Neural-HSS: Hierarchical Semi-Separable Neural PDE Solver

Pietro Sittoni [*1] **Emanuele Zangrando** [*1] **Angelo Alberto Casulli** [1] **Nicola Guglielmi** [1] **Francesco Tudisco** [1 2 3]

Abstract

Deep learning-based methods have shown remarkable effectiveness in solving PDEs, largely due to their ability to enable fast simulations once trained. However, despite the availability of high-performance computing infrastructure, many critical applications remain constrained by the substantial computational costs associated with generating large-scale, high-quality datasets and training models. In this work, inspired by studies on the structure of Green's functions for elliptic PDEs, we introduce Neural-HSS, a **parameter-efficient** architecture built upon the Hierarchical Semi-Separable (HSS) matrix structure that is provably **data-efficient** for a broad class of PDEs. We theoretically analyze the proposed architecture, proving that it satisfies exactness properties even in very low-data regimes. We also investigate its connections with other architectural primitives, such as the Fourier neural operator layer and convolutional layers. We experimentally validate the data efficiency of Neural-HSS on the three-dimensional Poisson equation over a grid of two million points, demonstrating its superior ability to learn from data generated by elliptic PDEs in the low-data regime while outperforming baseline methods. Finally, we demonstrate its capability to learn from data arising from a broad class of PDEs in diverse domains, including electromagnetism, fluid dynamics, and biology.

1. Introduction and Related Work

Machine learning is emerging as a powerful tool for accurately simulating complex physical phenomena (Brandstetter et al., 2022; Li et al., 2021; Boullé et al., 2023). Unlike traditional numerical methods, which rely on explicit mathematical models and require problem-specific implementations for spatial resolution, timescales, domain geometry, and boundary conditions, machine learning models learn directly from data—either from simulations or real-world observations—enabling greater flexibility and scalability. Moreover, these models leverage ongoing advances in GPU acceleration and large-scale parallelization, with continued improvements in both accuracy and efficiency.

A wide range of architectural primitives has been explored for modeling different physical systems, each leveraging structural biases that arise in the solution operator of specific classes of PDEs. As prominent examples, the Fourier neural operator (Li et al., 2021) learns a kernel integral operator through convolution in Fourier space, enabling efficient representation of global interactions; message passing neural networks (Brandstetter et al., 2022) capture localized interactions via graph-based message updates; U-Net ConvNets (Gupta & Brandstetter, 2022) exploit multiscale representations to couple fine- and coarse-scale features; while operator transformers (Hao et al., 2023) leverage transformers to handle challenging settings such as irregular meshes. Other methods directly approximate the action of linear operators in the elliptic setting (Boullé et al., 2023), bypassing explicit discretization of the underlying equations. Moreover, (Boullé et al., 2023) show that the underlying learning problem is well-posed for a very small number of datapoints and prove that the class of elliptic PDEs is "data efficient": the number of training points needed to learn the solution operator depends only on the problem dimension. This effect is due to the nature of the solution operator for elliptic-type PDEs, which is highly structured: the Green function $G(x, y)$ associated with the solution operator is a Hierarchical Semi-Separable (HSS) mapping that exhibits low rank when restricted to off-diagonal subdomains.

In fact, the Multipole Graph Neural Operator employed in (Boullé et al., 2023) is motivated by a hierarchical structure observed in Green functions (Boullé & Townsend, 2024, Section 2.4), known as the Hierarchical Off-Diagonal Low-Rank (HODLR) structure. In an HODLR matrix, off-diagonal blocks at different scales are well-approximated by low-rank factorizations. When used within a neural architecture, this flexible representation allows the network to capture both near-field and far-field interactions in a multiscale fashion and is one of the key factors behind the

[1]Gran Sasso Science Institute, L'Aquila, Italy [2]University of Edinburgh, Edinburgh, UK [3]Miniml.AI. Correspondence to: Pietro Sittoni <pietro.sittoni@gssi.it>, Emanuele Zangrando <emanuele.zangrando@gssi.it>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

data-efficiency property of the neural architecture.

The HSS structure is a special case of the more general HODLR format, a hierarchical domain-decomposition strategy that has been widely studied in the scientific computing literature after the seminal work (Tyrtyshnikov, 1996) for developing fast and memory-efficient solvers for elliptic-type PDEs, e.g., (Hackbusch et al., 2005; Martinsson & Rokhlin, 2005; Gillman et al., 2012). Compared to HODLR, the HSS model enforces a nested basis across levels in addition to low-rank off-diagonal blocks. This nested parameterization greatly reduces redundancy, yielding a more compact representation and faster matrix–vector products, improvements that are not available in the plain HODLR structure.

In this work, we propose Neural-HSS, a neural architecture that injects this type of structure into a neural network model for PDE learning. Thanks to the efficient data representation of HSS, the proposed model is able to approximate solution operators with far fewer parameters than baseline models while retaining better or comparable accuracy.

The geometric structure of HSS operators imposes an inductive bias that concentrates modeling capacity on local interaction effects in the physical system—modeled through full-rank submatrices—while approximating interaction effects between distant subdomains using low-rank matrices, resembling a mean-field approximation. This modeling strategy underlies many popular neural PDE solvers. In fact, architectures such as ResNets (He et al., 2016), Swin Transformers (Liu et al., 2021), and Message Passing Neural Networks (Gilmer et al., 2017) devote the majority of their representational capacity to modeling local interactions. This bias aligns well with the nature of PDE dynamics, in which the dominant behavior is driven by localized interactions, in contrast to integro-differential equations, which can incorporate global interaction effects. The effect of this implicit bias was also highlighted in (Holzschuh et al., 2025), where the authors demonstrate that in a Swin-Transformer-based model, increasing the attention window size leads to a rapid performance plateau.

Overall, our main contributions are as follows:

- We propose Neural-HSS, a **parameter-efficient** neural architecture with a novel type of layer inspired by HSS theory for PDEs (Hackbusch, 2015), and we establish its universal approximation property.
- We prove that the proposed architecture is **exact** and **data-efficient** on the class of elliptic PDEs, i.e., (a) the global minimizers of the empirical loss represent the discretized solution operator exactly, and (b) the number of data points required to learn the exact operator depends only on the intrinsic dimensionality of the problem. To the best of our knowledge, this is the first result for an architecture that can handle arbitrary types of PDEs while guaranteeing exactness and data efficiency for a broad class of PDEs.
- We highlight an intriguing connection between the proposed architecture and a discretized version of the convolutional-type layer used in FNO architectures, showing that an HSS layer can approximate it arbitrarily well with very few parameters.
- Under the assumptions of Theorem 2.3, which match the setting of (Boullé et al., 2023), we conduct two experiments in 1D and 3D. The 3D experiment is performed on a grid with 2M points, a well-known challenge for machine learning models. In both cases, we demonstrate the superior performance and scalability of Neural-HSS.
- We conduct extensive experiments on a broad class of PDEs arising from electromagnetism, fluid dynamics, and biology, showcasing strong performance relative to commonly used architectural primitives such as ResNet and FNO layers in terms of parameter efficiency, test error, and computational time. In particular, this last advantage becomes more evident for higher-dimensional PDEs. Moreover, we demonstrate the effectiveness of Neural-HSS beyond elliptic PDEs, showing that the same architecture is also effective for different classes of nonlinear PDEs.

Related Work. There has been a surge of interest in learning-based approaches for improving classical solvers for linear PDEs. Several works focus on accelerating iterative methods such as Conjugate Gradient for symmetric positive definite systems (Li et al., 2023a; Kaneda et al., 2023; Zhang et al., 2023), or GMRES-type solvers for specific applications like the Poisson equation (Luna et al., 2021). Others focus on learned preconditioning strategies: neural networks have been used to construct preconditioners that speed up convergence (Greenfeld et al., 2019; Luz et al., 2020; Taghibakhshi et al., 2021), or to optimize heuristics such as Jacobi and ILU variants (Flegar et al., 2021; Stanaityte, 2020). NeurKItt (Luo et al., 2024), for example, employs a neural operator to predict the invariant subspace of the system matrix and accelerate solution convergence. However, these approaches are primarily designed to enhance classical linear numerical pipelines. In contrast, our work takes a fundamentally different perspective: we aim to learn an end-to-end solver.

Moreover, recent work has focused on learning low-dimensional latent representations of PDE states for efficient simulation, extending classical projection-based reduction (Benner et al., 2015) with deep learning methods such as autoencoders (Wiewel et al., 2019; Maulik et al., 2021), graph embeddings (HAN et al., 2022), and implicit neural representations (Chen et al., 2023). Koopman-inspired

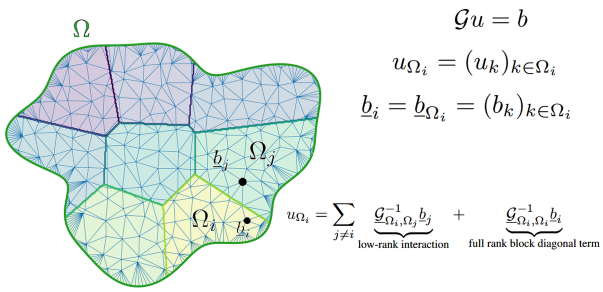

Figure 1. Illustration of the low-rank/high rank interaction bias of HSS structure.

methods enforce linear latent dynamics (Geneva & Zabaras, 2022; Yeung et al., 2019), while latent neural solvers and transformer-based ROMs have been developed for end-to-end modeling (Li et al., 2025a; Hemmasian & Barati Farimani, 2023). In addition, (Kissel & Diepold, 2023b) introduces a hierarchical (Fan et al., 2019b) network to model the nonlinear Schrödinger equation. Another key challenge remains long-horizon stability, motivating strategies like autoregressive training, architectural constraints, and spectral or stochastic regularization (Geneva & Zabaras, 2020; McCabe et al.; Stachenfeld et al., 2022). More recently, generative models, in particular using diffusion-based models, have shown promise for stable rollout and data assimilation by producing statistically consistent trajectories (Shysheya et al., 2024; Li et al., 2025b; Andry et al., 2025).

The use of structured matrices in neural network architectures is also highly relevant and used across different areas of deep learning. Recent approaches include the use of low-rank (Schotthöfer et al., 2022; Zangrando et al., 2024) and hierarchical decompositions (Fan et al., 2019a; Kissel & Diepold, 2023a), low-displacement rank (Thomas et al., 2019; Zhao et al., 2017; Choromanski et al., 2024), butterflies and monarchs (Fu et al., 2023; Dao et al., 2019; 2022).

2. Setting and Model

In this section, we present the necessary definitions and theoretical motivations for our proposed architecture. We start with the following formal definition of HSS structure:

Definition 2.1. (HSS structure (Casulli et al., 2024, Definition 3.1)) Let $\mathcal{T}$ be a cluster tree of depth L for the indices $[1,\dots,d]$. A matrix $A \in \mathbb{R}^{d\times d}$ belongs to $\mathrm{HSS}(r,\mathcal{T})$ or simply $\mathrm{HSS}(r)$ if there exist real matrices

$$\{U_\tau, V_\tau : \tau \in \mathcal{T},\, 1 \le \mathrm{depth}(\mathcal{T})\} \quad \text{and} \quad \{D_\tau : \tau \in \mathcal{T}\}$$

called *telescopic decomposition* and for brevity denoted by $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}}$ or simply $\{U_\tau, V_\tau, D_\tau\}$, with the following properties:

1. D_τ is of size $|\tau| \times |\tau|$ if $\mathrm{depth}(\tau) = L$ and $2r \times 2r$ otherwise;
2. U_τ, V_τ are of size $|\tau| \times r$ if $\mathrm{depth}(\tau) = L$ and $2r \times r$ otherwise;
3. if $L = 0$ (i.e., $\mathcal{T}$ consists only of the root γ) then $A = D_\gamma$;
4. if $L \ge 1$ then $A^{(L)} = \boldsymbol{D}^{(L)} + \boldsymbol{U}^{(L)} A^{(L-1)} (\boldsymbol{V}^{(L)})^T$ where

$$\begin{aligned}
\boldsymbol{U}^{(L)} &:= \mathrm{blkdiag}(U_\tau : \tau \in \mathcal{T}, \mathrm{depth}(\tau) = L),\\
\boldsymbol{V}^{(L)} &:= \mathrm{blkdiag}(V_\tau : \tau \in \mathcal{T}, \mathrm{depth}(\tau) = L),\\
\boldsymbol{D}^{(L)} &:= \mathrm{blkdiag}(D_\tau : \tau \in \mathcal{T}, \mathrm{depth}(\tau) = L)
\end{aligned}$$

and the matrix $A^{(L-1)} = (\boldsymbol{U}^{(L)})^T (A^{(L)} - \boldsymbol{D}^{(L)}) \boldsymbol{V}^{(L)}$ has the telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}^{(L-1)}_{2r}}$, where $\mathcal{T}^{(L-1)}_{2r}$ denotes a balanced cluster tree of depth $L-1$ for the indices $[1,\dots,2^L r]$, see Definition A.2.

This definition is a natural extension of the idea of a low-rank linear operator, when the matrix is not globally low-rank but admits a low-rank expansion on subdomains that do not intersect.

The utility and efficiency of these structures become immediately clear when considering elliptic-type PDEs. In fact, when one discretizes an elliptic partial differential operator—say by finite differences, finite elements, or boundary integral methods—a large algebraic system $Au = f$ arises, where A is associated with discretized Green's functions. It turns out that many of the off-diagonal blocks of A have low numerical rank because of the smoothing properties of elliptic operators, making distant interactions "weak" and therefore inducing a decay in the singular values, which depends on the regularity of the kernel (Bebendorf, 2000; Bebendorf & Hackbusch, 2003). HSS matrices exploit exactly this feature by organizing the matrix into a hierarchical block structure in which off-diagonal blocks are approximated by low-rank factorizations. Thus, HSS-based models are used to design fast numerical solvers or preconditioners for elliptic PDEs by efficiently approximating the operator, thereby reducing storage and computational complexity (Börm, 2010; Börm & Grasedyck, 2005; Gillman et al., 2012). As the HSS structure is invariant under matrix inversion, these properties demonstrate that HSS operators can effectively model the structure of the solution operator and offer a powerful modeling primitive for a neural PDE solver.

Based on this key observation, we propose below the Neural-HSS model.

2.1. Model Overview

In this section, we present the proposed Neural-HSS architecture, as illustrated in Figure 2.

One-dimensional HSS layer. For 1-dimensional problems, the HSS layer consists of an HSS structured matrix followed by a nonlinear activation. In Appendix D we describe the forward pass in more detail, along with input and output channels of the layers, the number of levels, and the rank of the layer. The rank controls the size of the low-rank coupling matrices between sub-blocks, enabling efficient compression of off-diagonal interactions. The number of levels determines the recursion depth, i.e., how many hierarchical splits the input undergoes. Clearly, the HSS layer is fully compatible with the backpropagation algorithm. The structure of the layer allows us to stack multiple layers, enabling the construction of deeper architectures.

Activation function. We employ the $\mathrm{LeakyReLU}_\alpha$ activation. α is typically a tunable hyperparameter; however, in our implementation, α is learnable. In particular, if the underlying PDE is linear, the model adapts $\alpha \to 1$, effectively recovering the identity function. Instead, for nonlinear PDEs, α can deviate from 1 to capture the nonlinearity where necessary. This activation function is also relevant because it fulfills the exactness guarantees presented in Theorem 2.3. While this is the activation of choice in our implementation, we emphasize that this choice is not restrictive: other activation functions can be used if better suited for the problem at hand. We also highlight that as long as the activation acts entrywise, the HSS-induced structural bias is maintained as the topology of the interlayer connections is not affected.

m-dimensional HSS layer. The complex hierarchical architecture of HSS operators in higher dimensions poses significant challenges, as it depends on the geometry of a partition of the domain, which can be arbitrarily complex. For this reason, different possible models can be used to extend the HSS structure to higher-dimensional tensors (Hackbusch, 2015). Here we approximate the higher-dimensional HSS by parametrizing each layer as a high-dimensional tensor obtained as a low-rank expansion of the outer product of one-dimensional HSS layers. Given the implementative and computational complications of higher-dimensional HSS layer, this approximation is a deliberate compromise to retain scalability and parameter efficiency. More precisely, an m-dimensional HSS layer is a tensor of outer CP-rank r_{out} parametrized as follows:

$$
\begin{aligned}
&H_\theta^m : \mathbb{R}^{\overbrace{d \times \cdots \times d}^{m}} \to \mathbb{R}^{d \times \cdots \times d}, \\
&H_\theta^m(\mathcal{Z}) := \sum_{k=1}^{r_{\text{out}}} \mathcal{Z} \bigtimes_{j=1}^{m} W_j^{(k)}, \\
&W_j^{(k)} \in \mathrm{HSS}(r_{k,j}) \subset \mathbb{R}^{d \times d}, \\
&\theta = (W_j^{(k)})_{j,k} \text{ trainable parameters.}
\end{aligned}
\tag{1}
$$

Where the definition of modal product $\bigtimes$ is recalled in Definition A.1 and $\mathcal{Z}$ is the layer's input tensor. This model is motivated by the effectiveness of the Canonic Polyadic decomposition in representing very high-dimensional tensors with a small number of variables (Hitchcock, 1927; Lebedev et al., 2015). In fact, while the parameter count on a generic linear map between tensors with m modes would scale as $O(d^{2m})$, the memory for this m-dimensional HSS layer scales as $O(r_{out} m r d)$ when $r_{k,j}$ are all equal to r. Thus, the architecture's parameter-efficiency increases as the dimensionality of the PDE grows.

2.2. Theoretical Results

In this section, we present our main theoretical results, showing that the proposed Neural-HSS enjoys some useful properties.

First of all, we notice in the next Theorem that the HSS structure is efficient in representing convolutional-type operators in which the kernel is regular. Thus, each HSS layer in the proposed model can be interpreted as a generalization of a (discretized) convolutional-type linear layer, such as those implemented in popular FNO or CNO architectures (Li et al., 2021; Raonic et al., 2023), as well as classical convolutional filters.

Theorem 2.2. *(Convolutional kernels are HSS approximable) Let $D \geq 1$, $\Omega \subseteq \mathbb{R}^D$ be a compact set, let $k : \Omega \to \mathbb{R}$ be an asymptotically smooth convolutional kernel (Definition A.4). Consider the operator*

$$
\begin{aligned}
&T : \mathcal{C}^0(\Omega; \mathbb{R}) \to \mathcal{C}^0(\Omega; \mathbb{R}), \\
&(Tf)(x) = \int_\Omega k(x-y) f(y)\, dy.
\end{aligned}
$$

For a set of basis functions $\{\phi_j\}_{j=1}^K$, consider the discretization matrix

$$
A_{ij} = \int_\Omega k(x-y)\, \phi_i(x)\, \phi_j(y)\, dx\, dy, \quad i,j = 1, \ldots, K.
$$

Then, for every η-admissible pair (Definition A.3) of well-separated clusters τ, τ', there exist $r = O(\log(1/\varepsilon)^D)$ such that

$$
\inf_{B\,:\,\mathrm{rank}(B)=r} \|\nu|_{\tau\times\tau'} - B\|_F \leq \varepsilon
$$

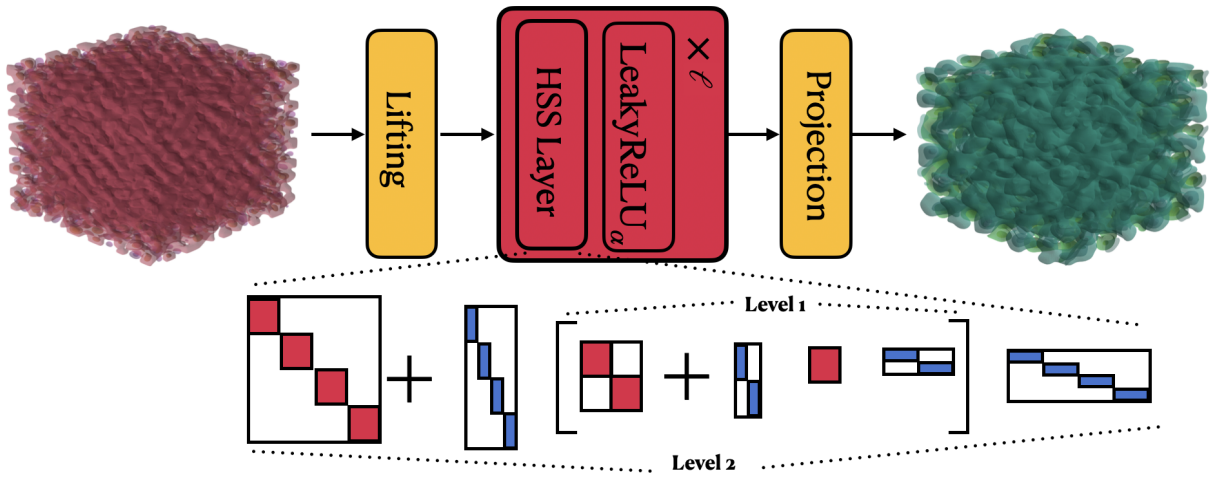

Figure 2. **Model overview.** The lifting and projection layers can be implemented either as HSS layers or as full-rank layers. We illustrate, as an example, the weight matrix structure with two different hierarchical levels.

where $A|_{\tau\times\tau'} := \Pi_{\tau'} A \Pi_\tau$ is the orthogonally projected operator in the subdomains spanned by the nodes τ, τ'.

The result in Theorem 2.2 shows that each regular convolutional operator on well-separated domains can be well approximated by a low-rank matrix, where the rank grows logarithmically with the tolerance. In other terms, if the domains are well-separated and the kernel is regular enough, then interactions between subdomains can be well-approximated through a low-rank expansion. Thus, the whole convolution can be approximated in $\mathrm{HSS}(r)$ by recursively partitioning the index set into clusters. The proof of Theorem 2.2 is included in Appendix B.

Next, we analyze approximation and data efficiency properties of the model. Note that by setting the number of hierarchical levels to its minimum and simultaneously increasing the rank to its maximum, the backbone effectively reduces to a standard multilayer perceptron (MLP). This observation immediately implies that Neural-HSS inherits the well-known **universal approximation** property of MLPs, at the cost of possibly increasing the number of levels of the tree and the rank.

Moreover, in combination with piecewise linear activation functions with learnable slope, it satisfies data-efficiency and exactness recovery properties, as formalized in the following:

Theorem 2.3. *(Exact Recovery and Data-Efficiency) Consider the model*

$$\begin{aligned} \mathcal{N}_\theta(b) &= H_\ell \circ \cdots \circ H_1(b), \\ H_i(z) &= \mathrm{LeakyReLU}_{\alpha_i}(W_i[z]), \\ \theta &= (W_1, \alpha_1, \ldots, W_\ell, \alpha_\ell) \end{aligned}$$

and for $\lambda > 0$, the loss function together with its set of global minimizers

$$\begin{aligned} \mathcal{L}_N(\theta) &= \sum_{i=1}^{N} \|\mathcal{N}_\theta(b_i) - u_i\|_2^2 + \frac{\lambda}{2}\sum_{i=1}^{\ell}(\alpha_i - 1)^2, \\ \mathcal{M}_N &:= \underset{W_i \in \mathrm{HSS}(r_i, \mathcal{T}), \alpha_i \in \mathbb{R}}{\arg\min} \mathcal{L}_N, \end{aligned}$$

where the datapoints $\{b_i\}_{i=1}^N \subseteq \mathbb{R}^{d^k}$ are standard-Gaussian distributed and $\{u_i\}_{i=1}^N \subseteq \mathbb{R}^{d^k}$ satisfying $\mathcal{G}u_i = b_i$ with $\mathcal{G}^{-1} \in \mathrm{HSS}(r, \mathcal{T})$. Furthermore, suppose $\max_i r_i \geq r$.

Then there exists a constant $C > 0$ such that, whenever $N \geq C \cdot \sum_{i=1}^{\ell} r_i$, the following exact recovery identity holds with high probability:

$$\sup_{\theta \in \mathcal{M}_N} \left\|\mathcal{N}_\theta(\cdot) - \mathcal{G}^{-1}(\cdot)\right\|_{L^\infty} = 0.$$

Consequently, any global minimizer $\theta^* \in \mathcal{M}_N$ has zero generalization gap, i.e., $\mathcal{N}_{\theta^*} = \mathcal{G}^{-1}$ and the model is data-efficient as exact recovery is possible with a number of data points N that depends only on the architecture's intrinsic dimensionalities r_i.

A proof of Theorem 2.3 is included Appendix C. This result shows that even in possession of only a possibly very small number of sufficiently informative/independent examples (which is guaranteed by the randomness of source terms with high probability), if the underlying operator is HSS, then all global minimizers of $\mathcal{L}_N$ recover the exact solution operator. This is, for example, the case for linear elliptic PDEs. Moreover, notice that Theorem 2.3 formalizes the "data-efficiency" for the proposed architectures, in the spirit of what was done in (Boullé et al., 2023). An experimental validation of this result is presented in Figure 3 and Figure 5, where we numerically compare the data-efficiency of our architecture against that of other baseline models.

Remark 2.4. (Approximation beyond the elliptic regime) Beyond the regime where exact HSS theory applies, the structured linear part still provides a strong locality/compression prior: nearby interactions are modeled with higher effective rank, while long-range interactions are compressed through low-rank couplings. Once combined with nonlinearities, this yields a more expressive model than the HSS matrix class alone. As an example, consider the parametric nonlinear operator

$$\mathcal{F}_\nu(u) = \nu\mathcal{F}(u) + \mathcal{G}u, \quad u_\nu := \mathcal{F}_\nu^{-1}(b), \quad u_0 := \mathcal{G}^{-1}b.$$

Assume that $\mathcal{G} : X \to X$ is boundedly invertible with $\mathcal{G}^{-1}$ HSS structured, that $\mathcal{F} : X \to X$ is Lipschitz continuous with Lipschitz constant K and let $M := \|\mathcal{G}^{-1}\|$. Let u_0 be a solution of $\mathcal{G}u_0 = b$ and consider the semilinear PDE $\mathcal{G}u_\nu + \nu\mathcal{F}(u_\nu) = b$. Subtracting the two equations gives $\mathcal{G}(u_\nu - u_0) + \nu\mathcal{F}(u_\nu) = 0$, which implies $u_\nu - u_0 = -\nu\mathcal{G}^{-1}\mathcal{F}(u_\nu)$. Therefore, by Lipschitz continuity, we have

$$\|u_\nu - u_0\| \leq |\nu|M\|\mathcal{F}(u_0)\| + |\nu|MK\|u_\nu - u_0\|.$$

Assuming $|\nu|MK < 1$, we can rearrange to get

$$(1 - |\nu|MK)\|u_\nu - u_0\| \leq |\nu|M\|\mathcal{F}(u_0)\|,$$

and consequently

$$\|u_\nu - u_0\| \leq \frac{|\nu|M}{1 - |\nu|MK}\|\mathcal{F}(u_0)\|.$$

In terms of b, this is

$$\|\mathcal{F}_\nu^{-1}(b) - \mathcal{G}^{-1}b\| \leq \frac{|\nu|\,\|\mathcal{G}^{-1}\|}{1 - |\nu|\,\|\mathcal{G}^{-1}\|K}\|\mathcal{F}(\mathcal{G}^{-1}b)\|.$$

Moreover, for every $R > 0$, if $\|b\| \leq R$, then

$$\|\mathcal{F}(\mathcal{G}^{-1}b)\| \leq \|\mathcal{F}(0)\| + K\|\mathcal{G}^{-1}b\| \leq \|\mathcal{F}(0)\| + KMR.$$

Therefore

$$\sup_{\|b\|\leq R}\|\mathcal{F}_\nu^{-1}(b) - \mathcal{G}^{-1}b\| \leq \frac{|\nu|M\left(\|\mathcal{F}(0)\| + KMR\right)}{1 - |\nu|MK}.$$

Thus, $\mathcal{F}_\nu^{-1} \underset{\nu\to 0}{\to} \mathcal{G}^{-1}$, uniformly for $\|b\| \leq R$. Therefore, the solution operator can be very close to an HSS one uniformly, despite the problem being nonlinear or non-elliptic. Moreover, by using a more refined structure that includes nonlinearities, this estimate can improve significantly. This is, in practice, what we observe numerically.

3. Experiments

In this section, we will present the numerical results. For a more complete description of the PDE problems considered and the data generation, we refer to Appendix F; for the hyperparameter settings and model implementation details, we refer to Appendix M; for more details on the training loss and evaluation metrics used, we refer to Appendix G.

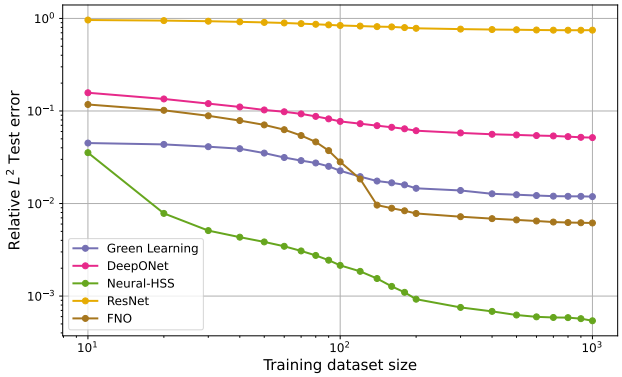

Figure 3. Train size vs relative test error for different models. The models are trained on a 1D Poisson equation.

3.1. Data Efficiency

We conduct two experiments analogous to those in (Boullé et al., 2023). The first experiment focuses on the one-dimensional Poisson equation, where we train our models on datasets of varying sizes, ranging from 10 to 10^3 samples. The second experiment considers the three-dimensional equation on a grid with resolution $128 \times 128 \times 128$, corresponding to approximately 2×10^6 grid points. Here, we vary the training set size from 16 to 256 samples. We use fewer training samples in three dimensions because, as shown in (Boullé et al., 2023), meaningful conclusions can already be drawn with training sets of around 200 samples. Moreover, generating three-dimensional data and training models on it is computationally expensive, further motivating the need for data-efficient models in 3D simulations.

In the 3D experiment in Figure 5, we do not report the performance of the Green Learning model (Boullé et al., 2023), since the memory of an NVIDIA A100 80GB was not sufficient already using a batch size of 1 and a depth of 1. We also omit DeepONet results for training set sizes smaller than 128, as its performance in this regime was poor and including it would clutter the visualization. The full plot can be found in Appendix H. Details on the models are provided in Appendix M.

To verify the data efficiency also for problems in which the HSS structure is not exact, we propose the same numerical experiment on the steady Burgers' equation. In particular, the aim is to approximate the operator $u_0 \mapsto u_\infty$, where u_∞ is the solution of $u_\infty\partial_x u_\infty = \nu\partial_x^2 u_\infty$ (with viscosity parameter $\nu = 10^{-3}$. As shown in Figure 4, Neural-HSS still demonstrates data efficiency even when the exact solution operator is not HSS.

1D Our findings in Fig. 3 are consistent with those reported in (Boullé et al., 2023). With a limited training sample budget, Green Learning outperforms the FNO model, as

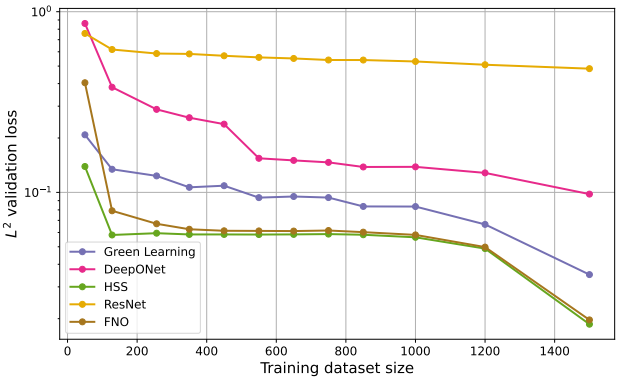

Figure 4. Train size vs relative test error for different models. The models are trained on a 1D Steady Burger equation.

observed in (Boullé et al., 2023); however, as the number of samples increases, FNO scales more effectively and surpasses Green Learning once the training size exceeds 10^2. In contrast, Neural-HSS consistently outperforms all baselines across all training budgets. Notably, with very small datasets (around ~ 10 samples), the performance gap between Neural-HSS and the Green Learning model remains small. As the training size increases, this gap grows substantially in favor of Neural-HSS, highlighting its superior scalability in this setting.

3D The 3D setting is particularly challenging for neural PDE solvers, as generating large, high-fidelity 3D datasets is costly, not only in terms of simulation but also in data storage and model training. This makes it of paramount importance to have a model that can be trained with a very reduced number of samples when the underlying solution operator is structured.

As shown in the results in Figure 5, Neural-HSS consistently outperforms all baselines. With only 16 samples, Neural-HSS matches the performance of FNO trained on 64 samples and ResNet trained on 128 samples. At 32 samples, Neural-HSS achieves performance comparable to FNO trained on 256 samples. Training time is a crucial factor Figure 5. Neural-HSS trains significantly faster than both FNO and ResNet, completing training in about two and a half hours, compared to six hours for FNO and nearly one day and eighteen hours for ResNet. DeepONet trains even faster, requiring only about one hour, but it needs a much larger training set to achieve comparable performance. With a training size of only 256 samples, DeepONet cannot match Neural-HSS's accuracy. Since generating new samples is highly expensive, Neural-HSS is overall the most efficient choice, also in terms of time.

Together with the results in Table 2, this shows that the choice of higher-order HSS layer for $m-$dimensional problems proposed in Equation (1) is both simple and effective, even with very small outer rank values r_{out}, which for this experiment was set to two (see Appendix M).

3.2. Additional Evaluation of PDE Learning Performance

1D PDEs. To showcase the effectiveness of Neural-HSS on time-dependent problems, we train the models to predict the dynamics of the Heat and Burgers' equations. Specifically, the models learn the time-stepping operator $\mathcal{G} : u_t \mapsto u_{t+\delta t}$. At this timescale, prior work has shown that learning the residual yields better performance than direct prediction (Li et al., 2022). Following this strategy, our model M is trained to predict $\delta u = u_{t+\delta t} - u_t$, so that, during inference, the state is updated as $u_{t+\delta t} = u_t + M(u_t)$. For training stability, we normalize the residuals by the maximum value in the training set, i.e., $\max \delta u$. We defer the model details with all the hyperparameters to Appendix M. We want to remark that we do not provide results for Green Learning on the Burgers equation, as the method requires the underlying PDE to be linear.

In both experiments, Neural-HSS outperforms the baselines while using fewer parameters, see Table 1. For the Heat equation, we observe trends consistent with the data efficiency experiments (Section 3.1). When trained on a larger dataset (approximately 10^4 samples), Green Learning underperforms compared to both Neural-HSS and FNO. DeepONet is the most parameter-hungry model, we fix it at 247K parameters, since smaller variants consistently yielded weaker performance. We notice, moreover, that by the end of training, the parameter α of the $\mathrm{LeakyReLU}$ converged to 1, reflecting that the model has learned the underlying relation to be linear.

For the Burgers' equation, we find that employing a full-rank lifting and projection improves performance. This observation is aligned with findings in the literature on training models with low-rank parameter matrices, where the last and sometimes first layers are typically kept full-rank (Schotthöfer et al., 2022; Zangrando et al., 2024; Wang et al., 2021). Moreover, with respect to the heat equation, it is necessary to employ a higher rank in the intermediate layers to achieve sufficient expressivity. In this case, Neural-HSS consistently outperforms all other baselines while using fewer parameters and, similarly to the heat equation, the FNO emerges as the second-best performing.

We run another series of experiments on a modified version of the Burgers' equation, where, using a parameter β, we shift the equation towards the elliptic setting. While the other models seem to be unaffected by the change in the PDE setting, Neural-HSS shows a significant performance improvement as the equation approaches the elliptic setting, without any change to the hyperparameter setting. We defer to Appendix J the detailed results.

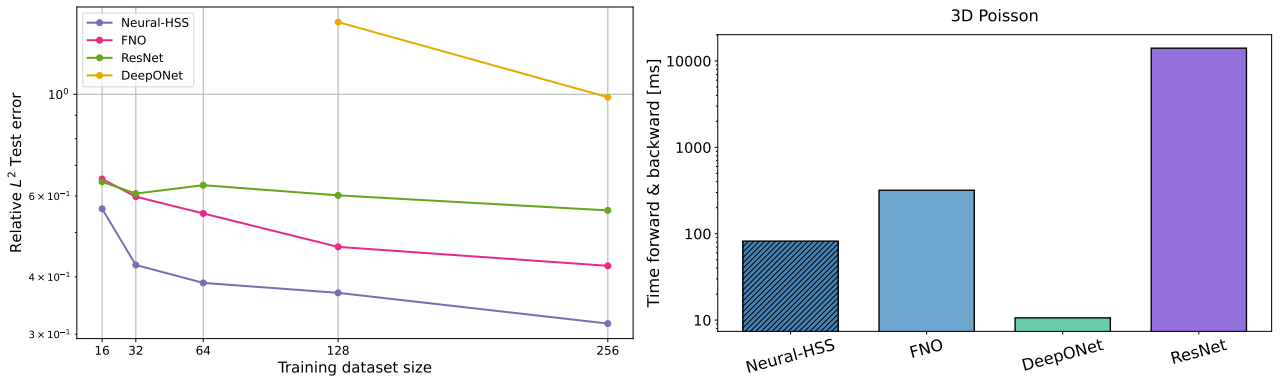

Figure 5. **Left**: Train size vs relative test error for different models. The models are trained on a 3D Poisson equation. **Right**: Timing of forward and backward for the different models.

As shown in Figure 10, Neural-HSS's time for a single optimization step is comparable with other baseline models, while being significantly more effective as shown in Table 1. The efficiency is comparable to that of FNO, for which most of the computations are performed in Fourier space while truncating the modes, which significantly reduces cost. DeepONet is the fastest model; however, it also has the highest test error. Timings of only the inference phase are reported in Appendix L.

To further validate Neural-HSS, we tested all the methods on the Poisson equation with Neumann-type boundary conditions. Coherently with the theory, as it is known that the solution operator still has HSS structure even with these boundary conditions (Hackbusch, 2015), we observe that Neural-HSS achieves the lowest test error and parameter count. In this setting, ResNet underperforms with respect to Dirichlet boundary conditions, and the test error of all the other baselines are comparable.

2D PDEs. In Table 2 and in Figure 11, we test model performance on 2D problems, when predicting the steady state of an equation or a specific time step. For the incompressible Navier-Stokes, we use a Z-score normalization as in the original papers (Yao et al., 2025; Huang et al., 2024); for the other equations, we use a max rescaling as for the 1D experiments. We defer the model details with all the hyperparameters to Appendix M. We do not provide results for Green Learning on the non-elliptic equation, as the method requires the underlying PDE to be linear. In addition to all the previous architectures, for the 2D case, we can also compare with the FactFormer architecture proposed in (Li et al., 2023b) and with the Transolver architecture (Wu et al., 2024).

For the Poisson equation, similar to the 1D case, smaller DeepONet architectures underperform. We clearly observe that the Neural-HSS significantly outperforms all baselines, with an even larger performance gap in the 2D experiments compared to the 1D case. As before, the FNO model ranks second, followed by the Green Learning model, consistent with the findings from the 1D experiments.

In the non-elliptic setting, Neural-HSS demonstrates competitive performance compared to other models. This is particularly evident for the incompressible Navier–Stokes equation, where Neural-HSS outperforms all baselines in the forward problem. For the Gray–Scott forward model, Neural-HSS ranks as the second-best model, with ResNet achieving a slightly lower test error, but using significantly fewer parameters and fewer training steps. A very similar situation is observed in the Gray-Scott inverse problem, where FactFormer and ResNets' test errors are essentially equal and slightly lower than Neural-HSS. As for the 1D experiments, we use full-rank layers for the lift and the projection.

Also for these experiments, we report in Figure 11 the time required for a training step. In this setting, Neural-HSS remains highly competitive. In particular, for the Poisson equation, it is the second most computationally efficient model after DeepONets, but with a gap of six orders of magnitude in performance. For the Gray–Scott model and incompressible Navier–Stokes equation, training step timing is comparable to that of the FNO layer, which benefits from a low-cost forward pass due to mode truncation. The inverse Gray–Scott experiment highlights an interesting trade-off: although ResNet achieves the lowest test error, it requires more than one order of magnitude more in terms of time compared to Neural-HSS, which attains the second-lowest error. Given that the test-error gap between the two models is very small, Neural-HSS may be preferable in practice due to its substantially lower memory and computational cost.

Moreover, we remark here that the theory of HSS operators does not strictly depend on the domain being rectangular or on the specific Dirichlet boundary conditions. For non-rectangular domains, the Green's function remains smooth

	Neural-HSS		FNO		ResNet		DeepONet		Green Learning	
Equation	Params	Test Err	Params	Test Err	Params	Test Err	Params	Test Err	Params	Test Err
Heat Eq.	45K	3×10^{-3}	150K	8×10^{-3}	165K	1×10^{-2}	247K	1×10^{-2}	83K	1×10^{-2}
Poisson Neumann BC	37K	2×10^{-4}	102K	3×10^{-3}	165K	4×10^{-1}	247K	9×10^{-3}	83K	3×10^{-3}
Burgers' Eq.	1.5M	0.12	1.5M	0.26	1.7M	0.57	1.7M	0.44	-	-

Table 1. Comparison between Neural-HSS and baseline models. We report the number of learnable parameters for each model and the test error Eq. (3).

	Neural-HSS		FNO		ResNet		DeepONet		Green Learning		FactFormer		Transolver	
Equation	Params	Test Err	Params	Test Err	Params	Test Err	Params	Test Err	Params	Test Err	Params	Test Err	Params	Test Err
Poisson Eq.	37K	7×10^{-8}	132K	7×10^{-6}	165K	3×10^{-4}	280K	2×10^{-2}	83K	5×10^{-5}	3.9M	6×10^{-3}	1.8M	3.7×10^{-3}
Poisson Eq. (Mixed BC)	37K	1×10^{-4}	132K	6×10^{-3}	165K	8×10^{-2}	280K	4×10^{-1}	83K	1×10^{-2}	3.9 M	7×10^{-2}	1.8M	1.2×10^{-1}
Poisson Eq. (L-shape domain)	37K	2×10^{-2}	132K	2×10^{-1}	165K	7×10^{-1}	280K	8×10^{-1}	83K	5×10^{-1}	3.9 M	2×10^{-1}	1.8M	1.01
Helmholtz equation ($k = 70$)	37K	6×10^{-3}	132K	5×10^{-2}	165K	1×10^{0}	280K	1×10^{0}	83K	1×10^{0}	3.9 M	7×10^{-2}	1.8M	7×10^{-2}
Helmholtz equation ($k = 7$)	37K	1.7×10^{-2}	132K	2.1×10^{-2}	165K	3.9×10^{-1}	280K	1.5×10^{-1}	83K	6×10^{-1}	3.9 M	1.3×10^{-1}	-	-
Gray–Scott Eq. (Forward)	329K	0.294	2.1M	0.331	1.99M	0.273	2.3M	0.315	-	-	3.9M	0.291	1.8M	0.356
Gray–Scott Eq. (Inverse)	329K	0.203	2.1M	0.208	1.99M	0.193	2.3M	0.276	-	-	3.9M	0.192	1.8M	0.159
NS Eq. (Forward)	329K	0.123	2.1M	0.483	1.99M	0.481	2.3M	0.409	-	-	3.9M	0.14	1.8M	0.183
NS Eq. (Inverse)	329K	0.208	2.1M	0.19	1.99M	0.383	2.3M	0.514	-	-	3.9M	0.21	1.8M	0.233

Table 2. Comparison between Neural-HSS and baseline models on 2D problems. We report the number of learnable parameters for each model and the test error Equation (2). With NS Eq., we refer to the incompressible Navier-Stokes equation.

for well-separated points, so the same low-rank approximation property applies (Hackbusch, 2015). For this purpose, we experimentally demonstrate the performance of Neural-HSS on an L-shaped domain, for which we include the results in Table 2.

We also tested Neural-HSS on the 2D Poisson equation on the unit square domain deprived of a small disk of radius $R = 0.2$ in the center, with mixed boundary conditions. More precisely, on the external boundary of the square, we imposed periodic boundary conditions, while on the interior border of the disk, we imposed Neumann boundary conditions. While the exact structure of the solution operator in this more complicated geometry is not guaranteed theoretically to be well captured by the HSS structure, we show numerically that the inductive bias induced by Neural-HSS still works as a good approximation, as shown in the results in Table 2.

Finally, compared to the one-dimensional experiments, we observe that the efficiency of Neural-HSS compared to the other baselines increases with the dimension as discussed in Section 2.1. We also refer to Section 3.1 for the three-dimensional case, in which this effect is even more evident.

Conclusion and Limitations

In this work, we present Neural-HSS, a novel hierarchical architecture inspired by the structure of the solution operator of Elliptic-type linear PDEs. Leveraging this very structured representation, we are able to produce lightweight neural PDE solvers with competitive performance with respect to state-of-the-art baselines and provable data-efficiency guarantees. The proposed architecture has an exact HSS structure for the one-dimensional case, while for higher-dimensional problems uses an approximate outer product expansion. Our model demonstrates superior scalability with respect to problem dimensionality compared to baseline approaches for data-efficient learning in the elliptic setting. Notably, the Green Learning model could not be trained in 3D at high resolution. Furthermore, in lower dimensions (1 and 2), our model also exhibits better scalability with respect to training set size, again outperforming all the baselines on linear and nonlinear PDEs. Concerning limitations and future work, the proposed structured parameterization remains effective beyond the simple box-domain setting as shown in Table 2, but remains not an ideal choice for fully unstructured meshes, which might require an ad-hoc treatment. Moreover, we further remark that for higher-dimensional layers, the outer product parametrization is an approximation of the real higher-dimensional HSS structure. We also remark that, while the numerical approximation might be good in nonlinear regimes as discussed in Remark 2.4, the strongest version of the theoretical results Theorem 2.3 holds in the elliptic regime.

Acknowledgments

The authors acknowledge compute support from Modal. The work of FT is partially funded by the PRIN-MUR project MOLE code: 2022ZK5ME7 and by the PRIN-PNRR project FIN4GEO within the European Union's Next Generation EU framework, Mission 4, Component 2, CUP P2022BNB97. The work of E. Zangrando was funded by the MUR-PNRR project "Low-parametric machine learning".

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Andry, G., Lewin, S., Rozet, F., Rochman, O., Mangeleer, V., Pirlet, M., Faulx, E., Grégoire, M., and Louppe, G. Appa: Bending weather dynamics with latent diffusion models for global data assimilation, 2025. URL https://arxiv.org/abs/2504.18720.

Bebendorf, M. Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589, 2000.

Bebendorf, M. and Hackbusch, W. Existence of $\mathscr{H}$-matrix approximants to the inverse fe-matrix of elliptic operators with L^∞-coefficients. *Numerische Mathematik*, 95(1): 1–28, 2003.

Benner, P., Gugercin, S., and Willcox, K. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

Börm, S. *Efficient numerical methods for non-local operators: H2-matrix compression, algorithms and analysis*, volume 14. European Mathematical Society, 2010.

Börm, S. and Grasedyck, L. Hybrid cross approximation of integral operators. *Numerische Mathematik*, 101(2): 221–249, 2005.

Boullé, N. and Townsend, A. A mathematical guide to operator learning. In *Handbook of Numerical Analysis*, volume 25, pp. 83–125. Elsevier, 2024.

Boullé, N., Nakatsukasa, Y., and Townsend, A. Rational neural networks. *Advances in neural information processing systems*, 33:14243–14253, 2020.

Boullé, N., Halikias, D., and Townsend, A. Elliptic pde learning is provably data-efficient. *Proceedings of the National Academy of Sciences*, 120(39):e2303904120, 2023.

Brandstetter, J., Worrall, D. E., and Welling, M. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU.

Casulli, A. A., Kressner, D., and Robol, L. Computing functions of symmetric hierarchically semiseparable matrices. *SIAM Journal on Matrix Analysis and Applications*, 45 (4):2314–2338, 2024.

Chandrasekaran, S., Gu, M., and Pals, T. A fast ulv decomposition solver for hierarchically semiseparable representations. *SIAM Journal on Matrix Analysis and Applications*, 28(3):603–622, 2006. doi: 10.1137/S0895479803436652. URL https://doi.org/10.1137/S0895479803436652.

Chen, P. Y., Xiang, J., Cho, D. H., Chang, Y., Pershing, G. A., Maia, H. T., Chiaramonte, M. M., Carlberg, K. T., and Grinspun, E. CROM: Continuous reduced-order modeling of PDEs using implicit neural representations. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FUORz1tG8Og.

Choromanski, K. M., Sehanobish, A., Chowdhury, S. B. R., Lin, H., Dubey, K. A., Sarlos, T., and Chaturvedi, S. Fast tree-field integrators: From low displacement rank to topological transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Eok6HbcSRI.

Dao, T., Gu, A., Eichhorn, M., Rudra, A., and Ré, C. Learning fast algorithms for linear transforms using butterfly factorizations. In *International conference on machine learning*, pp. 1517–1527. PMLR, 2019.

Dao, T., Chen, B., Sohoni, N. S., Desai, A., Poli, M., Grogan, J., Liu, A., Rao, A., Rudra, A., and Ré, C. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pp. 4690–4721. PMLR, 2022.

Driscoll, T. A., Hale, N., and Trefethen, L. N. *Chebfun Guide*. Pafnuty Publications, 2014.

Fan, Y., Lin, L., Ying, L., and Zepeda-Núnez, L. A multiscale neural network based on hierarchical matrices. *Multiscale Modeling & Simulation*, 17(4):1189–1213, 2019a. doi: 10.1137/18M1203602. URL https://doi.org/10.1137/18M1203602.

Fan, Y., Lin, L., Ying, L., and Zepeda-Núnez, L. A multiscale neural network based on hierarchical matrices. *Multiscale Modeling & Simulation*, 17(4):1189–1213, 2019b. doi: 10.1137/18M1203602. URL https://doi.org/10.1137/18M1203602.

Flegar, G., Anzt, H., Cojean, T., and Quintana-Ortí, E. S. Adaptive precision block-jacobi for high performance preconditioning in the ginkgo linear algebra software. *ACM Trans. Math. Softw.*, 47(2), April 2021. ISSN 0098-3500. doi: 10.1145/3441850. URL https://doi.org/10.1145/3441850.

Fu, D., Arora, S., Grogan, J., Johnson, I., Eyuboglu, E. S., Thomas, A., Spector, B., Poli, M., Rudra, A., and Ré, C. Monarch mixer: A simple sub-quadratic gemm-based architecture. *Advances in Neural Information Processing Systems*, 36:77546–77603, 2023.

Geneva, N. and Zabaras, N. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403: 109056, 2020.

Geneva, N. and Zabaras, N. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, 2022.

Gillman, A., Young, P. M., and Martinsson, P.-G. A direct solver with o (n) complexity for integral equations on one-dimensional domains. *Frontiers of Mathematics in China*, 7(2):217–247, 2012.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Greenfeld, D., Galun, M., Basri, R., Yavneh, I., and Kimmel, R. Learning to optimize multigrid PDE solvers. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2415–2423. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/greenfeld19a.html.

Gupta, J. K. and Brandstetter, J. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

Hackbusch, W. -matrices. In *Hierarchical Matrices: Algorithms and Analysis*, pp. 203–240. Springer, 2015.

Hackbusch, W., Khoromskij, B. N., and Tyrtyshnikov, E. E. Hierarchical kronecker tensor-product approximations. *Journal of Numerical Mathematics*, 13(2):119–156, 2005. doi: doi:10.1515/1569395054012767. URL https://doi.org/10.1515/1569395054012767.

HAN, X., Gao, H., Pfaff, T., Wang, J.-X., and Liu, L. Predicting physics in mesh-reduced space with temporal attention. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=XctLdNfCmP.

Hao, Z., Wang, Z., Su, H., Ying, C., Dong, Y., Liu, S., Cheng, Z., Song, J., and Zhu, J. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pp. 12556–12569. PMLR, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hemmasian, A. and Barati Farimani, A. Reduced-order modeling of fluid flows with transformers. *Physics of Fluids*, 35(5):057114, 2023.

Hitchcock, F. L. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927. doi: https://doi.org/10.1002/sapm192761164. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164.

Holzschuh, B., Liu, Q., Kohl, G., and Thuerey, N. PDE-transformer: Efficient and versatile transformers for physics simulations. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=3BaJMRaPSx.

Huang, J., Yang, G., Wang, Z., and Park, J. J. Diffusion-pde: Generative pde-solving under partial observation. *Advances in Neural Information Processing Systems*, 37: 130291–130323, 2024.

Kaneda, A., Akar, O., Chen, J., Kala, V. A. T., Hyde, D., and Teran, J. A deep conjugate direction method for iteratively solving linear systems. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 15720–15736. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kaneda23a.html.

Kassam, A.-K. and Trefethen, L. N. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

Kissel, M. and Diepold, K. Structured matrices and their application in neural networks: A survey. *New Gen. Comput.*, 41(3):697–722, July 2023a. ISSN 0288-3635. doi: 10.1007/s00354-023-00226-1. URL https://doi.org/10.1007/s00354-023-00226-1.

Kissel, M. and Diepold, K. Structured matrices and their application in neural networks: A survey. 41(3):697–722, 2023b. doi: 10.1007/s00354-023-00226-1. URL https://doi.org/10.1007/s00354-023-00226-1.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. Speeding-up convolutional neural networks using fine-tuned cp-decomposition, 2015. URL https://arxiv.org/abs/1412.6553.

Levitt, J. and Martinsson, P.-G. Linear-complexity black-box randomized compression of rank-structured matrices. *SIAM Journal on Scientific Computing*, 46(3):A1747–A1763, 2024.

Li, Y., Chen, P. Y., Du, T., and Matusik, W. Learning preconditioners for conjugate gradient PDE solvers. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19425–19439. PMLR, 23–29 Jul 2023a. URL https://proceedings.mlr.press/v202/li23e.html.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Li, Z., Liu-Schiaffini, M., Kovachki, N., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Learning dissipative dynamics in chaotic systems. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 16768–16781, 2022.

Li, Z., Shu, D., and Farimani, A. B. Scalable transformer for PDE surrogate modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b. URL https://openreview.net/forum?id=djyn8Q0anK.

Li, Z., Patil, S., Ogoke, F., Shu, D., Zhen, W., Schneier, M., Buchanan Jr, J. R., and Farimani, A. B. Latent neural pde solver: A reduced-order modeling framework for partial differential equations. *Journal of Computational Physics*, 524:113705, 2025a.

Li, Z., Zhou, A., and Farimani, A. B. Generative latent neural pde solver using flow matching, 2025b. URL https://arxiv.org/abs/2503.22600.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021. doi: 10.1109/ICCV48922.2021.00986.

Luna, K., Klymko, K., and Blaschke, J. P. Accelerating gmres with deep learning in real-time, 2021. URL https://arxiv.org/abs/2103.10975.

Luo, J., Wang, J., Wang, H., huanshuo dong, Geng, Z., Chen, H., and Kuang, Y. Neural krylov iteration for accelerating linear system solving. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=cqfE9eYMdP.

Luz, I., Galun, M., Maron, H., Basri, R., and Yavneh, I. Learning algebraic multigrid using graph neural networks. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6489–6499. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/luz20a.html.

Martinsson, P.-G. and Rokhlin, V. A fast direct solver for boundary integral equations in two dimensions. *Journal of Computational Physics*, 205(1):1–23, 2005.

Maulik, R., Lusch, B., and Balaprakash, P. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3), 2021.

McCabe, M., Harrington, P., Subramanian, S., and Brown, J. Towards stability of autoregressive neural operators. *Transactions on Machine Learning Research*.

Munafo, R. Reaction-diffusion by the gray-scott model: Pearson's parametrization. *Unpublished material, available at http://mrob. com/pub/comp/xmorphia/index. html, under Creative Commons Attribution Non-Commercial*, 2, 2022.

Ohana, R., McCabe, M., Meyer, L. T., Morel, R., Agocs, F. J., Beneitez, M., Berger, M., Burkhart, B., Dalziel, S. B., Fielding, D. B., Fortunato, D., Goldberg, J. A., Hirashima, K., Jiang, Y.-F., Kerswell, R., Maddu, S., Miller, J. M., Mukhopadhyay, P., Nixon, S. S., Shen, J., Watteaux, R., Blancard, B. R.-S., Rozet, F., Parker, L. H., Cranmer, M., and Ho, S. The well: a large-scale collection of diverse physics simulations for machine learning. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=00Sx577BT3.

Raonic, B., Molinaro, R., Rohner, T., Mishra, S., and de Bezenac, E. Convolutional neural operators. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023. URL https://openreview.net/forum?id=GT8p40tiVlB.

Schotthöfer, S., Zangrando, E., Kusch, J., Ceruti, G., and Tudisco, F. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing*

Systems, 2022. URL https://openreview.net/forum?id=IILJ0KWZMy9.

Shysheya, A., Diaconu, C., Bergamin, F., Perdikaris, P., Hernández-Lobato, J. M., Turner, R., and Mathieu, E. On conditional diffusion models for pde simulations. *Advances in Neural Information Processing Systems*, 37: 23246–23300, 2024.

Stachenfeld, K., Fielding, D. B., Kochkov, D., Cranmer, M., Pfaff, T., Godwin, J., Cui, C., Ho, S., Battaglia, P., and Sanchez-Gonzalez, A. Learned simulators for turbulence. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=msRBojTz-Nh.

Stanaityte, R. *ILU and Machine Learning Based Preconditioning for the Discretized Incompressible Navier-Stokes Equations*. PhD thesis, University of Houston, 2020.

Taghibakhshi, A., MacLachlan, S., Olson, L., and West, M. Optimization-based algebraic multigrid coarsening using reinforcement learning. *Advances in neural information processing systems*, 34:12129–12140, 2021.

Thomas, A. T., Gu, A., Dao, T., Rudra, A., and Ré, C. Learning compressed transforms with low displacement rank, 2019. URL https://arxiv.org/abs/1810.02309.

Tyrtyshnikov, E. Mosaic-skeleton approximations. *CALCOLO*, 33(1):47–57, Jun 1996. ISSN 1126-5434. doi: 10.1007/BF02575706. URL https://doi.org/10.1007/BF02575706.

Walker, B. J., Townsend, A. K., Chudasama, A. K., and Krause, A. L. Visualpde: rapid interactive simulations of partial differential equations. *Bulletin of Mathematical Biology*, 85(11):113, 2023.

Wang, H., Agarwal, S., and Papailiopoulos, D. Pufferfish: Communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems*, 3:365–386, 2021.

Wiewel, S., Becher, M., and Thuerey, N. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, volume 38, pp. 71–82. Wiley Online Library, 2019.

Wu, H., Luo, H., Wang, H., Wang, J., and Long, M. Transolver: A fast transformer solver for pdes on general geometries. In *International Conference on Machine Learning*, 2024.

Yao, J., Mammadov, A., Berner, J., Kerrigan, G., Ye, J. C., Azizzadenesheli, K., and Anandkumar, A. Guided diffusion sampling on function spaces with applications to pdes. *arXiv preprint arXiv:2505.17004*, 2025.

Yeung, E., Kundu, S., and Hodas, N. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pp. 4832–4839. IEEE, 2019.

Zangrando, E., Schotthöfer, S., Ceruti, G., Kusch, J., and Tudisco, F. Geometry-aware training of factorized layers in tensor tucker format. *Advances in Neural Information Processing Systems*, 37:129743–129773, 2024.

Zhang, X., Wang, L., Helwig, J., Luo, Y., Fu, C., Xie, Y., Liu, M., Lin, Y., Xu, Z., Yan, K., Adams, K., Weiler, M., Li, X., Fu, T., Wang, Y., Strasser, A., Yu, H., Xie, Y., Fu, X., Xu, S., Liu, Y., Du, Y., Saxton, A., Ling, H., Lawrence, H., Stärk, H., Gui, S., Edwards, C., Gao, N., Ladera, A., Wu, T., Hofgard, E. F., Tehrani, A. M., Wang, R., Daigavane, A., Bohde, M., Kurtin, J., Huang, Q., Phung, T., Xu, M., Joshi, C. K., Mathis, S. V., Azizzadenesheli, K., Fang, A., Aspuru-Guzik, A., Bekkers, E., Bronstein, M., Zitnik, M., Anandkumar, A., Ermon, S., Liò, P., Yu, R., Günnemann, S., Leskovec, J., Ji, H., Sun, J., Barzilay, R., Jaakkola, T., Coley, C. W., Qian, X., Qian, X., Smidt, T., and Ji, S. Artificial intelligence for science in quantum, atomistic, and continuum systems, 2023.

Zhao, L., Liao, S., Wang, Y., Li, Z., Tang, J., and Yuan, B. Theoretical properties for neural networks with weight matrices of low displacement rank. In *international conference on machine learning*, pp. 4082–4090. PMLR, 2017.

A. Notation and useful definition

Definition A.1. (Modal product)

Let $\mathcal{Z} \in \mathbb{R}^{d_1 \times \cdots \times d_m}, W \in \mathbb{R}^{D_k \times d_k}$. We denote with $(Z \bigtimes_k W) \in \mathbb{R}^{d_1 \times \ldots d_{k-1} \times D_K \times d_{k+1} \times \cdots \times d_m}$ the $k-$th modal product of Z and W as

$$(\mathcal{Z} \bigtimes_k W)_{i_1,\ldots,i_n} := \sum_{j_k=1}^{d_k} \mathcal{Z}_{i_1,\ldots,i_{k-1},j_k,i_{k+1},\ldots,i_m} W_{i_k,j_k}$$

Definition A.2 (Cluster tree)**.** Let $d \in \mathbb{N}$. A **cluster tree** is a perfect binary tree $\mathcal{T}$ that defines subsets of indices obtained by recursively subdividing $I = [1, \ldots, d]$.

The root γ of the tree corresponds to the full index set I. Each non-leaf node τ is associated with a consecutive set of indices I_τ, and its two children, α and β, correspond to two consecutive subsets I_α and I_β such that $I_\tau = I_\alpha \cup I_\beta$ and $I_\alpha \cap I_\beta = \emptyset$.

If $\text{depth}(\mathcal{T}) = L$ and $d = 2^L k$ for $k \in \mathbb{N}$, we say that $\mathcal{T}$ is a **balanced cluster tree** if every non leaf node $\tau = [a, \ldots, b]$ is split exactly in the middle, i.e.

$$I_\alpha = [a, \tfrac{a+b}{2}], \qquad I_\beta = [\tfrac{a+b}{2} + 1, b].$$

Definition A.3. (η- strong admissibility condition (Hackbusch, 2015, Definition 4.9))

Let $(\Omega, \|\cdot\|)$ be a normed space. We say that two subdomains $X, Y \subseteq \Omega$ satisfy the η-strong admissibility condition if

$$\max(\text{diam(X)}, \text{diam(Y)}) \leq \eta \, \text{dist}(X, Y)$$

where

$$\text{diam}(X) := \sup_{x,x' \in X} \|x - x'\|, \quad \text{dist}(X, Y) := \max\{\sup_{x \in X} \inf_{y \in Y} \|x - y\|, \sup_{y \in Y} \inf_{x \in X} \|x - y\|\}$$

Definition A.4. (Asymptotic-smoothness (Hackbusch, 2015, Definition 4.14))

We say that $k : \Omega \to \mathbb{R}$ is asymptoticall smoth if there exist constants $C_0, C_1 > 0$ and $\mu \geq 0$ such that for all multi-indices α

$$|\partial^\alpha k(z)| \leq C_0 \, C_1^{|\alpha|} |\alpha|! \, |z|^{-\mu - |\alpha|}, \quad z \neq 0.$$

B. Proof of Theorem 2.2

Consider the power expansion

$$k(x) = \sum_{|\alpha| \leq r} \frac{(x - x_0)^\alpha}{\alpha!} \partial_x^\alpha k(x_0) + R_r, \quad k^{(r)}(x) := \sum_{|\alpha| \leq r} \frac{(x - x_0)^\alpha}{\alpha!} \partial_x^\alpha k(x_0)$$

and denote with diam_∞ the diameter in L^∞ norm in the sense of Definition A.3. Thanks to (Hackbusch, 2015, Lemma 4.29), we have the bound

$$\inf_{B:\text{rank(B)}=r} \|A - B\|_F \leq \|k - k^{(r)}\|_{L^2(\Omega_\tau - \Omega_{\tau'})} \|R_r\|_2 \|R_{r'}\|_2 := C \|k - k^{(r)}\|_{L^2(\Omega_\tau - \Omega_{\tau'})} \leq$$
$$\leq C |\Omega_\tau - \Omega_{\tau'}| \|k - k^{(r)}\|_{L^\infty(\Omega_\tau - \Omega_{\tau'})}$$

In order for the upper bound to be controlled by ε, we need $\|k - k^{(r)}\| \leq \frac{\varepsilon}{C}$. Thanks to (Hackbusch, 2015, Theorem 4.22) we have that for $m = r^{1/D}$

$$\|k - k^{(r)}\|_{L^\infty(\Omega_\tau - \Omega_{\tau'})} \leq c_1 \left(\frac{c_2' \text{diam}_\infty(\Omega_\tau)}{\text{dist}(\Omega_\tau, \Omega_{\tau'})} \right)^m \leq c_1 (c_2' \eta)^m$$

Therefore we finally have

$$\inf_{B:\text{rank(B)}=r} \|A - B\|_F \leq C c_1 (c_2' \eta)^{r^{1/D}} \leq \varepsilon$$

Therefore, for $r = O(\log(1/\varepsilon)^D)$ the inequality is satisfied.

C. Proof of Theorem 2.3

Proof. As a first step we show the existence of a parameter choice achieving exact realization of $\mathcal{G}^{-1}$. First, we note that if we assume, without loss of generality, that $\max_i r_i = r_1 \geq r$, and we choose $W_1 = \mathcal{G}^{-1}$, set W_j equal to the identity matrix for all $j \geq 2$, and take $\alpha_i = 1$ for every i. Then, for these choice of parameters, remembering that $\mathrm{LeakyReLU}_{\alpha=1} = \mathrm{id}$, we obtain $N_\theta(x) = W_\ell \dots W_1 x = \mathcal{G}^{-1}x$ for all x, therefore

$$\sup_{x \in \mathbb{R}^d} \|\mathcal{N}_\theta(x) - \mathcal{G}^{-1}x\|_2 = 0.$$

As a second step we show that any global minimizer must satisfy the same identity. In particular, in order to satisfy $\mathcal{L}_N(\theta) = 0$, it is necessary that $\alpha_i = 1$ for each i. Under this condition, $\mathcal{N}_\theta$ can be expressed as a product of HSS matrices associated with the same cluster tree $\mathcal{T}$, which implies that $\mathcal{N}_\theta$ is itself an HSS matrix associated with $\mathcal{T}$, with HSS rank $\sum_{i=1}^{\ell} r_i$. By the results in (Levitt & Martinsson, 2024), such an HSS matrix can be recovered uniquely with high probability from $\mathcal{O}\left(\sum_{i=1}^{\ell} r_i\right)$ observations. Therefore, all global minimizers must satisfy $W_\ell \dots W_1 = \mathcal{G}^{-1}$, $\alpha_i = 1$, and therefore the claim.

□

D. Forward pass

We assume, for simplicity, that the input and output vectors are of equal length.

Algorithm 1 Forward pass of the Neural-HSS linear layer

Require: $x \in \mathbb{R}^n$, cluster tree $\mathcal{T}$ of depth L, rank r
Ensure: $y \in \mathbb{R}^n$
if $\mathrm{depth}(\mathcal{T}) > 0$ **then**
  Split $x = [x_1, \dots, x_{2^L}]$, with $x_i \in \mathbb{R}^{|\tau_i|}$
  for $i = 1, \dots, 2^L$ **do**
    $z_i \leftarrow W_i^{(V)} x_i$
  end for
  $z \leftarrow [z_1, \dots, z_{2^L}]$
  $y \leftarrow \mathrm{HSSForward}(z, \mathcal{T}_{2r}^{(L-1)}, r)$
  Split $y = [y_1, \dots, y_{2^L}]$, with $y_i \in \mathbb{R}^r$
  for $i = 1, \dots, 2^L$ **do**
    $y_i \leftarrow W_i^{(U)} y_i$
    $y_i \leftarrow y_i + W_i^{(D)} x_i$
  end for
  return $[y_1, \dots, y_{2^L}]$
else
  return Wx
end if

Complexity analysis A key advantage of Neural-HSS lies in its efficient memory footprint and inference cost. Under the simplifying assumption that the number of leaf indices equals $2r$, the storage requirement of an HSS operator scales as $O(nr)$, and the cost of a matrix–vector multiplication is likewise $O(nr)$, as shown in (Chandrasekaran et al., 2006).

E. Lifting for system of PDEs

Several PDEs have input and output multiple channels: to adapt our architecture for these kinds of problems, one needs to specify a lifting and projection architectures as depicted in Figure 2. While multiple choices are possible, we employed the

simplest possible choice as a backbone and, as a final layer, a (low-rank) linear tensor map

$$\phi_{\mathcal{W}} : \mathbb{R}^{D_1 \times \cdots \times D_M} \to \mathbb{R}^{d_1 \times \cdots \times d_m}, \quad \phi_{\mathcal{W}}(\mathcal{Z})_{\underline{\alpha}} = \sum_{\underline{\beta}} \mathcal{W}_{\underline{\alpha},\underline{\beta}} \mathcal{Z}_{\underline{\beta}},$$

$$\mathcal{W} = \sum_{i=1}^{r} c_i u_i^{(1)} \otimes \cdots \otimes u_i^{(m)} \otimes v_i^{(1)} \otimes \cdots \otimes v_i^{(M)}, \quad c_i \in \mathbb{R}, u_i^{(j)} \in \mathbb{R}^{d_j}, v_i^{(j)} \in \mathbb{R}^{D_j},$$

where $c_i, u_i^{(j)}, v_i^{(j)}$ are learnable parameters. This allows us to easily extend Neural-HSS for a system of vector PDEs.

The overall model architecture, including a potential lifting layer for a system of PDEs, is depicted in Figure 2.

F. Data Specification

For complete transparency and full reproducibility of our results in this section, we provide all the details about the data generation. If we did not generate them, we also include the source and the procedure for their generation.

F.1. Datasets

Poisson equation. The n-dimensional Poisson equation

$$-\nabla \cdot \big(a(x)\nabla u\big) = f,$$

models diffusion or conduction in heterogeneous media, where $a(x) > 0$ is a spatially varying diffusivity and f an external source. Although linear, the discretized problem is usually ill-conditioned, making it difficult to solve numerically. The Poisson equation is used to model different physical phenomena such as heat conduction, porous flow, and electrostatics in nonhomogeneous materials. During our experiments, we kept $a = 1$ and we trained the models to represent the mapping $\mathcal{G}\colon f \mapsto u$. To assess the quality of the model, we used the relative L^2 error on the test set.

Heat equation. The heat equation is a fundamental linear partial differential equation that describes the diffusion of heat over time. In one spatial dimension, it is written as

$$u_t = \kappa u_{xx},$$

where κ denotes the thermal diffusivity. The equation smooths out spatial inhomogeneities by dissipating gradients, leading to a monotone decay of energy in the system. In our experiments, we fixed the diffusion coefficient at $\kappa = 0.0002$. The objective was to learn the temporal dynamics of the system by approximating the mapping from the current state to its future evolution, i.e., $\mathcal{G}\colon u_t \mapsto u_{t+\delta t}$. In our experiments, we fixed $\delta t = 0.8$. Model performance is evaluated by rolling it through time and calculating the L^2 trajectory error on the test set.

Helmholtz equation Efficient simulation of acoustic, electromagnetic, and elastic wave phenomena is essential in many engineering applications, including ultrasound tomography, wireless communication, and geophysical seismic imaging. When the problem is linear, the wave field u typically satisfies the Helmholtz equation

$$-\Delta u - \kappa^2 u = f,$$

where the wavenumber $\kappa = \frac{\omega}{c}$ denotes the ratio between the (constant) angular frequency ω and the propagation speed c. Numerical solution of the Helmholtz equation is particularly challenging due to its non-symmetric, indefinite operator and the highly oscillatory nature of its solutions.

Burgers equation. The Burgers equation is a fundamental nonlinear partial differential equation that captures the competing effects of nonlinear convection and viscous diffusion. In one spatial dimension, it takes the form

$$u_t + u u_x = \nu u_{xx},$$

where ν denotes the kinematic viscosity. The viscous term νu_{xx} regularises these shocks, but introduces thin internal layers that are numerically stiff. In our experiments, we fixed the viscousity coefficient at $\nu = 0.001$, and the objective was to learn the temporal dynamics of the system, that is, to approximate the mapping from the current state to its future evolution, more formally learn the mapping $\mathcal{G}\colon u_t \mapsto u_{t+\delta t}$, to evaluate the models we use the L^2 trajectory error on the test set.

Incompressible Navier-Stokes equation. We consider the incompressible Navier–Stokes equations expressed in terms of the vorticity $w = \nabla \times v$:

$$\partial_\tau w(c,\tau) + v(c,\tau) \cdot \nabla w(c,\tau) = \nu \Delta w(c,\tau) + q(c),$$
$$\nabla \cdot v(c,\tau) = 0.$$

Here $v(c,\tau)$ is the velocity field, $q(c)$ a forcing term, and ν the viscosity; in the dataset, the viscosity is fixed $\nu = 10^{-3}$. These equations describe nonlinear, incompressible flow and are widely used to study vorticity dynamics, turbulence, and coherent structures. We use the data from (Yao et al., 2025; Huang et al., 2024) and we conduct a similar series of experiments, we learned the mapping $\mathcal{G}\colon w_0 \mapsto w_{10}$ and the related inverse problem $\mathcal{G}\colon w_{10} \mapsto w_0$. The model performances are evaluated using the relative L^2 error on the test set.

Gray–Scott model The Gray–Scott equations describe a prototypical reaction–diffusion system involving two interacting chemical species, A and B, whose concentrations vary in space and time. They are given by

$$\begin{cases} \dfrac{\partial A}{\partial t} = \delta_A \Delta A - AB^2 + f(1-A), \\ \dfrac{\partial B}{\partial t} = \delta_B \Delta B + AB^2 - (f+k)B. \end{cases}$$

Here, the parameters f and k regulate the *feed* and *kill* rates, respectively: f controls the rate at which A is supplied to the system, while k controls the rate at which B is removed. The diffusion constants δ_A and δ_B determine the spread of the two species in space. We used the dataset provided by (Ohana et al., 2024), where each trajectory consists of 1000 steps. Rolling out a machine learning model over so many steps becomes impractical due to the accumulation of errors. Therefore, as suggested in (Ohana et al., 2024), a relevant task is to predict the final state in order to understand the long-term behavior of the two chemical species. To this end, we model the mapping $\mathcal{G}\colon (A_0, B_0) \mapsto (A_{1000}, B_{1000})$, and we also pass the model the input parameters f and k. Similarly to the incompressible Navier-Stokes, we also made an experiment for the inverse problem. The model performance is evaluated using the relative L^2 error on the test set.

F.2. Poisson Equation (1D, 2D & 3D)

F.2.1. 1D (Data Efficiency)

We generate datasets of solutions to the one-dimensional Poisson equation with homogeneous Dirichlet boundary conditions. The spatial domain is discretized uniformly with 1024 grid points over $x \in [0,1]$, with grid spacing $h = 1/1024$. For each sample, the right-hand side $f(x)$ is drawn from a truncated Fourier sine series with 20 random modes:

$$f(x) = \sum_{k=1}^{10} c_k \, \sin\big(2k\pi x\big), \qquad c_k \sim \mathcal{U}(0,1),$$

with $f(x)$ set to zero at the first two and last two grid points to enforce the boundary conditions. The Poisson problem is discretized using a fourth-order central finite difference scheme for the second derivative. The resulting banded linear system (with five diagonals) is solved efficiently using a direct banded solver. We generate 1,000 samples for training and 1,000 for testing. For training and testing our model, we downsample to 256.

F.2.2. 2D

We generate datasets of solutions to the two-dimensional Poisson equation with homogeneous Dirichlet boundary conditions. The spatial domain is discretized uniformly with 64×64 grid points over $(x,y) \in [0,1]^2$, with grid spacing $h = 1/64$. For each sample, the right-hand side $f(x,y)$ is drawn from a truncated Fourier sine series with up to 10 random modes in each direction:

$$f(x,y) = \sum_{k_x=1}^{10} \sum_{k_y=1}^{10} c_{k_x,k_y} \, \sin(2\pi k_x x) \, \sin(2\pi k_y y) \qquad c_{k_x,k_y} \sim \mathcal{U}(0,1),$$

where $k_x, k_y \in \{1, \ldots, 10\}$ are drawn uniformly at random for each sample. The forcing term $f(x,y)$ is set to zero along the boundary to enforce Dirichlet conditions.

The Poisson problem

$$-\Delta u(x, y) = f(x, y), \qquad u|_{\partial\Omega} = 0,$$

is discretized using a fourth-order finite difference scheme (nine-point Laplacian stencil), with a spatial resolution of 128×128. This resulting linear system is solved using a direct solver. We generate $4,200$ samples for training and 800 for testing, and we downsample the spatial resolution to 64×64.

We further generate 2500 samples (2250 for training and 250 for testing) of the two-dimensional Poisson equation with homogeneous Dirichlet boundary conditions on a L-shape domain. We use the same strategy as the two-dimensional Poisson equation on the box to generate the data. We only change the domain shape, removing from the top right corner of the box a square with side $1/2$.

F.2.3. 3D (Data Efficiency)

We generate datasets of solutions to the three-dimensional Poisson equation with homogeneous Dirichlet boundary conditions. The spatial domain is discretized uniformly with 128^3 grid points over $(x, y, z) \in [0, 1]^3$, with grid spacing $h = 1/n$ where $n \in \{128\}$. For each sample, the right-hand side $f(x, y, z)$ is constructed as a truncated Fourier sine series with randomly selected modes:

$$f(x, y, z) = \sum_{k_x=1}^{20} \sum_{k_y=1}^{20} \sum_{k_z=1}^{20} c_{k_x,k_y,k_z} \sin(k_x \pi x) \sin(k_y \pi y) \sin(k_z \pi z), \qquad c_{k_x,k_y,k_z} \sim \mathcal{U}(0, 1),$$

where $k_x, k_y, k_z \in \{3, \ldots, 23\}$ are drawn uniformly at random for each sample. The forcing term $f(x, y, z)$ is set to zero on the boundary of the domain to impose Dirichlet conditions.

The Poisson problem

$$-\Delta u(x, y, z) = f(x, y, z), \qquad u|_{\partial\Omega} = 0,$$

is discretized using a higher-order finite difference scheme based on a 19-point stencil, yielding a sparse linear system of size $n^3 \times n^3$. The system is solved using a sparse direct solver. We generate one dataset of 256 samples for training and 200 for testing.

F.3. Heat Equation

For the one-dimensional Heat equation, we generate 2000 trajectories (1800 for training and 200 for testing) with a time horizon $T = 8$ and time step $\delta t = 0.2$. The spatial domain is $X = [0, 1]$ with spatial resolution $\Delta x = 1/1023$ (1024 grid points) and homogeneous Dirichlet boundary conditions $u(0, t) = u(1, t) = 0$. The initial conditions are sampled from a truncated Fourier sine series with 10 modes and random coefficients $c_k \sim \mathcal{U}(0, 1)$:

$$u_0(x) = \sum_{k=1}^{10} 2 \cdot c_k \cdot \sin(k\pi x)$$

Each initial condition is normalized to have a maximum absolute value of 1. We use a second-order finite difference scheme for spatial discretization with diffusion coefficient $D = 0.0002$, and solve the resulting system of ODEs using the BDF (Backward Differentiation Formula) method with relative tolerance 10^{-4} and absolute tolerance 10^{-6}. For training and testing our model, we downsample to 256.

F.4. Helmholtz equation

We further generate 2500 samples (2250 for training and 250 for testing) of the two-dimensional Helmholtz equation with periodic boundary conditions on a box domain. We sample the forcing term from the same distribution as the one used in the Poisson equation. Moreover, we use the same grid resolution. We fix the $\kappa = 70$.

F.5. Burgers Equation

For the one-dimensional Burgers equation, we generate 2000 trajectories (1800 for training and 200 for testing) with a time horizon $T = 15.0$ and output time step $\delta t = 0.2$. The spatial domain is $X = [0, 1)$ with spatial resolution $\Delta x = 1/1024$

(1024 grid points) and Dirichlet boundary conditions. The initial conditions are sampled using a random Fourier series with 10 modes:

$$u_0(x) = \sum_{k=1}^{10} c_k \sin(2\pi(k+1)x)$$

where the coefficients $c_k \sim \mathcal{U}(-1, 1)$ are drawn from a uniform distribution. Each initial condition is normalized to have a maximum absolute value of 1. We employ second-order finite difference schemes for spatial discretization and solve the resulting system using the BDF (Backward Differentiation Formula) method with relative tolerances of 10^{-4} and absolute tolerances of 10^{-6}. For training and testing our model, we downsample to 256.

F.6. Gray–Scott model

We use the dataset from (Ohana et al., 2024), the follwoing are the datails for generating the data. Many numerical methods exist to simulate reaction–diffusion equations. If low-order finite differences are used, real-time simulations can be carried out using GPUs, with modern browser-based implementations readily available (Munafo, 2022; Walker et al., 2023). They choose to simulate with a high-order spectral method for accuracy and stability purposes. Specifically, they simulate equations (15)–(16) in two dimensions on the doubly periodic domain $[-1, 1]^2$ using a Fourier spectral method implemented in the MATLAB package Chebfun (Driscoll et al., 2014). The implicit–explicit exponential time-differencing fourth-order Runge–Kutta method (Kassam & Trefethen, 2005) is used to integrate this stiff PDE in time. The Fourier spectral method is applied in space, with linear diffusion terms treated implicitly and nonlinear reaction terms treated explicitly and evaluated pseudospectrally.

Simulations are performed using a 128×128 bivariate Fourier series over a time interval of 10,000 seconds, with a simulation time step size of 1 second. Snapshots are recorded every 10 time steps. The simulation trajectories are seeded with 200 different initial conditions: 100 random Fourier series and 100 randomly placed Gaussians. In all simulations, they set $\delta_A = 2 \times 10^{-5}$ and $\delta_B = 1 \times 10^{-5}$.

Pattern formation is then controlled by the choice of the "feed" and "kill" parameters f and k. They choose six different (f, k) pairs which result in six qualitatively different patterns, summarized in the following table:

Pattern	f	k
Gliders	0.014	0.054
Bubbles	0.098	0.057
Maze	0.029	0.057
Worms	0.058	0.065
Spirals	0.018	0.051
Spots	0.030	0.062

On 40 CPU cores, it takes 5.5 hours per set of parameters, for a total of 33 hours across all simulations.

F.7. Navier–Stokes Equation

We use the data provided by (Huang et al., 2024) for the two-dimensional Navier–Stokes equations. They follow this procedure to generate the data: The initial condition w_0 is sampled from a Gaussian random field

$$w_0 \sim \mathcal{N}\big(0,\ 7^{1.5}(-\Delta + 49I)^{-2.5}\big)\,.$$

The external forcing term is defined as

$$q(x_1, x_2) = \tfrac{1}{10}\big(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2))\big).$$

We solve the Navier–Stokes equations in the stream-function formulation using a pseudo-spectral method. Specifically, the equations are transformed into the spectral domain via Fourier transforms, the vorticity equation is advanced in time in spectral space, and inverse Fourier transforms are applied to compute nonlinear terms in the physical domain. The system is simulated for 1 second with 10 time steps, and the vorticity field w_t is stored at a spatial resolution of 128×128. In the dataset $\nu = 10^{-3}$ therefore the Reynolds number corresponding to $\mathrm{Re} = 1000$.

G. Metrics

Training Loss. As a training objective, we use the Mean Squared Error (MSE), defined as

$$\mathcal{L}(pred, target) = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} ||pred_b - target_b||_2^2$$

Evaluation Metrics. We assess the performance of our model using two different metrics. For steady-state problems or predictions at a specific time step, we employ the relative L^2 error, formally defined as

$$\mathcal{L}(pred, target) = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \frac{\|pred_b - target_b\|_2}{\|target_b\|_2}. \tag{2}$$

For temporal rollouts, in which the model is evaluated over a sequence of time steps, we adopt the L^2 trajectory error, mathematically expressed as

$$\mathcal{L}(pred, target) = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \|pred_b - target_b\|_2 \,. \tag{3}$$

H. Complete plot data efficiency Poisson 3D

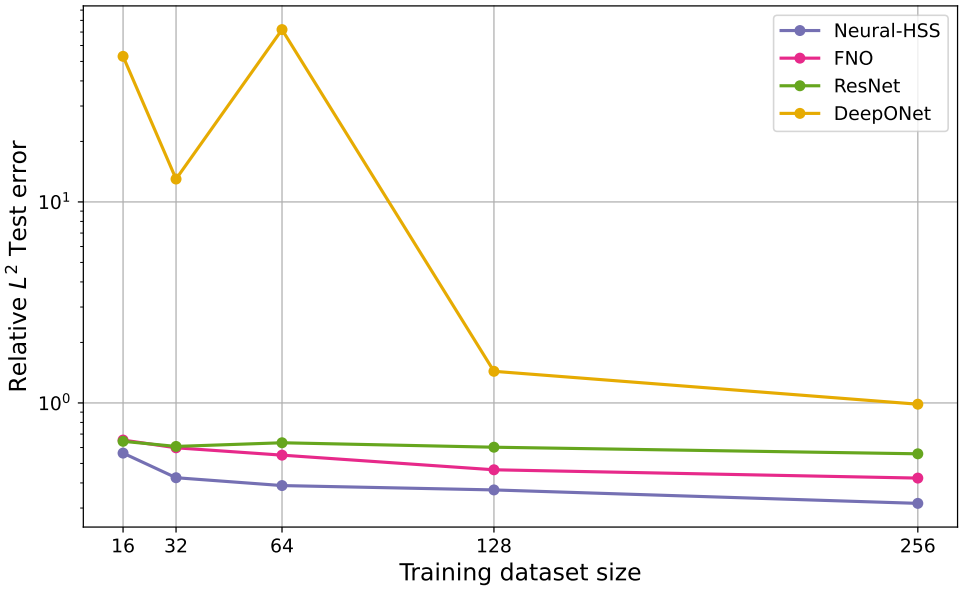

Figure 6. Train size vs relative test error for different models. The models are trained on a 3D Poisson equation

I. Ablation on hyperparameters

In this section we include an ablation study on the three main hyperparameters of the Neural-HSS layer, namely rank (r), outer rank r_{out} (for problems higher than $1D$), and levels t of the hierarchical structure. We considered a grid of $r \in \{2, 4, 8, 12, 16\}$, $r_{\text{out}} \in \{2, 4, 8, 16, 24, 32\}$, $t \in \{1, 2, 3\}$, and for each one of the triplets (r, r_{out}, t) of hyperparameters we trained a Neural-HSS model with those hyperparameters to predict the stationary state of the Gray-Scott problem. In Figure 7 we report the effect of the various couples of hyperparameters on the overall test performances. As we can observe in the plots, in the majority of the configurations, the model performs well, despite some small regions of high rank, outer rank, or levels in which the model started to overfit. Looking at the overall scale of the differences in performance, we can observe that the model is relatively stable with respect to all three hyperparameters, being the difference in terms of both test and train performance relatively small between the worst and best models observed.

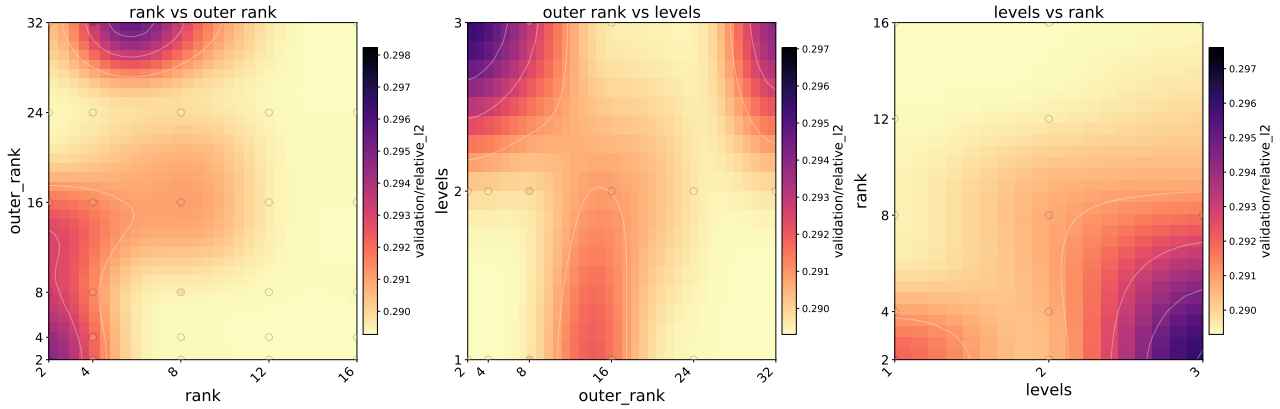

Figure 7. Ablation study on rank, outer rank, and levels for the Gray-Scott problem, where the color indicates the test error after training (rescaled from maximum to minimum observed, notice the scale in the colorbars).

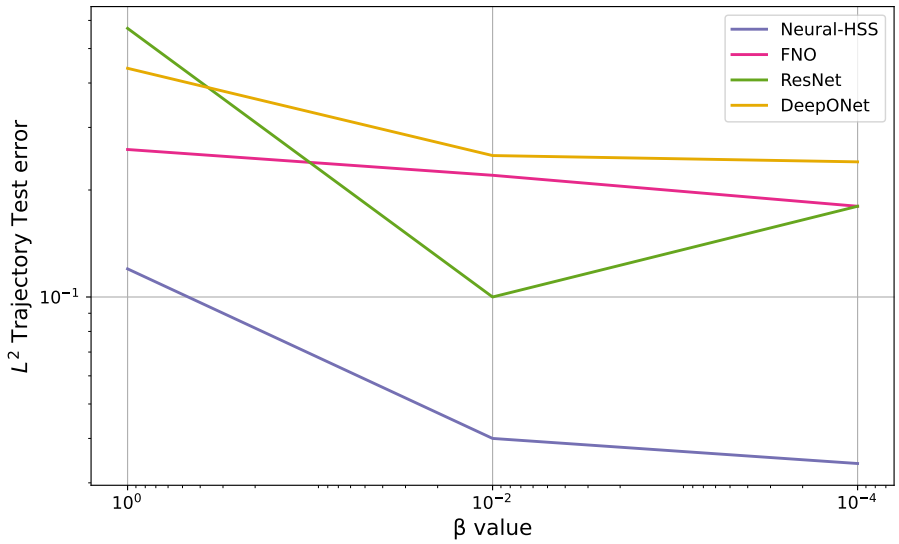

Figure 8. Trajectory test loss varying β

J. Additional Experiments on Burgers'

We run the following series of experiments using $\beta \in \{1, 10^{-2}, 10^{-4}\}$ on the modified Burgers' equation

$$u_t + \beta u u_x = \nu u_{xx}.$$

We adopt the same experimental setting used for the standard Burgers' equation, generating the data in the same manner and keeping the hyperparameters unchanged. When $\beta = 0$, the equation reduces to the heat equation, which is elliptic. The ResNet shows improved performance as β decreases from 1 to 10^{-2}, but its accuracy deteriorates at $\beta = 10^{-4}$. In contrast, the performance of FNO and DeepONet remains nearly constant across this range. The only model that benefits from the transition from the parabolic to the elliptic regime is Neural-HSS, highlighting its effectiveness in learning from data generated by elliptic PDEs.

K. Tuning DeepONet Hyperparameters

We study the effect of network depth and latent dimension on the performance of DeepONet for the heat equation benchmark. We consider architectures with 3 and 4 layers and latent dimensions in $\{32, 64, 128, 192\}$. In Table 3 we summarized the results.

	Latent dim 32	Latent dim 64	Latent dim 128	Latent dim 192
Depth 4	6×10^{-2}	3×10^{-2}	1.5×10^{-2}	1×10^{-2}
Depth 3	2×10^{0}	5×10^{-1}	1.5×10^{-1}	5×10^{-2}

Table 3. Hyperparameter tuning of DeepONet. Rows correspond to the number of layers, columns to the latent dimension.

L. Timing

In this section, we present the timing results. In particular, the setting is the same as Figures 10 and 11, for which we present the time of a training step for non-embedding layers with the same batch size used in the experiments (see Appendix M). In Figure 9, we present results in the same setting but just for the inference time.

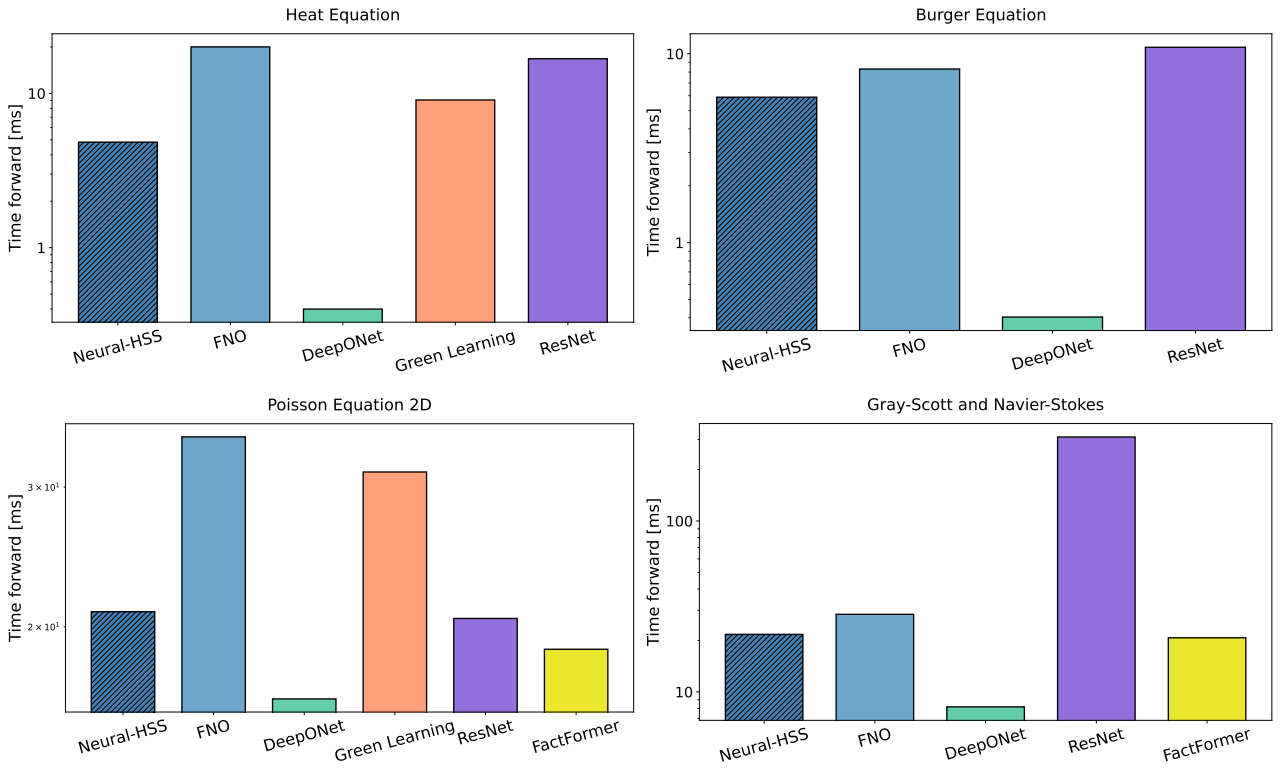

Figure 9. Timing of one forward pass, calculated on different datasets.

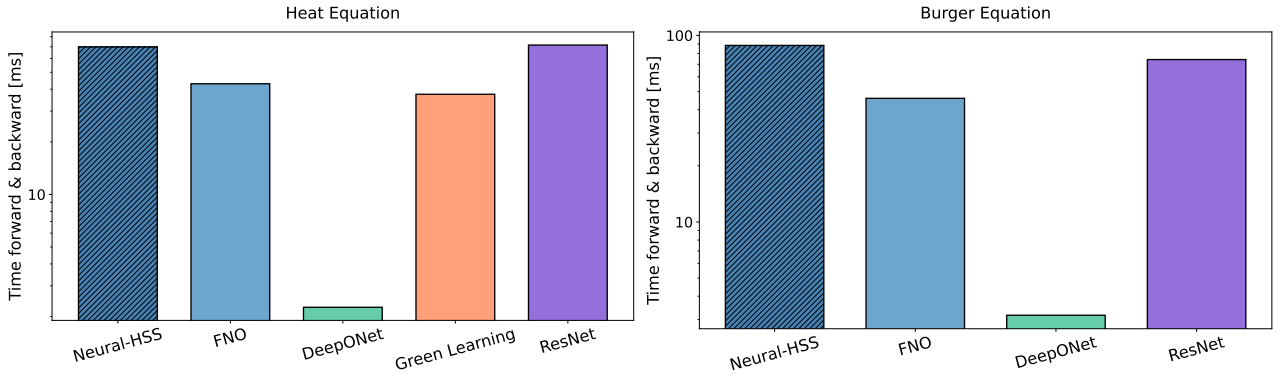

Figure 10. Timing of one forward+backward pass, calculated on two different datasets: **Left**: Heat equation. **Right**: Burgers' equation. We refer to Appendix L for inference only timings.

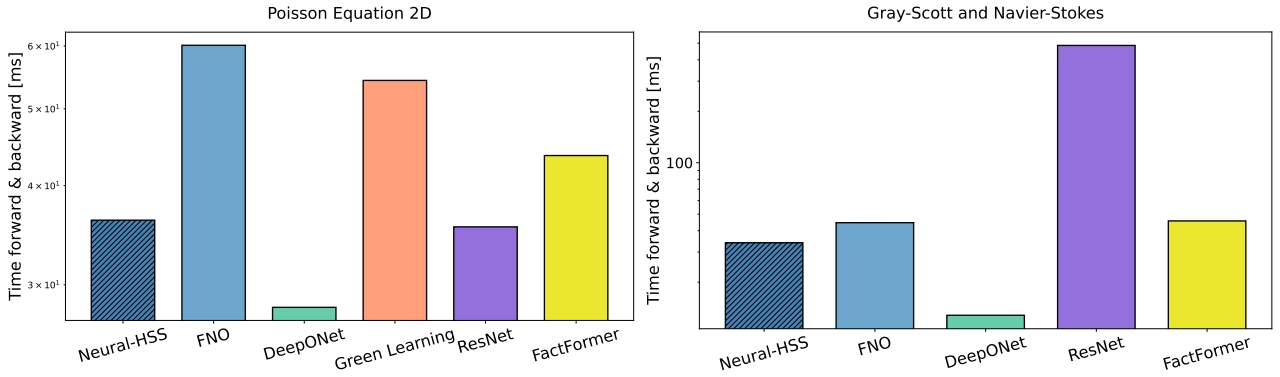

Figure 11. Timing of one forward+backward pass, calculated on two different datasets: **Left**: 2D Poisson Equation **Right**: Gray-Scott and Navier-Stokes.We refer to Appendix L for inference only timings.

M. Model & training details

We present in Tables 4 to 10 details on model and training parameters for all the datasets considered.

[1]The depth is the same for the Trunk network and the Branch network

[2](Boullé et al., 2020)

Configuration	Neural-HSS	FNO	ResNet	DeepONet	Green Learning
Depth	3	6	12	4[1]	6
Embedding Dimension	256	32	48	192	128
Activation	Trainable LeakyReLU	GeLU	GeLU	GeLU	Rational activation [2]
Levels	3	-	-	-	-
Rank	2	-	-	-	-
Modes	-	12	-	-	-
Peak learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Minimum learning rate	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Weight decay	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Learning rate schedule	Cosine	Cosine	Cosine	Cosine	Cosine
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
(β_1, β_2)	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$
Gradient clip norm	1	1	1	1	1
Epochs	1000	1000	1000	1000	1000
Batch size	256	256	256	256	256

Table 4. Models and training configuration for the **Heat Equation** and **Poisson Equation** with Neumann boundary conditions.

Configuration	Neural-HSS	FNO	ResNet	DeepONet
Depth	4	6	12	4
Embedding Dimension	1024	64	150	512
Activation	Trainable LeakyReLU	GeLU	GeLU	GeLU
Levels	3	-	-	-
Rank	32	-	-	-
Modes	-	32	-	-
Peak learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Minimum learning rate	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
Weight decay	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Learning rate schedule	Cosine	Cosine	Cosine	Cosine
Optimizer	AdamW	AdamW	AdamW	AdamW
(β_1, β_2)	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$
Gradient clip norm	1	1	1	1
Epochs	1500	1500	1500	1500
Batch size	256	256	256	256

Table 5. Models and training configuration for the **Burgers' Equation**.

Configuration	Neural-HSS	FNO	ResNet	DeepONet	Green Learning	FactFormer
Depth	3	4	4	4[3]	6	1
Embedding Dimension	64	16	64	64	64	384
Activation	Trainable LeakyReLU	GeLU	GeLU	GeLU	Rational activation[4]	GeLU
Levels	2	-	-	-	-	-
Rank	2	-	-	-	-	-
outer rank	8	-	-	-	-	-
Modes	-	8	-	-	-	-
Head dim	-	-	-	-	-	8
Head Num	-	-	-	-	-	1
Kern mult	-	-	-	-	-	1
Peak learning rate	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}
Minimum learning rate	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Weight decay	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Learning rate schedule	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
(β_1, β_2)	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$
Gradient clip norm	1	1	1	1	1	1
Epochs	500	500	500	500	500	500
Batch size	128	128	128	128	128	128

Table 6. Models and training configuration for the **2D Poisson Equation** and **2D Poisson Equation (L-shape domain)** .

Configuration	Neural-HSS	FNO	ResNet	DeepONet	Green Learning	
Depth	3	4	4	4[5]	6	1
Embedding Dimension	64	16	64	64	64	384
Activation	Trainable LeakyReLU	GeLU	GeLU	GeLU	Rational activation[6]	GeLU
Levels	2	-	-	-	-	-
Rank	2	-	-	-	-	-
outer rank	8	-	-	-	-	-
Modes	-	8	-	-	-	-
Head dim	-	-	-	-	-	8
Head Num	-	-	-	-	-	1
Kern mult	-	-	-	-	-	1
Peak learning rate	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}
Minimum learning rate	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Weight decay	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Learning rate schedule	Cosine	Cosine	Cosine	Cosine	Cosine	Cosine
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
(β_1, β_2)	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$
Gradient clip norm	1	1	1	1	1	1
Epochs	1000	1000	1000	1000	1000	1000
Batch size	64	64	64	64	64	64

Table 7. Models and training configuration for the **Helmholtz equation** .

Configuration	Neural-HSS	FNO	ResNet	DeepONet	FactFormer
Depth	3	4	4	4	1
Embedding Dimension	128×128	32	64	128	384
Activation	Trainable LeakyReLU	GeLU	GeLU	GeLU	GeLU
Levels	2	-	-	-	-
Rank	8	-	-	-	-
Outer Rank	8	-	-	-	-
Modes	-	32	-	-	-
Head dim	-	-	-	-	8
Head Num	-	-	-	-	1
Kern mult	-	-	-	-	1
Peak learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Minimum learning rate	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Weight decay	0	0	0	0	0
Learning rate schedule	Cosine	Cosine	Cosine	Cosine	Cosine
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
(β_1, β_2)	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$
Gradient clip norm	1	1	1	1	1
Epochs	10	10	10	10	10
Batch size	64	64	64	64	64

Table 8. Models and training configuration for the **Incompressible Navier-Stokes and Gray-Scott**.

Configuration	Neural-HSS	FNO	ResNet	DeepONet
Depth	1	2	2	4
Embedding Dimension	$128 \times 128 \times 128$	14	48	128
Activation	Trainable LeakyReLU	GeLU	GeLU	GeLU
Levels	2	-	-	-
Rank	4	-	-	-
Outer Rank	2	-	-	-
Modes	-	8	-	-
Peak learning rate	5×10^{-3}	5×10^{-3}	1×10^{-4}	1×10^{-3}
Minimum learning rate	1×10^{-3}	1×10^{-3}	1×10^{-5}	1×10^{-5}
Weight decay	0	0	0	0
Learning rate schedule	Cosine	Cosine	Cosine	Cosine
Optimizer	AdamW	AdamW	AdamW	AdamW
(β_1, β_2)	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$
Gradient clip norm	1	1	1	1
Trainig steps	16k	16k	16k	16k
Batch size	16	16	16	16

Table 9. Models and training configuration for the **data efficiency for 3D Poisson Equation**. We report the training steps since for each training size we match the training steps at 16k.

Configuration	Neural-HSS	FNO	ResNet	DeepONet	Green Learning
Depth	3	6	12	4[7]	6
Embedding Dimension	256	32	48	192	128
Activation	Trainable LeakyReLU	GeLU	GeLU	GeLU	Rational activation [8]
Levels	3	-	-	-	-
Rank	2	-	-	-	-
Modes	-	12	-	-	-
Peak learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Minimum learning rate	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Weight decay	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Learning rate schedule	Cosine	Cosine	Cosine	Cosine	Cosine
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
(β_1, β_2)	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$	$(0.9, 0.99)$
Gradient clip norm	1	1	1	1	1
Epochs	500	500	500	500	500
Batch size	256	256	256	256	256

Table 10. Models and training configuration for the **data efficiency for 1D Poisson Equation**.