundary conditions; see [87]. In order to address the issue of the low accuracy of the related flux transmission between neighboring nonoverlapping subdomains, the authors of [87] propose a compensated Deep Ritz method as subdomain solvers that enable reliable flux transmission in the presence of erroneous interface conditions. The compensated Deep Ritz method defines a weighted sum of the Ritz form of the loss function and the PINN residual loss given that PINNs are empirically found to provide more accurate estimates of partial derivatives. The authors provide numerical examples for elliptic problems and two and four subdomains.

6. Discussion and future work. In this paper, we have explored how established techniques and ideas from DDMs can effectively be combined with machine learning algorithms to develop major advances in SciML methods. In particular, we have categorized all reviewed work in the two main classes of DDMs for machine learning and machine learning-enhanced DDMs. By taking all cited work into account, we observe that, so far, higher efforts exist to implement ideas from DDMs into machine learning models than vice versa. However, in both areas, several ongoing challenges and opportunities for further developments and research exist.

In the field of DDMs for machine learning, a very large group of publications exists that use DDMs within physics-aware neural networks. Here, we have observed a strong focus of using DDMs as model or data parallelism to efficiently accelerate and distribute the training of physics-aware neural networks, or, in particular, PINNs. Furthermore, decomposing the training of one large global neural network into training several smaller neural networks in parallel can also have important impacts on the generalization properties of the underlying machine learning model. As investigated analytically in [40], decomposing the global optimization problem into smaller

subproblems decreases the complexity of each subproblem which can improve the generalization of the resulting global model. On the other hand, each subproblem is trained with a smaller amount of training data which tends to increase overfitting of the local models. Thus, a tradeoff between both effects has to be considered. Additionally, training decoupled local PINN models in a preliminary phase which are subsequently aggregated into a global model can also help to mitigate the spectral bias often observed for deep neural networks.

Besides, concepts from DDMs have also been integrated into classical machine learning models, for example, CNNs for image recognition, ResNets, Gaussian Processing, or PCA. Here, the primary objective is also to distribute the training of the respective machine learning model to different GPUs in order to accelerate the required training time. As a secondary effect, also for DDMs in classical machine learning models, in some cases, an improvement of the accuracy of the trained model can be observed. For some models, a mathematical comparison of the original global model versus the decomposed model already exists, for example, [29, 57], whereas for other cases, a mathematical proof of a corresponding correlation is still an open research question.

Within the second reviewed field of machine-learning enhanced DDMs, a range of work exists on learning interface conditions or restriction operators by regression neural networks. So far, all of these approaches are developed for concrete domain decomposition methods and test problems, for example, for FETI-DP and BDDC or optimized Schwarz methods for stationary diffusion problems, stochastic PDEs, or Maxwell's equations, respectively. Here, a possible priority for future work could be the aim to develop more universal approaches which can be applied to various problems and DDMs or are independent of the underlying finite element discretization. Currently, only some of the cited approaches partly fulfill such requirements. Additionally, some work on learning optimal parameters in DDMs exist, as, for example, learning the optimal width of overlap for overlapping subdomains, optimal stopping criterions, or optimal restriction operators. These efforts, considering the current state, are also restricted to very concrete cases where specific information with respect to the considered DDM and test problem are used as input data for different regression models. Thus, in this area, more generally valid approaches are also a topic of potential future research.

Finally, there are several efforts to use PINNs or Deep Ritz methods to replace the subdomain solvers or the discretization within classical DDMs. These approaches make use of the universal approximation capabilities of neural networks to represent, under certain assumptions on the activation function, etc., solutions of PDEs. However, there are some challenges within these approaches in order to be competitive with classical numerical methods for the solution of PDE problems. Most notably, most existing approaches require to be retrained for varying boundary and initial conditions which makes the training of such approaches computationally expensive and limits the applicability of the resulting models. Hence, practically relevant applications for such models have to be carefully thought of, for example, large-scale multi-physics problems, and more flexible surrogate models will much likely be an important topic of future research.

Neural PDE solvers or, more general, neural operators can offer a remedy for the described drawback that PINNs or Deep Ritz usually need to be retrained for varying boundary conditions. Neural operators are a type of model that is trained to approximate the PDE solution operator and to be evaluated for various instances of a boundary value problem with different boundary conditions. Practically, this means

that the neural operator is trained with a representative set of discretized boundary functions as input data. With respect to the aim of applying neural operators to large, complex domains with arbitrary boundary conditions, some efforts have been made to combine neural operators with ideas from DDMs to enable the efficient solution of complex boundary value problems and a scalable training of these models. The different avenues of ongoing research, so far, still face several challenges, for example, with respect to applicability to irregular meshes, complex PDE solutions, and computationally expensive training. Hence, combining DDMs and other promising concepts as, for example, transformers with neural operators [32] to address these challenges is likely to be a future research topic of rising interest and importance.

REFERENCES

[1] N. BAKER, F. ALEXANDER, T. BREMER, A. HAGBERG, Y. KEVREKIDIS, H. NAJM, M. PARASHAR, A. PATRA, J. SETHIAN, S. WILD, AND K. WILLCOX, *Brochure on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence. USDOE Office of Science (SC) (United States)*, (2018), https://doi.org/10.2172/1484362, https://www.osti.gov/biblio/1484362.

[2] S. BASIR AND I. SENOCAK, *Physics and equality constrained artificial neural networks: Application to forward and inverse problems with multi-fidelity data fusion*, Journal of Computational Physics, 463 (2022), p. 111301.

[3] S. BASIR AND I. SENOCAK, *A Generalized Schwarz-type Non-overlapping Domain Decomposition Method using Physics-constrained Neural Networks*, July 2023, http://arxiv.org/abs/2307.12435.

[4] A. G. BAYDIN, B. A. PEARLMUTTER, A. A. RADUL, AND J. M. SISKIND, *Automatic differentiation in machine learning: a survey*, The Journal of Machine Learning Research, 18 (2017), pp. 5595–5637.

[5] T. BEN-NUN AND T. HOEFLER, *Demystifying Parallel and Distributed Deep Learning: An In-depth Concurrency Analysis*, ACM Computing Surveys, 52 (2020), pp. 1–43.

[6] E. BENGIO, P.-L. BACON, J. PINEAU, AND D. PRECUP, *Conditional computation in neural networks for faster models*, 2015, https://arxiv.org/abs/1511.06297.

[7] S. BURROWS, J. FROCHTE, M. VÖLSKE, A. B. M. TORRES, AND B. STEIN, *Learning overlap optimization for domain decomposition methods*, in Advances in Knowledge Discovery and Data Mining, Springer, 2013, pp. 438–449.

[8] Y. CHEN, L. LU, G. E. KARNIADAKIS, AND L. DAL NEGRO, *Physics-informed neural networks for inverse problems in nano-optics and metamaterials*, Optics express, 28 (2020), pp. 11618–11633.

[9] F. CHOLLET, *Deep learning with Python*, Simon and Schuster, 2017.

[10] E. CHUNG, H.-H. KIM, M.-F. LAM, AND L. ZHAO, *Learning adaptive coarse spaces of BDDC algorithms for stochastic elliptic problems with oscillatory and high contrast coefficients*, Mathematical and Computational Applications, 26 (2021).

[11] S. CUOMO, V. S. DI COLA, F. GIAMPAOLO, G. ROZZA, M. RAISSI, AND F. PICCIALLI, *Scientific machine learning through physics–informed neural networks: Where we are and what's next*, Journal of Scientific Computing, 92 (2022), p. 88.

[12] T. DE RYCK, A. D. JAGTAP, AND S. MISHRA, *Error estimates for physics-informed neural networks approximating the Navier–Stokes equations*, IMA Journal of Numerical Analysis, (2023).

[13] V. DOLEAN, M. J. GANDER, AND L. GERARDO-GIORDA, *Optimized Schwarz methods for Maxwell's equations*, SIAM Journal on Scientific Computing, 31 (2009), pp. 2193–2213.

[14] V. DOLEAN, A. HEINLEIN, S. MISHRA, AND B. MOSELEY, *Finite basis physics-informed neural networks as a Schwarz domain decomposition method*, May 2023, http://arxiv.org/abs/2211.05560.

[15] V. DOLEAN, A. HEINLEIN, S. MISHRA, AND B. MOSELEY, *Multilevel domain decomposition-based architectures for physics-informed neural networks*, June 2023, http://arxiv.org/abs/2306.05486.

[16] 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.

[17] V. DWIVEDI, N. PARASHAR, AND B. SRINIVASAN, *Distributed physics informed neural network*

for data-efficient solution to partial differential equations, July 2019, http://arxiv.org/abs/1907.08967.

[18] W. E, *A proposal on machine learning via dynamical systems*, Communications in Mathematics and Statistics, 1 (2017), pp. 1–11.

[19] 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.

[20] M. EICHINGER, A. HEINLEIN, AND A. KLAWONN, *Surrogate convolutional neural network models for steady computational fluid dynamics simulations*, Electronic Transactions on Numerical Analysis, 56 (2022), pp. 235–255.

[21] M. EL BOUAJAJI, B. THIERRY, X. ANTOINE, AND C. GEUZAINE, *A quasi-optimal domain decomposition algorithm for the time-harmonic Maxwell's equations*, Journal of Computational Physics, 294 (2015), pp. 38–57.

[22] R. D. FALGOUT, S. FRIEDHOFF, T. V. KOLEV, S. P. MACLACHLAN, AND J. B. SCHRODER, *Parallel time integration with multigrid*, SIAM Journal on Scientific Computing, 36 (2014), pp. C635–C661.

[23] A. FEENEY, Z. LI, R. BOSTANABAD, AND A. CHANDRAMOWLISHWARAN, *Breaking Boundaries: Distributed Domain Decomposition with Scalable Physics-Informed Neural PDE Solvers*, Aug. 2023, http://arxiv.org/abs/2308.14258.

[24] M. J. GANDER, L. HALPERN, F. NATAF, ET AL., *Optimized Schwarz methods*, in Twelfth International Conference on Domain Decomposition Methods, Chiba, Japan, Citeseer, 2001, pp. 15–28.

[25] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep learning*, MIT press Cambridge, 2016.

[26] V. GRIMM, A. HEINLEIN, AND A. KLAWONN, *Learning the solution operator of two-dimensional incompressible Navier-Stokes equations using physics-aware convolutional neural networks*, arXiv preprint arXiv:2308.02137, (2023).

[27] V. GRIMM, A. HEINLEIN, A. KLAWONN, M. LANSER, AND J. WEBER, *Estimating the time-dependent contact rate of SIR and SEIR models in mathematical epidemiology using physics-informed neural networks*, Electronic Transactions on Numerical Analysis, 56 (2022), pp. 1–27.

[28] W. D. GROPP, S. HUSS-LEDERMAN, AND A. LUMSDAINE, *MPI-The Complete Reference, Volume 2: The MPI Extensions*, MIT Press, Cambridge, MA, USA, 1998.

[29] L. GU, W. ZHANG, J. LIU, AND X.-C. CAI, *Decomposition and composition of deep convolutional neural networks and training acceleration via sub-network transfer learning*, ETNA - Electronic Transactions on Numerical Analysis, 56 (2022), pp. 157–186.

[30] L. GU, W. ZHANG, J. LIU, AND X.-C. CAI, *Decomposition and Preconditioning of Deep Convolutional Neural Networks for Training Acceleration*, in Domain Decomposition Methods in Science and Engineering XXVI, vol. 145, Springer International Publishing, Cham, 2022, pp. 153–160. Series Title: Lecture Notes in Computational Science and Engineering.

[31] S. GUNTHER, L. RUTHOTTO, J. B. SCHRODER, E. C. CYR, AND N. R. GAUGER, *Layer-parallel training of deep residual neural networks*, SIAM Journal on Mathematics of Data Science, 2 (2020), pp. 1–23.

[32] Z. HAO, Z. WANG, H. SU, C. YING, Y. DONG, S. LIU, Z. CHENG, J. SONG, AND J. ZHU, *GNOT: A general neural operator transformer for operator learning*, in International Conference on Machine Learning, PMLR, 2023, pp. 12556–12569.

[33] S. HAYKIN, *Neural networks and learning machines*, Pearson Education India, 3. ed., 2010.

[34] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep Residual Learning for Image Recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[35] A. HEINLEIN, A. KLAWONN, J. KNEPPER, AND O. RHEINBACH, *An adaptive GDSW coarse space for two-level overlapping Schwarz methods in two dimensions*, in Domain Decomposition Methods in Science and Engineering XXIV, vol. 125 of LNCSE, Springer, Cham, 2018, pp. 373–382.

[36] A. HEINLEIN, A. KLAWONN, M. LANSER, AND J. WEBER, *Machine Learning in Adaptive Domain Decomposition Methods - Predicting the Geometric Location of Constraints*, SIAM J. Sci. Comput., 41 (2019), pp. A3887–A3912.

[37] A. HEINLEIN, A. KLAWONN, M. LANSER, AND J. WEBER, *Combining Machine Learning and Adaptive Coarse Spaces - A Hybrid Approach for Robust FETI-DP Methods in Three Dimensions*, SIAM J. Sci. Comput., 43 (2021), pp. S816–S838. Special Section Copper Mountain 2020.

[38] A. HEINLEIN, A. KLAWONN, M. LANSER, AND J. WEBER, *Combining Machine Learning and Domain Decomposition Methods for the Solution of Partial Differential Equations–A Re-*

view, GAMM-Mitt., 44 (2021), p. e202100001, https://doi.org/10.1002/gamm.202100001.

[39] A. HEINLEIN, A. KLAWONN, M. LANSER, AND J. WEBER, *Predicting the geometric location of critical edges in adaptive GDSW overlapping domain decomposition methods using deep learning*, in Domain Decomposition Methods in Science and Engineering XXVI, Springer, 2023, pp. 307–315.

[40] Z. HU, A. D. JAGTAP, G. E. KARNIADAKIS, AND K. KAWAGUCHI, *When Do Extended Physics-Informed Neural Networks (XPINNs) Improve Generalization?*, SIAM Journal on Scientific Computing, 44 (2022), pp. A3158–A3182.

[41] Z. HU, A. D. JAGTAP, G. E. KARNIADAKIS, AND K. KAWAGUCHI, *Augmented Physics-Informed Neural Networks (APINNs): A gating network-based soft domain decomposition methodology*, Engineering Applications of Artificial Intelligence, 126 (2023), p. 107183.

[42] G.-B. HUANG, Q.-Y. ZHU, AND C.-K. SIEW, *Extreme learning machine: theory and applications*, Neurocomputing, 70 (2006), pp. 489–501.

[43] A. D. JAGTAP AND G. E. KARNIADAKIS, *Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations*, Communications in Computational Physics, 28 (2020), pp. 2002–2041.

[44] A. D. JAGTAP, E. KHARAZMI, AND G. E. KARNIADAKIS, *Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems*, Computer Methods in Applied Mechanics and Engineering, 365 (2020), p. 113028.

[45] G. KARYPIS AND V. KUMAR, *METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0.* http://www.cs.umn.edu/~metis, 2009.

[46] 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.

[47] H. H. KIM, E. CHUNG, AND J. WANG, *BDDC and FETI-DP preconditioners with adaptive coarse spaces for three-dimensional elliptic problems with oscillatory and high contrast coefficients*, Journal of Computational Physics, 349 (2017), pp. 191–214.

[48] H. H. KIM AND E. T. CHUNG, *A BDDC algorithm with enriched coarse spaces for two-dimensional elliptic problems with oscillatory and high contrast coefficients*, Multiscale Model. Simul., 13 (2015), pp. 571–593.

[49] H. H. KIM AND H. J. YANG, *Domain Decomposition Algorithms for Physics-Informed Neural Networks*, in Domain Decomposition Methods in Science and Engineering XXVI, vol. 145, Springer International Publishing, Cham, 2022, pp. 697–704. Series Title: Lecture Notes in Computational Science and Engineering.

[50] A. KLAWONN, M. LANSER, AND J. WEBER, *Learning adaptive FETI-DP constraints for irregular domain decompositions.* TR series, Center for Data and Simulation Science, University of Cologne, Germany, Vol. 2022-3. https://kups.ub.uni-koeln.de/64122/. Accepted for publication in the proceedings of the International Conference on Domain Decomposition Methods 27, Springer LNCSE, November 2022.

[51] A. KLAWONN, M. LANSER, AND J. WEBER, *A domain decomposition-based CNN-DNN architecture for model parallel training applied to image recognition problems*, 2023, https://arxiv.org/abs/2302.06564.

[52] A. KLAWONN, M. LANSER, AND J. WEBER, *Learning adaptive coarse basis functions of FETI-DP*, Journal of Computational Physics, 496 (2024), p. 112587.

[53] T. KNOKE, S. KINNEWIG, S. BEUCHLER, A. DEMIRCAN, U. MORGNER, AND T. WICK, *Domain Decomposition with Neural Network Interface Approximations for time-harmonic Maxwell's equations with different wave numbers*, Mar. 2023, http://arxiv.org/abs/2303.02590.

[54] A. KOPANIČÁKOVÁ, H. KOTHARI, G. E. KARNIADAKIS, AND R. KRAUSE, *Enhancing training of physics-informed neural networks using domain-decomposition based preconditioning strategies*, June 2023, http://arxiv.org/abs/2306.17648.

[55] A. KOVACS, L. EXL, A. KORNELL, J. FISCHBACHER, M. HOVORKA, M. GUSENBAUER, L. BRETH, H. OEZELT, M. YANO, N. SAKUMA, ET AL., *Conditional physics informed neural networks*, Communications in Nonlinear Science and Numerical Simulation, 104 (2022), p. 106041.

[56] Y. LECUN ET AL., *Generalization and network design strategies*, Connectionism in perspective, 19 (1989), p. 18.

[57] J. LI AND X. CAI, *Summation pollution of principal component analysis and an improved algorithm for location sensitive data*, Numerical Linear Algebra with Applications, 28 (2021), p. e2370.

[58] K. Li, K. Tang, T. Wu, and Q. Liao, *D3M: A deep domain decomposition method for partial differential equations*, IEEE Access, 8 (2019), pp. 5283–5294.

[59] S. Li, Y. Xia, Y. Liu, and Q. Liao, *A deep domain decomposition method based on Fourier features*, May 2022, http://arxiv.org/abs/2205.01884.

[60] W. Li, X. Xiang, and Y. Xu, *Deep domain decomposition method: Elliptic problems*, in Mathematical and Scientific Machine Learning, PMLR, 2020, pp. 269–286.

[61] P.-L. Lions, *On the Schwarz alternating method. I*, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations (Paris, 1987), SIAM, Philadelphia, PA, 1988, pp. 1–42.

[62] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Mathematical programming, 45 (1989), pp. 503–528.

[63] L. Luo, W.-S. Shiu, R. Chen, and X.-C. Cai, *A nonlinear elimination preconditioned inexact Newton method for blood flow problems in human artery with stenosis*, Journal of Computational Physics, 399 (2019), p. 108926.

[64] T. Luo, Z. Ma, Z.-Q. J. Xu, and Y. Zhang, *Theory of the frequency principle for general deep neural networks*, 2019, https://arxiv.org/abs/1906.09235.

[65] N. Mai-Duy and T. Tran-Cong, *Mesh-free radial basis function network methods with domain decomposition for approximation of functions and numerical solution of Poisson's equations*, Engineering Analysis with Boundary Elements, 26 (2002), pp. 133–156.

[66] A. Malek and A. Emami Kerdabadi, *Solving Differential Equations by Artificial Neural Networks and Domain Decomposition*, Iranian Journal of Science, 47 (2023), pp. 1233–1244.

[67] J. Mandel and B. Sousedík, *Adaptive selection of face coarse degrees of freedom in the BDDC and the FETI-DP iterative substructuring methods*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1389–1399.

[68] J. Mandel, B. Sousedík, and J. Šístek, *Adaptive BDDC in three dimensions*, Math. Comput. Simulation, 82 (2012), pp. 1812–1831.

[69] V. Mercier, S. Gratton, and P. Boudier, *A coarse space acceleration of deep-DDM*, Dec. 2021, http://arxiv.org/abs/2112.03732.

[70] B. Moseley, A. Markham, and T. Nissen-Meyer, *Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations*, Advances in Computational Mathematics, 49 (2023), p. 62.

[71] C. Park, J. Z. Huang, and Y. Ding, *Domain Decomposition Approach for Fast Gaussian Process Regression of Large Spatial Data Sets*, Journal of Machine Learning Research, 12 (2011), pp. 1697–1728.

[72] D. Peteiro-Barral and B. Guijarro-Berdiñas, *A survey of methods for distributed machine learning*, Progress in Artificial Intelligence, 2 (2013), pp. 1–11.

[73] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, 2nd ed., 2008.

[74] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, *On the spectral bias of neural networks*, in International Conference on Machine Learning, PMLR, 2019, pp. 5301–5310.

[75] 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.

[76] B. Ronen, D. Jacobs, Y. Kasten, and S. Kritchman, *The convergence rate of neural networks for learned functions of different frequencies*, Advances in Neural Information Processing Systems, 32 (2019).

[77] F. Schwenker, H. A. Kestler, and G. Palm, *Three learning phases for radial-basis-function networks*, Neural networks, 14 (2001), pp. 439–458.

[78] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning*, Cambridge University Press, 2014.

[79] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, *Outrageously large neural networks: The sparsely-gated mixture-of-experts layer*, 2017, https://arxiv.org/abs/1701.06538.

[80] 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, 428 (2021), p. 110085.

[81] H. Sheng and C. Yang, *PFNN-2: A Domain Decomposed Penalty-Free Neural Network Method for Solving Partial Differential Equations*, Communications in Computational Physics, 32 (2022), pp. 980–1006.

[82] K. Shukla, A. D. Jagtap, and G. E. Karniadakis, *Parallel physics-informed neural net-*

works via domain decomposition, Journal of Computational Physics, 447 (2021), p. 110683.

[83] J. D. SMITH, K. AZIZZADENESHELI, AND Z. E. ROSS, *EikoNet: Solving the Eikonal Equation with Deep Neural Networks*, IEEE Transactions on Geoscience and Remote Sensing, 59 (2020), pp. 10685–10696.

[84] M. SNIR, S. OTTO, S. HUSS-LEDERMAN, D. WALKER, AND J. DONGARRA, *MPI-The Complete Reference, Volume 1: The MPI Core*, MIT Press, Cambridge, MA, USA, 2nd ed., 1998.

[85] W. SNYDER, I. TEZAUR, AND C. WENTLAND, *Domain decomposition-based coupling of physics-informed neural networks via the Schwarz alternating method*, (2023), https://arxiv.org/abs/2311.00224.

[86] P. STILLER, F. BETHKE, M. BÖHME, R. PAUSCH, S. TORGE, A. DEBUS, J. VORBERGER, M. BUSSMANN, AND N. HOFFMANN, *Large-scale Neural Solvers for Partial Differential Equations*, Sept. 2020, http://arxiv.org/abs/2009.03730.

[87] Q. SUN, X. XU, AND H. YI, *Domain Decomposition Learning Methods for Solving Elliptic Problems*, 2022, https://arxiv.org/abs/2207.10358.

[88] A. TAGHIBAKHSHI, N. NYTKO, T. ZAMAN, S. MACLACHLAN, L. OLSON, AND M. WEST, *Learning Interface Conditions in Domain Decomposition Solvers*, Oct. 2022, http://arxiv.org/abs/2205.09833.

[89] A. TAGHIBAKHSHI, N. NYTKO, T. U. ZAMAN, S. MACLACHLAN, L. OLSON, AND M. WEST, *MG-GNN: Multigrid Graph Neural Networks for Learning Multilevel Domain Decomposition Methods*, Mar. 2023, http://arxiv.org/abs/2301.11378.

[90] A. TOSELLI AND O. WIDLUND, *Domain Decomposition Methods—Algorithms and Theory*, vol. 34 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2005.

[91] J. VERBRAEKEN, M. WOLTING, J. KATZY, J. KLOPPENBURG, T. VERBELEN, AND J. S. RELLERMEYER, *A survey on distributed machine learning*, Acm computing surveys (csur), 53 (2020), pp. 1–33.

[92] H. WANG, R. PLANAS, A. CHANDRAMOWLISHWARAN, AND R. BOSTANABAD, *Mosaic flows: A transferable deep learning framework for solving PDEs on unseen domains*, Computer Methods in Applied Mechanics and Engineering, 389 (2022), p. 114424.

[93] J. WEBER, *Efficient and robust FETI-DP and BDDC methods – Approximate coarse spaces and deep learning-based adaptive coarse space*, PhD thesis, Universität zu Köln, 2022. Online available at http://kups.ub.uni-koeln.de/id/eprint/55179.

[94] C. K. WILLIAMS AND C. E. RASMUSSEN, *Gaussian processes for machine learning*, vol. 2, MIT press Cambridge, MA, 2006.

[95] R. XU, D. ZHANG, M. RONG, AND N. WANG, *Weak form theory-guided neural network (TgNN-wf) for deep learning of subsurface single- and two-phase flow*, Journal of Computational Physics, 436 (2021), p. 110318.

[96] H. J. YANG AND H. H. KIM, *Additive Schwarz algorithms for neural network approximate solutions*, 2022, https://arxiv.org/abs/2211.00225.

[97] H. J. YANG AND H. H. KIM, *Iterative algorithms for partitioned neural network approximation to partial differential equations*, Aug. 2023, http://arxiv.org/abs/2211.00225.

[98] D. ZHANG AND Z. LU, *An efficient, high-order perturbation approach for flow in random porous media via Karhunen–Loeve and polynomial expansions*, Journal of Computational Physics, 194 (2004), pp. 773–794.