Solving High-Dimensional PDEs with Latent Spectral Models

Haixu Wu [1] **Tengge Hu** [1] **Huakun Luo** [1] **Jianmin Wang** [1] **Mingsheng Long** [1]

Abstract

Deep models have achieved impressive progress in solving partial differential equations (PDEs). A burgeoning paradigm is learning neural operators to approximate the input-output mappings of PDEs. While previous deep models have explored the multiscale architectures and various operator designs, they are limited to learning the operators as a whole in the coordinate space. In real physical science problems, PDEs are complex coupled equations with numerical solvers relying on discretization into high-dimensional coordinate space, which cannot be precisely approximated by a single operator nor efficiently learned due to the curse of dimensionality. We present Latent Spectral Models (LSM) toward an efficient and precise solver for high-dimensional PDEs. Going beyond the coordinate space, LSM enables an attention-based *hierarchical projection network* to reduce the high-dimensional data into a compact latent space in linear time. Inspired by classical spectral methods in numerical analysis, we design a *neural spectral block* to solve PDEs in the latent space that approximates complex input-output mappings via learning multiple basis operators, enjoying nice theoretical guarantees for convergence and approximation. Experimentally, LSM achieves consistent state-of-the-art and yields a relative gain of 11.5% averaged on seven benchmarks covering both solid and fluid physics. Code is available at https://github.com/thuml/Latent-Spectral-Models.

1. Introduction

Extensive real-world phenomena are governed by underlying partial differential equations (PDEs), such as turbulence, atmospheric circulation and stress of deformed materials (Wazwaz, 2002; Roubíček, 2013). Thus, solving PDEs is the shared foundation problem among many scientific and engineering areas and can further benefit essential real-world applications, like airflow modeling for airfoil design, atmospheric simulation for weather forecasting and stress test in civil engineering. Recently, deep models have achieved great progress in various tasks (He et al., 2016; Devlin et al., 2019; Liu et al., 2021). In view of the great nonlinear modeling capability of deep models, they have been widely used to solve PDEs by approximating the mapping between input-output pairs in PDE-governed tasks (Hao et al., 2022; Raissi et al., 2019; Lu et al., 2021; Li et al., 2021; Cao, 2021).

Concretely, in real-world applications, PDEs are usually discretized into high-dimensional coordinate spaces, such as point cloud, mesh and grid. For example, as shown in Figure 1(c), the fluid simulation task governed by spatiotemporal continuous Navier-Stokes equations (Temam, 2001) can be discretized into successive grid frames, where the dimension of coordinate space is equal to the number of pixels in all frames. However, this high-dimensionality will bring thorny challenges to the solving process. Firstly, according to the phenomenon of curse of dimensionality (Trunk, 1979; Han et al., 2017), the solving process will cause huge computation cost in the high-dimensional space. Secondly, due to intricate interactions among multiple physical variates of coupled equations in high-dimensional coordinate space, the input-output mappings will be too complex to be approximated by a rough deep model (Trunk, 1979; Karniadakis et al., 2021). Thus, *how to efficiently and precisely approximate complex mappings between high-dimensional input-output pairs* is the key problem to solving PDEs.

In previous works, the well-acknowledged paradigm is to learn neural operators to approximate the complex input-output mappings (Li et al., 2020; Lu et al., 2021). Extensive designs of operators have been proposed, such as approximating the integral operator in Fourier space (Li et al., 2021; Tran et al., 2023), capturing the global information by Transformers (Vaswani et al., 2017; Cao, 2021; Liu et al., 2022) and etc. Note that all these designs attempt to learn the operator as a whole to approximate input-output mappings. However, in high-dimensional space, the input-output mappings can be too complex to be covered by a single operator, which may suffer from optimization problems and limited performance. Besides, some works introduce the

[1]School of Software, BNRist, Tsinghua University. Haixu Wu <whx20@mails.tsinghua.edu.cn>. Correspondence to: Mingsheng Long <mingsheng@tsinghua.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

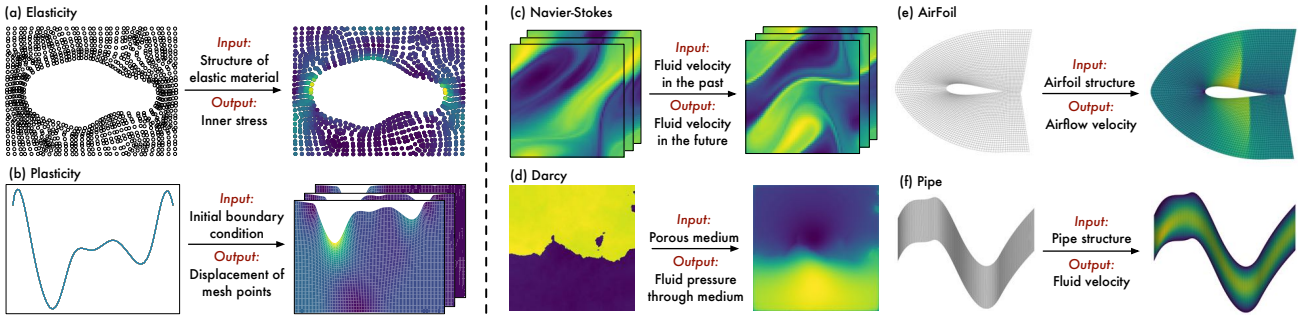

Figure 1. Examples of PDE-governed tasks, including solid (left) and fluid (right) physics, whose solving processes are approximating complex input-output mappings in discretized high-dimensional coordinate spaces. All the tasks are covered in our experiments.

multiscale architecture into deep models (Wen et al., 2021; Rahman et al., 2022). Although they can downsample the input into various scales, they are still limited to learning operators in the coordinate space, thereby still undergoing high-dimensionality challenges to some extent.

To tackle the above challenges, we start from the inherent property of PDE-governed tasks. It is observed that all their inputs and outputs follow certain PDE constraints, indicating that these high-dimensional data can be projected to a more compact latent space. Based on this insight, we propose the Latent Spectral Models (LSM) with a *hierarchical projection network*. Different from solely downsampling data like previous methods, by leveraging the attention mechanism with latent tokens as physical prompts, our projection network can reduce the high-dimensional data into compact latent space in linear time, which will also highlight the physics properties and remove the redundant coordinate information. Benefiting from this projection, LSM can get rid of the unwieldy coordinate space and solve PDEs in the latent space. Besides, to tackle the complex mappings, inspired by the classical spectral methods in numerical analysis (Gottlieb & Orszag, 1977), we present the *neural spectral block* to decompose complex nonlinear mappings into multiple basis operators, which also holds the universal approximation capacity with theoretical guarantees. Experimentally, LSM achieves consistent state-of-the-art on seven well-established benchmarks and also presents good transferability between PDEs of different conditions. Our contributions are summarized as follows:

- Instead of solving PDEs in the coordinate space, we present the LSM with a hierarchical projection network, which can reduce high-dimensional data into compact latent space with linear complexity.
- Inspired by spectral methods, we propose the *neural spectral block* to tackle complex mappings by learning multiple basis operators, which holds the universal approximation capacity under theoretical guarantees.
- LSM achieves an 11.5% relative error reduction with respect to the previous state-of-the-art models averaged from seven PDE-solving benchmarks, covering representative PDEs in both solid and fluid physics, and also presents favorable efficiency and transferability.

2. Preliminaries

2.1. Spectral Methods

Spectral methods are widely acknowledged in applied mathematics and scientific computing in solving PDEs numerically (Gottlieb & Orszag, 1977; Fornberg, 1998; Kopriva, 2009). The key idea is to approximate the solution f of a certain PDE as a finite sum of N orthogonal basis functions $\{f_1, f_2, \cdots, f_N\}$. Concretely, the approximation solution f^N can be formulized as follows:

$$f \approx f^N = \sum_{i=1}^{N} w_i f_i, \tag{1}$$

where N is the hyperparameter and w_i is the coefficient for $f_i, i \in \{1, \cdots, N\}$. With the above approximation, the solving process can be simplified as optimizing coefficients $\{w_1, w_2, \cdots, w_N\}$ to make f^N satisfy the PDE better. The spectral methods hold nice approximation and convergence properties in solving PDEs (Gottlieb & Orszag, 1977).

2.2. Deep Models for PDEs

Due to the immense importance in extensive scientific and engineering areas, solving PDEs has attached great interest. Since it is usually impossible to work out explicit formulas for PDE solutions, many numerical methods have been explored (Ŝolín, 2005; Grossmann et al., 2007). However, these classical methods need to recalculate for different instances, such as different initial velocity fields in fluid simulation or different meshes in solid stress estimation. Besides, these classical methods also suffer from poor computation efficiency, especially in processing the high-dimensional data. Recently, various deep models have been developed. The mainstream works can be roughly categorized into equation-constraint and operator-learning methods.

Equation-constraint methods. This category of works directly parameterizes the PDE solution as a deep model and formalizes equation constraints, e.g. the PDEs and their corresponding initial and boundary conditions, as the objective function (Weinan & Yu, 2017; Raissi et al., 2019; Wang et al., 2020a;b). By doing this, they can directly obtain the solution for a certain PDE through model optimization. However, these methods require the exact formalization of underlying PDEs, which is hard to acquire in real-world applications. Thus, instead of the equation-constraint methods, this paper focuses on the operator-learning paradigm, which does not need explicit PDE formalizations.

Operator-learning methods. This paradigm attempts to present deep models with novel architectures to approximate the mapping between input-output pairs, such as from past observations of fluid velocity to future prediction or from the structure of elastic material to inner stress. Technically, by rewriting inputs and outputs as functions w.r.t. coordinates, the solving process can be formulized as learning operators between input-output Banach spaces.

Some previous works have presented various designs for operators. Lu et al. present the DeepONet as a branch-trunk architecture derived from the universal approximation theorem (Chen & Chen, 1995). FNO (Li et al., 2021) adopts the linear transformation in the Fourier domain to approximate the integral operator. Further, geo-FNO (Li et al., 2022) is proposed to handle tasks with complex geometrics (e.g. point cloud) by transforming the data into and back from a latent uniform mesh. F-FNO (Tran et al., 2023) improves FNO with the separable Fourier transform and residual connection. KNO (Xiong et al., 2023a) enhances the temporal dynamic modeling of FNO based on the Koopman theory (Brunton et al., 2021). Besides, MWT (Gupta et al., 2021) introduces the multiwavelet-based operator, which can capture complex dependencies at various scales. SNO (Fanaskov & Oseledets, 2022) reformulates the input and output functions as coefficients of basis functions and adopts the neural network to learn the mapping between coefficients. Recently, Cao explored the self-attention mechanism (Vaswani et al., 2017) and presented a Galerkin-type attention with linear complexity for solving PDEs. Unlike previous methods, instead of approximating mappings with a single operator, LSM decomposes the complex nonlinear operator into several basis operators by the neural spectral block, thereby benefiting complex PDEs solving.

Other works attempt to enhance deep models with the multiscale architecture. U-FNO (Wen et al., 2021) and U-NO (Rahman et al., 2022) integrate U-Net (Ronneberger et al., 2015) and FNO to empower the model with multiscale processing capability. HT-Net (Liu et al., 2022) incorporates the advanced Transformers (Vaswani et al., 2017; Liu et al., 2021) into a hierarchical framework to capture high-frequency components in PDEs. In contrast to previous methods, LSM presents an attention-based hierarchical projection network to project high-dimensional data into compact latent space, which is free from the redundant coordinate space and focuses on the essential physical information.

3. Latent Spectral Models

As aforementioned, we highlight the difficulties of solving high-dimensional PDEs as huge computation costs and complex input-output mappings. To tackle these challenges, we present LSM with a *hierarchical projection network* to project the high-dimensional data into compact latent space with favorable efficiency. Further, inspired by spectral methods, we design the *neural spectral block* to approximate complex mappings with multiple basis operators, which holds nice approximation and convergence properties.

Problem setup. For a PDE-governed task, given the coordinates in a bounded open set $\mathcal{D} \subset \mathbb{R}^d$, both inputs and outputs can be rewritten as functions w.r.t. coordinates, which are in the Banach spaces $\mathcal{X} = \mathcal{X}(\mathcal{D}; \mathbb{R}^{d_{\boldsymbol{x}}})$ and $\mathcal{Y} = \mathcal{Y}(\mathcal{D}; \mathbb{R}^{d_{\boldsymbol{y}}})$ respectively (Lu et al., 2021; Li et al., 2021). $\mathbb{R}^{d_{\boldsymbol{x}}}$ and $\mathbb{R}^{d_{\boldsymbol{y}}}$ are the range of input and output functions. For example, as Figure 2 shows, both inputs and outputs are in the regular grid. Thus, $\mathcal{D}$ is a finite set of grid points within the rectangle area in $\mathbb{R}^2$. For each coordinate $\mathbf{s} \in \mathcal{D}$, $\boldsymbol{x}(\mathbf{s}) \in \mathbb{R}^{d_{\boldsymbol{x}}}$ and $\boldsymbol{y}(\mathbf{s}) \in \mathbb{R}^{d_{\boldsymbol{y}}}$ represent the input and output function values at position $\mathbf{s}$, corresponding to pixel values in the case of Figure 2. With the above formalization, the solving process is to approximate the optimal operator $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ with deep model $\mathcal{F}_\theta$, which is learned from observed samples $\{(\boldsymbol{x}, \boldsymbol{y})\}$ and $\theta \in \Theta$ is the parameter set.

Overall framework. Instead of directly solving PDEs in high-dimensional coordinate space like previous methods, by introducing latent space, LSM can get rid of redundant coordinate information. As shown in Figure 2, LSM breaks the PDE solving process into three modules as follows:

$$\mathcal{F}_\theta = \mathcal{F}_{\theta_{\text{LatentToCoord}}} \circ \mathcal{F}_{\theta_{\text{Solve}}} \circ \mathcal{F}_{\theta_{\text{CoordToLatent}}}, \tag{2}$$

where $\circ$ denotes the operator composition. In LSM, the hierarchical projection network provides an attention-based instantiation for $\mathcal{F}_{\theta_{\text{CoordToLatent}}} : \mathcal{X} \to \mathcal{T}_{\mathcal{X}}$ and $\mathcal{F}_{\theta_{\text{LatentToCoord}}} : \mathcal{T}_{\mathcal{Y}} \to \mathcal{Y}$, where $\mathcal{T}_{\mathcal{X}}(\mathcal{D}; \mathbb{R}^{d_{\text{latent}}})$ and $\mathcal{T}_{\mathcal{Y}}(\mathcal{D}; \mathbb{R}^{d_{\text{latent}}})$ are the latent input-output Banach spaces respectively. And the neural spectral block instantiates $\mathcal{F}_{\theta_{\text{Solve}}} : \mathcal{T}_{\mathcal{X}} \to \mathcal{T}_{\mathcal{Y}}$ to approximate complex nonlinear mappings in the latent space.

3.1. Hierarchical Projection Network

To make the solving process free from unwieldy coordinate space, we present the hierarchical projection network by embedding attention-based projectors in a patchified multi-

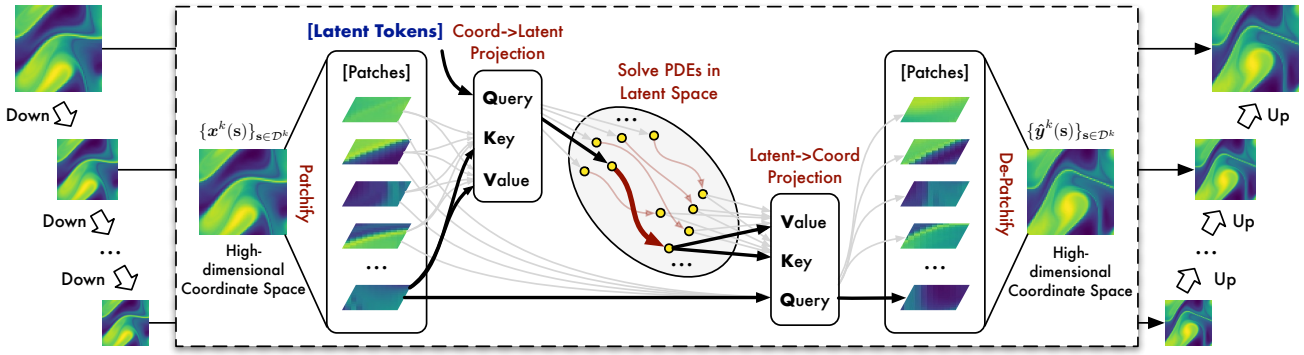

Figure 2. Overview of LSM. The solving process is applied to each patch of each scale with three successive modules: projecting coordinate space into latent space (CoordToLatent), solving PDEs in latent space and projecting back to coordinate space (LatentToCoord).

scale architecture, which can reduce high-dimensional data into compact latent space for efficient PDE solving.

Attention-based projectors. If we directly apply self-attention (Vaswani et al., 2017) among observations at multiple coordinates, the results will still be in high-dimensional coordinate space. Thus, to extract essential physical information of PDEs from redundant high-dimensional data, we propose attention-based projectors with latent tokens. The latent tokens are shared among all input-output pairs, initialized as learnable model parameters, and optimized to cover the common properties of data, namely PDE constraints, thereby providing physical prompts for projection.

Concretely, given the coordinates set $\mathcal{D}$ and the deep representations of inputs $\{\boldsymbol{x}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}}, \boldsymbol{x}(\mathbf{s}) \in \mathbb{R}^{1\times d_{\text{model}}}$, we will randomly initialize C latent tokens $\{\mathbf{T}_i\}_{i=1}^C, \mathbf{T}_i \in \mathbb{R}^{1\times d_{\text{latent}}}$ to provide physical prompts. As shown in Figure 2, we adopt the latent tokens as queries and deep representations as keys and values in the attention mechanism. The residual connection is also used to ease model optimization (He et al., 2016). This process can be formulized as:

$$\mathbf{T}_{\boldsymbol{x},i} = \mathbf{T}_i + \sum_{\mathbf{s}\in\mathcal{D}} \frac{\text{Sim}\left(\mathbf{T}_i, \boldsymbol{x}(\mathbf{s})\mathbf{W}_{\text{K}}\right)}{\sum_{\mathbf{s}'\in\mathcal{D}} \text{Sim}\left(\mathbf{T}_i, \boldsymbol{x}(\mathbf{s}')\mathbf{W}_{\text{K}}\right)} \left(\boldsymbol{x}(\mathbf{s})\mathbf{W}_{\text{V}}\right), \tag{3}$$

where $i \in \{1, \cdots, C\}$ and $\mathbf{W}_{\text{K}}, \mathbf{W}_{\text{V}} \in \mathbb{R}^{d_{\text{model}}\times d_{\text{latent}}}$ are linear layers for keys and values. $\text{Sim}\left(\mathbf{T}_i, \boldsymbol{x}(\mathbf{s})\mathbf{W}_{\text{K}}\right) = \exp\left(\mathbf{T}_i \left(\boldsymbol{x}(\mathbf{s})\mathbf{W}_{\text{K}}\right)^{\mathsf{T}}\right)$ is for the similarity calculation. Under the physical prompts of learned latent tokens $\{\mathbf{T}_i\}_{i=1}^C$, the deep representations $\{\boldsymbol{x}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}}$ in the high-dimensional coordinate space are projected to C tokens $\{\mathbf{T}_{\boldsymbol{x},i}\}_{i=1}^C$ in latent space, where the latter is free from redundant coordinate information. To simplify notations, we summarize Eq. (3) as $\{\mathbf{T}_{\boldsymbol{x},i}\}_{i=1}^C = \text{CoordToLatent}(\{\mathbf{T}_i\}_{i=1}^C, \{\boldsymbol{x}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}})$.

After solving PDEs in latent space by the neural spectral block, the latent input tokens $\{\mathbf{T}_{\boldsymbol{x},i}\}_{i=1}^C$ are mapped to the latent output tokens $\{\mathbf{T}_{\boldsymbol{y},i}\}_{i=1}^C$. We summarize the solving process in latent space as $\{\mathbf{T}_{\boldsymbol{y},i}\}_{i=1}^C = \text{Solve}(\{\mathbf{T}_{\boldsymbol{x},i}\}_{i=1}^C)$.

Finally, we need to project latent output tokens back to high-dimensional coordinate space as the final output. Similar to Eq. (3), by taking input representations as queries to provide coordinate information and latent output tokens as keys and values, this process can be formulized as follows:

$$\widehat{\boldsymbol{y}}(\mathbf{s}) = \boldsymbol{x}(\mathbf{s}) + \sum_{i=1}^{C} \frac{\text{Sim}\left(\boldsymbol{x}(\mathbf{s}), \mathbf{T}_{\boldsymbol{y},i}\mathbf{W}'_{\text{K}}\right)}{\sum_{i'=1}^{C} \text{Sim}\left(\boldsymbol{x}(\mathbf{s}), \mathbf{T}_{\boldsymbol{y},i'}\mathbf{W}'_{\text{K}}\right)} \left(\mathbf{T}_{\boldsymbol{y},i}\mathbf{W}'_{\text{V}}\right), \tag{4}$$

where $\mathbf{s} \in \mathcal{D}$ and $\mathbf{W}'_{\text{K}}, \mathbf{W}'_{\text{V}} \in \mathbb{R}^{d_{\text{latent}}\times d_{\text{model}}}$ are linear layers for keys and values. Eq. (4) is summarized as $\{\widehat{\boldsymbol{y}}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}} = \text{LatentToCoord}(\{\boldsymbol{x}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}}, \{\mathbf{T}_{\boldsymbol{y},i}\}_{i=1}^C)$. The computation complexity of projectors in Eq. (3) and (4) is linear w.r.t. the size of coordinate set $\mathcal{D}$, namely $\mathcal{O}(|\mathcal{D}|)$.

Patchified multiscale architecture. It is notable that PDEs always present different physical states according to the observed scales and regions (Karniadakis et al., 2021). For example, in turbulent flow, unsteady vortices appear of many sizes, which interact with each other, leading to a very complex phenomenon (Morrison, 2013). To fit the intrinsic multiscale property and complex interactions of PDEs, we present a patchified multiscale architecture and attempt to solve PDEs in different regions and scales.

Technically, for the raw inputs in $\mathbb{R}^{d_{\boldsymbol{x}}}$, we firstly map them into deep representations $\{\boldsymbol{x}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}}, \boldsymbol{x}(\mathbf{s}) \in \mathbb{R}^{1\times d_{\text{model}}}$ by the linear layer with parameters in $\mathbb{R}^{d_{\boldsymbol{x}}\times d_{\text{model}}}$. As shown in Figure 2, we employ the parameterized downsample layer to obtain deep representations $\{\{\boldsymbol{x}^k(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^k}\}_{k=1}^K$ in K scales by aggregating the local observations with learnable parameters, where $\boldsymbol{x}^k(\mathbf{s}) \in \mathbb{R}^{1\times d^k_{\text{model}}}$ and $\{\boldsymbol{x}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}} = \{\boldsymbol{x}^1(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^1}$ is in the finest resolution. For the k-th scale, we further adopt the patchify operation (Dosovitskiy et al., 2021) to split the coordinate set $\mathcal{D}^k$ into P_k nonoverlapping patches $\{\mathcal{D}^k_j\}_{j=1}^{P_k}$ for different regions, where $\mathcal{D}^k_j \subset \mathcal{D}^k$ denotes coordinate set of the j-th patch. More details about downsample and patchify operations are in Appendix I.

By randomly initializing $\left\{\{\mathbf{T}^k_i\}_{i=1}^C\right\}_{k=1}^K$, $\mathbf{T}^k_i \in \mathbb{R}^{1\times d^k_{\text{latent}}}$

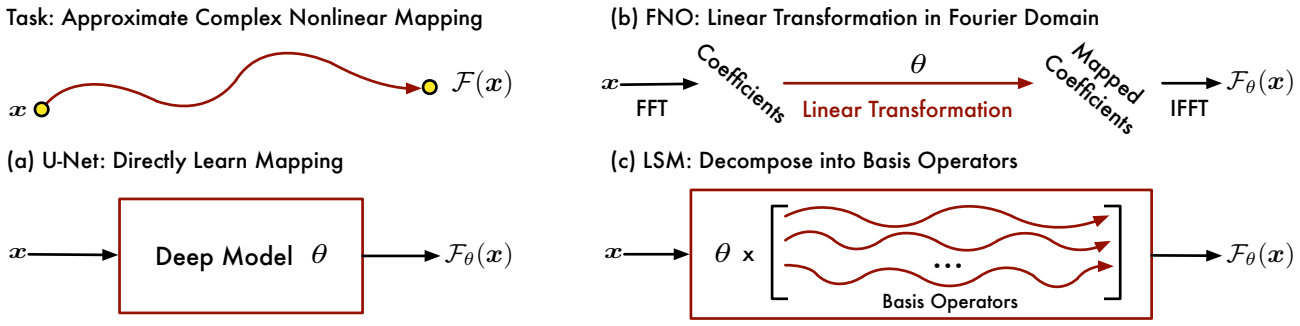

Figure 3. Comparison in approximating complex input-output mapping. For clarity, we only keep key components for approximation.

as latent tokens in K scales, the solving process for the j-th patch in the k-th scale can be formulized as follows:

$$\begin{aligned}\{\mathbf{T}^k_{\boldsymbol{x},i,j}\}_{i=1}^C &= \text{CoordToLatent}\left(\{\mathbf{T}^k_i\}_{i=1}^C, \{\boldsymbol{x}^k(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^k_j}\right)\\ \{\mathbf{T}^k_{\boldsymbol{y},i,j}\}_{i=1}^C &= \text{Solve}\left(\{\mathbf{T}^k_{\boldsymbol{x},i,j}\}_{i=1}^C\right)\\ \{\widehat{\boldsymbol{y}}^k(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^k_j} &= \text{LatentToCoord}\left(\{\boldsymbol{x}^k(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^k_j}, \{\mathbf{T}^k_{\boldsymbol{y},i,j}\}_{i=1}^C\right).\end{aligned} \tag{5}$$

More details of $\text{Solve}(\cdot)$ are deferred into the next section. Note that the patches in the same scale are governed by the same underlying PDEs, while in different scales, the coefficients of PDEs will change. Thus, the model parameters, e.g. latent tokens and linear layers, are shared in patches of the same scale but independent in different scales.

After the de-patchify operation, we splice patches into the output for the k-th scale as $\{\widehat{\boldsymbol{y}}^k(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^k}$. Then, we successively upsample the outputs in different scales from coarse to fine. Concretely, for the k-th scale, $\{\widehat{\boldsymbol{y}}^k(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^k}$ is concatenated with the interpolation-upsampled $\{\widehat{\boldsymbol{y}}^{k+1}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^{k+1}}$ and further projected to $\mathbb{R}^{d^k_{\text{model}}}$ with a linear layer parameterized in $\mathbb{R}^{(d^{k+1}_{\text{model}}+d^k_{\text{model}})\times d^k_{\text{model}}}$. Finally, we obtain the finest output $\{\widehat{\boldsymbol{y}}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}}$ with $\widehat{\boldsymbol{y}}(\mathbf{s})\in\mathbb{R}^{d_{\text{model}}}$. After the linear layer with parameters in $\mathbb{R}^{d_{\text{model}}\times d_{\boldsymbol{y}}}$, we can obtain the final output.

3.2. Neural Spectral Block

Benefitting from the hierarchical projection network, we can solve PDEs by approximating the complex mapping between latent input-output tokens as described in Eq. (5).

As shown in Figure 3, instead of learning a single operator, inspired by classical spectral methods in numerical analysis (Section 2.1), we present the neural spectral block by decomposing complex mappings into multiple basis operators:

$$\mathcal{F}_{\theta_{\text{Solve}}} = \sum_{i=1}^{N} w_i \mathcal{F}_{\theta_{\text{Solve},i}}, \tag{6}$$

where N is the hyperparameter and $\{\mathcal{F}_{\theta_{\text{Solve},i}}\}_{i=1}^N$ are orthogonal basis operators with learnable parameters $\{w_i\}_{i=1}^N$.

Following the classical design in spectral methods (Jackson, 1934; Tolstov, 2012), we select the trigonometric basis operators. Thus, for $\boldsymbol{t}_{\boldsymbol{x}}:\mathcal{D}\to\mathbb{R}^{d_{\text{latent}}}\in\mathcal{T}_{\mathcal{X}}, \forall \mathbf{s}\in\mathcal{D}$, we define the multiple basis operators as follows:

$$\begin{aligned}\mathcal{F}_{\theta_{\text{Solve}},(2k-1)}\left(\boldsymbol{t}_{\boldsymbol{x}}(\mathbf{s})\right) &= \sin\left(k\boldsymbol{t}_{\boldsymbol{x}}(\mathbf{s})\right)\\ \mathcal{F}_{\theta_{\text{Solve}},(2k)}\left(\boldsymbol{t}_{\boldsymbol{x}}(\mathbf{s})\right) &= \cos\left(k\boldsymbol{t}_{\boldsymbol{x}}(\mathbf{s})\right),\end{aligned} \tag{7}$$

where $k\in\{1,\cdots,\frac{N}{2}\}$ and N is even. Technically, given the latent input token $\mathbf{T}_{\boldsymbol{x}}\in\mathbb{R}^{d_{\text{latent}}}$, the latent output token $\mathbf{T}_{\boldsymbol{y}}$ of the neural spectral block is calculated as follows:

$$\mathbf{T}_{\boldsymbol{y}} = \mathbf{T}_{\boldsymbol{x}} + \mathbf{w}_0 + \mathbf{w}_{\sin}\begin{bmatrix}\sin(\mathbf{T}_{\boldsymbol{x}})\\ \vdots \\ \sin(\frac{N}{2}\mathbf{T}_{\boldsymbol{x}})\end{bmatrix} + \mathbf{w}_{\cos}\begin{bmatrix}\cos(\mathbf{T}_{\boldsymbol{x}})\\ \vdots \\ \cos(\frac{N}{2}\mathbf{T}_{\boldsymbol{x}})\end{bmatrix}, \tag{8}$$

where $\mathbf{w}_0\in\mathbb{R}^{d_{\text{latent}}}$, $\mathbf{w}_{\sin}\in\mathbb{R}^{1\times\frac{N}{2}}$, $\mathbf{w}_{\cos}\in\mathbb{R}^{1\times\frac{N}{2}}$ are learnable parameters. Residual connection is also adopted to facilitate optimization (He et al., 2016). We summarize the process of the neural spectral block as $\mathbf{T}_{\boldsymbol{y}}=\text{Solve}(\mathbf{T}_{\boldsymbol{x}})$, which is applied to the latent input tokens of every patch at every scale. Also according to the analysis in Eq. (5), like latent tokens, $\mathbf{w}_0, \mathbf{w}_{\sin}, \mathbf{w}_{\cos}$ is shared in patches of the same scale but independent in different scales.

Since PDE constraints have already been involved in input-output pairs, during the model training, $\mathbf{w}_0, \mathbf{w}_{\sin}, \mathbf{w}_{\cos}$ will be optimized to satisfy the PDEs better, namely solving PDEs in latent space. Besides, the neural spectral block also holds the universal approximation capacity with favorable convergence property guaranteed by the following theorems.

Assumption 3.1 (Finite Coordinate Set). In real-world applications, the analysis or numerical simulation of the PDE-governed task is mainly in the regular grid, mesh or point cloud, where the input is only observed on finite coordinates. Thus, to simplify the following theoretical derivations, we assume that $\mathcal{D}=\{\mathbf{s}_1,\cdots,\mathbf{s}_M\}$ is a finite set with size M, e.g. for a frame with height H and weight W, M is $H\times W$.

Remark 3.2 (**Simplification w.r.t. Finite Coordinate Set**). By assuming that $\mathcal{D}$ is a finite set with M coordinates, the

learning process of operator $\mathcal{F}: \mathcal{X}(\mathcal{D}; \mathbb{R}^{d_{\boldsymbol{x}}}) \to \mathcal{Y}(\mathcal{D}; \mathbb{R}^{d_{\boldsymbol{y}}})$ is simplified to solve the function $\boldsymbol{f}: \mathbb{R}^{M \times d_{\boldsymbol{x}}} \to \mathbb{R}^{M \times d_{\boldsymbol{y}}}$, where $\mathcal{F}(\boldsymbol{x}) = \boldsymbol{f} \circ \boldsymbol{x}, \forall \boldsymbol{x} \in \mathcal{X}$. Since the channel dimension can be seen as independent, we only focus on the coordinate dimension M in the following derivations.

Theorem 3.3 (**Convergence of Trigonometric Approximation in High-dimensional Space**). *(Dyachenko, 1995) Let $\boldsymbol{f}: \mathbb{R}^M \to \mathbb{R}^M$ be a 2π-periodic function w.r.t. the variable on each dimension, where $\boldsymbol{f} \in L_p\left([-\pi, \pi)^M\right), M \geq 2$, $1 \leq p \leq \infty$ and $p \neq 2$. For $\boldsymbol{f}$ defined on the M-dimension space, its trigonometric approximation $\boldsymbol{f}^N$ is defined as*

$$\boldsymbol{f}^N(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbb{Z}^M, |\mathbf{k}| \leq N} \left(\frac{1}{2\pi} \int_{[-\pi, \pi)^M} \boldsymbol{f}(\mathbf{t}) e^{-i\mathbf{k}\mathbf{t}} \mathrm{d}\mathbf{t} \right) e^{i\mathbf{k}\mathbf{x}}, \tag{9}$$

If $\boldsymbol{f}$ satisfies the Lipschitz condition, namely there is a non-negative constant K_1 such that

$$\|\boldsymbol{f}(\mathbf{x}) - \boldsymbol{f}(\mathbf{y})\|_p \leq K_1 \|\mathbf{x} - \mathbf{y}\|_p, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^M, \tag{10}$$

and if $(M-1)|\frac{1}{2} - \frac{1}{p}| < 1$, then there exists a constant K_2 such that

$$\left\|\boldsymbol{f} - \boldsymbol{f}^N\right\|_p \leq K_2 N^{(M-1)|\frac{1}{2} - \frac{1}{p}| - 1}. \tag{11}$$

Remark 3.4 (**Slow Convergence Rate in High-dimensional Space**). As demonstrated in Theorem 3.3, the convergence rate of trigonometric approximation is directly related to the dimension M, indicating that the spectral methods suffer from the slow convergence rate for high-dimensional space, e.g. $M = H \times W$ for a frame with height H and width W. Actually, the convergence properties of spectral methods in high-dimensional spaces are still under explored as an open problem (Brandolini et al., 2020). These results also support our design in solving PDEs in latent space instead of high-dimensional coordinate space.

Remark 3.5 (**Solving Process in Latent Space**). After projecting the M-dimension data into independent latent tokens and further restricting each latent token within $[0, \pi]$ through proper normalization, the solving process in the latent space is to approximate $f: [0, \pi] \to \mathbb{R}$.

Theorem 3.6 (**Approximation and Convergence Properties of Neural Spectral Block**). *Given $f: [0, \pi] \to \mathbb{R}$, if f satisfies the Lipschitz condition, there is a choice of model parameters such that the approximation f^N defined in neural spectral block (trigonometric approximation with residual) can uniformly converge to f with the speed as*

$$|f(x) - f^N(x)| \leq \frac{K_3 \ln N}{N}, \forall x \in [0, \pi], \tag{12}$$

where K_3 is a constant that does not depend on f nor N.

Proof. See Appendix A. □

4. Experiments

We extensively evaluate the proposed LSM on seven benchmarks, covering the typical PDEs in both solid and fluid physics and samples in various geometrics.

Table 1. Summary of experiment benchmarks.

PHYSICS	BENCHMARKS	GEOMETRY	#DIM
SOLID	ELASTICITY-P	POINT CLOUD	2D
	ELASTICITY-G	REGULAR GRID	2D
	PLASTICITY	STRUCTURED MESH	3D
FLUID	NAVIER–STOKES	REGULAR GRID	3D
	DARCY	REGULAR GRID	2D
	AIRFOIL	STRUCTURED MESH	2D
	PIPE	STRUCTURED MESH	2D

Benchmarks. As shown in Table 1, the experimental samples of seven benchmarks are recorded in various geometrics, including the regular grid, point cloud and structured mesh in the 2D or 3D space. These benchmarks are generated by different PDEs for different tasks. For clearness, we summarize the tasks of all benchmarks in Figure 1. Specifically, Elasticity-G is interpolated from Elasticity-P. More details can be found in Appendix B, including the governing PDEs, size of benchmarks and input-output resolutions.

Baselines. We compare the LSM with fourteen well-acknowledged and advanced models in all seven benchmarks, including three baselines proposed for vision tasks: U-Net (2015), ResNet (2016), Swin Transformer (2021), and ten baselines presented for PDEs: DeepONet (2021), TF-Net (2019), FNO (2021), U-FNO (2021), WMT (2021), Galerkin Transformer (2021), SNO (2022), U-NO (2022), HT-Net (2022), F-FNO (2023), KNO (2023a). U-NO and HT-Net are previous state-of-the-art models in solving PDEs. Note that all the above baselines are proposed for regular grid or structured mesh. Thus, for the Elasticity-P benchmark in point cloud, we adopt the special transformation proposed by geo-FNO (2022) at the beginning and end of these models, which can transform irregular input domain into or back from a uniform mesh.

Implementation. For fairness, all the methods are trained with L2 loss and 500 epochs, using the ADAM (Kingma & Ba, 2015) optimizer with an initial learning rate of 10^{-3}. The batch size is set to 20. We adopt the sum of mean squared error (MSE) on each coordinate as the metric. A comprehensive description is provided in Appendix I.

4.1. Main Results

Results. As shown in Table 2, LSM achieves consistent state-of-the-art performance on all seven benchmarks, covering both solid and fluid physics, justifying the generality of LSM on different PDEs, geometrics and dimensions. Overall, LSM averagely outperforms the previous best method on

Table 2. Performance comparison with fourteen baselines on all benchmarks. MSE is recorded. A smaller MSE indicates better performance. For clarity, the best result is in bold and the second best is underlined. Promotion refers to the relative error reduction w.r.t. the second best model on each benchmark. We only compare KNO (2023a; 2023b) and TF-Net (2019) on the Navier–Stokes benchmark, since they are proposed for auto-regressive tasks in fluid simulation. In addition to the quantitative performance, we also rank the models on each benchmark. See Table 13 for the performance rankings.

Model	Solid Physics*			Fluid Physics†			
	Elasticity-P ‡	Elasticity-G	Plasticity	Navier–Stokes	Darcy	Airfoil	Pipe
U-Net (2015)	0.0235	0.0531	0.0051	0.1982	0.0080	0.0079	0.0065
ResNet (2016)	0.0262	0.0843	0.0233	0.2753	0.0587	0.0391	0.0120
TF-Net (2019)	/	/	/	0.1801	/	/	/
Swin (2021)	0.0283	0.0819	0.0170	0.2248	0.0397	0.0270	0.0109
DeepONet (2021)	0.0965	0.0900	0.0135	0.2972	0.0588	0.0385	0.0097
FNO (2021)	<u>0.0229</u>	0.0508	0.0074	0.1556	0.0108	0.0138	0.0067
U-FNO (2021)	0.0239	0.0480	0.0039	0.2231	0.0183	0.0269	<u>0.0056</u>
WMT (2021)	0.0359	0.0520	0.0076	<u>0.1541</u>	0.0082	0.0075	0.0077
Galerkin (2021)	0.0240	0.1681	0.0120	0.2684	0.0170	0.0118	0.0098
SNO (2022)	0.0390	0.0987	0.0070	0.2568	0.0495	0.0893	0.0294
U-NO (2022)	0.0258	<u>0.0469</u>	<u>0.0034</u>	0.1713	0.0113	0.0078	0.0100
HT-Net (2022)	0.0372	0.0472	0.0333	0.1847	0.0079	<u>0.0065</u>	0.0059
F-FNO (2023)	0.0263	0.0475	0.0047	0.2322	<u>0.0077</u>	0.0078	0.0070
KNO (2023a)	/	/	/	0.2023	/	/	/
LSM	**0.0218**	**0.0408**	**0.0025**	**0.1535**	**0.0065**	**0.0059**	**0.0050**
Promotion	4.8%	13.0%	26.5%	0.4%	15.6%	9.2%	10.7%

∗ Top 5 ranking methods of solid benchmarks: LSM (ours), U-NO (2022), U-FNO (2021), FNO (2021), F-FNO (2023).
† Top 5 ranking methods of fluid benchmarks: LSM (ours), HT-Net (2022), WMT (2021), U-Net (2015), F-FNO (2023).
‡ All the experiments in Elasticity-P adopt the special transformation from geo-FNO (2022) to handle the point cloud geometric. Especially, FNO (2021) with the special transformation is just equivalent to geo-FNO (2022).

each benchmark by 11.5%. Specifically, our method accomplishes remarkable promotions on tasks with semantically heterogeneous input and output, such as 13.0% on Elasticity-G (0.0469→0.0408), 15.6% on Darcy (0.0077→0.0065). Note that these two tasks require the model to capture complex mappings between input and output, e.g. mapping from structure to inner stress on Elasticity-G or from the porous medium to flow on Darcy. From Table 2, we can find that the well-acknowledged FNO performs mediocrely on these complex tasks, verifying the advantages of LSM in approximating complex mappings of PDEs.

Ablations. To verify the effectiveness of each component in LSM, we provide detailed ablations, covering both removing components (*w/o*) and replacing projector (*rep*) experiments. From Table 3, we have the following observations.

In removing experiments, we can find that all components are essential to the final performance. Without the projector, model performance on both benchmarks will drop seriously, demonstrating the necessity of solving PDEs in latent space. Besides, the neural spectral block also reduces the estimation error significantly: 13.8% (0.0253→0.0218) in Elasticity-P and 13.3% (0.0075→0.0065) in Darcy. We can also find that the multiscale design can fit the Darcy benchmark well and the patchify operation is essential to the Elasticity-P benchmark, where the former always presents the multiphase flow and the latter mainly relies on the local information, showing that LSM can cover physical states in different scales and regions adaptively.

Table 3. Ablations on hierarchical projection network (*Projection, Multiscale, Patchify*) and neural spectral block (*Spectral*). We conduct two types of experiments: *replacing our attention-based projector with other designs (rep)* and *removing components (w/o)*. Efficiency is calculated on inputs with size 256×256 and batch size as 1. See Appendix D for full results.

Designs		#Param (MB)	#Mem (MB)	#Time (s/iter)	MSE Elas-P	MSE Darcy
rep	Conv	1.947	2.793	0.037	0.0236	0.0081
	AvgPool	1.836	1.748	0.028	0.0243	0.0077
	Self-Attn	2.002	7.188	0.064	0.0245	0.0082
w/o	Projector	1.836	2.793	0.035	0.0563	0.0080
	Multiscale	0.079	1.757	0.020	0.0269	0.0123
	Patchify	2.002	1.748	0.062	0.0545	0.0068
	Spectral	1.990	1.913	0.034	0.0253	0.0075
ours		2.002	1.914	0.041	**0.0218**	**0.0065**

In experiments of replacing our hierarchical projector, we observe that the convolution (*Conv*) and canonical self-attention (*Self-Attn*, 2017) will damage both efficiency and accuracy, since they still solve PDEs in the high-dimensional coordinate space. Although average pooling (*AvgPool*) can efficiently eliminate coordinate information, without latent tokens as physics prompts, it cannot capture the essential

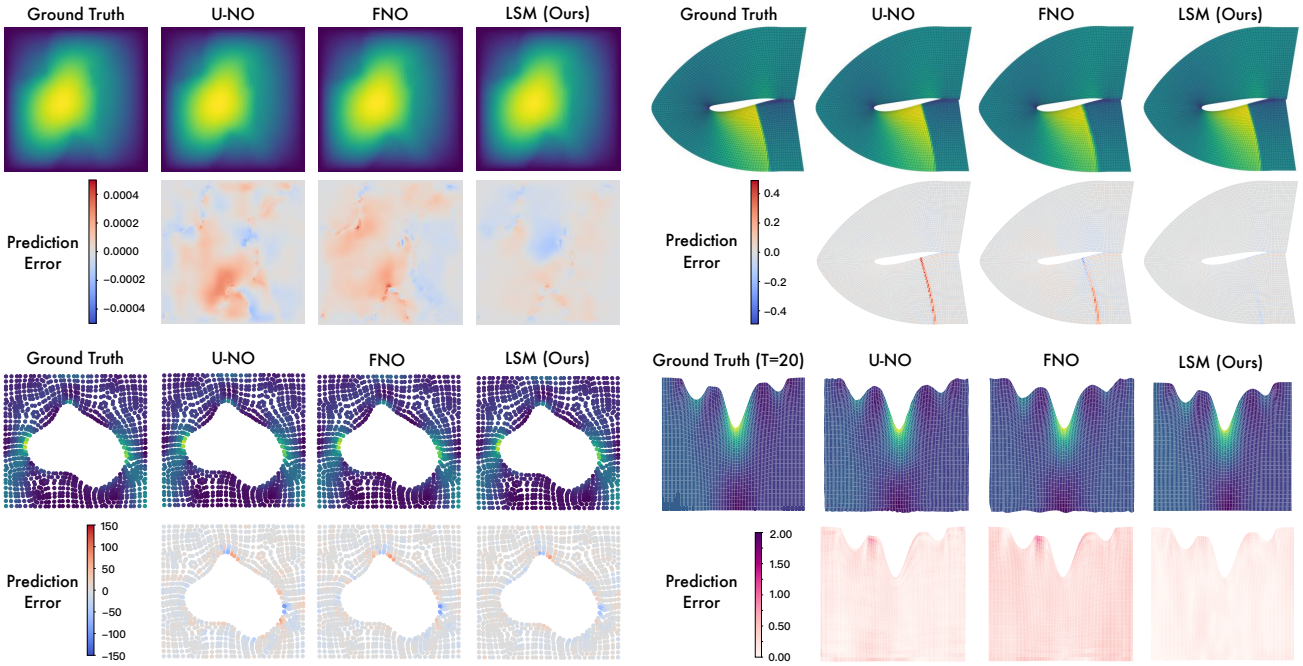

Figure 4. Top: showcases of fluid physics on Darcy (left) and Airfoil (right); bottom: showcases of solid physics on Elasticity-P (left) and Plasticity (right). We present the last timestamp ($T = 20$) for Plasticity here, which is a time-dependent task. For clearness, we also plot the prediction error, namely $\{\boldsymbol{y}(\mathbf{s}) - \widehat{\boldsymbol{y}}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}}$. See Appendix C for more showcases.

physical information and thus impairs accuracy. This verifies the efficacy of our hierarchical projection network.

Showcases. To present an intuitive comparison among different methods, we provide several showcases from representative benchmarks in Figure 4. Generally, LSM achieves impressive performance on both solid and fluid benchmarks. Especially, for the Airfoil benchmark, LSM is the only model that precisely captures the shock wave around the airfoil, which is vital for practical design. Note that the Airfoil benchmark is to estimate the airflow velocity from the airfoil structure, where the input and output are semantically heterogeneous, demonstrating the universal approximation capacity of LSM. Besides, LSM also surpasses FNO and U-NO in estimating the inner stress of elastic materials and the future mesh deformation in plastic materials, verifying the model capability in processing complex geometrics.

4.2. Model Analysis

Efficiency. From Figure 5, we can find that LSM achieves a good trade-off between accuracy and efficiency. For solid physics, although U-NO (Rahman et al., 2022) is the second-best model and slightly more efficient than LSM, LSM surpasses U-NO by a large margin, concretely 15.6%, 12.8% and 26.5% relative promotion in Elasticity-P, Elasticity-G and Plasticity respectively. For fluid physics, LSM is more accurate and efficient than the previous top three baselines: HT-Net, WMT and U-Net. It is notable that F-FNO is

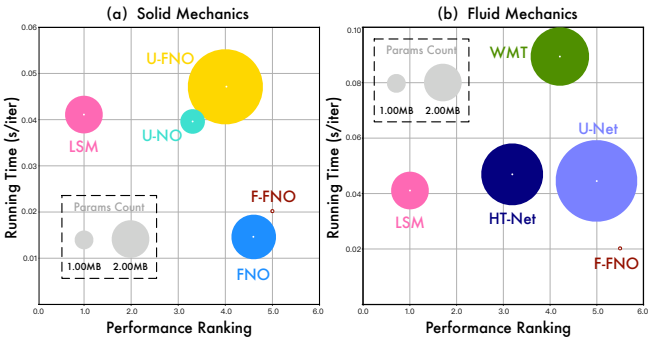

Figure 5. Efficiency comparison for the top 5 models on the benchmarks of solid and fluid physics. Running time is evaluated on inputs with size 256×256 and batch size as 1.

much more lightweight than others, but its running time and accuracy are still comparable to other baselines. Thus, in comparison to the lightweight model F-FNO, LSM is still more favorable for real-world applications due to the remarkable accuracy advantage. See Table 13 in Appendix for a comprehensive comparison.

Solving process visualization. We visualize the solving process of LSM in Figure 6. From Figure 6(a) and (b), we can easily recognize the projection and the PDE-solving process. Especially, for the Darcy benchmark, whose input and output are semantically heterogeneous, empowered by neural spectral block, LSM can present a distinct transformation in latent space to capture this complex mapping. Besides, we also provide a case for the time-dependent task from

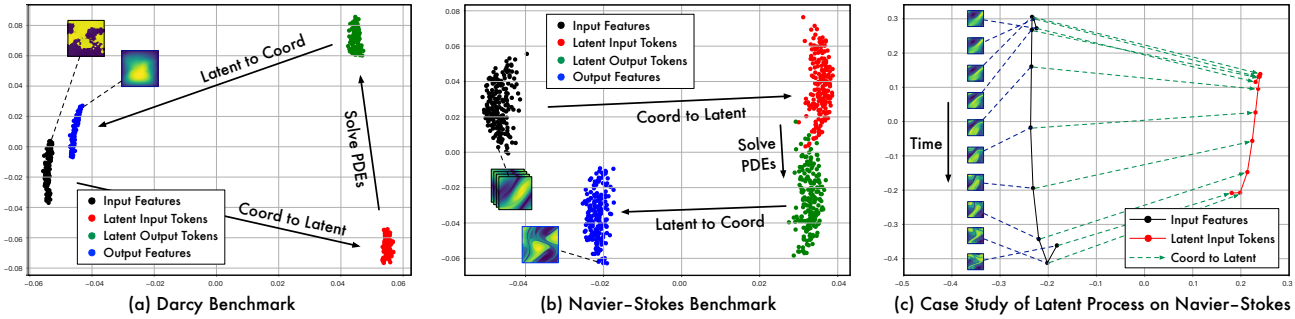

Figure 6. Visualization of solving process. Through PCA algorithm (Jolliffe & Cadima, 2016), we plot the input features $\{\boldsymbol{x}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}}$, latent input tokens $\{\mathbf{T}_{\boldsymbol{x}}\}$, latent output tokens $\{\mathbf{T}_{\boldsymbol{y}}\}$ and output features $\{\widehat{\boldsymbol{y}}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}}$ into a 2D plane. All the data are from the test set.

Table 4. Transfer the model pre-trained from full-data Pipe to limited-data Airfoil. The results are presented in the formalization of $a \to b$, where a is the model performance when it is trained from scratch and b is the performance finetuned from the Pipe pre-trained model. Since U-NO degenerates seriously in limited data situations, we do not take its 20% and 40% cases into comparison (colored in gray).

MSE ($\times 10^{-2}$)	20% Airfoil Data	40% Airfoil Data	60% Airfoil Data	80% Airfoil Data	100% Airfoil Data
U-Net (2015)	1.88→1.93 (-2.7%)	1.38→1.14 (+17.3%)	0.96→0.90 (+6.3%)	0.85→0.81 (+4.7%)	0.79→0.77 (+2.5%)
U-NO (2022)	6.30→1.72	2.39→1.73	1.10→1.00 (+9.1%)	0.86→0.82 (+4.7%)	0.78→0.82 (-5.1%)
HT-Net (2022)	1.73→1.43 (+17.3%)	1.08→0.82 (**+24.1%**)	0.75→0.69 (+8.0%)	0.70→0.65 (+7.1%)	0.65→0.61 (+6.2%)
LSM	**1.66→1.31 (+21.1%)**	**0.91→0.75** (+17.6%)	**0.69→0.61 (+11.6%)**	**0.63→0.58 (+7.9%)**	**0.59→0.55 (+6.8%)**

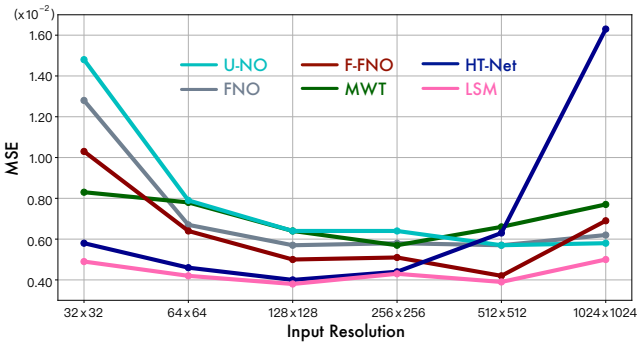

Figure 7. Model performance of Darcy under different resolutions.

the Navier-Stokes benchmark. As shown in Figure 6(c), by plotting the learned features over time, we can find that the latent input tokens present a similar process as the input features, demonstrating that LSM can precisely capture the latent process from high-dimensional coordinate space.

Performance under various resolutions. We also evaluate the model performance on the Darcy benchmark with various resolutions ranging from 32×32 to 1024×1024 in Figure 7. LSM presents a stable performance w.r.t. different inputs and consistently surpasses other baselines in all resolutions, presenting good capacity in solving high-dimensional PDEs. Besides, it is also notable that HT-Net degenerates in extremely high-dimensional setting, which is presented as a hierarchical Transformer, while FNO and its variants perform well. This indicates that there exist complex mappings between input-output pairs of high-dimensional PDEs, where even the most advanced deep models may fail without specific designs for mapping approximation.

Transferability. As shown in Table 4, we evaluate the model transferability by finetuning the model trained on Pipe to Airfoil. We can find that LSM consistently presents the positive transfer under all limited data situations, which is meaningful for applications. Besides, it is also observed that LSM performs best in both with and without pre-training cases. Note that both two benchmarks are governed by Navier-stokes equations but with distinct boundary conditions, indicating that LSM can learn the intrinsic physical information from unwieldy high-dimensional data.

5. Conclusions and Future Work

In this paper, we present LSM for solving high-dimensional PDEs. Instead of directly solving PDEs in coordinate space, LSM can efficiently reduce the high-dimensional data into compact latent space by a hierarchical projection network and approximate complex mappings by neural spectral block under theoretical guarantees. Benefiting from the above designs, LSM achieves consistent state-of-the-art in both solid and fluid benchmarks and presents a good trade-off between accuracy and efficiency, making itself a promising PDE solver for real-world applications. In the future, we further explore the generalization capability of LSM among different PDEs to pursue a foundation model.

Acknowledgements

This work was supported by the National Key Research and Development Plan (2021YFC3000905), National Natural Science Foundation of China (62022050 and 62021002), and Beijing Nova Program (Z201100006820041).

References

Brandolini, L., Colzani, L., Robins, S., and Travaglini, G. Pick's theorem and convergence of multiple fourier series. *Am. Math. Mon.*, 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020.

Brunton, S. L., Budisic, M., Kaiser, E., and Kutz, J. N. Modern koopman theory for dynamical systems. *SIAM Rev.*, 2021.

Cao, S. Choose a transformer: Fourier or galerkin. In *NeurIPS*, 2021.

Chen, T. and Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Netw. Learn. Syst.*, 1995.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Dyachenko, M. The rate of u-convergence of multiple fourier series. *Acta Mathematica Hungarica*, 1995.

Evans, L. C. *Partial differential equations*. American Mathematical Soc., 2010.

Fanaskov, V. and Oseledets, I. Spectral neural operators. *arXiv preprint arXiv:2205.10573*, 2022.

Fornberg, B. *A practical guide to pseudospectral methods*. Cambridge university press, 1998.

Gottlieb, D. and Orszag, S. A. *Numerical analysis of spectral methods: theory and applications*. SIAM, 1977.

Grossmann, C., Roos, H.-G., and Stynes, M. *Numerical treatment of partial differential equations*. Springer, 2007.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. In *NeurIPS*, 2021.

Han, J., Jentzen, A., and E, W. Solving high-dimensional partial differential equations using deep learning. *PNAS*, 2017.

Hao, Z., Liu, S., Zhang, Y., Ying, C., Feng, Y., Su, H., and Zhu, J. Physics-informed machine learning: A survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, 2016.

Jackson, D. The convergence of fourier series. *American Mathematical Monthly*, 1934.

Jolliffe, I. T. and Cadima, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2016.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nat. Rev. Phys.*, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kopriva, D. A. *Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers*. Springer Science & Business Media, 2009.

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 *ICLR*, 2021.

Li, Z.-Y., Kovachki, N. B., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

Li, Z.-Y., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022.

Liu, X., Xu, B., and Zhang, L. HT-net: Hierarchical transformer based operator learning model for multiscale PDEs. *arXiv preprint arXiv:2210.10890*, 2022.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. C.-F., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, 2021.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nat. Mach. Intell*, 2021.

McLean, D. Continuum fluid mechanics and the navier-stokes equations. *Understanding Aerodynamics: Arguing from the Real Physics*, 2012.

Morrison, F. A. *An introduction to fluid mechanics*. Cambridge University Press, 2013.

Nayak, L., Das, G., and Ray, B. An estimate of the rate of convergence of fourier series in the generalized hölder metric by deferred cesàro mean. *J. Math. Anal. Appl*, 2014.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 2020.

Rahman, M. A., Ross, Z. E., and Azizzadenesheli, K. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 2019.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

Roubíček, T. *Nonlinear partial differential equations with applications*. Springer Science & Business Media, 2013.

Ŝolín, P. *Partial differential equations and the finite element method*. John Wiley & Sons, 2005.

Temam, R. *Navier-Stokes equations: theory and numerical analysis*. American Mathematical Soc., 2001.

Tolstov, G. P. *Fourier series*. Courier Corporation, 2012.

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized fourier neural operators. In *ICLR*, 2023.

Trunk, G. V. A problem of dimensionality: A simple example. *TPAMI*, 1979.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. Towards physics-informed deep learning for turbulent flow prediction. *KDD*, 2019.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.*, 2020a.

Wang, S., Yu, X., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *J. Comput. Phys.*, 2020b.

Wazwaz, A. M. Partial differential equations : methods and applications. 2002.

Weinan, E. and Yu, T. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.*, 2017.

Wen, G., Li, Z.-Y., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. U-fno - an enhanced fourier neural operator based-deep learning model for multiphase flow. *arXiv preprint arXiv:2109.03697*, 2021.

Xiong, W., Huang, X., Zhang, Z., Deng, R., Sun, P., and Tian, Y. Koopman neural operator as a mesh-free solver of non-linear partial differential equations. *arXiv preprint arXiv:2301.10022*, 2023a.

Xiong, W., Ma, M., Huang, X., Zhang, Z., Sun, P., and Tian, Y. Koopmanlab: machine learning for solving complex physics equations. *arXiv preprint arXiv:2301.01104*, 2023b.

A. Proofs of Theorems 3.6

First, we would like to present a well-established theorem, whose proof can be found in the cited paper.

Theorem A.1. *(Nayak et al., 2014) Let $f : \mathbb{R} \to \mathbb{R}$ be a 2π-periodic function. Its trigonometric approximation f^N is defined as:*

$$f^N(x) = \sum_{k=-N}^{N} \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) e^{-ikt} \mathrm{d}t \right) e^{ikx}, \tag{13}$$

If f satisfies the Lipschitz condition, then there is a constant K that does not depend on f nor N, such that:

$$|f(x) - f^N(x)| \leq \frac{K \ln N}{N}, \forall x \in \mathbb{R}. \tag{14}$$

Lemma A.2. *Given $f : [0, \pi] \to \mathbb{R}$ and $g(x) = f(x) - x, \forall x \in [0, \pi]$. If f satisfies the Lipschitz condition, then g also satisfies the Lipschitz condition.*

Proof. Suppose that f satisfies the Lipschitz condition, then there is a constant K, such that

$$|f(x) - f(y)| \leq K|x - y|, \forall x, y \in [0, \pi].$$

Then, we have the following inequations:

$$|g(x) - g(y)| = |f(x) - x - (f(y) - y)\,| \leq |f(x) - f(y)| + |x - y| \leq (K + 1)|x - y|, \forall x, y \in [0, \pi].$$

Thus, g also satisfies the Lipschitz condition. □

Lemma A.3. *Given $f : [-\pi, \pi] \to \mathbb{R}$ and $f(x) = f(-x), \forall x \in [0, \pi]$. If f satisfies the Lipschitz condition within $[0, \pi]$, then f also satisfies Lipschitz condition in $[-\pi, \pi]$.*

Proof. Suppose that f satisfies the Lipschitz condition in $[0, \pi]$, then there is a constant K, such that

$$|f(x) - f(y)| \leq K|x - y|, \forall x, y \in [0, \pi].$$

$\forall x, y \in [-\pi, \pi]$, if $xy \geq 0$, we obvisouly have $|f(x) - f(y)| \leq K|x - y|$.

If $xy < 0$, we have $|f(x) - f(y)| = |f(x) - f(-y)| \leq K|x + y| \leq K|x - y|$. □

Next, we will prove Theorem 3.6, which shows the convergence property of trigonometric approximation with residual.

Proof. For simplification, we define $g(x) = f(x) - x, \forall x \in [0, \pi]$. From Lemma A.2, g holds the Lipschitz condition as f. Then we would like to extend $g : [0, \pi] \to \mathbb{R}$ to a 2π-periodic function $g_{\text{extend}} : \mathbb{R} \to \mathbb{R}$. Firstly, we define $\widehat{g}_{\text{extend}} : [-\pi, \pi] \to \mathbb{R}$ as:

$$\widehat{g}_{\text{extend}}(x) = \begin{cases} g(x), & \text{If } x \in [0, \pi] \\ g(-x), & \text{If } x \in [-\pi, 0), \end{cases} \tag{15}$$

Further, we define the 2π-periodic function $g_{\text{extend}} : \mathbb{R} \to \mathbb{R}$ as follows:

$$\begin{aligned} g_{\text{extend}}(x) &= \widehat{g}_{\text{extend}}\Big(\text{Normalize}(x) \Big), \text{ where} \\ \text{Normalize}(x) &= \begin{cases} x - \text{sgn}(x)\big(\lceil \frac{|x| - \pi}{2\pi} \rceil \times 2\pi \big), & \text{if } |x| > \pi \\ x, & \text{otherwise,} \end{cases} \end{aligned} \tag{16}$$

where $\text{sgn}(\cdot)$ is the sign function, whose values is 1 for positive inputs, -1 for negative inputs, 0 for zero inputs.

Considering the definition of neural spectral block in Eq. (8), we can find the following parameters in the neural spectral block will satisfy Eq. (12):

$$\begin{aligned}
\mathbf{w}_0 &= \Big[\frac{1}{2\pi}\int_{-\pi}^{\pi} g_{\text{extend}}(t)\mathrm{d}t\Big] \\
\mathbf{w}_{\sin} &= \Big[\frac{1}{\pi}\int_{-\pi}^{\pi} g_{\text{extend}}(t)\sin(t)\mathrm{d}t, \cdots, \frac{1}{\pi}\int_{-\pi}^{\pi} g_{\text{extend}}(t)\sin(\frac{N}{2}t)\mathrm{d}t\Big] \\
\mathbf{w}_{\cos} &= \Big[\frac{1}{\pi}\int_{-\pi}^{\pi} g_{\text{extend}}(t)\cos(t)\mathrm{d}t, \cdots, \frac{1}{\pi}\int_{-\pi}^{\pi} g_{\text{extend}}(t)\cos(\frac{N}{2}t)\mathrm{d}t\Big].
\end{aligned}$$

Then, we have the canonical trigonometric approximation of g_{extend} as g^N_{extend}, which is defined as follows:

$$g^N_{\text{extend}}(x) = \mathbf{w}_0 + \mathbf{w}_{\sin}\begin{bmatrix}\sin(x)\\ \vdots \\ \sin(\frac{N}{2}x)\end{bmatrix} + \mathbf{w}_{\cos}\begin{bmatrix}\cos(x)\\ \vdots \\ \cos(\frac{N}{2}x)\end{bmatrix} = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}}\left(\frac{1}{2\pi}\int_{-\pi}^{\pi} g_{\text{extend}}(t)e^{-ikt}\mathrm{d}t\right)e^{ikx}, \forall x\in\mathbb{R}.$$

If f satisfies the Lipschitz condition, from Lemma A.2 and Lemma A.3, we have that $\widehat{g}_{\text{extend}}$ satisfies the Lipschitz condition. Since $\widehat{g}_{\text{extend}} : [-\pi,\pi]\to\mathbb{R}$ satisfies $\widehat{g}_{\text{extend}}(x) = \widehat{g}_{\text{extend}}(-x)$, then $\forall x, y\in\mathbb{R}$, there is a constant K', such that:

$$\begin{aligned}
|g_{\text{extend}}(x) - g_{\text{extend}}(y)| &= \left|\widehat{g}_{\text{extend}}\left(|\operatorname{Normalize}(x)|\right) - \widehat{g}_{\text{extend}}\left(|\operatorname{Normalize}(y)|\right)\right| \\
&= \left|g\left(|\operatorname{Normalize}(x)|\right) - g\left(|\operatorname{Normalize}(y)|\right)\right| \\
&\le K'\Big||\operatorname{Normalize}(x)| - |\operatorname{Normalize}(y)|\Big| \\
&\le K'|x-y|. \quad \text{(Similar discussion as Lemma A.3)}
\end{aligned} \tag{17}$$

For the last inequation of Eq. (17), if $|x-y|\ge\pi$, the inequation obviously holds. If $|x-y|<\pi$ and $x,y\in[n\pi,(n+1)\pi], n\in\mathbb{Z}$, then we have $\Big||\operatorname{Normalize}(x)| - |\operatorname{Normalize}(y)|\Big| = |x-y|$. As for $|x-y|<\pi$ and $x\le 2n\pi\le y, n\in\mathbb{Z}$ (suppose $x\le y$ without loss of generality), we have

$$\Big||\operatorname{Normalize}(x)| - |\operatorname{Normalize}(y)|\Big| = \Big|(2n\pi - x) - (y - 2n\pi)\Big| \le \Big|(2n\pi - x) + (y - 2n\pi)\Big| = |x-y|.$$

As for $|x-y|<\pi$ and $x\le(2n+1)\pi\le y, n\in\mathbb{Z}$, we have

$$\begin{aligned}
\Big||\operatorname{Normalize}(x)| - |\operatorname{Normalize}(y)|\Big| &= \Big|\big(\pi - ((2n+1)\pi - x)\big) - \big(\pi - (y - (2n+1)\pi)\big)\Big| \\
&= \Big|(y - (2n+1)\pi) - ((2n+1)\pi - x)\Big| \\
&\le \Big|(y - (2n+1)\pi) + ((2n+1)\pi - x)\Big| \\
&= |x-y|.
\end{aligned}$$

Thus, g_{extend} also satisfies the Lipschitz condition.

Thus, from Theorem A.1, we have that g^N_{extend} converges to g_{extend} with the speed as follows:

$$|g_{\text{extend}}(x) - g^N_{\text{extend}}(x)| \le \frac{K\ln\frac{N}{2}}{\frac{N}{2}}, \forall x\in\mathbb{R}, \tag{18}$$

where K is a constant. From the definition of Eq. (8), $\forall x\in[0,\pi]$, we have $f^N(x) = x + g^N_{\text{extend}}(x)$, then

$$\begin{aligned}
|f(x) - f^N(x)| &= \Big|\big(g_{\text{extend}}(x) + x\big) - \big(g^N_{\text{extend}}(x) + x\big)\Big| \\
&= \Big|g_{\text{extend}}(x) - g^N_{\text{extend}}(x)\Big| \le \frac{K\ln\frac{N}{2}}{\frac{N}{2}} \le \frac{(2K)\ln N}{N}, \forall x\in[0,\pi].
\end{aligned} \tag{19}$$

□

Table 5. Details for benchmarks. All the settings follow FNO (Li et al., 2021) and geo-FNO (Li et al., 2022). The input-output resolutions are presented in the shape of (temporal, spatial, variate). "/" means without this dimension.

	Descriptions	Solid Physics			Fluid Physics			
		Elasticity-P	Elasticity-G	Plasticity	Navier–Stokes	Airfoil	Pipe	Darcy
Physics	PDEs	PDEs of solid material			Navier-Stokes Equation			Darcy's law
	Task	Estimate stress		Model deformation	Predict future	Estimate velocity		Estimate Pressure
	Input	Material structure		Boundary condition	Past velocity	Structure		Porous medium
	Output	Inner stress		Mesh displacement	Future velocity	Fluid velocity		Fluid Pressure
Data	Train set size	1000	1000	900	1000	1000	1000	1000
	Test set size	200	200	80	200	100	200	200
	Input Tensor	$(/, 972, 2)$	$(/, 41 \times 41, 1)$	$(/, 101 \times 31, 2)$	$(10, 64 \times 64, 1)$	$(/, 200 \times 50, 2)$	$(/, 129 \times 129, 2)$	$(/, 85 \times 85, 1)$
	Output Tensor	$(/, 972, 1)$	$(/, 41 \times 41, 1)$	$(20, 101 \times 31, 4)$	$(10, 64 \times 64, 1)$	$(/, 200 \times 50, 1)$	$(/, 129 \times 129, 1)$	$(/, 85 \times 85, 1)$

B. Details for Benchmarks

We have summarized benchmark configurations in Table 5. Here are the generation details categorized by governing PDEs.

B.1. Solid Material

The governing equation of solid material is:

$$\rho^s \frac{\partial^2 \boldsymbol{u}}{\partial t^2} + \nabla \cdot \boldsymbol{\sigma} = 0, \tag{20}$$

where $\rho^s \in \mathbb{R}$ means the solid density, ∇ denotes the nabla operator. $\boldsymbol{u}$ is a function that represents the displacement vector of material over time t. $\boldsymbol{\sigma}$ denotes the stress tensor. Elasticity-P, Elasticity-G and Plasticity (Li et al., 2022) share the same governing equation as shown in Eq. (20).

Elasticity-P and Elasticity-G. These benchmarks are to estimate the inner stress of an incompressible material with an arbitrary void at the center of the material. Besides, an external tension is applied to the material. The input is the structure of the material, and the output is inner stress. Elasticity-P and Elasticity-G differ in the way modeling the geometric of material: Elasticity-P uses a point cloud with 972 points, while Elasticity-G presents the data in a regular grid with the size of 41×41, which is interpolated from Elasticity-P.

Plasticity. This benchmark focuses on the plastic forging problem, where a plastic material is impacted from above by an arbitrary-shaped die. The input is the shape of the die, which is recorded in structured mesh. And the output is the deformation of each mesh point in the future 20 time steps. The resolution of the structured mesh is 101×31.

B.2. Navier-Stokes Equation

The differential form of fluid dynamics equations are:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}) = 0 \tag{21}$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{U} \cdot \nabla \boldsymbol{U} = \boldsymbol{f} + \frac{1}{\rho} \nabla \cdot (\boldsymbol{T}_{ij} \boldsymbol{e}_i \boldsymbol{e}_j) \tag{22}$$

$$\frac{\partial (e + \frac{1}{2}\boldsymbol{U}^2)}{\partial t} + \boldsymbol{U} \cdot \nabla (e + \frac{1}{2}\boldsymbol{U}^2) = \boldsymbol{f} \cdot \boldsymbol{U} + \frac{1}{\rho} \nabla \cdot (\boldsymbol{U} \cdot \boldsymbol{T}_{ij} \boldsymbol{e}_i \boldsymbol{e}_j) + \frac{\lambda}{\rho} \Delta T, \tag{23}$$

where Eq. (21), Eq. (22) and Eq. (23) describe the mass, momentum and energy conservation respectively. Here ρ is the density, $\boldsymbol{U}$ is the velocity vector, $\boldsymbol{f}$ is the external force, e is the internal energy. And $\boldsymbol{T}$ is the stress tensor in the fluid, $\boldsymbol{e}$ is the basis vector and $\boldsymbol{T}_{ij} \boldsymbol{e}_i \boldsymbol{e}_j$ follows the Einstein summation convention. All above variates are related to both space and time. $\frac{\lambda}{\rho} \Delta T$ is for heat conduction. For a Newtonian fluid, the stress tensor $\boldsymbol{T}$ is related to the pressure p, viscosity coefficient ν and velocity vector $\boldsymbol{U}$. Thus, for the Newtonian fluid, Eq. (22) can be rewritten as:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{U} \cdot \nabla \boldsymbol{U} = \boldsymbol{f} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \boldsymbol{U}. \tag{24}$$

Besides, Eq. (23) can also be deduced in a similar way, but the result is too complex to be presented in this paper. See (McLean, 2012) for more details. The dynamics equations for Newtonian fluid are well-known as Navier-Stokes equations. Next, we will detail the underlying PDEs for our fluid benchmarks.

Navier-Stokes. We take the Navier-Stokes dataset from (Li et al., 2021). This dataset simulates incompressible and viscous flow on the unit torus, where the density of fluid is unchangeable (ρ in Eq. (21)). In this situation, the energy conservation presented in Eq. (23) is independent of mass and momentum conservation. Hence, the fluid dynamics can be deduced with Eq. (21) and Eq. (24):

$$\begin{aligned} \nabla \cdot \boldsymbol{U} &= 0 \\ \frac{\partial w}{\partial t} + \boldsymbol{U} \cdot \nabla w &= \nu \nabla^2 w + f \\ w|_{t=0} &= w_0, \end{aligned} \tag{25}$$

where $\boldsymbol{U} = (u, v)$ is a velocity vector in 2D field, $w = |\nabla \times \boldsymbol{U}| = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}$ is the vorticity, $w_0 \in \mathbb{R}$ is the initial vorticity at $t = 0$. In this dataset, viscosity ν is set as 10^{-5} and the resolution of the 2D field is 64×64. Each generated sample contains 20 successive frames and the task is to predict the future 10 frames based on the past 10 frames.

Pipe. This dataset (Li et al., 2022) focuses on the incompressible flow through a pipe. The governing equations are similarly deduced with Eq. (21) and Eq. (24):

$$\begin{aligned} \nabla \cdot \boldsymbol{U} &= 0 \\ \frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{U} \cdot \nabla \boldsymbol{U} &= \boldsymbol{f} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \boldsymbol{U}. \end{aligned} \tag{26}$$

The dataset is generated in the geometric of structured mesh with the resolution of 129×129. For experiments, we adopt the mesh structure as the input data, and the output is the horizonal fluid velocity within the pipe.

Airfoil. The airfoil dataset (Li et al., 2022) is about the transonic flow over an airfoil. Since the viscosity of air is quite small, the viscous term $\nu \nabla^2 \boldsymbol{U}$ can be ignored in the Navier-Stokes equation. Thus, the governing equations for this situation can be presented as follows:

$$\begin{aligned} \frac{\partial \rho^f}{\partial t} + \nabla \cdot (\rho^f \boldsymbol{U}) &= 0 \\ \frac{\partial \rho^f \boldsymbol{U}}{\partial t} + \nabla \cdot (\rho^f \boldsymbol{U}\boldsymbol{U} + p\mathbb{I}) &= 0 \\ \frac{\partial E}{\partial t} + \nabla \cdot ((E + p)\boldsymbol{U}) &= 0, \end{aligned} \tag{27}$$

where ρ^f denotes the fluid density, and E represents the total energy. The data is generated in the geometric of structured mesh with resolution of 200×50. The locations of these mesh points are adopted as inputs. And the Mach number of each mesh point is the output.

B.3. Darcy Flow

Darcy. The Darcy's law describes the flow of fluid through a porous medium, for example, water goes through sand. We use the Darcy dataset proposed in (Li et al., 2021), where 2-D Darcy flow equations in a unit box are formulized as:

$$\begin{aligned} -\nabla \cdot (a \nabla u) &= f \\ u|_{x \in \partial(0,1)^2} &= 0, \end{aligned} \tag{28}$$

where $a \in \mathbb{R}^+$ is the diffusion coefficient. f means the externel force, which is fixed as 1 in this dataset. This dataset takes a as input, and the output is the solution u. The samples in this dataset are in the regular grid with resolution as 85×85.

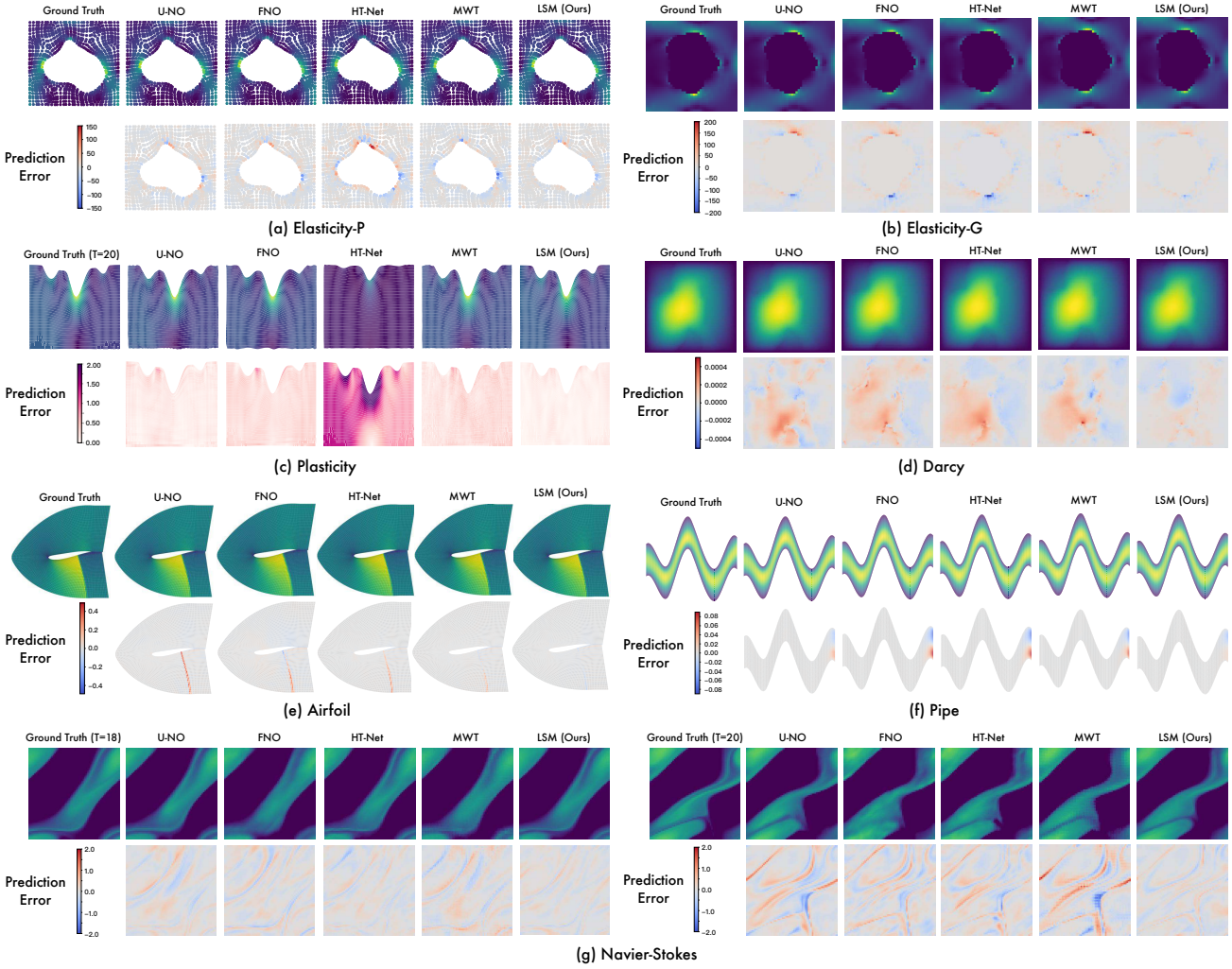

Figure 8. Showcases on all seven benchmarks. Especially for the sub-figure (c) Plasticity, we plot the last timestamp of output ($T = 20$). As for the sub-figure (g) Navier-Stokes, we plot the frames at $T = 18$ and $T = 20$ to present the model performance over time.

C. More Showcases

As a complement to Figure 4, we present showcases for all benchmarks in Figure 8 and also plot the coordinate-wise prediction error for comparison. As demonstrated in above showcases, LSM achieves a remarkable prediction performance in extensive tasks. By investigating each case, we can obtain the following observations:

- Performance on the boundary. From Figure 8(a)(b)(c)(f), we can find that LSM significantly surpasses other baselines on the boundary of different geometrics, demonstrating the model capability in learning physical constraints.
- Performance in time-dependent tasks. As shown in Figure 8(g), LSM can precisely predict the future velocity for the fluid in the Navier-Stokes benchmark. Especially, the performances of other methods drop seriously from $T = 18$ to $T = 20$, while LSM can simulate the fluid accurately even in the long-term future.

D. Full Ablations

As a complement to Table 3 of main text, we provide the comprehensive ablation results for all seven benchmarks here. From Table 6, we can observe that all the components in LSM are effective to the final performance. Besides, we also present detailed ablations on the neural spectral block in Table 7. Here are the analyses.

Table 6. Full ablation results on hierarchical projection network (*Projection, Multiscale, Patchify*) and neural spectral block (*Spectral*). We conduct two types of experiments: *replacing our attention-based projector with other designs (rep)* and *removing components (w/o)*. Efficiency is calculated on inputs with size 256×256 and batch size as 1. "/" indicates the out-of-memory situation.

Designs		#Param (MB)	#Mem (MB)	#Time (s/iter)	Solid Physics			Fluid Physics			
					Elasticity-P	Elasticity-G	Plasticity	Navier-Stokes	Darcy	Airfoil	Pipe
rep	Conv	1.947	2.793	0.037	0.0236	0.00429	0.0029	0.1571	0.0081	0.0077	0.0052
	AvgPool	1.836	1.748	0.028	0.0243	0.0413	0.0031	0.1564	0.0077	0.0072	0.0056
	Self-Attn	2.002	7.188	0.064	0.0245	0.0424	/	0.1567	0.0082	0.0062	0.0056
w/o	Projector	1.836	2.793	0.035	0.0563	0.0419	/	0.1609	0.0080	0.0085	0.0059
	Multiscale	0.079	1.757	0.020	0.0269	0.0479	0.0044	0.1667	0.0123	0.0097	0.0091
	Patchify	2.002	1.748	0.062	0.0545	0.0414	0.0040	0.1576	0.0068	0.0062	0.0055
	Spectral	1.990	1.913	0.034	0.0253	0.0421	0.0034	0.1618	0.0075	0.0107	0.0053
ours		2.002	1.914	0.041	**0.0218**	**0.0408**	**0.0025**	**0.1535**	**0.0065**	**0.0059**	**0.0050**

Table 7. Detailed ablations on neural spectral block. MSE is recorded.

Type	Model	Elasticity-P	Darcy
Replace neural spectral block	LSM w/o neural spectral block	0.0253	0.0075
	LSM but replace neural spectral block with MLP	0.0249	0.0075
	LSM but replace neural spectral block with FNO	0.0356	0.0073
Replace basis operators	LSM with polynomial basis operators	0.0261	0.0073
Final version	LSM	**0.0218**	**0.0065**

Replace neural spectral block with other global operators. To verify advantages in learning multiple basis operators, we also conduct experiments on replacing the neural spectral block with multilayer perceptrons (MLP) and FNO (Li et al., 2021), where the latter ones are global operators without basis operator decomposition design. As shown in Table 7, compared to learning basis operators, it is harder to learn a global operator, whose performance is close to removing the neural spectral block directly. It is also notable that replacing neural spectral block with FNO means applying FFT in the latent space, which is unreasonable since the latent tokens are independent. Thus, directly replacing neural spectral block with FNO damages performance seriously (Table 7), sometimes even worse than the case without neural spectral block.

Replace basis operators in neural spectral block. Note that the classical spectral method is a general framework, which is to decompose the complex solution into several orthogonal basis functions. Thus, replacing the trigonometric approximation in LSM with other basis is also implementable. However, other basis may not achieve the nice approximation and optimization properties as the trigonometric basis functions. For example, as shown in Table 7, directly replacing trigonometric basis with polynomial basis will decrease the model performance. Thus, we would like to leave the exploration of other basis operators as the future work, including the corresponding model design and theoretical derivation.

E. Performance Under Various Resolutions

As shown in Table 8 and 9, we also evaluate the model performance on the newly-generated Darcy and Navier-Stokes datasets with various resolutions, where we can obtain the following observations:

- For the Darcy benchmark, U-Net (2015) and HT-Net (Liu et al., 2022) that are proposed based on advanced deep models U-Net and Transformer (Vaswani et al., 2017), degenerate a lot on the inputs with large resolutions, e.g. 1024×1024, indicating that there exist complex mappings between input-output pairs of high-dimensional PDEs. In contrast, LSM presents a stable performance w.r.t. different inputs and consistently surpasses other baselines in all resolutions, presenting good capacity in solving high-dimensional PDEs.
- As for the Navier-Stokes benchmark, whose task is to predict the future 10 frames based the past 10 frames, we can find that in comparison with other baselines, LSM presents more significant advantage in higher input resolutions.

Table 8. Model performance comparison on Darcy under different resolutions.

RESOLUTION	U-NET (2015)	FNO (2021)	MWT (2021)	U-NO (2022)	F-FNO (2023)	HT-NET (2022)	**LSM (OURS)**
32×32	0.0059	0.0128	0.0083	0.0148	0.0103	0.0058	**0.0049**
64×64	0.0052	0.0067	0.0078	0.0079	0.0064	0.0046	**0.0042**
128×128	0.0054	0.0057	0.0064	0.0064	0.0050	0.0040	**0.0038**
256×256	0.0251	0.0058	0.0057	0.0064	0.0051	0.0044	**0.0043**
512×512	0.0496	0.0057	0.0066	0.0057	0.0042	0.0063	**0.0039**
1024×1024	0.0754	0.0062	0.0077	0.0058	0.0069	0.0163	**0.0050**

Table 9. Model performance comparison on the Navier-Stokes benchmark under different resolutions. "/" indicates the poor performance.

RESOLUTION	U-NET (2015)	FNO (2021)	MWT (2021)	U-NO (2022)	F-FNO (2023)	HT-NET (2022)	**LSM (OURS)**
64×64	0.1982	0.1556	0.1541	0.1713	0.2322	0.1847	**0.1535**
128×128	/	0.1028	0.1099	0.1068	0.1506	0.1088	**0.0961**

F. Additional Experiments on Burger's Equation

As a fundamental partial differential equation for convection-diffusion processes occurring in various areas of applied mathematics, Burger's equation is widely used in modeling fluid mechanics, nonlinear acoustics and gas dynamics. Following the experiment settings in FNO (Li et al., 2021), we also test LSM in solving 1D Burger's equation. Especially, to fit the 1D input data, we need to implement the following changes to LSM and other baselines:

- By conducting the up-down sampling and patchify in the 1D space, LSM can handle the 1D inputs.
- As for the F-FNO, we replace its factorized 2D FFT with 1D FFT.
- For the U-NO, we replace both 2D up-down sampling and 2D FFT with 1D versions.

From Table 10, we can find that LSM still performs well in this equation under various resolutions, verifying the model capacity in solving high-dimensional PDEs.

Table 10. Model performance comparison on 1D Burger's equation.

RESOLUTION	FNO (2021)	MWT (2021)	U-NO (2022)	F-FNO (2023)	**LSM (OURS)**
256	0.00332	0.00199	0.00450	0.00414	**0.00123**
512	0.00333	0.00185	0.00488	0.00347	**0.00124**
1024	0.00377	0.00185	0.00508	0.00319	**0.00126**
2048	0.00346	0.00186	0.00574	0.00313	**0.00115**
4096	0.00324	0.00185	0.00571	0.00314	**0.00122**
8192	0.00336	0.00178	0.00575	0.00315	**0.00105**

G. Hyperparameter Sensitivity

As shown in Table 11, we test the hyperparameter sensitivity of our model by changing one hyperparameter and fixing the other. Here are the details:

- Change the number of latent tokens C and fix $N = 24, K = 5$. we can find that the performance of LSM is stable w.r.t. different choices of C, which may come from the equivalence of different latent tokens.
- Change the number of basis operators N and fix $C = 4, K = 5$. Generally, larger N will bring better results, while larger N will also cause more computation cost and optimization problems, which explains why the model performance drops slightly at $N = 40$ and $N = 48$. Note that the $N = 0$ setting is equivalent to the without neural spectral block situation, where the model performance will drop seriously.
- Change the number of scales K and fix $C = 4, N = 24$. In general, adding scales K will improve the model's performance. But the model with too many scales is unimplementable due to the limitation of input resolution.

Table 11. Model performances on Elasticity-G with different number of latent tokens C, number of basis operators N and number of scales K. "/" means that the experiment is unimplementable.

NUMBER OF LATENT TOKENS C	0	1	2	3	4	5	6
MSE	/	0.0415	0.0415	0.0409	0.0408	0.0411	0.0415
NUMBER OF BASIS OPERATORS N	0	8	16	24	32	40	48
MSE	0.0433	0.0415	0.0418	0.0408	0.0406	0.0413	0.0416
NUMBER OF SCALES K	3	4	5	6	7	8	9
MSE	0.0428	0.0412	0.0408	0.0400	0.0402	/	/

Overall, LSM is stable to these three hyperparameters, where C is robust and easy to tune in the range of 3 to 5, N is robust in 24 to 40 and K is stable in 5 to 7. Thus, the setting of number of latent tokens C as 4, number of basis operators N as 24 and number of scales K as 5 can aptly trade off the efficiency and performance.

H. Training Stability

We provide the model training curves on different benchmarks in Figure 9. From Figure 9, we can observe that in addition to the consistent state-of-the-art performance in all benchmarks, LSM also presents comparable training stability w.r.t. the well-acknowledged FNO (Li et al., 2021).

Besides, we also repeat all the experiments five times, where the standard deviations of LSM performance are within 0.0001 for Elasticity-P, Elasticity-G and Plasticity, Darcy and Airfoil, and within 0.0002 for Navier-Stokes and Pipe.

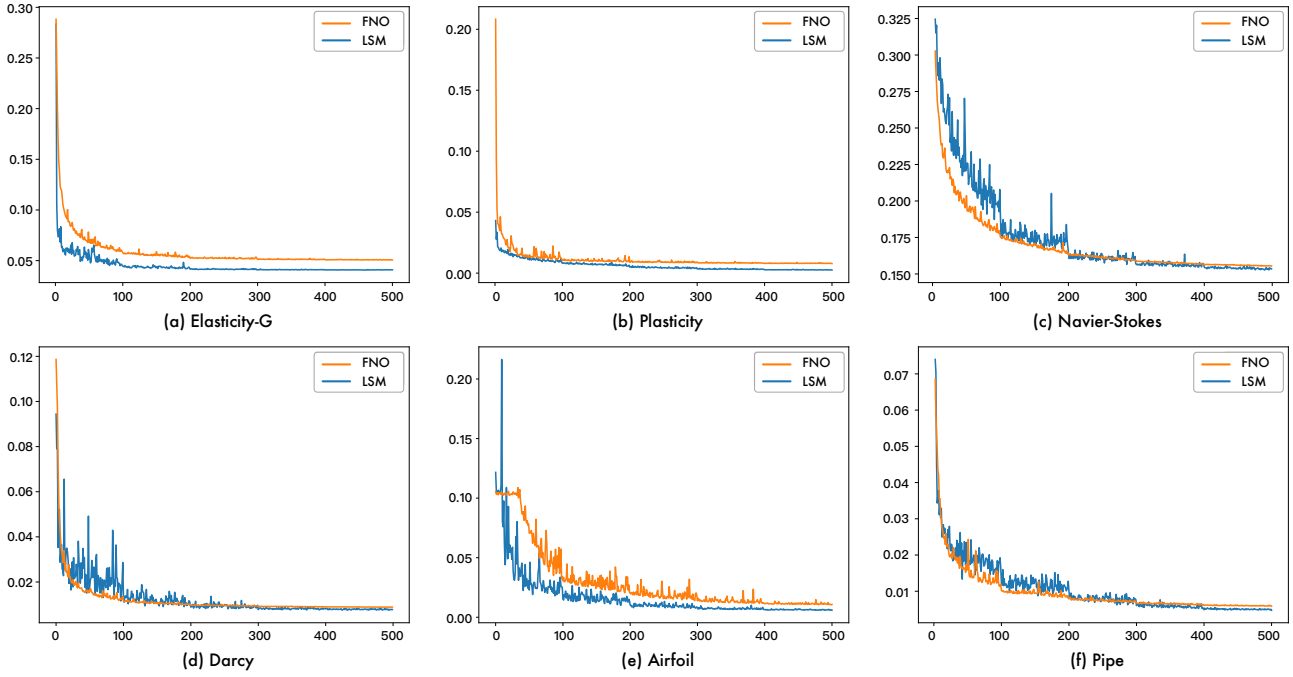

Figure 9. Model training curves, where the x-axis means the number of epochs and the y-axis is MSE performance in the test set.

I. Implementation Details

All experiments are repeated five times, implemented in PyTorch (Paszke et al., 2019) and conducted on a single NVIDIA RTX 3090 24GB GPU. We have provided the training curves and standard deviations in Appendix H. For all methods, the performance at the final epoch is recorded as the final result. Here are the implementation details of the LSM model.

I.1. Model Configurations

Here, we present the detailed model configurations for LSM. In the beginning, we will pad the input with zeros properly to resolve the division problem in model configurations.

Table 12. Model configurations for LSM.

MODEL DESIGNS	HYPERPARAMETERS	VALUES
HIERARCHICAL PROJECTION NETWORK	NUMBER OF LATENT TOKENS C	4
	NUMBER OF SCALES K	5
	DOWNSAMPLE RATIO $r = \frac{\lvert\mathcal{D}^{k+1}\rvert}{\lvert\mathcal{D}^{k}\rvert}$	0.5
	CHANNELS OF EACH SCALE $\{d^1_{\text{MODEL}}, \cdots, d^K_{\text{MODEL}}\}$	$\{32, 64, 128, 128, 128\}$
	CHANNELS OF LATENT TOKENS AT EACH SCALE $\{d^1_{\text{LATENT}}, \cdots, d^K_{\text{LATENT}}\}$	$\{32, 64, 128, 128, 128\}$
	PATCHES OF EACH SCALE $\{P_1, \cdots, P_K\}$	$\{256, 64, 16, 4, 1\}$
NEURAL SPECTRAL BLOCK	NUMBER OF BASIS OPERATORS N	24

I.2. Model Architecture

In this section, we will illustrate the operations in the patchified multiscale architecture.

Downsample. Given deep features $\{\boldsymbol{x}^k(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^k}$ at the k-th scale, the downsample operation is to aggregate deep features in a local region with maximum pooling and convolution operations, which can be formulized as follows:

$$\{\boldsymbol{x}^{k+1}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^{k+1}} = \mathrm{Conv}\left(\mathrm{MaxPool}\left(\{\boldsymbol{x}^k(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^k}\right)\right),\ k \text{ from 1 to } (K-1). \tag{29}$$

Upsample. Given the deep features $\{\widehat{\boldsymbol{y}}^{k+1}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^{k+1}}$, $\{\widehat{\boldsymbol{y}}^{k}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^{k}}$ at the $(k+1)$-th and k-th scales respectively, which have been projected from latent space back to coordinate space, the upsample process is to fuse the interpolated $k+1$-th features and the k-th features with local convolution, which can be formulized as follows:

$$\{\widehat{\boldsymbol{y}}^{k}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^{k}} = \mathrm{Conv}\left(\mathrm{Concat}\left(\left[\mathrm{Interpolation}\left(\{\widehat{\boldsymbol{y}}^{k+1}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^{k+1}}\right), \{\widehat{\boldsymbol{y}}^{k}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{D}^{k}}\right]\right)\right),\ k \text{ from } (K-1) \text{ to } 1, \tag{30}$$

where we adopt the bilinear $\mathrm{Interpolation}(\cdot)$ for 2D data and the trilinear $\mathrm{Interpolation}(\cdot)$ for 3D data.

Patchify and De-Patchify. The patchify operation is to split the coordinate set into several non-overlapping local regions with an equal number of coordinates. This process of patchify at the k-th scale is formulized as follows:

$$\{\mathcal{D}^k_j\}_{j=1}^{P_k} = \mathrm{Patchify}(\mathcal{D}^k). \tag{31}$$

And the depacthify operation is just to splice the patches in different local regions, that is $\mathcal{D}^k = \text{De-Patchify}(\{\mathcal{D}^k_j\}_{j=1}^{P_k})$.

I.3. Benchmark Construction

Time-dependent tasks. In our benchmarks, both Plasticity and Navier-Stokes are time-dependent. For the Plasticity benchmark, since its input is the boundary condition and output is the mesh displacement over time, we adopt the 3D-version LSM for Plasticity experiments, where all the convolution (Conv), max pooling ($\mathrm{MaxPool}$), interpolation ($\mathrm{Interpolation}$) and patchify ($\mathrm{Patchify}$) operations are in the 3D space. As for the Navier-Stokes, since it is an autoregressive task, we still adopt the 2D-version LSM like other benchmarks and predict the next frame step by step. Note that the neural spectral block is applied to the independent latent tokens, hence it is unchanged for both 2D- and 3D-versions.

Baselines. We implement all the baselines based on their official code. Note that we focus on the operator-learning paradigm, thus we only adopt their model and uniformly use the L2 loss during training for fairness. Especially, for the MWT (Gupta et al., 2021), we pad the inputs with zeros to make the input resolutions as integer power of two.

J. Efficiency

To present a clear efficiency comparison among different models, we fix the model input as a regular grid with the resolution of $S \times S$, the input channel as 1 and the batch size as 1. Then, we record the model parameter, GPU memory and running time under different choices of S, which are selected from $\{64, 128, 256, 512, 1024\}$. Besides, we also calculate the performance ranking of baselines on each benchmark and present the model efficiency in the order of averaged ranking.

From Table 13, we can obtain the following observations:

- *There is an evident gap between solid and fluid physics*. From Table 13, we can find that the performances of baselines are quite different in solid and fluid physics. Concretely, the top 5 models in solid and fluid benchmarks are distinct, except LSM (rank 1st) and F-FNO (rank 5th). This result also indicates that there is a large gap between solid and fluid physics and previous methods cannot cover different disciplines of physics well.
- *LSM presents favorable generality in varied physics.* From the performance ranking on seven tasks, it is observed that other baselines fluctuate greatly in different benchmarks. In contrast, it is impressive that our proposed LSM can achieve consistent state-of-the-art on these varied physics, demonstrating the model generality.
- *LSM presents competitive efficiency in high-dimensional inputs.* We have provided the efficiency comparison for 256×256 inputs in Figure 5 of the main text. If we focus on the running time for the inputs with the resolution of 512×512 and 1024×1024, we can find that LSM is clearly faster than U-NO (rank 2nd), U-Net (rank 3rd).

K. Limitations

As we discussed in the main results (Table 2), performance under various resolutions (Table 8), efficiency comparison (Table 13) and transfer learning tasks (Table 4), LSM can precisely solve the PDEs with good efficiency and transferability, covering both solid and fluid physics and various geometrics. Although LSM can achieve advanced performance, it still holds some limitations. Here are the discussions.

One potential limitation of LSM may lie in the model generality among different PDEs, where we expect a zero-shot PDE solver like the foundation models in natural language processing, such as GPT-3 (Brown et al., 2020), T5 (Raffel et al., 2020) and etc. Note that due to the inherent complexity of PDEs, a small perturbation to the coefficients of PDEs may change their property seriously, such as condition number, with or without explicit solution, the convergence of infinite values (Evans, 2010). Thus, to tackle this potential limitation, we need first to explore the fundamental question of "whether there is a universal solution to all PDEs or not," which is clearly far beyond the scope of our paper. Thus, we would like to leave this problem as future work.

L. Societal Impacts

Real-world applications. In this paper, we present the LSM as a practical deep solver for high-dimensional PDEs. Given the state-of-the-art performance of LSM, this paper may help many PDE-related applications, such as airfoil design, the load-bearing tests of civil engineering, weather forecasting, etc. Especially, LSM can also present a favorable transferability to limited data scenarios (Table 4), which is important for fast-adaption to new scenarios.

Academic research. Unlike previous methods, LSM attempts to solve high-dimensional PDEs through a new technology roadmap: going beyond high-dimensional coordinate information and solving PDEs in latent space, which can be a good supplement for the operator learning community.

This paper only focuses on the scientific problem. All the datasets are generated by public tools and strictly follow the corresponding licenses (Appendix B). Thus, there is no potential ethical risk or negative social impact.

Table 13. Model efficiency comparison and their rankings in solid, fluid and all seven benchmarks, where we select the top 10 methods. A smaller ranking means better performance. Efficiency is evaluated on inputs with size $S \times S$ during the training phase. The batch size is set to 1. Running time is averaged from 10^3 iterations. "/" indicates the out-of-memory situation.

Input Size ($S \times S$)		Parameter	GPU Memory	Running Time	Ranking			
Model	S	(MB)	(MB)	(s / iter)	Seven tasks*	Solid*	Fluid†	Averaged‡
LSM (ours)	64	2.002	1409	0.0353	(1, 1, 1, 1, 1, 1, 1)	1.0	1.0	1.0
	128	2.002	1679	0.0359				
	256	2.002	1959	0.0411				
	512	2.002	3019	0.0602				
	1024	2.002	7859	0.2002				
U-NO	64	1.307	1345	0.0347	(6, 2, 2, 4, 7, 4, 9)	3.3	8.0	4.9
	128	1.307	1381	0.0354				
	256	1.307	1603	0.0397				
	512	1.307	2473	0.0989				
	1024	1.307	6833	0.3335				
U-Net	64	4.332	1171	0.0321	(3, 8, 5, 6, 4, 6, 4)	5.3	5.0	5.1
	128	4.332	1243	0.0307				
	256	4.332	1515	0.0450				
	512	4.332	2429	0.1589				
	1024	4.332	6235	0.8100				
FNO	64	2.368	1137	0.0202	(2, 6, 6, 3, 6, 8, 5)	4.6	7.3	5.1
	128	2.368	1179	0.0203				
	256	2.368	1349	0.0147				
	512	2.368	1975	0.0401				
	1024	2.368	4591	0.1270				
HT-Net	64	3.285	1175	0.0406	(10, 3, 10, 5, 3, 2, 3)	7.6	3.2	5.1
	128	3.285	1283	0.0415				
	256	3.285	1749	0.0469				
	512	3.285	3267	0.1300				
	1024	3.285	9259	0.4581				
F-FNO	64	0.218	1089	0.0303	(7, 4, 4, 9, 2, 5, 6)	5.0	5.5	5.3
	128	0.218	1169	0.0303				
	256	0.218	1437	0.0202				
	512	0.218	2457	0.0825				
	1024	0.218	6443	0.3248				
U-FNO	64	3.990	1169	0.0400	(4, 5, 3, 7, 9, 9, 2)	4.0	6.7	5.6
	128	3.990	1241	0.0400				
	256	3.990	1499	0.0471				
	512	3.990	2537	0.1047				
	1024	3.990	6869	0.3217				
WMT	64	3.106	1145	0.0615	(9, 7, 7, 2, 5, 3, 7)	7.6	4.2	5.7
	128	3.106	1201	0.0720				
	256	3.106	1407	0.0900				
	512	3.106	2165	0.1118				
	1024	3.106	5241	0.3120				
Galerkin Trasnformer	64	6.319	1233	0.0252	(5, 10, 8, 10, 8, 7, 8)	7.6	8.2	8.0
	128	6.319	1277	0.0260				
	256	6.319	1675	0.0681				
	512	6.319	3175	0.2225				
	1024	2.002	9333	0.8688				
Swin Trasnformer	64	0.538	1135	0.0615	(8, 9, 9, 8, 10, 10, 10)	8.6	9.5	9.1
	128	0.538	1261	0.0333				
	256	0.538	1789	0.0355				
	512	0.538	3759	0.1150				
	1024	/	/	/				

$\star$ The rankings are presented in the order of (Elasticity-P, Elasticity-G, Plasticity, Navier-Stokes, Darcy, Airfoil, Pipe).
$*$ Top 5 methods of solid benchmarks: LSM (ours), U-NO (2022), U-FNO (2021), FNO (2021), F-FNO (2023).
† Top 5 methods of fluid benchmarks: LSM (ours), HT-Net (2022), WMT (2021), U-Net (2015), F-FNO (2023).
‡ Top 5 methods of all benchmarks: LSM (ours), U-NO (2022), U-Net (2015), FNO (2021), HT-Net (2022).