Weierstraß-Institut
für Angewandte Analysis und Stochastik
Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

Multilevel CNNs for parametric PDEs

Cosmas Heiß[1], Ingo Gühring[1], Martin Eigel[2]

submitted: July 27, 2023

[1] Technische Universität Berlin
Straße des 17. Juni 136
10623 Berlin
Germany
E-Mail: cosmas.heiss@gmail.com
guehring@math.tu-berlin.de

[2] Weierstrass Institute
Mohrenstr. 39
10117 Berlin
Germany
E-Mail: martin.eigel@wias-berlin.de

No. 3035
Berlin 2023

2020 *Mathematics Subject Classification.* 65D40, 65N55, 65F10, 68Q32.

Key words and phrases. deep learning, partial differential equations, parametric PDE, multilevel, expressivity, CNN, uncertainty quantification.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Mohrenstraße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Multilevel CNNs for parametric PDEs

Cosmas Heiß, Ingo Gühring, Martin Eigel

Abstract

We combine concepts from multilevel solvers for partial differential equations (PDEs) with neural network based deep learning and propose a new methodology for the efficient numerical solution of high-dimensional parametric PDEs. An in-depth theoretical analysis shows that the proposed architecture is able to approximate multigrid V-cycles to arbitrary precision with the number of weights only depending logarithmically on the resolution of the finest mesh. As a consequence, approximation bounds for the solution of parametric PDEs by neural networks that are independent on the (stochastic) parameter dimension can be derived.

The performance of the proposed method is illustrated on high-dimensional parametric linear elliptic PDEs that are common benchmark problems in uncertainty quantification. We find substantial improvements over state-of-the-art deep learning-based solvers. As particularly challenging examples, random conductivity with high-dimensional non-affine Gaussian fields in 100 parameter dimensions and a random cookie problem are examined. Due to the multilevel structure of our method, the amount of training samples can be reduced on finer levels, hence significantly lowering the generation time for training data and the training time of our method.

1 Introduction

The application of deep learning (DL) to many problems of the natural sciences has become one of the most promising emerging research areas, sometimes coined *scientific machine learning* [4, 66, 43, 58, 54]. In this work, we focus on the challenging problem of parametric partial differential equations (PDEs) that are used to describe complex physical systems e.g. in engineering applications and the natural sciences. The parameters determine the data or domain and are used to introduce uncertainties into the model, e.g. based on assumed or measured statistical properties. The resulting high-dimensional problems are analyzed and solved numerically in the research area of uncertainty quantification (UQ) [53]. Such parametric models are of importance in automated design, weather forecasting, risk simulations, and the material sciences, to name but a few.

To make the problem setting concrete, for a possibly countably infinite parameter space $\Gamma \subseteq \mathbb{R}^{\mathbb{N}}$ and a spatial domain $D \subseteq \mathbb{R}^d$, $d \in \{1, 2, 3\}$, we are concerned with learning the solution map $u : \Gamma \times \overline{D} \to \mathbb{R}$ of the *parametric stationary diffusion PDE* that satisfies the following equation for all parameters $\mathbf{y} \in \Gamma$

$$\begin{cases} -\operatorname{div}_x \kappa(\mathbf{y}, x) \nabla_x u(\mathbf{y}, x) &= f(x) \quad \text{on } D, \\ u(\mathbf{y}, x) &= 0 \qquad \text{on } \partial D, \end{cases} \tag{1.1}$$

where $\kappa : \Gamma \times D \to \mathbb{R}$ is the parameterized diffusion coefficient describing the diffusivity, i.e. the media properties. Note that differential operators act with respect to the physical variable x.

DOI 10.20347/WIAS.PREPRINT.3035 Berlin 2023

As a straight-forward but computationally ineffective solution, Equation (1.1) could be solved numerically for each $\mathbf{y} \in \Gamma$ individually, using e.g. the finite element (FE) method and a Monte Carlo approach to estimate statistical quantities of interest. More refined methods such as stochastic Galerkin [19, 20], stochastic collocation [3, 57, 23] or least squares methods [13, 21, 17] exploit smoothness and low-dimensionality of the solution manifold $\{u(\mathbf{y}) : \mathbf{y} \in \Gamma\}$ by projection or interpolation of a finite set of discrete solutions $\{u_h(\mathbf{y}_n)\}_{n=1}^N$. After the numerical solution of Equation (1.1) at sample points $\mathbf{y}_n$ and the evaluation of the projection or interpolation scheme, solutions for arbitrary $\mathbf{y} \in \Gamma$ can quickly be obtained approximately simply by evaluating the constructed functional "surrogate". However, while compression techniques such as low-rank approximations or reduced basis methods foster the exploitation of low-dimensional structures, these approaches still become practically infeasible for large parameter dimensions rather quickly. Multilevel approaches represent another concept that can be applied beneficially for this problem class as was e.g. shown with multilevel stochastic collocation (MLSC) [71] and multilevel quadrature [39, 5]. The central idea is to equilibrate the error components of the physical and the statistical approximations. This can be achieved by performing a frequency decomposition of the solution in the physical domain based on a hierarchy of nested FE spaces determined by appropriate spatial discretizations. An initial approximation on the coarsest level is successively refined by additive corrections on finer levels. Corrections generally contain strongly decreasing information for finer levels, which can be exploited in terms of a level dependent fineness of the stochastic discretization.

For the numerical solution of (usually non-parametric) PDEs, a plethora of NN architectures has been devised [see e.g., 6, 7, 61, 70, 18, 62, 44, 76, 38, 49]. Similar to classical methods, DL-based solvers for parametric PDEs typically learn the mapping $\mathbf{y} \mapsto u_h(\mathbf{y})$ in a supervised way from computed (or observed) samples $\{u^h(\mathbf{y}_n)\}_{n=1}^N$ [see e.g., 27, 41, 2]. This results in a computationally expensive data generation process, consisting of approximately computing the solutions at $\mathbf{y}_n$ and subsequently training the NN on this data. After training, only a cheap forward pass is necessary for the approximate solution of (1.1) at arbitrary $\mathbf{y}$. We note that in contrast to many classical machine learning problems, a dataset can be generated on the fly and in principle with arbitrary precision.

Complementing numerical results, significant progress has been made to understand the approximation power, also called *expressivity*, of NNs for representing solutions of PDEs [47, 68, 8, 56, 31, 35, 33]. These works show that NNs are at least as expressive as "classical" best-in-class methods like the ones mentioned before. However, numerical results do often not reflect this theoretical advantage [1]. Instead, the convergence of the error typically stagnates early during training and the accuracy of classical approaches is not reached [27, 44, 54, 32].

We propose a DL-based NN approach that overcomes the performance shortcomings of other NN architectures for parametric PDEs by borrowing concepts from MLSC and multigrid algorithms. Coarse approximation and successive corrections on finer levels are learned in a supervised way by sub-networks and combined for the final prediction. Analogously to multilevel Monte Carlo (MLMC) and MLSC methods, we show that the fineness of sampling the stochastic parameter space (in terms of the amount of training data per level) can be (reciprocally) equilibrated with the fineness of the spatial discretization. The developed architecture consists of a composition of UNets [64], which inherently are closely related to multiresolution approaches.

In the following, we summarize our main contributions.

- *Theory:* We conduct an extensive analysis of the expressivity of UNet-like convolutional

NNs (CNNs) for the solution of parametric PDEs. Our results show that under the uniform ellipticity assumption, the number of required parameters is independent on the parameter dimension with increasing accuracy. In detail, we show that common multilevel CNN architectures are able to approximate a V-cycle multigrid solver iteration with Richardson smoother [63] with less than $\log(1/h)$ weights up to arbitrary accuracy, where h is the mesh size of the underlying FE space (Theorem 5.1). Using this result, we approximate the solution map of (1.1) up to expected H^1-norm error h by an NN with $\log(1/h)^2$ weights.

- *Numerical Results:* We conduct an in-depth numerical study of our method and, more generally, UNet-based approaches on common benchmark problems for parametric PDEs in UQ. Strikingly, the tested methods show significant improvements over the state-of-the-art NN architectures for solving parametric PDEs. To reduce computing complexity during training data generation and training the NN, we shift the bulk of the training data to coarser levels, which allows our method to be scaled to finer grids, ultimatively resulting in higher accuracy.

1.1 Related Work

[68, 30] derived expressivity results for approximating the solution of parametric PDEs based on a sparse generalized polynomial chaos representation of the solution known from UQ [67, 16] and a central result for approximating polynomials by NNs [74]. [47] translated a reduced basis method into an NN architecture. None of these results show bounds that are independent on the (stochastic) parameter dimension. In contrast, translated to our setting (and simplified), [47] show an asymptotic upper bound of $h^{-2}\log(h^{-1})^p + p\log(h^{-1})^{3p}$ weights. Numerical experiments are either not provided or do not support the positive theoretical results [27]. For general overview of expressivity results for NNs, we refer the interested reader to [36].

Related to our method and instructive for its derivation is the MLSC method presented in [71] and also [39, 5]. Another related work is [55], where a training process with multilevel structure is presented. Under certain assumptions on the achieved accuracy of the optimization, an error analysis is carried out that could in part also be transferred to our architecture.

The connection between differential operators and convolutional layers is well known [see 12] and CNNs have already been applied to approximate parametric PDEs for problems like predicting the airflow around a given geometry in the works [9, 34]. In [77] Bayesian CNNs were used to assess uncertainty in processes governed by stochastic PDEs. [72] extend the concept to continuous convolutions in mesh-free Lagrangian approaches for solving problems in fluid dynamics. The concept of a multilevel decomposition has also been applied to CNNs to approximate solutions of PDEs or in [26] for general generative modeling, or based on hierarchical matrix representations in [24, 25].

Such hierarchical matrix representations have even been extended to graph CNNs as multipole kernels to allow for a mesh-free approximation of the solutions to parametric PDEs [48]. This approach is similar to our method. However, the mathematical connection to classical multigrid algorithms is not explored. The kernels are based on fast multipole modeling without a direct link to the FE method.

(author?) [40] present a multilevel approach called MgNet, where various parts of a classical multigrid solver are replaced by NNs resulting in a UNet-like NN and use it for image

classification. Meta-MgNet [12] is a meta-learning approach using MgNet to predict solutions for parametric PDEs and is most closely related to our method. The predicted solution is successively refined and stochastic parameters are incorporated by computing encodings of the parametric discretized operator on each level using additional smaller NNs. The output of these meta-networks then influences the smoothing operation in the MgNet architecture. However, in contrast to our method, Meta-MgNet still depends on the assembled operator matrix. Furthermore, while using an iterative approach yields more accurate approximations, the speedup promised by a single application of a CNN is mitigated and the authors report runtimes on the same order of magnitude as classical solvers. Lastly, the authors test their method on problems with uniform diffusivity using two or three degrees of freedom instead of a continuous diffusivity field. Our approach differs from the Meta-MgNet in the following ways: (i) We show that a general UNet is already able to approximate parametric differential operators without the need for a meta-network approach; (ii) we improve on the training procedure by incorporating a multilevel optimization scheme; (iii) we provide a theoretical complexity analysis; (iv) we provide experiments showing high accuracies with a single application of our method for a variety of random parameter fields.

1.2 Outline

In the first section we introduce the parametric Darcy model problem and its FE discretization. In Section 3, we give a brief introduction to multigrid methods. In Section 4, we describe our DL-based solution approach. Section 5 is concerned with the theoretical analysis of the expressivity of the presented architecture. Section 6 contains our numerical study on several parametric PDE problems. We discuss and summarize our findings in Section 7. Implementation details and our proofs can be found in the appendix.

2 Problem setting

A standard model problem in UQ is the stationary linear diffusion equation (1.1) where the diffusion coefficient $\kappa : \Gamma \to L^\infty(D)$ is a random field with finite variance determined by a possibly high-dimensional random parameter $\mathbf{y} \in \Gamma \subseteq \mathbb{R}^{\mathbb{N}}$, hence also parametrizing the solution. The parameters are images of random vectors distributed according to some product measure $\pi = \otimes_{k\geq 1}\pi_1$, where π_1 usually either is the uniform (denoted by U) or the standard normal measure (denoted by N) on $\Gamma_1 \subseteq \mathbb{R}$. For each parameter realization $\mathbf{y} \in \Gamma$, the *variational formulation* of the problem is given by: For a bounded Lipschitz domain $D \subseteq \mathbb{R}^d$ and source term $f \in V^* = H^{-1}(D)$, find $u \in V := H_0^1(D)$ such that for all test functions $w \in V$,

$$\int_D \kappa(\mathbf{y}, x)\nabla u(x) \cdot \nabla w(x)\,\mathrm{d}x = \int_D f(x)w(x)\,\mathrm{d}x. \tag{2.1}$$

We assume that the above equation always exhibits a unique solution. In fact, this holds true with high probability even for the case of unbounded (lognormal) coefficients, where uniform ellipticity of the bilinear form is not given. This yields the solution operator v, mapping a parameter realization $\mathbf{y} \in \Gamma$ to its corresponding solution $v(\mathbf{y})$:

$$v\colon \Gamma \to V, \qquad \mathbf{y} \mapsto v(\mathbf{y}).$$

In this work, we strive to develop an efficient and accurate NN approximation $\tilde{v}\colon \Gamma \to V$, which minimizes the expected H^1-distance

$$\int_\Gamma \|\nabla(v(\mathbf{y}) - \tilde{v}(\mathbf{y}))\|^2_{L^2(D)} \,\mathrm{d}\pi(\mathbf{y}).$$

With the assumed boundedness (with high probability) of the coefficient κ, this is equivalent to optimizing with respect to the canonical parameter-dependent norm.

We recall the truncated *Karhunen-Loève expansion* (KLE) [46, 53] with stochastic parameter dimension p, which is a common parametric representation of random fields. It is given by

$$\kappa(\mathbf{y}, x) = a_0(x) + \sum_{k=1}^{p} \mathbf{y}_k a_k(x). \tag{2.2}$$

Here, the $p \in \mathbb{N}$ basis functions $a_k \in L^2(D), 1 \le k \le p$ are determined by assumed statistical properties of the field κ. When Gaussian random fields are considered, they are completely characterized by their first two moments and the a_k are the eigenfunctions of the covariance integral operator while the parameter vector $\mathbf{y}$ is the image of a standard normal random vector. To become computationally feasible, a sufficient decay of the norms of these functions is required for the truncation to be reasonable. Note that in the numerical experiments, we use a well-known "artificial" KLE.

2.1 Discretization

To solve the infinite dimensional variational formulation (2.1) numerically, a finite dimensional subspace has to be defined in which an approximation of the solution is computed. As is common with PDEs, we use the FE method, which is based on a disjoint decomposition of the computational domain D into simplices (also called elements or cells). Setting the domain $D = [0,1]^2$, we assume a subdivision into congruent triangles such that D is exactly represented by a uniform square mesh with identical connectivity at every node (see also Remark C.1). The discrete space V_h is then spanned by conforming P1 FE hat functions $\varphi_j : \mathbb{R}^2 \to \mathbb{R}$, i.e., $V_h = \operatorname{span}\{\varphi_j\}_{j=1}^{\dim V_h} \subseteq V$ [see e.g., 10, 15]. We write $u_h, w_h \in V_h$ for the discretized versions of $u, w \in V$ of Equation (2.1) with FE coefficient vectors $\mathbf{u}, \mathbf{w} \in \mathbb{R}^{\dim V_h}$ and denote by κ_h the nodal interpolation of κ such that $\kappa_h(\mathbf{y}, \cdot) \in V_h$ with coefficient vector $\boldsymbol{\kappa}_\mathbf{y} \in \mathbb{R}^{\dim V_h}$ for every $\mathbf{y} \in \Gamma$. Hence,

$$u_h = \sum_{i=1}^{\dim V_h} \mathbf{u}_i \varphi_i, \quad w_h = \sum_{i=1}^{\dim V_h} \mathbf{w}_i \varphi_i, \quad \kappa_h(\mathbf{y}, \cdot) = \sum_{i=1}^{\dim V_h} (\boldsymbol{\kappa}_\mathbf{y})_i \varphi_i.$$

The discretized version of the parametric Darcy problem (2.1) reads:

Problem 2.1 (Parametric Darcy problem, discretized)**.** *For $\mathbf{y} \in \Gamma$, find $u_h \in V_h$ such that for all $w_h \in V_h$*

$$a_{\mathbf{y},h}(u_h, v_h) = f(w_h),$$

where $a_{\mathbf{y},h}(u_h, v_h) \coloneqq \int_D \kappa_h(\mathbf{y}, x) \nabla u_h(x) \cdot \nabla w_h(x) \,\mathrm{d}x$. Using the FE coefficient vectors leads to the system of linear equations

$$A_\mathbf{y} \mathbf{u} = \mathbf{f}, \tag{2.3}$$

where $\mathbf{f} \coloneqq (f(\varphi_i))_{i=1}^{\dim V_h} = (\int_D f(x)\varphi_i(x) \,\mathrm{d}x)_{i=1}^{\dim V_h}$ and $A_\mathbf{y} \coloneqq (a_{\mathbf{y},h}(\varphi_j, \varphi_i))_{i,j=1}^{\dim V_h}$.

We assume that there always exists a unique solution to this problem and define the discretized parameter to solution operator by

$$v_h \colon \Gamma \to V_h, \qquad \mathbf{y} \mapsto v_h(\mathbf{y}), \tag{2.4}$$

where $v_h(\mathbf{y})$ solves Problem 2.1 for any $\mathbf{y} \in \Gamma$. Furthermore, $\mathbf{v}_h \colon \Gamma \to \mathbb{R}^{\dim V_h}$ maps $\mathbf{y}$ to the FE coefficient vector of $v_h(\mathbf{y})$.

Remark 2.1. *We neglect the influence of the approximation of the diffusivity field κ as κ_h in our analysis since it would only introduce unnecessary technicalities. It is well-known from FE analysis how this has to be treated theoretically [see e.g., 15, Chapter 4]. Notably, we always assume that the coefficient is discretized on the finest level of the multigrid algorithm that we analyze and evaluate in the experiments.*

3 A primer on multigrid methods

Our proposed method relies fundamentally on the notion of multigrid solvers for algebraic equations. These iterative solvers exhibit an optimal complexity in the sense that they converge linearly in terms of degrees of freedom. Their mathematical properties are well understood and they are commonly used for solving PDEs discretized, in particular, with FEs and a hierarchy of meshes [see 10, 75, 37]. This section provides a brief primer on the multigrid setting.

The starting point is a hierarchy of nested spaces indexed by the level $\ell \in \{1, \dots, L\}$ with finest level L, which determines the accuracy of the approximation. In our setting, this finest level is chosen in advance and is assumed to yield a sufficient FE accuracy. Similar to the previous section, we define V_ℓ as the FE space of conforming P1 elements on the associated dyadic quasi-uniform triangulations $\mathcal{T}_\ell$ with mesh size $h_\ell \sim 2^{-\ell}$ and degrees of freedom $\dim V_\ell \sim 2^{\ell d}$. This yields a sequence of nested FE spaces

$$V_1 \subseteq \ldots \subseteq V_L \subseteq H^1_0(D).$$

These spaces are spanned by the the FE basis functions on the respective level, i.e., $V_\ell = \operatorname{span}\{\varphi_j^{(\ell)}\}_{j=1}^{\dim V_\ell}$. A frequency decomposition of the target function $u \in H^1_0(D)$ is obtained by approximating low frequency parts on the coarsest mesh $\mathcal{T}_1$ and subsequently calculating corrections for higher frequencies on finer meshes.

For each level $\ell = 1, \dots, L$ we define the discretized solution operator $v_\ell \colon \Gamma \to V_\ell$ as in (2.4). The correction $\hat{v}_\ell$ with respect to the next coarser level $\ell - 1$ for $\ell \in \{2, \dots, L\}$ is given by

$$\hat{v}_\ell \colon \Gamma \to V_\ell, \qquad \mathbf{y} \mapsto v_\ell(\mathbf{y}) - v_{\ell-1}(\mathbf{y}). \tag{3.1}$$

Additionally, we define $\mathbf{v}_\ell$ and $\hat{\mathbf{v}}_\ell$ as the functions mapping the parameter $\mathbf{y}$ to the corresponding coefficient vectors of $v_\ell(\mathbf{y})$ and $\hat{v}_\ell(\mathbf{y})$ in V_ℓ. If the solution of the considered PDE is sufficiently smooth, the corrections decay exponentially in the level, namely $\|\hat{v}_\ell(\mathbf{y})\|_{H^1} \lesssim 2^{-\ell} \|f\|_*$ [see 10, 75, 37].

Central to the multigrid method is a transfer of discrete functions between adjacent levels of the hierarchy of spaces. This is achieved in terms of prolongation and restriction operators, the discrete versions of which are defined in the following.

Definition 3.1 (Prolongation & weighted restriction matrices)**.** *Let $L \in \mathbb{N}$ and for some $\ell \in \{1, \dots, L-1\}$ the spaces V_ℓ and $V_{\ell+1}$ be defined as above. The* prolongation matrix *$P_\ell \in \mathbb{R}^{\dim V_{\ell+1} \times \dim V_\ell}$ is the matrix representation of the canonical embedding of V_ℓ into $V_{\ell+1}$ under the respective FE basis functions. $P_\ell^T \in \mathbb{R}^{\dim V_\ell \times \dim V_{\ell+1}}$ defines the weighted restriction.*

The theoretical analysis of our multilevel NN architecture hinges on the multigrid V-cycle which is depicted in Algorithm 3.1. It exploits the observation that simple iterative solvers reduce high-frequency components of the residual. In addition to these so-called *smoothing iterations*, low-frequency correction are recursively computed on coarser grids. In fact, if the approximation error by the recursive calls for the coarse grid correction is sufficiently small, a grid independent contraction rate is achieved [see 11, 10].

For our analysis, we use a damped Richardson smoother [63], which is rarely used in practice due to subpar convergence rates, but theoretically accessible because of its simplicity. For a suitable damping parameter $\omega > 0$, the Richardson iteration for solving Equation (2.3) is given by

$$\begin{aligned} \mathbf{u}^0 &:= \mathbf{0}, \\ \mathbf{u}^{n+1} &:= \mathbf{u}^n + \omega(\mathbf{f} - A_{\mathbf{y}}\mathbf{u}^n). \end{aligned} \tag{3.2}$$

We denote by $\mathrm{MG}_{k_0,k}^m \colon \mathbb{R}^{3\times\dim V_h} \to \mathbb{R}^{\dim V_h}$ the function mapping an initial guess, the diffusivity field and the right-hand side to the result of m multigrid V-cycles (Algorithm 3.1) with k_0 smoothing iterations on the coarsest level and k smoothing iterations on the finer levels.

Based on this iteration, we use a fundamental convergence result for the multigrid algorithm shown in [11, Thm. 4.2] together with the assumption of uniform ellipticity to get the uniform contraction property of the V-cycle multigrid algorithm. For $\mathbf{x} \in \mathbb{R}^{\dim V_h}$ and the discretized operator $A \in \mathbb{R}^{\dim V_h \times \dim V_h}$, we use the standard notation for the energy norm $\|\mathbf{x}\|_A = \|A^{1/2}\mathbf{x}\|_{\ell^2}$.

Theorem 3.1. *Let $k \in \mathbb{N}$. There exists $\omega > 0$ and mesh-independent $k_0 \in \mathbb{N}$, $C < 0$, such that the V-cycle iteration with k pre- and post-smoothing iterations, k_0 iterations on the coarsest grid, and Richardson damping parameter ω applied to Problem 2.1 yields a grid independent contraction of the residual $\mathbf{e}_{\mathbf{y}}^i := \mathrm{MG}_{k_0,k}^i(\mathbf{0}, \boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}) - \mathbf{v}_h(\mathbf{y})$ at the i-th iteration, namely*

$$\|\mathbf{e}_{\mathbf{y}}^{i+1}\|_{A_{\mathbf{y}}} \leq \mu(k)\|\mathbf{e}_{\mathbf{y}}^i\|_{A_{\mathbf{y}}}, \quad \mu(k) \leq \frac{C}{C+2k} < 1,$$

for all $\mathbf{y} \in \Gamma$.

In the next remark, we elaborate on the number of smoothing steps on the coarsest grid k_0.

Remark 3.1. *In the V-cycle algorithm it is often assumed that the equation system solved on the coarsest level is solved exactly (by a direct solver). However, the proof of [11, Thm. 4.2] shows that this is not necessary. In fact, a contraction of the residual smaller than $C/(C+2k)$ on the coarsest level suffices. This shows that k_0 is independent on the fineness of the finest grid. For a rigorous derivation, we refer to [73, Theorem 4.4]. In our setting, it is possible to use the damped Richardson iteration for the correction on the coarsest grid as well, effectively carrying out additional smoothing steps.*

Algorithm 3.1 `V-cycle` function

Input: $\mathbf{u}$, $\mathbf{f}$, A, ℓ
for k pre-smoothing steps **do**
  $\mathbf{u} \leftarrow \mathbf{u} + \omega(\mathbf{f} - A\mathbf{u})$ // perform smoothing steps
end for
if $\ell = 1$ **then**
  solve $A\mathbf{u} = \mathbf{f}$ for $\mathbf{u}$ // coarsest grid step
else
  $P \leftarrow P_{\ell-1}$
  $\overline{\mathbf{r}} \leftarrow P^\intercal(\mathbf{f} - A\mathbf{u})$ // compute restricted residual
  $\overline{A} \leftarrow P^\intercal A P$ // compute restricted operator
  $\overline{\mathbf{e}} \leftarrow \mathbf{0}$
  $\overline{\mathbf{e}} \leftarrow \texttt{V-cycle}(\overline{\mathbf{e}}, \overline{\mathbf{r}}, \overline{A}, \ell - 1)$ // solve for coarse correction $\overline{\mathbf{e}}$
  $\mathbf{u} \leftarrow \mathbf{u} + P\overline{\mathbf{e}}$ // add coarse correction
end if
for m post-smoothing steps **do**
  $\mathbf{u} \leftarrow \mathbf{u} + \omega(\mathbf{f} - A\mathbf{u})$ // perform smoothing steps
end for
return $\mathbf{u}$

4 Methodology of multilevel neural networks

This section gives an overview of our multilevel NN architecture for the efficient numerical solution of parametric PDEs. Some of the design choices are tailored specifically to the stationary diffusion equation (1.1) but can easily be adapted to other settings, in particular to different linear and presumably also nonlinear PDEs. Furthermore, in our analysis and experiments we fix the number of spatial dimensions to two. We note again that these results can be extended to more dimensions in a straightforward way e.g. by using higher-dimensional convolutions.

The general procedure of the proposed approach can be described as follows. Initially, a dataset of solutions is generated using classical finite element solvers for randomly drawn parameters (datapoints), i.e., realizations of coefficient fields leading to respective FE solutions. Then, for each datapoint a multilevel decomposition of the solution is generated as formalized in (3.1). A NN is trained to output the coarse grid solution and the subsequent level corrections from the input parameters, which are given as coefficient vectors of the coefficient realizations on the respective FE space. The network prediction is obtained by summing up these contributions. Finally, to test the accuracy of the network, a set of independent FE solutions is generated and error metrics of the NN predictions are computed.

The centerpiece of our methodology is called $\mathrm{ML\text{-}Net}$ (short for *multilevel network*). For a prescribed number of levels $L \in \mathbb{N}$, it consists of L jointly trained subnetworks $\mathrm{ML\text{-}Net}_\ell$. Each subnetwork is optimized to map its input – the discretized parameter dependent diffusion coefficient $\kappa_{\mathbf{y}}$ and the output of the previous level $\mathbf{z}^{(\ell-1)}$ – to the level ℓ correction $\hat{\mathbf{v}}_\ell(\mathbf{y})$ as in (3.1). The $\mathrm{ML\text{-}Net}_\ell$ subnetwork architecture is inspired by a cascade of multigrid V-cycles, which is implemented by a sequence of UNets [64]. Using UNets in a sequential manner to correct outputs of previous modules successively is a well-established strategy in computer vision [see e.g., 28]. We provide a theoretical foundation for this design choice in the next section. Since we assume the right-hand side $f \in H^{-1}(D)$ to be independent of $\mathbf{y}$, we do not

include it as an additional input. More precisely,

$$\text{ML-Net}[\boldsymbol{\theta}; L] : \mathbb{R}^{\dim V_L} \to \bigotimes_{\ell=1}^{L} \mathbb{R}^{\dim V_\ell}, \quad \boldsymbol{\kappa} \mapsto (\mathbf{z}^{(\ell)})_{\ell=1}^{L},$$

where $\mathbf{z}^{(\ell)} \in \mathbb{R}^{\dim V_\ell}$ is the output of the level ℓ subnetwork and $\boldsymbol{\theta}$ are the learnable parameters. For the subnetworks, we have

$$\begin{aligned}
&\text{ML-Net}_1[\boldsymbol{\theta}_1; L] \colon \mathbb{R}^{\dim V_L} \to \mathbb{R}^{\dim V_1}, \\
&\text{ML-Net}_\ell[\boldsymbol{\theta}_\ell; L] \colon \mathbb{R}^{\dim V_L} \times \mathbb{R}^{\dim V_{\ell-1}} \to \mathbb{R}^{\dim V_\ell}, \quad \text{for } \ell = 2, \dots, L.
\end{aligned}$$

The outputs are given by

$$\begin{aligned}
\mathbf{z}^{(1)} &= \text{ML-Net}_1[\boldsymbol{\theta}_1; L](\boldsymbol{\kappa}), \\
\mathbf{z}^{(\ell)} &= \text{ML-Net}_\ell[\boldsymbol{\theta}_\ell; L](\boldsymbol{\kappa}, \mathbf{z}^{(\ell-1)}), \quad \text{for } \ell = 2, \dots, L.
\end{aligned}$$

Again, $\boldsymbol{\theta}_\ell$ are the learnable parameters and $\boldsymbol{\theta} = (\boldsymbol{\theta}_\ell)_{\ell=1}^{L}$.

Each level consists of a sequence of UNets of depth ℓ with input dimension equal to $\dim V_\ell$ (arranged on a 2D grid). By coupling the UNet depth with the level, we ensure that the UNets on each level internally subsample the data to the coarse grid resolution $\dim V_1$ (again arranged on a 2D grid). Consequently, the number of parameters per UNet increases concurrently with the level. To distribute the compute effort (roughly) equally across levels, we decrease the number of UNets per level, which we denote by $\boldsymbol{R}_\ell$, where $\boldsymbol{R} \in \mathbb{N}^L$. The NN architecture on level ℓ is given by

$$\text{ML-Net}_\ell[\boldsymbol{\theta}_\ell; L](\boldsymbol{\kappa}, \boldsymbol{z}) := \left[\bigcirc_{r=1}^{\boldsymbol{R}_\ell} \text{UNet}[\boldsymbol{\theta}_{\ell,r}; \ell]([\downarrow \boldsymbol{\kappa}, \cdot])\right] (\uparrow \boldsymbol{z}),$$

where $\downarrow$ ($\uparrow$) denotes the up-sampling (down-sampling) operator to level ℓ resolution $\dim V_\ell$. These operators are motivated by the prolongation and restriction operators of Definition 3.1. However, they are comprised of trainable strided (or transpose-strided) convolutions instead of an explicit construction. The parameters of the level ℓ subnetwork $\boldsymbol{\theta}_\ell$ consist of the parameters of the learnable up- and down-sampling and the UNet-sequence $(\boldsymbol{\theta}_{\ell,r})_{r=1}^{\boldsymbol{R}_\ell}$.

Our complete approach to tackle a parametric PDE with $\mathrm{ML\text{-}Net}$ consists of the following multi-step procedure:

Step 1: Sampling and computing the data. We denote by $N \in \mathbb{N}$ the training dataset size and sample parameter realizations $\mathbf{y}_1, \dots, \mathbf{y}_N$ drawn according to π. For each parameter realization $\mathbf{y}_i$, $1 \le i \le N$, the grid corrections $\hat{\mathbf{v}}_\ell(\mathbf{y}_i)$ on each level $1 \le \ell \le L$ as in (3.1) and the FE coefficient vector $\boldsymbol{\kappa}_{\mathbf{y}_i} \in \mathbb{R}^{\dim V_L}$ of $\kappa_L(\cdot, \mathbf{y}_i)$ on the finest level L are computed. Note that the evaluation of the grid corrections requires the PDE solution for each parameter on the finest grid.

Step 2: Training the $\mathrm{ML\text{-}Net}$. For each level $\ell \in \{1, \dots, L\}$, compute the normalization constants $\delta_\ell^2 := \frac{1}{N} \sum_{i=1}^{N} \|\hat{\mathbf{v}}_\ell(\mathbf{y}_i)\|_2^2$ and the H^1 mass matrix $\boldsymbol{M}_\ell \in \mathbb{R}^{\dim V_\ell \times \dim V_\ell}$ by

$$(\boldsymbol{M}_\ell)_{kj} := \left\langle \varphi_k^{(\ell)}, \varphi_j^{(\ell)} \right\rangle_{H^1(D)}, \quad \text{for} \quad k, j \in \{1, \dots, \dim V_\ell\},$$

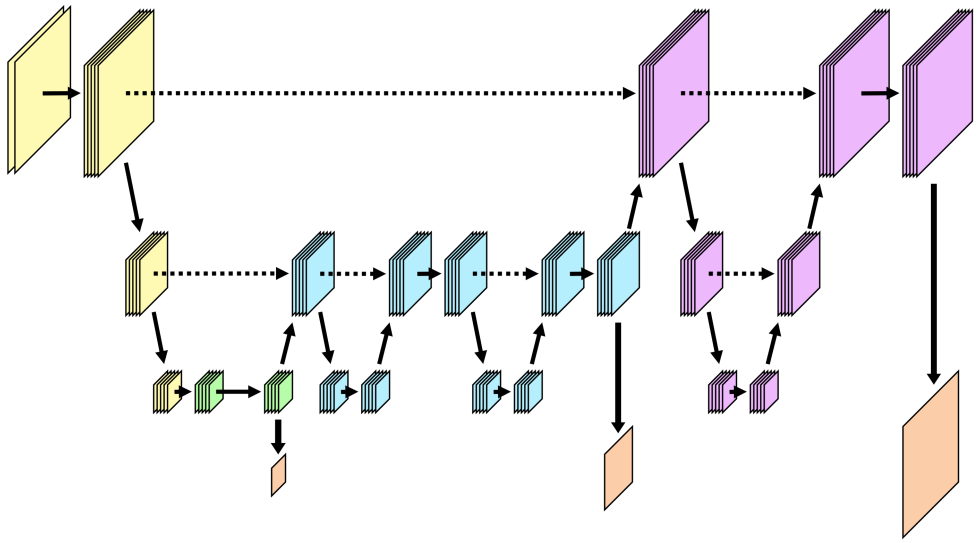

Figure 1: Illustration of the structure of the proposed ML-Net. The down-sampling cascade is shown in yellow. In this example, the network is constructed for $L = 3$ with the green, cyan and magenta parts representing the individual stages on each level. The outputs are shown in orange. Solid arrows represent convolutions and dashed arrows skip connections.

where $\{\varphi_1^{(\ell)}, \dots, \varphi_{\dim V_\ell}^{(\ell)}\}$ constitutes the conforming P1 FE basis of V_ℓ. We optimize the parameters $\boldsymbol{\theta}$ by approximately solving

$$\min_{\boldsymbol{\theta}} \sum_{\ell=1}^{L} \sum_{i=1}^{N} \mathbf{d}_{\ell i}^T \boldsymbol{M}_\ell \mathbf{d}_{\ell i}, \quad \text{where} \quad \mathbf{d}_{\ell i} \coloneqq \left[\text{ML-Net}[\boldsymbol{\theta}; L](\boldsymbol{\kappa}_{\mathbf{y}_i})\right]_\ell - \frac{1}{\delta_\ell} \hat{\mathbf{v}}_\ell(\mathbf{y}_i)$$

with mini-batch stochastic gradient descent. Our loss computes the H^1-distance between the subnetwork output as a coefficient of the FE basis of V_ℓ and the normalized grid corrections directly on the coefficients for each level ℓ, i.e.,

$$\mathbf{d}_{\ell i}^T \boldsymbol{M}_\ell \mathbf{d}_{\ell i} = \left\| \sum_{j=1}^{\dim V_\ell} \left[\text{ML-Net}[\boldsymbol{\theta}^*; L](\boldsymbol{\kappa}_{\mathbf{y}_i})\right]_{\ell j} \varphi_j^{(\ell)} - \frac{1}{\delta_\ell} \hat{v}_\ell(\mathbf{y}_i) \right\|_{H^1(D)}.$$

Normalization via δ_ℓ assures that each level is weighted equally in the loss function.

Step 3: Inference. Denoting the optimized parameters by $\boldsymbol{\theta}^*$, the approximate solution (i.e., the NN prediction) for $\mathbf{y} \in \Gamma$ is computed by

$$v(\mathbf{y}) \approx \sum_{\ell=1}^{L} \delta_\ell \sum_{j=1}^{\dim V_\ell} \left[\text{ML-Net}[\boldsymbol{\theta}^*; L](\boldsymbol{\kappa}_{\mathbf{y}})\right]_{\ell j} \varphi_j^{(\ell)}.$$

In the following remark, a simpler baseline method is introduced that can be seen as a special case of ML-Net.

Remark 4.1. *As a comparison and to assess the benefits of the described multilevel approach, we propose a simpler NN, which directly maps the discretized parameter dependent diffusion coefficient $\kappa_{\mathbf{y}}$ to the coefficients on the finest level. For this, we denote by $UNetSeq$ a sequence*

of $UNet$s which can be seen as a variation of $ML\text{-}Net$ where $\boldsymbol{R}_\ell = 0$ for levels $\ell = 1, \dots, L-1$, i.e., only the highest level sub-module is used. This architecture is a hybrid approach between single-level and multi-level because of the internal multi-resolution structure of the $UNet$ blocks. For this architecture, we choose the number of $UNet$s such that the total parameter count roughly matches $ML\text{-}Net$.

5 Theoretical analysis

In this section, we show that UNet-like NN architectures are able to approximate multigrid V-cycles on V_h up to arbitrary accuracy $\varepsilon > 0$ with the number of parameters independent on ε and growing only logarithmically in $1/h$ (Theorem 5.1). We use this result in Corollary 5.1 to approximate the discrete solution $\mathbf{v}_h \colon \Gamma \to \mathbb{R}^{\dim V_h}$ of the parametric PDE Problem 2.1 with arbitrary accuracy ε by a NN with the number of parameters bounded from above by $\log(1/h)\log(1/\varepsilon)$ independent of the stochastic parameter dimension p. This is achieved by a NN that emulates a multigrid solver applied to each $\mathbf{y}$ individually and, thus, approximately computes the solution $\mathbf{v}_h(\mathbf{y})$. Eventually, we derive similar approximation bounds for the $\mathrm{ML\text{-}Net}$ architecture and point out computational advantages of $\mathrm{ML\text{-}Net}$ over pure multigrid approximations of Corollary 5.1. Note that while only the two-dimensional setting is considered in this section, the results can be translated to arbitrary spatial dimensions.

The rigorous mathematical analysis of NNs requires an extensive formalism, which can e.g., be found in [60, 35, 33]. We assume that readers interested in the proofs are familiar with common techniques such as concatenation/parallelisation of NNs and the approximation of polynomials, the identity, and algebraic operations. To make our presentation more accessible from a "practitioner point of view", we present a streamlined exposition which still includes sufficient details to follow the arguments of the proofs. All proofs can be found in Appendix C.

We consider a CNN as any computational graph consisting of (various kinds of) convolutions in combination with standard building blocks from common DL libraries (e.g., skip connections and pooling layers). Note that this conception also includes the prominent UNet architecture [64] and other UNet-like architectures with the same recursive structure.

For our theoretical analysis, we consider activation functions $\varrho \in L^\infty_{\mathrm{loc}}(\mathbb{R})$ such that there exists $x_0 \in \mathbb{R}$ with ϱ three times continuously differentiable in a neighborhood of some $x_0 \in \mathbb{R}$ and $\varrho''(x_0) \neq 0$. This includes many standard activation functions such as exponential linear unit, softsign, swish, and sigmoids. For (leaky) ReLUs a similar but more involved analysis would be possible, which we avoid for the sake of simplicity.

For a CNN Ψ, we denote the number of weights by $M(\Psi)$. Since we consider the FE spaces V_h in the two-dimensional setting on uniformly refined square meshes, FE coefficient vectors $\mathbf{x} \in \mathbb{R}^{\dim V_h}$ can be viewed as two-dimensional arrays. Whenever $\mathbf{x}$ is processed by a CNN, we implicitly assume a 2D matrix representation. We define $\|\mathbf{x}\|_{H^1} \coloneqq \left\| \sum_{i=1}^{\dim V_h} \mathbf{x}_i \varphi_i \right\|_{H^1}$. For this section, we set $D = [0,1]^2$ and assume uniform ellipticity for Problem 2.1.

We start with the approximation of the solution of the multigrid V-cycle $\mathrm{MG}^m_{k_0,k}$ with initialization $\mathbf{u}_0$, diffusion coefficient $\boldsymbol{\kappa}$, and right-hand side $\mathbf{f}$ by UNet-like CNNs.

Theorem 5.1. *Let $V_h \subseteq H^1_0(D)$ be the P1 FE space on a uniform square mesh. Then there exists a constant $C > 0$ such that for any $M, \varepsilon > 0$ and $m, k, k_0 \in \mathbb{N}$ there exists a CNN $\Psi \colon \mathbb{R}^{3\times\dim V_h} \to \mathbb{R}^{\dim V_h}$ with*

(i) $\left\|\Psi(\mathbf{u}_0, \boldsymbol{\kappa}, \mathbf{f}) - \mathrm{MG}^m_{k_0,k}(\mathbf{u}_0, \boldsymbol{\kappa}, \mathbf{f})\right\|_{H^1(D)} \leq \varepsilon$ *for all* $\boldsymbol{\kappa}, \mathbf{u}_0, \mathbf{f} \in [-M, M]^{\dim V_h}$,

(ii) number of weights bounded by $M(\Psi) \leq C\left(k_0 + k \log\left(\frac{1}{h}\right)\right) m$.

In the next remark, we provide more details about the architecture of Ψ and its implications.

Remark 5.1. *The proof of Theorem 5.1 reveals that* Ψ *resembles the concatenation of multiple UNet-like subnetworks, each approximating one V-cycle. We draw two conclusions from that:*

- *Generally, this supports the heuristic relation between UNets and multigrid methods with a rigorous mathematical analysis indicating a possible reason for their success in multiscale-related tasks e.g., found in medical imaging [51] or PDE-related problems [see 14, 42].*
- *Together with our numerical results in the following section, this underlines the suitability of* $UNetSeq$ *(see Remark 4.1) for the solution of Problem 2.1. This is made more specific in the next corollary.*

The following corollary is an easy consequence of Theorem 5.1 for the parametric PDE Problem 2.1 by applying the multigrid solver to solve Equation (2.4) for each $\mathbf{y} \in \Gamma$ individually. We use that the required number of smoothing iterations in Theorem 3.1 is grid independent and choose the number of V-cycles as $m \approx \log(1/\varepsilon)$. An application of the triangular inequality yields the following result.

Corollary 5.1. *Consider the discretized Problem 2.1 with the conforming P1 FE space* $V_h \subseteq H^1_0(D)$. *Assume that* $\boldsymbol{\kappa}_\mathbf{y}$ *is uniformly bounded over all* $\mathbf{y} \in \Gamma$. *Then there exists a constant* $C > 0$ *such that for any* $\varepsilon > 0$ *there exists a CNN* $\Psi : \mathbb{R}^{2\times \dim V_h} \to \mathbb{R}^{\dim V_h}$ *with*

(i) $\left\|\Psi(\boldsymbol{\kappa}_\mathbf{y}, \mathbf{f}) - \mathbf{v}_h(\mathbf{y})\right\|_{H^1(D)} \leq \varepsilon \left\|f\right\|_*$ *for all* $\mathbf{y} \in \Gamma$,

(ii) number of weights bounded by $M(\Psi) \leq C \log\left(\frac{1}{h}\right) \log\left(\frac{1}{\varepsilon}\right)$.

Setting the NN approximation error equal to the FE discretization error ($\varepsilon = h$), we get the bound $M(\Psi) \leq C \log(1/h)^2$. This shows that the number of parameters at most grows polylogarithmically in $1/h$ and is independent of the stochastic parameter dimension p.

Finally, we provide complexity estimates for the $\mathrm{ML\text{-}Net}$ architecture in the next corollary. Here, at each level of $\mathrm{ML\text{-}Net}$ a multigrid V-cycle (on the respective level) is approximated by using Theorem 5.1.

Corollary 5.2. *Consider the discretized Problem 2.1. Let the spaces* $V_1, \dots, V_L$ *be defined as in Section 3 for* $L \in \mathbb{N}$, $f \in H^{-1}(D)$ *and its discretization denoted by* $\mathbf{f} \in \mathbb{R}^{\dim V_L}$. *Assume that the discretized diffusion coefficient* $\boldsymbol{\kappa}_\mathbf{y} \in \mathbb{R}^{\dim V_L}$ *is uniformly bounded over all* $\mathbf{y} \in \Gamma$. *Then, there exists a constant* $C > 0$ *such that for every* $\varepsilon > 0$ *there exists an* $ML\text{-}Net$ Ψ *(as in Section 4) with*

(i) $\left\|\Psi(0, \boldsymbol{\kappa}_\mathbf{y}, \mathbf{f}) - \mathbf{v}_L(\mathbf{y})\right\|_{H^1(D)} \leq \varepsilon \left\|f\right\|_*$ *for all* $\mathbf{y} \in \Gamma$,

(ii) $M(\Psi) \leq CL \log\left(\frac{1}{\varepsilon}\right) + CL^2$.

In the following remark, we discuss the relation of the previous two corollaries.

Remark 5.2. *Equilibrating approximation and discretization error, i.e., setting $\varepsilon = h = 2^{-L}$, we make two observations:*

- *Both, Corollary 5.1 and 5.2, yield an upper bound for the number weights of $M(\Psi) \leq CL^2$. We conclude that from our upper bounds no advantage in terms of expressiveness for using $ML\text{-}Net$ over $UNetSeq$ is evident. In Section 6.2 we show that our numerical experiments support this finding.*

- *Despite a similar number of parameters in relation to the approximation accuracy, a forward pass of ML-net is more efficient than a forward pass of $UNetSeq$. This is due to the multilevel structure of the $ML\text{-}Net$ that shifts the computational load more evenly across resolutions. Note that the number of parameters of a convolutional filter is independent on the input resolution, whereas the computational complexity of convolving the input with the filter is not. In detail, the number of operations for a forward pass of $ML\text{-}Net$ is $\mathcal{O}(2^{2L}) = \mathcal{O}(h^{-2})$, while for $UNetSeq$, we have $\mathcal{O}(L2^{2L}) = \mathcal{O}(\log(h^{-1})h^{-2})$ computations. This is one advantage of $ML\text{-}Net$ that results in shorter training times (see Section 6).*

6 Numerical results

This section is concerned with the evaluation of $\mathrm{ML\text{-}Net}$ and the simpler $\mathrm{UNetSeq}$ on several common benchmark problems in terms of the mean relative H^1-distance of the predicted solution. Moreover, the performance of $\mathrm{ML\text{-}Net}$ is tested with a declining number of training samples for successive grid levels. This is motivated by multilevel theory and leads to a beneficial training complexity (see Remark 5.2 for a more detailed evaluation).

We assess the performance of our methods for different choices of varying computational complexity for the coefficient functions a_k, comprising the diffusivity field κ by (2.2) and different parameter distributions π. For all test cases, we choose the unit square domain $D = [0,1]^2$, $a_0(x) \coloneqq a_0 \in \mathbb{R}$ constant and the right-hand side $f \equiv 1$. The test problems are defined as follows.

1 **Uniform smooth field.** Let $\Gamma = [-1,1]^p$ and $\pi = U[-1,1]^p$. Moreover, let $a_0 := 1$ and choose a_k as planar Fourier modes as described in [22] and [27] with decay $\|a_k\|_\infty = 0.1k^{-2}$ for $1 \leq k \leq p$.

2 **Log-normal smooth field.** Let $\Gamma = \mathbb{R}^p$ and $\pi = N(\mathbf{0}, \mathrm{Id}_p)$. Define $\kappa(\mathbf{y}, x) := \exp(\tilde{\kappa}(\mathbf{y}, x))$, where $\tilde{\kappa}$ equals κ from the uniform case with $a_0 := 0$.

3 **Cookie problem with fixed radii inclusions.** Let $\Gamma = [-1,1]^p$, with p having a natural square root and $\pi = U[-1,1]^p$. Moreover, define $a_0 := 0.1$ and $a_k(x) = \mathcal{X}_{D_k}(x)$, where D_k are disks centered in a cubic lattice with a fixed radius $r = 0.3p^{-1/2}$ for $1 \leq k \leq p$.

4 **Cookie problem with variable radii inclusions.** As an extension of the cookie problem, we also assume the disk radii to vary. To this end, let p' be the (quadratic) number

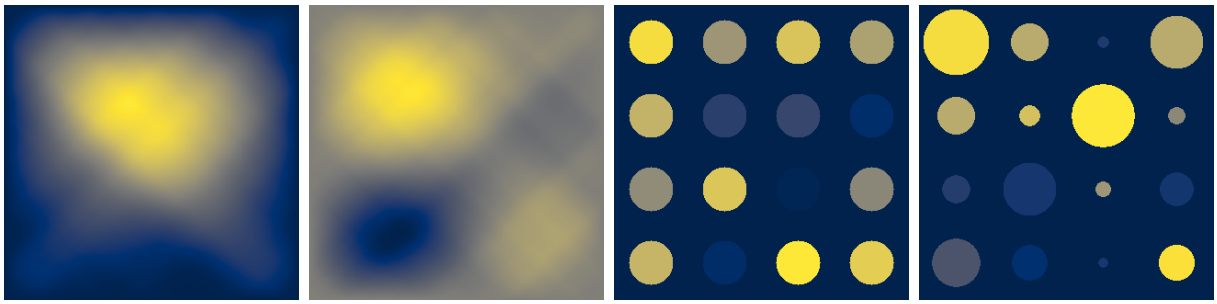

Figure 2: Example realizations of $\kappa(\cdot, \mathbf{y})$ for the four test cases. From left to right: uniform case ($p = 100$), log-normal case ($p = 100$), cookie problem ($p = 16$), cookie problem with variable radii ($p = 32$). The colormaps are normalized for visibility.

of cookies, $p = 2p'$, $\Gamma = [-1, 1]^p$ and $\pi = U[-1, 1]^p$. The diffusion coefficient is defined by

$$\kappa(\mathbf{y}, x) := a_0 + \sum_{k=1}^{p'} \mathbf{y}_{2k-1} \mathcal{X}_{\mathscr{D}_{k,r(\mathbf{y}_{2k})}}(x),$$

where the $\mathscr{D}_{k,r}$ are disks centered in a cubic lattice with parameter dependent radius r. We choose $r(y)$ for $y \in [-1, 1]$ such that the radii are uniformly distributed in $[0.5(p')^{-1/2}, 0.9(p')^{-1/2}]$.

Figure 2 depicts a random realization for each of the 4 problem cases. Note that not all of the above problems satisfy the theoretical assumptions for the results in Section 5. Notably, the log-normal case is not uniformly elliptic and our theoretical guarantees thus only hold with (arbitrarily) high probability as indicated above. Moreover, in case of the cookie problem with variable radii, the diffusion coefficient depends non-linearly on the parameter vector since also the basis functions a_k are parameter dependent. We include these cases to study our methodology with diverse and particularly challenging setups.

We use the open source package `FEniCS` [52] with the GMRES [65] solver for carrying out the FE simulations to generate training data and `PyTorch` [59] for the NN models.

If not stated otherwise, we use a training dataset with 10^4 samples and 1024 samples for independent validation computed from i.i.d. parameter vectors. For testing, the performance of the methods is evaluated on 1024 independently generated test samples.

For the multilevel decomposition, we consider $L = 7$ resolution levels starting with a coarse resolution of 5×5 cells. Iterative uniform mesh refinement with factor $\eta = 2$ results in 320×320 cells on the finest level. To reduce computational complexity, we only compute solutions on the finest level and derive the coarser solutions as well as the corrections in (3.1) using nodal interpolation. Additionally, we compute reference test solutions on a high-fidelity mesh with 1280×1280 cells, resulting in about 1.6×10^6 degrees of freedom.

For $\mathrm{UNetSeq}$, we choose the number of UNets such that the total parameter count roughly matches the 5.6×10^6 parameters of $\mathrm{ML\text{-}Net}$. For training, we use the Adam [45] optimizer and train for 200 epochs with learning rate decay. After each run, we evaluate the model with the best validation performance. Training took approximately 33 GPU-hours on an NVIDIA Tesla P100 for $\mathrm{ML\text{-}Net}$ and 46 GPU-hours for $\mathrm{UNetSeq}$ (see also Remark 5.2).

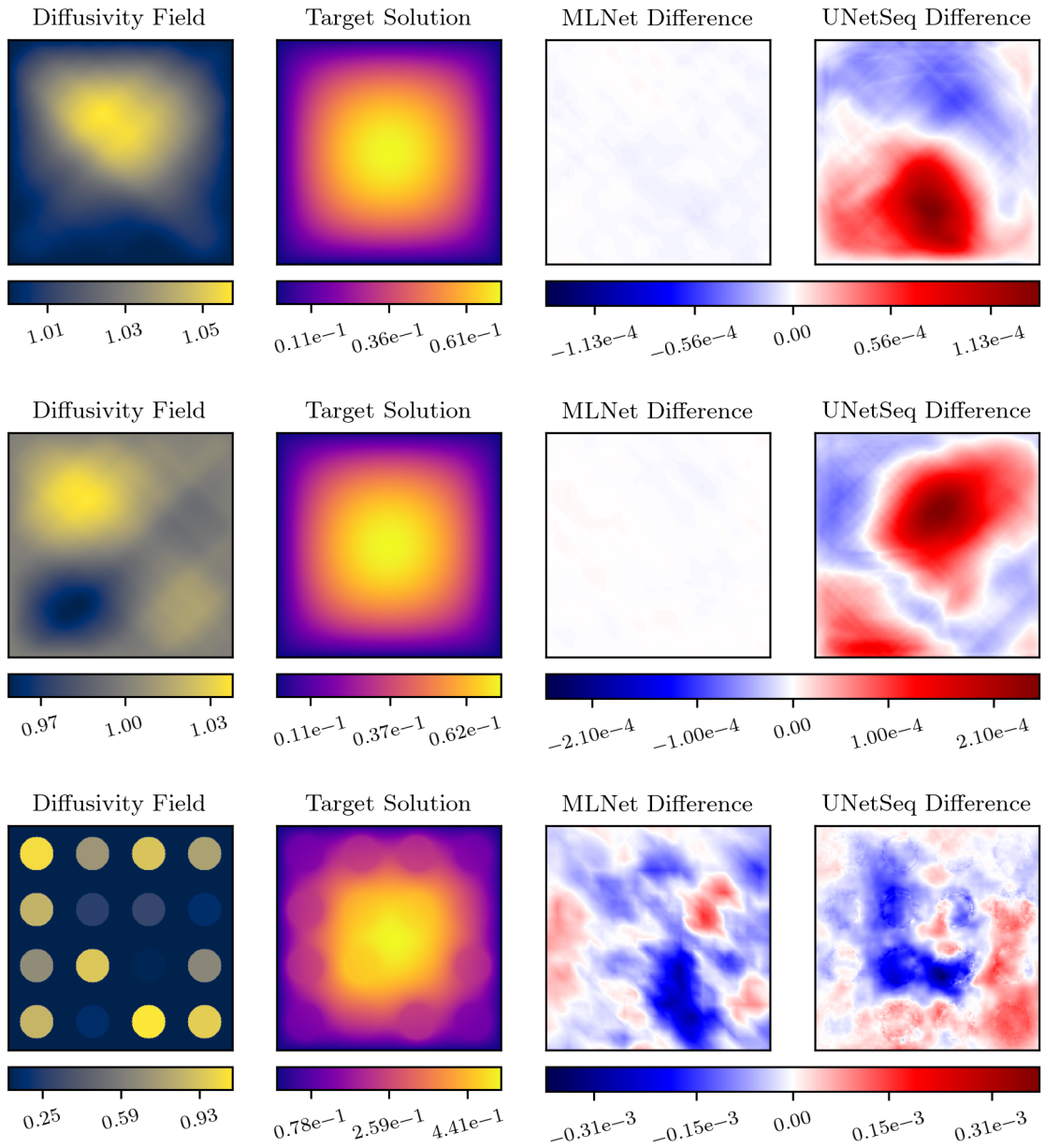

Figure 3: Realizations for different benchmark problems (top to bottom: uniform $p = 100$, log-normal $p = 100$, cookie fixed). Left-hand side columns show field realizations and respective solutions. Right-hand side pictures the error of ML-Net and UNetSeq predictions w.r.t. exact target solutions.

6.1 Error metrics

We examine the mean relative H^1 error (MRH^1) and mean relative L^2 error (MRL^2) with respect to the solutions on the finest resolution level L as well as the high-fidelity reference solutions. For parameters $\mathbf{y}_1, \dots, \mathbf{y}_N \in \Gamma$, predictions $u_1, \dots, u_N \in V_L$, and the solution operators $v_L \colon \Gamma \to V_L$ and $v_{\mathsf{ref}} \colon \Gamma \to H^1(D)$ mapping to the discrete space on the finest grid and the reference grid, respectively, we define

$$\mathcal{E}_{\mathsf{MR}*} := \sqrt{\frac{\sum_{i=1}^N \|u_i - v_L(\mathbf{y}_i)\|_*^2}{\sum_{i=1}^N \|v_L(\mathbf{y}_i)\|_*^2}} \quad \text{and} \quad \mathcal{E}^{\mathsf{ref}}_{\mathsf{MR}*} := \sqrt{\frac{\sum_{i=1}^N \|u_i - v_{\mathsf{ref}}(\mathbf{y}_i)\|_*^2}{\sum_{i=1}^N \|v_{\mathsf{ref}}(\mathbf{y}_i)\|_*^2}},$$

with $* \in \{H^1, L^2\}$.

6.2 Results for the test cases

Tables 1 and 2 depict the $\mathcal{E}_{\mathsf{MR}H^1}$ and $\mathcal{E}^{\mathsf{ref}}_{\mathsf{MR}H^1}$ for $\mathrm{ML\text{-}Net}$ and $\mathrm{UNetSeq}$ for the numerical test cases described above with varying stochastic parameter dimensions. $\mathcal{E}_{\mathsf{MR}L^2}$ and $\mathcal{E}^{\mathsf{ref}}_{\mathsf{MR}L^2}$ are shown in Tables 4 and 5 in Appendix B. A random selection of solutions and NN predictions is also shown in Figure 3.

In comparison to previously reported performances of DL-based approaches for these benchmark problems in [27, 50, 54, 32], we observe an improvement of one to two orders of magnitude with our methodology.

[27] observed a strong dependence of the performance of their DL-based method on the stochastic parameter dimension. In contrast, both $\mathrm{ML\text{-}Net}$ and $\mathrm{UNetSeq}$ perform very consistently with increasing parameter dimensions although the problems become more involved. This is in line with the parameter independent bounds for CNNs derived in Section 5.

$\mathcal{E}_{\mathsf{MR}H^1}$ is generally significantly lower than $\mathcal{E}^{\mathsf{ref}}_{\mathsf{MR}H^1}$. This illustrates that for these cases the NN approximation error on the finest grid $L = 7$ can be several magnitudes lower than the FE approximation error. Conversely, the discrepancy between $\mathcal{E}_{\mathsf{MR}L^2}$ and $\mathcal{E}^{\mathsf{ref}}_{\mathsf{MR}L^2}$ seen in Tables 4 and 5 is far less pronounced.

Comparing $\mathrm{ML\text{-}Net}$ to $\mathrm{UNetSeq}$, we observe that both models generally exhibit a comparable performance. Lower variances in model accuracy for $\mathrm{ML\text{-}Net}$ indicate an improved training stability when using a suitable multilevel decomposition.

6.3 Training with successively fewer samples on finer levels

A striking advantage of stochastic multilevel methods is that the majority of sample points can be computed on coarser levels with low effort while only few samples are needed on the finest grid [71, 55, 39, 5]. We transfer this concept to $\mathrm{ML\text{-}Net}$ by exponentially decreasing the number of training samples from level to level. To this end, we train each level of our $\mathrm{ML\text{-}Net}$ only with a fraction of the dataset. More concretely, we divide the number of samples in each subsequent level by two, i.e for level $\ell = 1, \dots, L$, we use $N_\ell \coloneqq 2^{1-\ell} \times 10^4$ training samples. We observe that alternating between low level samples and samples for which all corrections are known during training is needed for stable optimization. The computational budget of generating a multiscale dataset with 1000 samples on the coarsest level and exponentially decaying number of samples on subsequent levels corresponds to 232 full resolution samples.

problem	parameter dimension p	$\mathcal{E}_{\mathrm{MR}H^1}$	
		ML-Net	UNetSeq
uniform	10	$2.24\mathrm{e}{-4} \pm 9.62\mathrm{e}{-5}$	$9.75\mathrm{e}{-5} \pm 2.10\mathrm{e}{-5}$
	50	$2.21\mathrm{e}{-4} \pm 2.96\mathrm{e}{-5}$	$2.08\mathrm{e}{-3} \pm 2.79\mathrm{e}{-3}$
	100	$2.27\mathrm{e}{-4} \pm 2.72\mathrm{e}{-5}$	$8.89\mathrm{e}{-5} \pm 1.00\mathrm{e}{-5}$
	200	$2.31\mathrm{e}{-4} \pm 3.50\mathrm{e}{-5}$	$2.05\mathrm{e}{-3} \pm 2.76\mathrm{e}{-3}$
log-normal	10	$2.15\mathrm{e}{-4} \pm 6.18\mathrm{e}{-5}$	$2.83\mathrm{e}{-3} \pm 3.70\mathrm{e}{-3}$
	50	$9.73\mathrm{e}{-4} \pm 1.04\mathrm{e}{-3}$	$3.71\mathrm{e}{-3} \pm 4.91\mathrm{e}{-3}$
	100	$2.64\mathrm{e}{-4} \pm 1.73\mathrm{e}{-5}$	$2.82\mathrm{e}{-3} \pm 4.31\mathrm{e}{-3}$
	200	$3.00\mathrm{e}{-4} \pm 1.43\mathrm{e}{-5}$	$1.97\mathrm{e}{-4} \pm 4.22\mathrm{e}{-5}$
cookie fixed	16	$9.41\mathrm{e}{-4} \pm 1.12\mathrm{e}{-4}$	$1.10\mathrm{e}{-3} \pm 3.76\mathrm{e}{-4}$
	64	$1.85\mathrm{e}{-3} \pm 1.38\mathrm{e}{-4}$	$7.20\mathrm{e}{-4} \pm 1.43\mathrm{e}{-4}$
cookie variable	32	$4.69\mathrm{e}{-3} \pm 1.41\mathrm{e}{-3}$	$3.69\mathrm{e}{-3} \pm 4.53\mathrm{e}{-4}$
	128	$5.98\mathrm{e}{-3} \pm 1.13\mathrm{e}{-4}$	$3.08\mathrm{e}{-3} \pm 5.07\mathrm{e}{-4}$

Table 1: $\mathcal{E}_{\mathrm{MR}H^1}$ for ML-Net and UNetSeq evaluated on all test cases.

problem	parameter dimension p	$\mathcal{E}^{\mathrm{ref}}_{\mathrm{MR}H^1}$	
		ML-Net	UNetSeq
uniform	10	$5.33\mathrm{e}{-3} \pm 4.50\mathrm{e}{-6}$	$5.33\mathrm{e}{-3} \pm 7.01\mathrm{e}{-7}$
	50	$5.33\mathrm{e}{-3} \pm 1.15\mathrm{e}{-6}$	$6.24\mathrm{e}{-3} \pm 1.28\mathrm{e}{-3}$
	100	$5.33\mathrm{e}{-3} \pm 1.63\mathrm{e}{-6}$	$5.33\mathrm{e}{-3} \pm 8.55\mathrm{e}{-7}$
	200	$5.33\mathrm{e}{-3} \pm 1.46\mathrm{e}{-6}$	$6.22\mathrm{e}{-3} \pm 1.25\mathrm{e}{-3}$
log-normal	10	$5.33\mathrm{e}{-3} \pm 2.93\mathrm{e}{-6}$	$6.78\mathrm{e}{-3} \pm 2.04\mathrm{e}{-3}$
	50	$5.51\mathrm{e}{-3} \pm 2.49\mathrm{e}{-4}$	$7.53\mathrm{e}{-3} \pm 3.10\mathrm{e}{-3}$
	100	$5.33\mathrm{e}{-3} \pm 5.54\mathrm{e}{-7}$	$6.90\mathrm{e}{-3} \pm 2.71\mathrm{e}{-3}$
	200	$5.34\mathrm{e}{-3} \pm 3.03\mathrm{e}{-6}$	$5.33\mathrm{e}{-3} \pm 1.69\mathrm{e}{-6}$
cookie fixed	16	$7.09\mathrm{e}{-2} \pm 1.87\mathrm{e}{-5}$	$7.09\mathrm{e}{-2} \pm 7.39\mathrm{e}{-6}$
	64	$9.73\mathrm{e}{-2} \pm 1.16\mathrm{e}{-5}$	$9.73\mathrm{e}{-2} \pm 5.62\mathrm{e}{-6}$
cookie variable	32	$7.83\mathrm{e}{-2} \pm 3.22\mathrm{e}{-4}$	$7.81\mathrm{e}{-2} \pm 2.39\mathrm{e}{-4}$
	128	$1.12\mathrm{e}{-1} \pm 2.58\mathrm{e}{-4}$	$1.12\mathrm{e}{-1} \pm 2.76\mathrm{e}{-5}$

Table 2: $\mathcal{E}^{\mathrm{ref}}_{\mathrm{MR}H^1}$ for ML-Net and UNetSeq evaluated on all test cases.

method	error	dataset	uniform	log-normal
ML-Net	$\mathcal{E}_{\mathrm{MR}H^1}$	decaying	$2.86\mathrm{e}{-4} \pm 2.24\mathrm{e}{-5}$	$4.05\mathrm{e}{-4} \pm 2.48\mathrm{e}{-5}$
UNetSeq	$\mathcal{E}_{\mathrm{MR}H^1}$	fixed	$1.14\mathrm{e}{-3} \pm 6.47\mathrm{e}{-4}$	$5.92\mathrm{e}{-3} \pm 4.66\mathrm{e}{-3}$
ML-Net	$\mathcal{E}^{\mathrm{ref}}_{\mathrm{MR}H^1}$	decaying	$5.34\mathrm{e}{-3} \pm 1.22\mathrm{e}{-6}$	$5.34\mathrm{e}{-3} \pm 1.61\mathrm{e}{-6}$
UNetSeq	$\mathcal{E}^{\mathrm{ref}}_{\mathrm{MR}H^1}$	fixed	$5.49\mathrm{e}{-3} \pm 1.59\mathrm{e}{-4}$	$8.53\mathrm{e}{-3} \pm 3.53\mathrm{e}{-3}$
ML-Net	$\mathcal{E}_{\mathrm{MR}L^2}$	decaying	$5.75\mathrm{e}{-5} \pm 4.43\mathrm{e}{-6}$	$9.39\mathrm{e}{-5} \pm 5.42\mathrm{e}{-6}$
UNetSeq	$\mathcal{E}_{\mathrm{MR}L^2}$	fixed	$5.38\mathrm{e}{-4} \pm 3.93\mathrm{e}{-4}$	$3.76\mathrm{e}{-3} \pm 3.23\mathrm{e}{-3}$
ML-Net	$\mathcal{E}^{\mathrm{ref}}_{\mathrm{MR}L^2}$	decaying	$6.65\mathrm{e}{-5} \pm 2.16\mathrm{e}{-6}$	$9.91\mathrm{e}{-5} \pm 4.59\mathrm{e}{-6}$
UNetSeq	$\mathcal{E}^{\mathrm{ref}}_{\mathrm{MR}L^2}$	fixed	$5.40\mathrm{e}{-4} \pm 3.91\mathrm{e}{-4}$	$3.76\mathrm{e}{-3} \pm 3.23\mathrm{e}{-3}$

Table 3: All errors for ML-Net and UNetSeq trained with a reduced dataset size on two problem cases with parameter dimension $p = 100$. The ML-Net is trained with the number of training samples halved for each level and UNetSeq is trained using 232 samples in total.

Table 3 depicts the errors for two test cases (uniform and log-normal with $p = 100$) for ML-Net trained on the decayed multiscale dataset and UNetSeq trained on the full resolution dataset generated with a comparable compute budget (232 samples). We make two observations: First, training ML-Net with a decaying number of samples per level hardly decreases its performance when compared to the full dataset of 1000 samples from Table 4. Second, UNetSeq trained on a full-resolution dataset of comparable compute budget significantly reduces performance compared to training UNetSeq on 1000 samples (Table 4) and compared to ML-Net on the decayed dataset (Table 3).

The MRL^2 errors depicted in Table 3 illustrate the low error that can be achieved, which is at least one order of magnitude lower than what is reported in other papers, see also Tables 4 and 5 in Appendix B. Note that the observed MRH^1 error in our experiments is bounded from below by the FE approximation, i.e. the resolution of the finest mesh. For the multilevel advantage to fully take effect, training on finer grids would be required to decrease the FE approximation errors to be comparable to the smaller NN approximation errors. Hence, using fewer high fidelity training samples is a crucial step to train models which achieve an overall MRH^1 accuracy comparable to traditional FE solvers.

We conclude that ML-Net can effectively be trained with fewer samples for finer levels, significantly reducing the overall training and data generation costs and, thus, enabling the application of our model with much finer grids.

7 Conclusion

In this work, we combine concepts from established multilevel algorithms with NNs for efficiently solving challenging high-dimensional parametric PDEs. We provide a theoretical complexity analysis revealing the relation between UNet-like architectures approximate classical multigrid algorithms arbitrarily well. Moreover, we show that the ML-Net architecture presents an advantageous approach for learning the parameter-to-solution operator of the parametric Darcy problems. The performance of our method is illustrated by several numerical benchmark

experiments showing a significant improvement over the state-of-the-art of previous DL-based approaches for parametric PDEs. In fact, the shown approximation quality matches the best-in-class techniques such as low-rank least squares methods, stochastic Galerkin approaches, compressed sensing and stochastic collocation. Additionally, we show that the multilevel architecture allows to use fewer data points for finer corrections during training with negligible impact on practical performance. This enables the extension to much finer resolutions without a prohibitive increase of computational complexity. Finally, we note that while we consider a scalar linear elliptic equation in this work, our methodology can be used for a variety of possibly nonlinear and vector-valued problems. Future research directions could include the analysis of such problems as well as the extension of our method to adaptive grids.

The authors would like to thank Reinhold Schneider for fruitful discussions on the topic. I. Gühring acknowledges support from the Research Training Group "Differential Equation- and Data-driven Models in Life Sciences and Fluid Dynamics: An Interdisciplinary Research Training Group (DAEDALUS)" (GRK 2433) funded by the German Research Foundation (DFG) and a post-doctoral scholarship from Technical University of Berlin for "Deep Learning for (Parametric) PDEs from a Theoretical and Practical Perspective". M. Eigel acknowledges partial support from DFG SPP 1886 "Polymorphic uncertainty modelling for the numerical design of structures" and DFG SPP 2298 "Theoretical foundations of Deep Learning". We acknowledge the computing facilities and thank the IT support of the Weierstrass Institute.

References

[1] Ben Adcock and Nick Dexter. The gap between theory and practice in function approximation with deep neural networks. *SIAM Journal on Mathematics of Data Science*, 3(2):624–655, 2021.

[2] Anima Anandkumar, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Nikola Kovachki, Zongyi Li, Burigede Liu, and Andrew Stuart. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[3] Ivo Babuska, Fabio Nobile, and Raul Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45:1005–1034, 2007.

[4] Pierre Baldi. Deep learning in biomedical data science. *Annual Review of Biomedical Data Science*, 1(1):181–205, 2018.

[5] Jonas Ballani, Daniel Kressner, and Michael Peters. Multilevel tensor approximation of pdes with random data. *Stochastics and Partial Differential Equations: Analysis and Computations*, 5:400–427, 2016.

[6] Christian Beck, Martin Hutzenthaler, Arnulf Jentzen, and Benno Kuckuck. An overview on deep learning-based approximation methods for partial differential equations. *Discrete and Continuous Dynamical Systems - B*, 28(6):3697–3746, 2023.

[7] Julius Berner, Markus Dablander, and Philipp Grohs. Numerically solving parametric families of high-dimensional kolmogorov partial differential equations via deep learning. *Advances in Neural Information Processing Systems*, 33:16615–16627, 2020.

[8] Julius Berner, Philipp Grohs, and Arnulf Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black–scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020.

[9] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

[10] Dietrich Braess. *Finite elements: Theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, 2007.

[11] Dietrich Braess and Wolfgang Hackbusch. A new convergence proof for the multigrid method including the v-cycle. *Siam Journal on Numerical Analysis*, 20:967–975, 1983.

[12] Yuyan Chen, Bin Dong, and Jinchao Xu. Meta-mgnet: Meta multigrid networks for solving parameterized partial differential equations. *Journal of Computational Physics*, page 110996, 2022.

[13] Abdellah Chkifa, Albert Cohen, Giovanni Migliorati, Fabio Nobile, and Raul Tempone. Discrete least squares polynomial approximation with random evaluations- application to parametric and stochastic elliptic pdes. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 49(3):815–837, 2015.

[14] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, and Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4621–4630, 2021.

[15] Philippe G. Ciarlet. *The Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics, 2002.

[16] Albert Cohen and Ronald DeVore. Approximation of high-dimensional parametric pdes. *Acta Numerica*, 24:1–159, 2015.

[17] Albert Cohen and Giovanni Migliorati. Near-optimal approximation methods for elliptic pdes with lognormal coefficients. *Mathematics of Computation*, 2023.

[18] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[19] Martin Eigel, Claude Jeffrey Gittelson, Christoph Schwab, and Elmar Zander. Adaptive stochastic galerkin FEM. *Computer Methods in Applied Mechanics and Engineering*, 270:247–269, March 2014.

[20] Martin Eigel, Manuel Marschall, Max Pfeffer, and Reinhold Schneider. Adaptive stochastic galerkin FEM for lognormal coefficients in hierarchical tensor representations. *Numerische Mathematik*, 145(3):655–692, 6 2020.

[21] Martin Eigel, Reinhold Schneider, Philipp Trunschke, and Sebastian Wolf. Variational monte carlo—bridging concepts of machine learning and high-dimensional partial differential equations. *Advances in Computational Mathematics*, 45(5-6):2503–2532, October 2019.

[22] Martin Eigel, Reinhold Schneider, Philipp Trunschke, and Sebastian Wolf. Variational monte carlo—bridging concepts of machine learning and high-dimensional partial differential equations. *Advances in Computational Mathematics*, 45(5):2503–2532, 2019.

[23] Oliver G Ernst, Bjorn Sprungk, and Lorenzo Tamellini. Convergence of sparse collocation for functions of countably many gaussian random variables (with application to elliptic pdes). *SIAM Journal on Numerical Analysis*, 56(2):877–905, 2018.

[24] Yuwei Fan, Jordi Feliu-Faba, Lin Lin, Lexing Ying, and Leonardo Zepeda-Núnez. A multiscale neural network based on hierarchical nested bases. *Research in the Mathematical Sciences*, 6(2):1–28, 2019.

[25] Yuwei Fan, Lin Lin, Lexing Ying, and Leonardo Zepeda-Núnez. A multiscale neural network based on hierarchical matrices. *Multiscale Modeling & Simulation*, 17(4):1189–1213, 2019.

[26] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative convnets via multi-grid modeling and sampling. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9155–9164, 2018.

[27] Moritz Geist, Philipp Petersen, Mones Raslan, Reinhold Schneider, and Gitta Kutyniok. Numerical solution of the parametric diffusion equation by deep neural networks. *Journal of Scientific Computing*, 88(1):1–37, 2021.

[28] Martin Genzel, Ingo Gühring, Jan MacDonald, and Maximilian März. Near-exact recovery for tomographic inverse problems via deep learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 7368–7381. PMLR, 2022.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, Cambridge, 2016.

[30] Philipp Grohs and Lukas Herrmann. Deep neural network approximation for high-dimensional elliptic pdes with boundary conditions. *IMA Journal of Numerical Analysis*, 42(3):2055–2082, 2022.

[31] Philipp Grohs, Fabian Hornung, Arnulf Jentzen, and Philippe Von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of black-scholes partial differential equations. *arXiv preprint arXiv:1809.02362*, 2018.

[32] Tamara G Grossmann, Urszula Julia Komorowska, Jonas Latz, and Carola-Bibiane Schönlieb. Can physics-informed neural networks beat the finite element method? *arXiv preprint arXiv:2302.04107*, 2023.

[33] Ingo Gühring and Mones Raslan. Approximation rates for neural networks with encodable weights in smoothness spaces. *Neural Networks*, 134:107–130, 2021.

[34] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 481–490, New York, NY, USA, 2016. Association for Computing Machinery.

[35] Ingo Gühring, Gitta Kutyniok, and Philipp Petersen. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *Analysis and Applications (Singap.)*, 18(05):803–859, 2020.

[36] Ingo Gühring, Mones Raslan, and Gitta Kutyniok. Expressivity of deep neural networks. In Philipp Grohs and Gitta Kutyniok, editors, *Mathematical Aspects of Deep Learning*, page 149–199. Cambridge University Press, 2022.

[37] Wolfgang Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[38] Jiequn Han, Arnulf Jentzen, et al. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in mathematics and statistics*, 5(4):349–380, 2017.

[39] Helmut Harbrecht, Michael Peters, and Markus Siebenmorgen. Multilevel accelerated quadrature for pdes with log-normally distributed diffusion coefficient. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):520–551, 2016.

[40] Juncai He and Jinchao Xu. Mgnet: A unified framework of multigrid and convolutional neural network. *Science China Mathematics*, 62(7):1331–1354, 2019.

[41] J.S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

[42] Zhihao Jiang, Pejman Tahmasebi, and Zhiqiang Mao. Deep residual u-net convolution neural networks with autoregressive strategy for fluid flow predictions in large-scale geosystems. *Advances in Water Resources*, 150:103878, 02 2021.

[43] John A Keith, Valentin Vassilev-Galindo, Bingqing Cheng, Stefan Chmiela, Michael Gastegger, Klaus-Robert Müller, and Alexandre Tkatchenko. Combining machine learning and computational chemistry for predictive insights into chemical systems. *Chemical Reviews*, 121(16):9816–9872, 2021.

[44] Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*, 2019.

[45] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[46] Omar M Knio and OP Le Maitre. Uncertainty propagation in cfd using polynomial chaos decomposition. *Fluid dynamics research*, 38(9):616, 2006.

[47] Gitta Kutyniok, Philipp Petersen, Mones Raslan, and Reinhold Schneider. A theoretical analysis of deep neural networks and parametric pdes. *Constructive Approximation*, 55(1):73–125, 2022.

[48] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

[49] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[50] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[51] Liangliang Liu, Jianhong Cheng, Quan Quan, Fang-Xiang Wu, Yu-Ping Wang, and Jianxin Wang. A survey on u-shaped networks in medical image segmentations. *Neurocomputing*, 409:244–258, 2020.

[52] Anders Logg and Garth N. Wells. Dolfin. *ACM Transactions on Mathematical Software*, 37(2):1–28, 2010.

[53] Gabriel J Lord, Catherine E Powell, and Tony Shardlow. *An introduction to computational stochastic PDEs*, volume 50. Cambridge University Press, 2014.

[54] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[55] Kjetil O Lye, Siddhartha Mishra, and Roberto Molinaro. A multi-level procedure for enhancing accuracy of machine learning algorithms. *European Journal of Applied Mathematics*, 32(3):436–469, 2021.

[56] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA Journal of Numerical Analysis*, 43(1):1–43, 01 2022.

[57] Fabio Nobile, Raúl Tempone, and Clayton G Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2309–2345, 2008.

[58] Frank Noé, Alexandre Tkatchenko, Klaus-Robert Müller, and Cecilia Clementi. Machine learning for molecular simulation. *Annual Review of Physical Chemistry*, 71(1):361–390, 2020.

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[60] Philipp Petersen and Felix Voigtländer. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108:296–330, 2018.

[61] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[62] Amuthan A Ramabathiran and Prabhu Ramachandran. Spinn: sparse, physics-based, and partially interpretable neural networks for pdes. *Journal of Computational Physics*, 445:110600, 2021.

[63] Lewis Fry Richardson and Richard Tetley Glazebrook. Ix. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210(459-470):307–357, 1911.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[65] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 7(3):856–869, 1986.

[66] Peter Sadowski and Pierre Baldi. Deep learning in the natural sciences: Applications to physics. In Lev Rozonoer, Boris Mirkin, and Ilya Muchnik, editors, *Braverman Readings in Machine Learning. Key Ideas from Inception to Current State: International Conference Commemorating the 40th Anniversary of Emmanuil Braverman's Decease, Boston, MA, USA, April 28-30, 2017, Invited Talks*, pages 269–297. Springer International Publishing, Cham, 2018.

[67] Christoph Schwab and Claude Jeffrey Gittelson. Sparse tensor discretizations of high-dimensional parametric and stochastic pdes. *Acta Numerica*, 20:291–467, 2011.

[68] Christoph Schwab and Jakob Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq. *Analysis and Applications*, 17(01):19–55, 2019.

[69] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[70] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[71] A. L. Teckentrup, P. Jantsch, C. G. Webster, and M. Gunzburger. A multilevel stochastic collocation method for partial differential equations with random input data. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1046–1074, 2015.

[72] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2020.

[73] Xuefeng Xu and Chen-Song Zhang. Convergence analysis of inexact two-grid methods: A theoretical framework. *SIAM Journal on Numerical Analysis*, 60(1):133–156, 2022.

[74] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

[75] Harry Yserentant. Old and new convergence proofs for multigrid methods. *Acta numerica*, 2:285–326, 1993.

[76] Bing Yu et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[77] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

A Exact $\mathrm{ML\text{-}Net}$ architecture and training details

In our experiments, we construct $\mathrm{ML\text{-}Net}$ and $\mathrm{UNetSeq}$ with $L = 7$ levels. Convolutional layers use 3×3 kernels with zero padding. Upsampling ($\uparrow$) and downsampling ($\downarrow$) layers are implemented by 5×5 transpose-strided and strided convolutions with 2-strides, respectively. We do not employ any normalization layers and use the ReLU activation function throughout all architectures. Apart from up- and downsampling layers, UNet subunits of both $\mathrm{ML\text{-}Net}$ and $\mathrm{UNetSeq}$ contain two convolutional layers on each scale. Skip-connections are realized by concatenating the output of preceding layers.

For $\mathrm{ML\text{-}Net}$, we choose the number of UNets per level $\mathbf{R}_\ell$, $\ell = 1, \dots, L$, such that each subnetwork $\mathrm{ML\text{-}Net}_\ell$ has approximately 8×10^5 trainable parameters. This number was empirically found to produce satisfactory results and yields $\mathbf{R} = [11, 9, 5, 4, 3, 2, 2]$. To downsample the diffusivity field $\boldsymbol{\kappa}$ to the input resolution required by the subnetwork $\mathrm{ML\text{-}Net}_\ell$, $L - \ell + 1$ strided convolutions followed by ReLU activations are used. We use 32 channels for convolutional layers of $\mathrm{ML\text{-}Net}$, except for the coarsest level where 64 channels are used. For all levels $\ell = 2, \dots, L$ except the coarsest one, the $\mathrm{ML\text{-}Net}_\ell$ subnetworks output the predictions as four channels at half the resolution and use pixel un-shuffle layers [69] to assemble the full predictions. The reason for this is that fine grid corrections for a dyadic subdivision yield a higher similarity between every second value than neighbouring values in the array. For $\mathrm{UNetSeq}$, 64 channels are used for all convolutional layers.

$\mathrm{ML\text{-}Net}$ and $\mathrm{UNetSeq}$ are trained for 200 epochs with an initial learning rate of 10^{-3} during the first 60 epochs. The learning rate is then linearly decayed to 2×10^{-5} over the next 100 epochs, where it was held for the rest of the training. Due to memory constraints, the batch sizes were chosen to be 20 for $\mathrm{ML\text{-}Net}$ and 16 for $\mathrm{UNetSeq}$. The Adam optimizer [45] was used in the standard configuration with parameters $\beta_1 = 0.99, \beta_2 = 0.999$, and without weight decay.

B Additional tables

The following tables depict relative L^2 errors of the considered NN architectures for all presented test cases.

problem case	parameter dimension p	$\mathcal{E}_{\mathrm{MR}L^2}$	
		ML-Net	UNetSeq
uniform	10	$9.28\mathrm{e}{-5} \pm 6.92\mathrm{e}{-5}$	$3.72\mathrm{e}{-5} \pm 9.18\mathrm{e}{-6}$
	50	$4.88\mathrm{e}{-5} \pm 6.82\mathrm{e}{-6}$	$1.21\mathrm{e}{-3} \pm 1.66\mathrm{e}{-3}$
	100	$4.90\mathrm{e}{-5} \pm 9.56\mathrm{e}{-6}$	$3.21\mathrm{e}{-5} \pm 3.06\mathrm{e}{-6}$
	200	$4.81\mathrm{e}{-5} \pm 7.38\mathrm{e}{-6}$	$1.17\mathrm{e}{-3} \pm 1.61\mathrm{e}{-3}$
log-normal	10	$7.44\mathrm{e}{-5} \pm 1.75\mathrm{e}{-5}$	$1.66\mathrm{e}{-3} \pm 2.22\mathrm{e}{-3}$
	50	$7.46\mathrm{e}{-4} \pm 9.61\mathrm{e}{-4}$	$2.30\mathrm{e}{-3} \pm 3.13\mathrm{e}{-3}$
	100	$6.54\mathrm{e}{-5} \pm 6.71\mathrm{e}{-6}$	$1.72\mathrm{e}{-3} \pm 2.80\mathrm{e}{-3}$
	200	$7.35\mathrm{e}{-5} \pm 2.82\mathrm{e}{-6}$	$7.94\mathrm{e}{-5} \pm 2.14\mathrm{e}{-5}$
cookie fixed	16	$3.29\mathrm{e}{-4} \pm 3.26\mathrm{e}{-5}$	$2.35\mathrm{e}{-4} \pm 1.37\mathrm{e}{-5}$
	64	$5.32\mathrm{e}{-4} \pm 1.80\mathrm{e}{-5}$	$1.40\mathrm{e}{-4} \pm 4.32\mathrm{e}{-5}$
cookie variable	32	$1.33\mathrm{e}{-3} \pm 1.05\mathrm{e}{-4}$	$7.68\mathrm{e}{-4} \pm 2.45\mathrm{e}{-5}$
	128	$2.01\mathrm{e}{-3} \pm 8.14\mathrm{e}{-5}$	$7.14\mathrm{e}{-4} \pm 7.74\mathrm{e}{-5}$

Table 4: $\mathcal{E}_{\mathrm{MR}L^2}$ for ML-Net and UNetSeq evaluated on all test cases.

problem case	parameter dimension p	$\mathcal{E}^{\mathrm{ref}}_{\mathrm{MR}L^2}$	
		ML-Net	UNetSeq
uniform	10	$1.05\mathrm{e}{-4} \pm 6.75\mathrm{e}{-5}$	$4.77\mathrm{e}{-5} \pm 6.80\mathrm{e}{-6}$
	50	$5.86\mathrm{e}{-5} \pm 7.02\mathrm{e}{-6}$	$1.22\mathrm{e}{-3} \pm 1.66\mathrm{e}{-3}$
	100	$5.50\mathrm{e}{-5} \pm 5.07\mathrm{e}{-6}$	$4.19\mathrm{e}{-5} \pm 3.41\mathrm{e}{-6}$
	200	$5.67\mathrm{e}{-5} \pm 5.78\mathrm{e}{-6}$	$1.18\mathrm{e}{-3} \pm 1.60\mathrm{e}{-3}$
log-normal	10	$7.76\mathrm{e}{-5} \pm 1.29\mathrm{e}{-5}$	$1.66\mathrm{e}{-3} \pm 2.22\mathrm{e}{-3}$
	50	$7.59\mathrm{e}{-4} \pm 9.68\mathrm{e}{-4}$	$2.30\mathrm{e}{-3} \pm 3.13\mathrm{e}{-3}$
	100	$7.06\mathrm{e}{-5} \pm 4.38\mathrm{e}{-6}$	$1.72\mathrm{e}{-3} \pm 2.80\mathrm{e}{-3}$
	200	$8.31\mathrm{e}{-5} \pm 3.91\mathrm{e}{-6}$	$8.69\mathrm{e}{-5} \pm 1.97\mathrm{e}{-5}$
cookie fixed	16	$6.19\mathrm{e}{-3} \pm 6.60\mathrm{e}{-5}$	$6.05\mathrm{e}{-3} \pm 3.50\mathrm{e}{-5}$
	64	$9.63\mathrm{e}{-3} \pm 4.33\mathrm{e}{-5}$	$9.48\mathrm{e}{-3} \pm 2.79\mathrm{e}{-5}$
cookie variable	32	$8.81\mathrm{e}{-3} \pm 3.41\mathrm{e}{-5}$	$8.49\mathrm{e}{-3} \pm 9.10\mathrm{e}{-5}$
	128	$1.64\mathrm{e}{-2} \pm 5.36\mathrm{e}{-5}$	$1.62\mathrm{e}{-2} \pm 6.70\mathrm{e}{-5}$

Table 5: $\mathcal{E}^{\mathrm{ref}}_{\mathrm{MR}L^2}$ for ML-Net and UNetSeq evaluated on all test cases.

C Proofs for the results of Section 5

This section is devoted to the proofs of our theoretical analysis in Section 5. For this, we start with some mathematical notation.

C.1 Mathematical notation

For some function $f\colon D \subseteq \mathbb{R}^d \to \mathbb{R}^n$, we denote $\|f\|_{L^\infty(D)} := \operatorname{ess\,sup}_{x\in D} \|f(x)\|_\infty$, where $\|\cdot\|_\infty$ denotes the maximum norm of a vector. For two tensors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{W\times H}$ with $W, H \in \mathbb{N}$,

we denote their pointwise multiplication by $\mathbf{x} \odot \mathbf{y} \in \mathbb{R}^{W \times H}$. Since we consider the FE spaces V_h in the two-dimensional setting on uniformly refined square meshes, FE coefficient vectors $\mathbf{x} \in \mathbb{R}^{\dim V_h}$ can be viewed as two-dimensional arrays. Whenever x is processed by a CNN, we implicitly assume a 2D matrix representation.

The bilinear form in (2.1) induces the problem related and parameter dependent *energy norm* given by

$$\|w\|_{A_{\mathbf{y}}}^2 := \int_D \kappa(\mathbf{y}, x) |\nabla w(x)|^2 \, \mathrm{d}x \quad \text{for} \quad w \in V. \tag{C.1}$$

Consider Problem 2.1. Under the uniform ellipticity assumption[1], the energy norm is equivalent to the H^1 norm (see e.g. [16]). We denote by $c_{H^1}, C_{H^1} > 0$ the grid independent constants such that for all $u \in V$ and $\mathbf{y} \in \Gamma$

$$c_{H^1} \|u\|_{A_{\mathbf{y}}} \leq \|u\|_{H^1} \leq C_{H^1} \|u\|_{A_{\mathbf{y}}} . \tag{C.2}$$

Moreover, for a multilevel decomposition up to level $L \in \mathbb{N}$ and $\hat{v}_1, \dots, \hat{v}_L$ defined as in (3.1), denote by $C_{\mathsf{corr}} > 0$ the constant such that for all $\mathbf{y} \in \Gamma$ and $\ell = 1, \dots, L$

$$\|\hat{v}_\ell(\mathbf{y})\|_{H^1} \leq C_{\mathsf{corr}} 2^{-\ell} \|f\|_* . \tag{C.3}$$

Note that for any $\mathbf{y} \in \Gamma$, the solution $v_h(\mathbf{y})$ of Problem 2.1 satisfies

$$\|v_h(\mathbf{y})\|_{A_{\mathbf{y}}} \leq C_{H^1} \|f\|_* . \tag{C.4}$$

To limit excessive mathematical overhead in our proofs, we restrict ourselves to a certain type of uniform square grid specified in the following remark.

Remark C.1. *For $D = [0,1]^2$, mesh width $h > 0$ and $m := 1/h \in \mathbb{N}$, we exclusively consider the uniform square grid comprised of m^2 identical squares each subdivided into two triangles as illustrated in Figure 4.*

C.2 CNNs: terminology and basic properties

In this section, we briefly introduce basic convolutional operations together with their corresponding notation. We mostly focus on the two-dimensional case. However, all definitions and results can easily be extended to arbitrary dimensions. For a more in-depth treatment of the underlying concepts, we refer to [29, Chapter 9].

Convolutions used in CNNs deviate in some details from the established mathematical definition of a convolution. In the two-dimensional case, the input consists of a tensor $\mathbf{x}_{\mathsf{in}} \in \mathbb{R}^{C_{\mathsf{in}} \times W_{\mathsf{in}} \times H_{\mathsf{in}}}$ with spatial dimensions $W_{\mathsf{in}}, H_{\mathsf{in}} \in \mathbb{N}$ and number of channels $C_{\mathsf{in}} \in \mathbb{N}$, and the output is given by a tensor $\mathbf{x}_{\mathsf{out}} \in \mathbb{R}^{C_{\mathsf{out}} \times W_{\mathsf{out}} \times H_{\mathsf{out}}}$. Here, $C_{\mathsf{out}} \in \mathbb{N}$ denotes the number of output channels and $W_{\mathsf{out}}, H_{\mathsf{out}} \in \mathbb{N}$ the potentially transformed spatial dimensions. $\mathbf{x}_{\mathsf{out}}$ is the result of convolving $\mathbf{x}_{\mathsf{in}}$ with a learnable *convolutional kernel* $K \in \mathbb{R}^{C_{\mathsf{in}} \times C_{\mathsf{out}} \times W_K \times W_K}$ with kernel width $W_K \in \mathbb{N}$, and the channel-wise addition of a learnable *bias* $B \in \mathbb{R}^{C_{\mathsf{out}}}$. We use three different types of convolutions: (i) *vanilla*, denoted by $\mathbf{x}_{\mathsf{in}} * K$, here $W_{\mathsf{out}} = W_{\mathsf{in}}$ and $H_{\mathsf{out}} = H_{\mathsf{in}}$; (ii) *two-strided* denoted by $\mathbf{x}_{\mathsf{in}} *^{\mathsf{2s}} K$, here $W_{\mathsf{out}} = \lfloor W_{\mathsf{in}}/2 \rfloor - 1$ and $H_{\mathsf{out}} = \lfloor H_{\mathsf{in}}/2 \rfloor - 1$; (iii) *two-transpose-strided*, denoted by $\mathbf{x}_{\mathsf{in}} *^{\mathsf{2ts}} K$, here $W_{\mathsf{out}} := 2W_{\mathsf{in}} + 1$ and $H_{\mathsf{out}} := 2H_{\mathsf{in}} + 1$. Moreover, we

[1] which always is satisfied with high probability in our settings

write $M(\Phi) \in \mathbb{N}$ for the number of trainable parameters and $L(\Phi) \in \mathbb{N}$ for the number of layers of a (possibly convolutional) NN Φ.

Many previous works have focused on approximating functions by fully connected NNs. With a simple trick, we transfer these approximation results to CNNs.

Theorem C.1. *Let $D \subseteq \mathbb{R}^d$ and $f : D \to \mathbb{R}$ be some function. Furthermore, let $\varepsilon > 0$ and Φ be a fully connected NN with d-dimensional input and one-dimensional output such that $\|f - \Phi\|_{L^\infty(D)} \leq \varepsilon$, then there exists a CNN Ψ with*

(i) d input channels and one output channel;

(ii) the spatial dimension of all convolutional kernels is 1×1;

(iii) the same activation function as Φ;

(iv) $M(\Phi) = M(\Psi)$ and $L(\Phi) = L(\Psi)$;

(v) for all $W \in \mathbb{N}$, we have

$$\left\|\Psi - \hat{f}\right\|_{L^\infty(D^{W\times W})} \leq \varepsilon,$$

where $\hat{f} : D^{W\times W} \to \mathbb{R}^{W\times W}$ is the component-wise application of f.

Proof. The proof follows directly by using the affine linear transformations in each layer of Φ in the channel dimension as 1×1 convolutions. □

In the next corollary, Theorem C.1 is used to approximate the pointwise multiplication function of two input tensors by a CNN.

Corollary C.1. *Let $\varrho \in L^\infty_{\mathrm{loc}}(\mathbb{R})$ such that there exists $x_0 \in \mathbb{R}$ with ϱ is three times continuously differentiable in a neighborhood of some $x_0 \in \mathbb{R}$ and $\varrho''(x_0) \neq 0$. Let $W \in \mathbb{N}$, $B > 0$, and $\varepsilon \in (0, 1/2)$, then there exists a CNN $\widetilde{\times}$ with activation function ϱ, a two-channel input and one-channel output that satisfies the following properties:*

1 $\left\|\widetilde{\times}(\mathbf{x}, \mathbf{y}) - \mathbf{x} \odot \mathbf{y}\right\|_{L^\infty([-B,B]^{2\times W\times W}, d\mathbf{x}d\mathbf{y})} \leq \varepsilon$;

2 $L(\widetilde{\times}) = 2$ *and* $M(\widetilde{\times}) \leq 9$;

3 the spatial dimension of all convolutional kernels is 1×1.

Proof. The proof follows from Theorem C.1 together with [33, Corollary C.3]. □

C.3 Approximating isolated V-cycle building blocks

One of the main intermediate steps to approximate the full multigrid cycle by CNNs is the approximation of the operator A_κ. The theoretical backbone of this section is the observation that $A_\kappa \mathbf{u}$ can be approximated by a CNN acting on κ (in an integral representation defined in Definition C.1) and $\mathbf{u}$. We start by defining some basic concepts used for our proofs. For the rest of this section, we set $D = [0, 1]^2$.

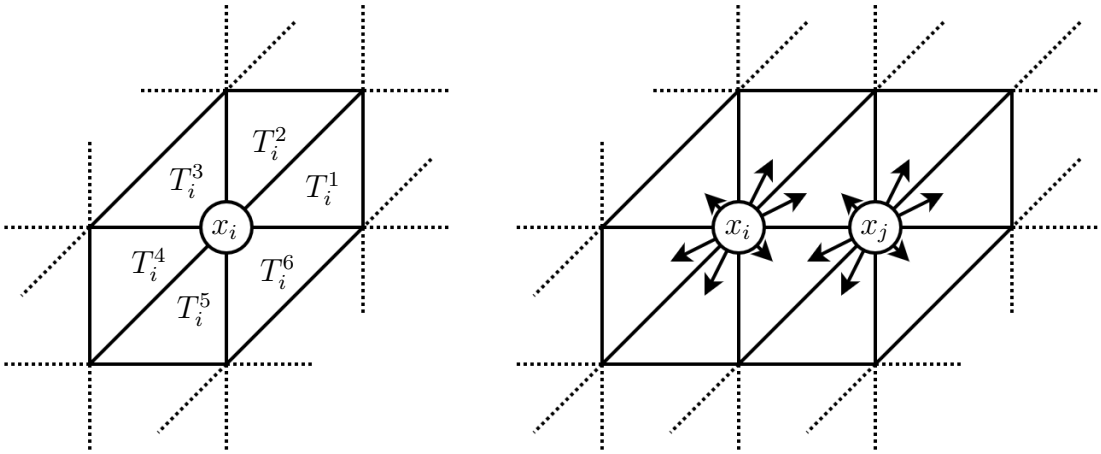

Figure 4: Illustration showing a possible enumeration of the six adjacent triangles of a grid point on the uniform square mesh from Definition C.1 (ii). Note that this assignment is redundant, as on the right, one can see that $T_i^{(6)} = T_j^{(4)}$ and $T_i^{(1)} = T_j^{(3)}$.

Definition C.1. *Let $\mathcal{T}$ be a uniform triangulation of D with corresponding conforming P1 FE space V_h with basis $\{\varphi_1, \dots, \varphi_{\dim V_h}\}$. Furthermore, let $\kappa \in H_0^1(D)$.*

(i) For $i \in \{1, \dots, \dim V_h\}$ we set $\mathrm{Neigh2D}(i) \coloneqq \{j \in \{1, \dots, \dim V_h\} : \operatorname{supp} \varphi_i \cap \operatorname{supp} \varphi_j \neq \emptyset\}$.

(ii) For each vertex $i \in \{1, \dots, \dim V_h\}$ of the triangulation, we enumerate the six adjacent triangles (arbitrarily but in the same order for every i) and denote them by $T_i^{(k)}$ with $k = 1, \dots, 6$ (see Figure 4 for an illustration).

(iii) For $i = 1, \dots, \dim V_h$ and $k = 1, \dots, 6$, we set

$$\boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k, i) \coloneqq \int_{T_i^k} \kappa \, \mathrm{d}x$$

and use the notation

$$\boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k) \coloneqq [\boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k, i)]_{i=1}^{\dim V_h} \in \mathbb{R}^{\dim V_h}$$

and

$$\boldsymbol{\Upsilon}(\kappa, \mathcal{T}) \coloneqq [\boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k)]_{k=1}^{6} \in \mathbb{R}^{6 \times \dim V_h}.$$

In the next lemma, we show that the integrals $\boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k, i)$ can be computed via a convolution from (a discretized) κ and, furthermore, that integrals over a coarse grid can be computed from fine-grid-integrals, again via a convolution.

Lemma C.1. *Let $\mathcal{T}_h, \mathcal{T}_{2h}$ be nested triangulations of fineness h and $2h$, respectively, with corresponding FE spaces $V_{2h} \subseteq V_h \subseteq H_0^1(D)$. Then, the following holds:*

(i) There exists a convolutional kernel $K \in \mathbb{R}^{1 \times 6 \times 3 \times 3}$ such that for every $\kappa \in V_h$, we have for $k \in \{1, \dots, 6\}$

$$(\boldsymbol{\kappa} * K)[k] = \boldsymbol{\Upsilon}(\kappa, \mathcal{T}_h, k).$$

(ii) There exists a convolutional kernel $K \in \mathbb{R}^{6\times6\times3\times3}$ *such that for every* $\kappa \in H_0^1(D)$ *and* $k = 1, \dots, 6$*, we have*

$$([\boldsymbol{\Upsilon}(\kappa, \mathcal{T}_h, 1), \dots, \boldsymbol{\Upsilon}(\kappa, \mathcal{T}_h, 6)] * K)[k] = \boldsymbol{\Upsilon}(\kappa, \mathcal{T}_{2h}, k).$$

Proof. We start by showing (i). For $k = 1, \dots 6$ and $i = 1, \dots, \dim V_h$,

$$\boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k, i) = \int_{T_i^k} \kappa \,\mathrm{d}x = \sum_{j=1}^{\dim V_h} \boldsymbol{\kappa}_j \int_{T_i^k} \varphi_j^{(h)} \,\mathrm{d}x = \sum_{\{j:\operatorname{supp}\varphi_j^{(h)} \cap T_i^k \neq \emptyset\}} \boldsymbol{\kappa}_j \frac{h^2}{3},$$

where we used in the last step that $\int_{T_i^k} \kappa \,\mathrm{d}x = {}^{h^2}/_3$ if $\operatorname{supp}\varphi_j^{(h)} \cap T_i^k \neq \emptyset$. Clearly, $\operatorname{supp}\varphi_j^{(h)} \cap T_i^k = \emptyset$ for all $j \in \{1, \dots, \dim V_h\}$ with $j \notin \mathrm{Neigh2D}(i)$. Furthermore, the position of the relevant neighbors $\{j : \operatorname{supp}\varphi_j^{(h)} \cap T_i^k \neq \emptyset\}$ relative to index i is invariant to 2D-translations of i. Combining these two observations concludes the proof.

(ii) Intuitively it is clear that a strided convolution is able to locally compute the mean of all coefficients from the finer triangulation. However, to prove this rigorously would be a tedious exercise. We hence omit the formal proof and refer to Figure 5 for an illustration of the neighboring sub-triangles in each grid point. □

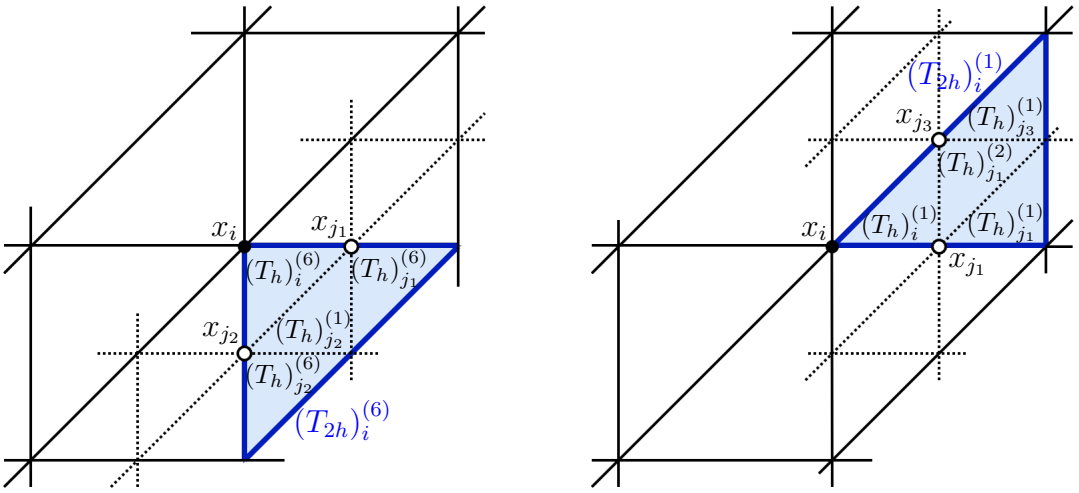

Figure 5: Illustration of the neighboring triangles and sub-triangles of a finer mesh at some grid point. This figure shows how an adjacent triangle (blue) to the grid point x_i in the coarse grid is subdivided. Note that all contained finer triangles can be assigned to a grid point directly adjacent to x_i in the fine mesh. Hence, summing over the diffusivity values associated with the finer triangles can be represented using a convolutional (3×3)-kernel in the fine mesh.

The next theorem shows that using the integral representation of κ from Definition C.1, we can now represent $A_\kappa \mathbf{u}$ by the pointwise multiplication of the κ integrals with the output of a convolution applied to $\mathbf{u}$.

Theorem C.2 (Representation of $A_\kappa \mathbf{u}$)**.** *For* $k = 1, \dots, 6$*, there exist convolutional kernels* $K^{(k)} \in \mathbb{R}^{1\times1\times3\times3}$*, such that for the function*

$$F : \mathbb{R}^{7\times\dim V} \to \mathbb{R}^{\dim V}, \quad (\mathbf{u}, \overline{\boldsymbol{\kappa}}^{(1)}, \dots, \overline{\boldsymbol{\kappa}}^{(6)}) \mapsto \sum_{k=1}^{6} \overline{\boldsymbol{\kappa}}^{(k)} \odot (\mathbf{u} * K^{(k)}),$$

it holds that F is continuous and for any $\kappa \in H_0^1(D)$, we have

$$F(\mathbf{u}, \boldsymbol{\Upsilon}(\kappa, \mathcal{T}, 1), \dots, \boldsymbol{\Upsilon}(\kappa, \mathcal{T}, 6)) = A_\kappa \mathbf{u}.$$

Proof. We start by introducing some notation: Let $i, j \in \{1, \dots, \dim V_h\}$ and $k \in \{1, \dots, 6\}$ and set $C_{ijk} \in \mathbb{R}$ as the constant that $\langle \nabla\varphi_i, \nabla\varphi_j \rangle$ attains on T_i^k. Note that $C_{ijk} = 0$ if $i \notin \operatorname{Neigh2D}(j)$. We now represent each entry of A_κ as a sum of multiplications: For $i, j \in \{1, \dots, \dim V_h\}$, we have

$$(A_\kappa)_{ij} = \int_D \kappa \langle \nabla\varphi_i, \nabla\varphi_j \rangle \, \mathrm{d}x = \sum_{k=1}^{6} \int_{T_i^k} \kappa \langle \nabla\varphi_i, \nabla\varphi_j \rangle \, \mathrm{d}x = \sum_{k=1}^{6} \Upsilon(\kappa, \mathcal{T}, k, i) C_{ijk},$$

where we use Definition C.1 (ii) for the last step. For $j \in \{1, \dots, \dim V_h\}$, we can now rewrite $(A_\kappa \mathbf{u})_j$ as

$$\begin{aligned}
(A_\kappa \mathbf{u})_j &= \sum_{i=1}^{\dim V_h} \mathbf{u}_i \sum_{k=1}^{6} \boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k, j) C_{ijk} \\
&= \sum_{k=1}^{6} \boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k, j) \sum_{i=1}^{\dim V_h} \mathbf{u}_i C_{ijk} \\
&= \sum_{k=1}^{6} \boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k, j) \sum_{i \in \operatorname{Neigh2D}(j)} \mathbf{u}_i C_{ijk}.
\end{aligned}$$

Here, we use in the first step the definition of A_κ and in the third step that $\operatorname{supp}\varphi_i \cap \operatorname{supp}\varphi_j \neq \emptyset$ only for neighboring indices. Since C_{ijk} only depends on the relative position of i to j (in the 2D sense) and is independent on j, the inner sum can be expressed in the vectorized form as a zero-padded convolution with a 3×3 kernel (the number of neighbors) where the values depend on k, i.e.,

$$A_\kappa \mathbf{u} = \sum_{k=1}^{6} \boldsymbol{\Upsilon}(\kappa, \mathcal{T}, k) \odot (\mathbf{u} * K^{(k)}) = F(\mathbf{u}, \boldsymbol{\Upsilon}(\kappa, \mathcal{T}, 1), \dots, \boldsymbol{\Upsilon}(\kappa, \mathcal{T}, 6)),$$

where $K^{(k)}$ are the respective kernels. □

In the following remark, we elaborate on the treatment of the boundary conditions.

Remark C.2. *In our setting, we only consider basis functions on inner grid points. Alternatively, one could include the boundary basis functions and constrain their coefficients to zero to enforce homogeneous Dirichlet boundary conditions. Representing the FE operator as a zero-padded convolution on the inner grid vertices naturally realizes homogeneous Dirichlet boundary conditions by implicitly including boundary basis coefficients as zeros through padding.*

Using the representation of $A_\kappa \mathbf{u}$ from Theorem C.2 as a convolution followed by a pointwise multiplication, Corollary C.1 can now be applied to obtain an approximation by a CNN.

Theorem C.3 (Approximation of $A_\kappa \mathbf{u}$)**.** *Let $\mathcal{T}$ be a uniform triangulation of D with corresponding conforming P1 FE space V. Furthermore, let the activation function $\varrho \in L^\infty_{\mathrm{loc}}(\mathbb{R})$ be such that there exists $x_0 \in \mathbb{R}$ with ϱ is three times continuously differentiable in a neighborhood of some $x_0 \in \mathbb{R}$ and $\varrho''(x_0) \neq 0$. Then, for any $\varepsilon > 0$ and $M > 0$, there exists a CNN $\Psi_{\varepsilon,M} : \mathbb{R}^{7 \times \dim V_h} \to \mathbb{R}^{\dim V_h}$ with size independent of ε and M such that*

$$\|\Psi_{\varepsilon,M} - F\|_{L^\infty([-M,M]^{7 \times \dim V_h})} \leq \varepsilon.$$

Proof. A three-step procedure leads to the final result:

(i) Clearly, there exists a one-layer CNN that realizes the mapping

$$[\mathbf{u}, \overline{\boldsymbol{\kappa}}^{(1)}, \dots, \overline{\boldsymbol{\kappa}}^{(6)}] \mapsto [\mathbf{u} * K^{(k)}, \overline{\boldsymbol{\kappa}}^{(k)}]_{k=1}^{6}.$$

(ii) Corollary C.1 yields the existence of a two-layer CNN $\tilde{\Psi}$ with at most 9 weights and 1×1 kernels such that

$$\left\| \tilde{\Psi}([\mathbf{u} * K^{(k)}, \overline{\boldsymbol{\kappa}}^{(k)}]_{k=1}^{6}) - [\overline{\boldsymbol{\kappa}}^{(k)} \odot (\mathbf{u} * K^{(k)})]_{k=1}^{6} \right\|_{\infty} \leq \frac{\varepsilon}{6}.$$

(iii) The channel-wise addition can trivially be constructed with a one-layer CNN with a 1×1 kernel.

A concatenation of the above three CNNs concludes the proof. □

Recall that the prolongation and its counterpart, the weighted restriction, are essential operations in the multigrid cycle. The following remark states that these operations can naturally be expressed as convolutions.

Remark C.3 (Weighted restriction and prolongation)**.** *Consider a two-level decomposition using nested conforming P1 FE spaces $V_1 \subseteq V_2 \subseteq H_0^1(D)$ on uniform square meshes. Let P be the corresponding prolongation matrix (Definition 3.1). Then, there exist convolutional kernels $K_1, K_2 \in \mathbb{R}^{1\times1\times3\times3}$ such that for any FE coefficient vector*

(i) $\mathbf{u} \in \mathbb{R}^{\dim V_2}$*, we have* $\mathbf{u} *^{2s} K_1 = P^T \mathbf{u}$*;*

(ii) $\mathbf{u} \in \mathbb{R}^{\dim V_1}$*, we have* $\mathbf{u} *^{2ts} K_2 = P\mathbf{u}$*.*

C.4 Putting it all together

In this section, we combine the approximation of individual computation steps of the multigrid algorithm by CNNs to approximate an arbitrary number of multigrid cycles by a CNN. For the composition of the individual approximations, we first provide an auxiliary lemma.

Lemma C.2. *Let $n \in \mathbb{N}$ and $d_1, \dots, d_{n+1} \in \mathbb{N}$. For $i = 1, \dots, n$ let $f_i \colon \mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$ be continuous functions and define F as their concatenation, i.e.,*

$$F \colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_{n+1}}, \quad F \coloneqq f_n \circ \dots \circ f_1.$$

Then, for every $M, \varepsilon > 0$, there exist $M_1, \dots, M_n > 0$ and $\varepsilon_1, \dots, \varepsilon_n > 0$, such that the following holds true: If $\tilde{f}_i \colon \mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$ are functions such that $\|f_i - \tilde{f}_i\|_{L^\infty([-M_i, M_i]^{d_i})} \leq \varepsilon_i$ for $i = 1, \dots, n$, then

$$\left\| F - \tilde{f}_n \circ \dots \circ \tilde{f}_1 \right\|_{L^\infty([-M,M]^{d_1})} \leq \varepsilon.$$

Proof. We prove the statement by induction over $n \in \mathbb{N}$. For $n = 1$, the statement is clear. Define the function $g\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_n}$ for $n > 1$ by

$$g := f_{n-1} \circ \ldots \circ f_1.$$

We set $\varepsilon_n := \varepsilon/2$ and $M_n := \|g\|_{L^\infty([-M,M]^{d_1})} + \varepsilon < \infty$ (since g is continuous). As f_n is uniformly continuous on compact sets, there exists $0 < \delta \leq \varepsilon$ such that for all $z, \tilde{z} \in [-M_n, M_n]^{d_n}$ with $\|z - \tilde{z}\|_\infty \leq \delta$ we have $\|f_n(z) - f_n(\tilde{z})\|_\infty \leq \frac{\varepsilon}{2}$.

Let now $M_1, \ldots, M_{n-1}$ and $\varepsilon_1, \ldots, \varepsilon_{n-1}$ be given by the induction assumption (applied with $M = M$ and $\varepsilon = \delta$) and let $\tilde{f}_i\colon \mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$ be corresponding approximations for $i = 1, \ldots, n-1$. Then (again by the induction assumption), it holds that

$$\left\|g - \tilde{f}_{n-1} \circ \ldots \circ \tilde{f}_1\right\|_{L^\infty([-M,M]^{d_1})} \leq \delta.$$

Setting $\tilde{g} := \tilde{f}_{n-1} \circ \ldots \circ \tilde{f}_1$, we get $\|\tilde{g}\|_{L^\infty([-M,M]^{d_1})} \leq \|g\|_{L^\infty([-M,M]^{d_1})} + \delta \leq M_n$. The proof is concluded by deriving that

$$\begin{aligned}
&\left\|F - (\tilde{f}_n \circ \ldots \circ \tilde{f}_1)\right\|_{L^\infty([-M,M]^{d_1})} \\
&\leq \|(f_n \circ g) - (f_n \circ \tilde{g})\|_{L^\infty([-M,M]^{d_1})} + \left\|(f_n \circ \tilde{g}) - (\tilde{f}_n \circ \tilde{g})\right\|_{L^\infty([-M,M]^{d_1})} \\
&\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.
\end{aligned}$$

□

With the preceding preparations, we are now ready to prove our main results. We start with the approximation of the multigrid algorithm by a CNN.

Proof of Theorem 5.1 The overall proof strategy consists of decomposing the function $\mathrm{MG}^m_{k_0,k} : \mathbb{R}^{3\times \dim V_L} \to \mathbb{R}^{\dim V_L}$, where $V_L = V_h$, into a concatenation of a finite number of continuous functions that can either be implemented by a CNN or arbitrarily well approximated by a CNN. An application of Lemma C.2 then yields that the concatenation of these CNNs (which is again a CNN) approximates $\mathrm{MG}^m_{k_0,k}$. We make the following definitions:

(i) **Integrating the diffusion coefficient.** Let $K \in \mathbb{R}^{1\times 6\times 3\times 3}$ be the convolutional kernel from Lemma C.1(i). For the function

$$f_{\mathsf{in}}\colon \mathbb{R}^{3\times \dim V_L} \to \mathbb{R}^{8\times \dim V_L}, \quad \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\kappa} \\ \mathbf{f} \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\kappa} * K \\ \mathbf{f} \end{pmatrix},$$

it then follows from Lemma C.1(i) that $f_{\mathsf{in}}([\mathbf{u}, \boldsymbol{\kappa}, \mathbf{f}]) = [\mathbf{u}, \boldsymbol{\Upsilon}(\kappa_h, \mathcal{T}^L), \mathbf{f}]$.

(ii) **Smoothing iterations.** For $\ell = 1, \ldots, L$ let $F_\ell : \mathbb{R}^{7\times \dim V_\ell} \to \mathbb{R}^{\dim V_\ell}$ be defined as in Theorem C.2. Then we define

$$f^\ell_{\mathsf{sm}}\colon \mathbb{R}^{8\times \dim V_\ell} \to \mathbb{R}^{8\times \dim V_\ell}, \quad \begin{pmatrix} \mathbf{u} \\ \overline{\boldsymbol{\kappa}}^{(1)} \\ \vdots \\ \overline{\boldsymbol{\kappa}}^{(6)} \\ \mathbf{f} \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{u} + \omega(\mathbf{f} - F_\ell(\mathbf{u}, \overline{\boldsymbol{\kappa}}^{(1)}, \ldots, \overline{\boldsymbol{\kappa}}^{(6)})) \\ \overline{\boldsymbol{\kappa}}^{(1)} \\ \vdots \\ \overline{\boldsymbol{\kappa}}^{(6)} \\ \mathbf{f} \end{pmatrix}.$$

It follows from Theorem C.2 that

$$f^\ell_{\mathsf{sm}}([\mathbf{u}, \boldsymbol{\Upsilon}(\kappa_h, \mathcal{T}^\ell), \mathbf{f}]) = [\mathbf{u} + \omega(\mathbf{f} - A^\ell_{\kappa_h}\mathbf{u}), \boldsymbol{\Upsilon}(\kappa_h, \mathcal{T}^\ell), \mathbf{f}].$$

(iii) **Restricted residual.** Similar as in the previous step, we define

$$f^\ell_{\mathsf{resi}}\colon \mathbb{R}^{8\times\dim V_\ell}\to\mathbb{R}^{9\times\dim V_\ell},\quad \begin{pmatrix}\mathbf{u}\\ \overline{\boldsymbol{\kappa}}^{(1)}\\ \vdots\\ \overline{\boldsymbol{\kappa}}^{(6)}\\ \mathbf{f}\end{pmatrix}\mapsto\begin{pmatrix}\mathbf{u}\\ \mathbf{f}-F_\ell(\mathbf{u},\overline{\boldsymbol{\kappa}}^{(1)},\dots,\overline{\boldsymbol{\kappa}}^{(6)})\\ \overline{\boldsymbol{\kappa}}^{(1)}\\ \vdots\\ \overline{\boldsymbol{\kappa}}^{(6)}\\ \mathbf{f}\end{pmatrix},$$

and get from Theorem C.2 that

$$f^\ell_{\mathsf{resi}}([\mathbf{u},\boldsymbol{\Upsilon}(\kappa_h,\mathcal{T}^\ell),\mathbf{f}])=[\mathbf{u},\mathbf{f}-A^\ell_{\kappa_h}\mathbf{u},\boldsymbol{\Upsilon}(\kappa_h,\mathcal{T}^\ell),\mathbf{f}].$$

Let now $[K_1,\dots,K_6]\in\mathbb{R}^{6\times6\times3\times3}$ be the convolutional kernel from Lemma C.1(ii) and $P_{\ell-1}\in\mathbb{R}^{\dim V_\ell\times\dim V_{\ell-1}}$ the prolongation matrix from Definition 3.1 (where its transpose acts as the weighted restriction), then we define

$$f^\ell_{\mathsf{rest}}\colon \mathbb{R}^{9\times\dim V_\ell}\to\mathbb{R}^{8\times\dim V_\ell}\times\mathbb{R}^{8\times\dim V_{\ell-1}},\quad \begin{pmatrix}\mathbf{u}\\ \mathbf{r}\\ \overline{\boldsymbol{\kappa}}^{(1)}\\ \vdots\\ \overline{\boldsymbol{\kappa}}^{(6)}\\ \mathbf{f}\end{pmatrix}\mapsto\begin{bmatrix}\begin{pmatrix}\mathbf{u}\\ \overline{\boldsymbol{\kappa}}^{(1)}\\ \vdots\\ \overline{\boldsymbol{\kappa}}^{(6)}\\ \mathbf{f}\end{pmatrix}\\ \begin{pmatrix}\mathbf{0}\\ \overline{\boldsymbol{\kappa}}^{(1)}*K_1\\ \vdots\\ \overline{\boldsymbol{\kappa}}^{(6)}*K_6\\ P_{\ell-1}^T\mathbf{r}\end{pmatrix}\end{bmatrix}.$$

Note that $\boldsymbol{\Upsilon}(\kappa_h,\mathcal{T}^\ell,k)*K_k=\boldsymbol{\Upsilon}(\kappa_h,\mathcal{T}^{\ell-1},k)$ for $k=1,\dots,6$.

(iv) **Add coarse grid correction to fine grid.** As above, let $P_{\ell-1}\in\mathbb{R}^{\dim V_\ell\times\dim V_{\ell-1}}$ be the prolongation matrix from Definition 3.1. We define the function

$$f^\ell_{\mathsf{prol}}\colon \mathbb{R}^{8\times\dim V_\ell}\times\mathbb{R}^{8\times\dim V_{\ell-1}}\to\mathbb{R}^{8\times\dim V_\ell},\quad \begin{bmatrix}\begin{pmatrix}\mathbf{u}\\ \overline{\boldsymbol{\kappa}}^{(1)}\\ \vdots\\ \overline{\boldsymbol{\kappa}}^{(6)}\\ \mathbf{f}\end{pmatrix}\\ \begin{pmatrix}\mathbf{e}\\ \vdots\end{pmatrix}\end{bmatrix}\mapsto\begin{pmatrix}\mathbf{u}+P_{\ell-1}\mathbf{e}\\ \overline{\boldsymbol{\kappa}}^{(1)}\\ \vdots\\ \overline{\boldsymbol{\kappa}}^{(6)}\\ \mathbf{f}\end{pmatrix}.$$

(v) **Return solution.** We define

$$f_{\mathsf{out}}\colon \mathbb{R}^{8\cdot\dim V_\ell}\to\mathbb{R}^{\dim V_\ell},\quad \begin{pmatrix}\mathbf{u}\\ \overline{\boldsymbol{\kappa}}^{(1)}\\ \vdots\\ \overline{\boldsymbol{\kappa}}^{(6)}\\ \mathbf{f}\end{pmatrix}\mapsto\mathbf{u}.$$

We assemble a single V-cycle $\mathsf{VC}^{\ell}_{k_0,k}$ on level $\ell = 1, \dots, L$ as follows

$$\begin{aligned}\mathsf{VC}^{\ell}_{k_0,k} &:= \left(\bigcirc_{i=1}^{k} f^{\ell}_{sm}\right) \circ f^{\ell}_{\mathsf{prol}} \circ (\mathsf{Id}, \mathsf{VC}^{\ell-1}_{k_0,k}) \circ f^{\ell}_{\mathsf{rest}} \circ f^{\ell}_{\mathsf{resi}} \circ \left(\bigcirc_{i=1}^{k} f^{\ell}_{sm}\right), \\ \mathsf{VC}^{1}_{k_0,k} &:= \bigcirc_{i=1}^{k_0} f^{1}_{sm}.\end{aligned}$$

Finally, $\mathrm{MG}^{m}_{k_0,k}$ is given by:

$$\mathsf{MG}^{m}_{k_0,k} = f_{\mathsf{out}} \circ \left(\bigcirc_{i=1}^{m} \mathsf{VC}^{L}_{k_0,k}\right) \circ f_{\mathsf{in}}.$$

It is easy to see that the above defined functions are continuous and, thus, Lemma C.2 dictates the accuracy with which to approximate each part of the multigrid algorithm (f^{ℓ}_{sm}, f^{ℓ}_{prol}, f^{ℓ}_{rest}, f^{ℓ}_{resi}, f^{L}_{in}, f^{L}_{out}) on each level $\ell = 1, \dots, L$ to yield a final approximation accuracy of $\delta > 0$ for inputs bounded by $M > 0$.

On each level $\ell = 1, \dots, L$, we derive the approximations fulfilling these requirements for f^{ℓ}_{sm} and f^{ℓ}_{resi} using Theorem C.3, f^{ℓ}_{prol} and f^{ℓ}_{rest} as in Remark C.3, and f^{ℓ}_{rest} using Lemma C.1. The approximation of f^{L}_{in} is derived using Lemma C.1 and f^{L}_{out} is trivially represented using a single convolution. Concatenating the individual CNNs yields a CNN Ψ with an architecture resembling multiple concatenated UNets, such that

$$\left\| \Psi - \mathsf{MG}^{m}_{k_0,k} \right\|_{L^{\infty}([-M,M]^{3\times n})} \leq \delta.$$

The norm equivalence of $\|\cdot\|_{L^{\infty}([-M,M]^{3\times n})}$ and $\|\cdot\|_{H^1}$ in the finite dimensional setting directly implies inequality (i) with a suitable choice of δ. The parameter bound (ii) follows from the construction of Ψ. □

Proof of Corollary 5.1 Fix the damping parameter $\omega > 0$ and $k \in \mathbb{N}$ such that $\mu(k) < 1$ and let $m \in \mathbb{N}$ be chosen as small as possible such that

$$m \geq \log\left(\frac{1}{\mu(k)}\right)^{-1} \log\left(\frac{2C^2_{H^1}}{\varepsilon}\right).$$

Let $k_0 \in \mathbb{N}$ be given such that the contraction factor of k_0 damped Richardson iterations on the coarsest grid is smaller than $\mu(k)$. Let Ψ be the CNN from Theorem 5.1 setting $M = \max\left\{\sup_{\mathbf{y}\in\Gamma} \|\boldsymbol{\kappa}_{\mathbf{y}}\|_{\infty}, \|\mathbf{f}\|_{\infty}\right\}$ such that for all $\boldsymbol{\kappa}, \mathbf{f} \in [-M, M]^{n^2}$

$$\left\| \Psi(\mathbf{0}, \boldsymbol{\kappa}, \mathbf{f}) - \mathrm{MG}^{m}_{k_0,k}(\mathbf{0}, \boldsymbol{\kappa}, \mathbf{f}) \right\|_{H^1} \leq \frac{1}{2}\varepsilon \|f\|_{*}.$$

Using the contraction property of the multigrid algorithm from Theorem 3.1, we get

$$\begin{aligned}\left\| \mathrm{MG}^{m}_{k_0,k}(\mathbf{0}, \boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}) - \mathbf{v}_h(\mathbf{y}) \right\|_{A_{\mathbf{y}}} &\leq \mu(k)^m \|\mathbf{v}_h(\mathbf{y})\|_{A_{\mathbf{y}}} \\ &\leq \frac{1}{2}\varepsilon C^{-1}_{H^1} \|f\|_{*}.\end{aligned}$$

This yields

$$\begin{aligned}&\|\Psi(\mathbf{0}, \boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}) - \mathbf{v}_h(\mathbf{y})\|_{H^1} \\ &\leq \left\| \Psi(\mathbf{0}, \boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}) - \mathrm{MG}^{m}_{k_0,k}(\mathbf{0}, \boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}) \right\|_{H^1} + \left\| \mathrm{MG}^{m}_{k_0,k}(\mathbf{0}, \boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}) - \mathbf{v}_h(\mathbf{y}) \right\|_{H^1} \\ &\leq \frac{1}{2}\varepsilon \|f\|_{*} + C_{H^1} \left\| \mathrm{MG}^{m}_{k_0,k}(\mathbf{0}, \boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}) - \mathbf{v}_h(\mathbf{y}) \right\|_{A_{\mathbf{y}}} \\ &\leq \varepsilon \|f\|_{*}.\end{aligned}$$

Assuming $\varepsilon \leq 1$, there is a constant $C > 0$ such that $m \leq C \log\left(\frac{1}{\varepsilon}\right)$. Together with the parameter count from Theorem 5.1, this concludes the proof. □

Proof of Corollary 5.2 This proof is structured as follows. We consider a multilevel structure of V-Cycles to approximate the fine grid solution and derive error estimates for this approximation. Similarly to Corollary 5.1, we show how a CNN of sufficient size is able to represent this algorithm and translate the error estimates to the network approximation.

To this end, we introduce some constants and fix the number of V-Cycle iterations done on each grid level: Let $c := \max\{C_{H^1}, C_{\text{corr}}\}$. Fix $k \in \mathbb{N}$ such that $\mu(k) < 1$ and set $m_1, \dots, m_L \in \mathbb{N}$ such that

$$m_L \geq \log\left(\frac{1}{\mu(k)}\right)^{-1} \left(\log\left(\frac{2c^2 L}{\varepsilon}\right) - L\log(2)\right),$$
$$m_\ell \geq \log\left(\frac{1}{\mu(k)}\right)^{-1} \log(2), \text{ for } \ell = 1, \dots, L-1.$$

For any $\mathbf{y} \in \Gamma$, we denote by $A^\ell_{\mathbf{y}} \in \mathbb{R}^{\dim V_\ell \times \dim V_\ell}$ be the discretized operator (2.3) and $\mathbf{f}_\ell$ the right hand side from Problem 2.1 on level $\ell = 1, \dots, L$ such that $A^\ell_{\mathbf{y}} \mathbf{v}_\ell(\mathbf{y}) = \mathbf{f}_\ell$. Moreover, for ease of notation, we adapt the functions introduced in the proof of Theorem 5.1 and set

(i) for each level $\ell \in 1, \dots, L$, $\boldsymbol{\kappa}^\ell_{\mathbf{y}} \in \mathbb{R}^{6 \times \dim V_\ell}$ as the six channel tensor

$$\overline{\boldsymbol{\kappa}}^\ell_{\mathbf{y}} := \left[\Upsilon(\kappa_{\mathbf{y}}, \mathcal{T}^\ell, 1), \dots, \Upsilon(\kappa_{\mathbf{y}}, \mathcal{T}^\ell, 6)\right].$$

(ii) the function $\widetilde{\mathrm{MG}}^{m_\ell}_{k_0,k} \colon \mathbb{R}^{8 \times \dim V} \to \mathbb{R}^{\dim V}$ as

$$\widetilde{\mathrm{MG}}^{m_\ell}_{k_0,k} := f^\ell_{\text{out}} \circ \left(\bigcirc_{i=1}^{m_\ell} \mathrm{VC}^\ell_{k_0,k}\right),$$

such that $\widetilde{\mathrm{MG}}^{m_\ell}_{k_0,k} \circ f^\ell_{\text{in}} = \bigcirc_{i=1}^{m_\ell} \mathrm{MG}^{m_\ell}_{k_0,k}$.

We outline the following multilevel procedure for solving Problem 2.1 using multigrid with $\mathbf{y} \in \Gamma$ and $\boldsymbol{\kappa}_{\mathbf{y}} \in \mathbb{R}^{\dim V_L}$ given as input:

- $\boldsymbol{\kappa}_{\mathbf{y}}$ **subsampling.** Using $\boldsymbol{\kappa}_{\mathbf{y}}$, compute $\overline{\boldsymbol{\kappa}}^L_{\mathbf{y}}$ as outlined in Lemma C.1 (i). Subsequently, compute $\overline{\boldsymbol{\kappa}}^\ell_{\mathbf{y}}$ for each level $\ell = 2, \dots, L-1$ as shown in Lemma C.1 (ii).
- $\boldsymbol{\ell = 1}$. Solve for $\mathbf{v}_1(\mathbf{y})$ using m_1 iterations of the multigrid cycle. Let

$$\tilde{\mathbf{v}}_1(\mathbf{y}) := \widetilde{\mathrm{MG}}^{m_1}_{k_0,k}(0, \overline{\boldsymbol{\kappa}}^1_{\mathbf{y}}, \mathbf{f}).$$

- $\boldsymbol{\ell = 2, \dots, L}$. While $\hat{\mathbf{v}}_\ell(\mathbf{y})$ is the true correction with respect to the Galerkin approximation $\mathbf{v}_{\ell-1}(\mathbf{y})$, we introduce $\check{\mathbf{v}}_\ell(\mathbf{y})$ as the correction with respect to $\tilde{\mathbf{v}}_{\ell-1}(\mathbf{y})$. To this end, set

$$\check{\mathbf{v}}_\ell(\mathbf{y}) := \mathbf{v}_\ell(\mathbf{y}) - P_\ell \tilde{\mathbf{v}}_{\ell-1}(\mathbf{y}),$$
$$\tilde{\mathbf{f}}_\ell := \mathbf{f}_\ell - A^\ell_{\mathbf{y}} P_\ell \tilde{\mathbf{v}}_{\ell-1}(\mathbf{y}) = A^\ell_{\mathbf{y}} \check{\mathbf{v}}_\ell(\mathbf{y}).$$

Solve $A^\ell_{\mathbf{y}} \check{\mathbf{v}}_\ell(\mathbf{y}) = \tilde{\mathbf{f}}_\ell$ using m_ℓ iterations of the multigrid V-cycle and set

$$\tilde{\mathbf{v}}_\ell(\mathbf{y}) := \widetilde{\mathrm{MG}}^{m_\ell}_{k_0,k}(0, \overline{\boldsymbol{\kappa}}^\ell_{\mathbf{y}}, \tilde{\mathbf{f}}_\ell) + P_\ell \tilde{\mathbf{v}}_{\ell-1}(\mathbf{y}).$$

This procedure yields the following estimates using the contraction from Theorem 3.1.

- $\boldsymbol{\ell = 1}$. Using the contraction property from Theorem 3.1 and $m_1 \geq -\log(2)/\log(\mu(k))$ yields

$$\|\tilde{\mathbf{v}}_1(\mathbf{y}) - \mathbf{v}_1(\mathbf{y})\|_{A^1_{\mathbf{y}}} \leq \frac{1}{2} c \|f\|_* .$$

- $\boldsymbol{\ell = 2, \dots, L-1}$. Using again Theorem 3.1, the definitions for $m_\ell, \tilde{\mathbf{v}}_\ell, \check{\mathbf{v}}_\ell$ and $\hat{\mathbf{v}}_\ell = \mathbf{v}_\ell - P_\ell \mathbf{v}_{\ell-1}$, it holds

$$\begin{aligned}
\|\tilde{\mathbf{v}}_\ell(\mathbf{y}) - \mathbf{v}_\ell(\mathbf{y})\|_{A^\ell_{\mathbf{y}}} &= \left\| \widetilde{\mathrm{MG}}^{m_\ell}_{k_0,k}(0, \overline{\boldsymbol{\kappa}}^\ell_{\mathbf{y}}, \tilde{\mathbf{f}}_\ell) + P_\ell \tilde{\mathbf{v}}_{\ell-1}(\mathbf{y}) - \check{\mathbf{v}}_\ell(\mathbf{y}) - P_\ell \tilde{\mathbf{v}}_{\ell-1}(\mathbf{y}) \right\|_{A^\ell_{\mathbf{y}}} \\
&= \left\| \widetilde{\mathrm{MG}}^{m_\ell}_{k_0,k}(0, \overline{\boldsymbol{\kappa}}^\ell_{\mathbf{y}}, \tilde{\mathbf{f}}_\ell) - \check{\mathbf{v}}_\ell(\mathbf{y}) \right\|_{A^\ell_{\mathbf{y}}} \\
&\leq \frac{1}{2} \|\check{\mathbf{v}}_\ell(\mathbf{y})\|_{A^\ell_{\mathbf{y}}} \\
&\leq \frac{1}{2} \|\hat{\mathbf{v}}_\ell(\mathbf{y})\|_{A^\ell_{\mathbf{y}}} + \frac{1}{2} \|P_\ell \tilde{\mathbf{v}}_{\ell-1}(\mathbf{y}) - P_\ell \mathbf{v}_{\ell-1}(\mathbf{y})\|_{A^\ell_{\mathbf{y}}} \\
&\leq \frac{1}{2} c 2^{-\ell} \|f\|_* + \frac{1}{2} \|\tilde{\mathbf{v}}_{\ell-1}(\mathbf{y}) - \mathbf{v}_{\ell-1}(\mathbf{y})\|_{A^{\ell-1}_{\mathbf{y}}} .
\end{aligned}$$

  Here, the bound on $\hat{\mathbf{v}}_\ell(\mathbf{y})$ follows from Equation (C.3).

- $\boldsymbol{\ell = L}$. By the same argument as for the case $\ell = 2, \dots, L-1$ together with the norm inequality in (C.2) and the definition of m_L, we arrive at the following estimate for the H^1-distance on the finest level.

$$\begin{aligned}
&\|\tilde{\mathbf{v}}_L(\mathbf{y}) - \mathbf{v}_L(\mathbf{y})\|_{H^1} \\
&\leq c \|\tilde{\mathbf{v}}_L(\mathbf{y}) - \mathbf{v}_L(\mathbf{y})\|_{A^L_{\mathbf{y}}} \\
&= c \left\| \widetilde{\mathrm{MG}}^{m_L}_{k_0,k}(0, \overline{\boldsymbol{\kappa}}^L_{\mathbf{y}}, \tilde{\mathbf{f}}_L) - \check{\mathbf{v}}_L(\mathbf{y}) \right\|_{A^L_{\mathbf{y}}} \\
&\leq \frac{1}{2cL} 2^L \varepsilon \|\check{\mathbf{v}}_L(\mathbf{y})\|_{A^L_{\mathbf{y}}} \\
&\leq \frac{1}{cL} 2^L \varepsilon \left(\frac{1}{2} \|\hat{\mathbf{v}}_L(\mathbf{y})\|_{A^L_{\mathbf{y}}} + \frac{1}{2} \|P_L \tilde{\mathbf{v}}_{L-1}(\mathbf{y}) - P_L \mathbf{v}_{L-1}(\mathbf{y})\|_{A^L_{\mathbf{y}}} \right) \\
&\leq \frac{1}{cL} 2^L \varepsilon \left(\frac{1}{2} c 2^{-L} \|f\|_* + \frac{1}{2} \|\tilde{\mathbf{v}}_{L-1}(\mathbf{y}) - \mathbf{v}_{L-1}(\mathbf{y})\|_{A^{L-1}_{\mathbf{y}}} \right)
\end{aligned}$$

  Recursively, this yields

$$\|\tilde{\mathbf{v}}_L(\mathbf{y}) - \mathbf{v}_L(\mathbf{y})\|_{H^1} \leq \frac{1}{cL} 2^L \varepsilon \sum_{\ell=1}^{L} \left(\prod_{n=\ell}^{L} \frac{1}{2} \right) 2^{-\ell} c \|f\|_* = \frac{1}{2} \varepsilon \|f\|_* .$$

In total, the procedure is comprised of downsampling operations, multigrid V-cycles and intermediate upsampling operations, all of which can be approximated arbitrarily well by CNNs: The downsampling of the diffusivity field is realized following Lemma C.1, the V-cycles are realized as shown in Theorem 5.1 and the prolongation operators for upsampling are realized as in Remark C.3. Hence, the procedure poses a concatenation of finitely many functions each

representable by CNNs up to any accuracy. Using Lemma C.2 (with $M = M$ and $\varepsilon = \frac{1}{2}\varepsilon \|f\|_*$) implies the existence of a CNN Ψ such that for all $\mathbf{y} \in \Gamma$ and all right hand sides $\mathbf{f}$, we have

$$\begin{aligned}\|\Psi(\boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}) - \mathbf{v}_L(\mathbf{y})\|_{H^1} &\leq \|\Psi(\boldsymbol{\kappa}_{\mathbf{y}}, \mathbf{f}) - \tilde{\mathbf{v}}_L(\mathbf{y})\|_{H^1} + \|\tilde{\mathbf{v}}_L(\mathbf{y}) - \mathbf{v}_L(\mathbf{y})\|_{H^1} \\ &\leq \varepsilon \|f\|_* .\end{aligned}$$

The structure of the presented multilevel algorithm together with the parameter bounds from Theorem 5.1, Lemma C.1 and Remark C.3 implies that Ψ is an $\mathrm{ML\text{-}Net}$ architecture and that there exists a constant $C > 0$ with

$$\begin{aligned}M(\Psi) &\leq C \sum_{\ell=1}^{L-1} \ell + C \left(\log\left(\frac{2L}{\varepsilon}\right) - L \log(2) \right) L \\ &\leq CL \left(L + \log\left(\frac{1}{\varepsilon}\right) + \log(2L) - L \log(2) \right) \\ &\leq CL \log\left(\frac{1}{\varepsilon}\right) + CL^2.\end{aligned}$$

□