A METHODS

A.1 NOTATION

Symbol	Used for
$\boldsymbol{X}$	A discretization of the domain D, which is used to train the model.
$\boldsymbol{X'}$	A discretization of the domain D, which is used to evaluate the model.
$\boldsymbol{x}$	Coordinate of spatial point in D.
n_s	Number of spatial points of seen meshes for training, as $\lvert\boldsymbol{X}\rvert = n_s$.
n_s'	Number of spatial points of unseen meshes for inference, as $\lvert\boldsymbol{X'}\rvert = n_s'$.
n_t	Number of input timestamps as number of input time-dependent PDE's initial states.
n_t'	Number of output timestamps as number of output time-dependent PDE's future states.
a	Input function, where $a \in \mathcal{A}(D; \mathbb{R}^{d_a})$ means for $\boldsymbol{x} \in D$, $a(\boldsymbol{x}) \in \mathbb{R}^{d_a}$. In time-dependent PDEs, $a = u(\cdot, \boldsymbol{T})$, where $\boldsymbol{T} = \{t_i\}_{i=1}^{n_t}$.
u	Target function for approximation, where $u \in \mathcal{U}(D; \mathbb{R}^{d_u})$ means for $\boldsymbol{x} \in D$, $u(\boldsymbol{x}) \in \mathbb{R}^{d_u}$.
v	Representation function, where $v \in \mathcal{U}(D; \mathbb{R}^{d_v})$ means for $\boldsymbol{x} \in D$, $v(\boldsymbol{x}) \in \mathbb{R}^{d_v}$, which is a function obtained by a lifter which project a into a higher dimensional space.
$\mathcal{G}_\theta$	Approximation opeartor, where $\mathcal{G}_\theta(a) \approx u$.
μ	The probability measure for sampling Spatial points $\boldsymbol{x}$, which supported on D.
$\mathcal{C}$	Cost functional as the minimum optimization target.
$\mathcal{F}$	Discrete Fourier transform for equispaced spatial points.
$\mathcal{F}^{-1}$	Discrete Inverse Fourier transform for equispaced spatial points.
P	Project operator, with $P(a)(\boldsymbol{x}) \in \mathbb{R}^{d_v}$.
Q	Project operator, with $Q(v)(\boldsymbol{x}) \in \mathbb{R}^{d_u}$.
v^{t}	t-th iterative representation function after kernel operators' update.
$\mathcal{K}_\phi$	Kernel integral operator mapping, which maps a to a bounded linear opeartor, with parameter ϕ.
W	Linear transform on the d_v dimension (chennel) of $v(\boldsymbol{x}) \in d_v$.
R_ϕ	Fourier transform of a periodic kernel function, which is learnable parameters in a single iterative process in FNO.
ChannelMix	Channel-mixing operator as a linear transform in the dimensoion of channel (d_v).
TokenMix	Token-mixing operator as a linear transform in the dimensoion of spatial points (n_s).
$\tilde{\mathcal{F}}$	Proposed non-equispaced Fourier transform, where $\tilde{\mathcal{F}} = (\mathcal{F} \circ \mathcal{H}(a))$.
m_s	Spatial points' number on resampled equi-spaced points.
$\mathcal{H}$	Interpolation opeartor to interpolate the non-equispaced spatial points on equispaced spatial grids.
τ	Parameter in Gaussian interpolation kernels controlling smoothness of the kernel.
$\boldsymbol{\mu}$	Parameter in Gaussian interpolation kernels, as the mean of Gaussian kernels
$\mathcal{H}_\eta$	Interpolation opeartor mapping, where $\mathcal{H}_\eta(a)$ is a interpolation opeartor used to map signals on the non-equispaced spatial points to equispaced spatial grids.
$\mathcal{H'}_\zeta$	Interpolation opeartor mapping, where $\mathcal{H}'_\zeta(a)$ is a interpolation opeartor used to map signals on the equispaced spatial points to non-equispaced spatial grids.
$\mathcal{N}(\boldsymbol{x})$	Neighborhood of spatial point $\boldsymbol{x}$.

Table A1: Glossary of Notations used in this paper.

A.2 GRAPH CONSTRUCTION

Neighborhood construction. Instead of using K-nearest neighborhood method, the neighborhood system in the interpolation layer is constructed by ϵ-ball, because in equispace scenarios, there will be multiple points as K-th nearest neighbor at the same time. For point $\boldsymbol{x}$, its neighbor is defined according to

$$\begin{cases} d(\boldsymbol{x}, \boldsymbol{x}_i) \leq \epsilon & \boldsymbol{x}_i \in \mathcal{N}(\boldsymbol{x}); \\ d(\boldsymbol{x}, \boldsymbol{x}_i) > \epsilon & \boldsymbol{x}_i \notin \mathcal{N}(\boldsymbol{x}). \end{cases} \tag{11}$$

For given c defined in Sec. 3.2, we can restict ϵ so that $\mathbb{E}_{x\sim\mu}[|\mathcal{N}(\boldsymbol{x})|] < c\log(n_s)$.

A.3 PROOF OF THEOREM 3.1.

Our proof is mostly based on Chen & Chen (1995) and Kovachki et al. (2021). For notation simplicity, in the proof, we directly write $\mathcal{H}_\eta(a)$ as $\mathcal{H}_\eta$ as the linear operator.

Lemma A1. *Let $\mathcal{X}$ be a Banach space, and $\mathcal{U} \subseteq \mathcal{X}$ a compact set, and $\mathcal{K} \subset \mathcal{X}$ a dense set. Then, for any $\epsilon > 0$, there exists a number $n \in \mathbb{N}$, and a series of continuous, linear functionals $G_1, G_2, \ldots, G_n \in C(\mathcal{U}; \mathbb{R})$, and elements $\varphi_1, \ldots, \varphi_n \in \mathcal{K}$, such that*

$$\sup_{u\in\mathcal{U}} ||v - \sum_{j=1}^{n} G_j(v)\varphi_j||_{\mathcal{X}} \leq \epsilon \tag{12}$$

The proof is given in **Lemma 7.** in Kovachki et al. (2021), and **Theorem 3.** and **4.** for reference .

Theorem A2. *Let $D \subseteq \mathbb{R}^d$ be compact domain. Let $\mathcal{U}$ be a separable Banach space of real-valued functions on D, such that $C(D, \mathbb{R}) \subseteq \mathcal{U}$ is dense. Suppose $\mathcal{U} = L^p(D; \mathbb{R})$ for any $1 < p < \infty$. ν is a probability measure supported on $\mathcal{U}$ and assume that, $\mathbb{E}_{v\sim\nu}||v||_{\mathcal{U}} < \infty$ for any $v \in \mathcal{U}$. μ is a probabilistic measure supported on D, which defines the inner porduct of Hilbert space $\mathcal{U}$ as $< f, g >_{\mathcal{U}} = \int_D f(\boldsymbol{x})g(\boldsymbol{x})d\mu(\boldsymbol{x})$. Then, there exists a neural network $h_\eta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ whose activation functions are of the Tauber-Wiener class, such that*

$$||v - \mathcal{H}(v)||_{\mathcal{U}} \leq \epsilon,$$

where $\mathcal{H}(v)(\boldsymbol{x}) = \int_D h_\eta(\boldsymbol{x}, \boldsymbol{y})v(\boldsymbol{y})d\mu(\boldsymbol{y})$.

Proof. Since $\mathcal{U}$ is a Polish sreipace, we can find a compact set $\mathcal{K}$, such that $\nu(\mathcal{U} \setminus \mathcal{K}) \leq \epsilon$. Therefore, **Lemma A1** can be applied, to find a number $n \in \mathbb{N}$, a series of continuous linear functionals $G_j \in C(\mathcal{U}; \mathbb{R})$ and functions $\varphi_j \in C(D; \mathbb{R})$ such that

$$\sup_{v\in\mathcal{K}} ||v - \sum_{j=1}^{n} G_j(v)\varphi_j||_{\mathcal{U}} \leq \epsilon$$

Denote $\hat{\mathcal{H}}_n(v) = \sum_{j=1}^n G_j(v)\varphi_j$, and let $1 < q < \infty$ be the Hölder conjugate of p. Since $\mathcal{U} = L^p(D; \mathbb{R})$, by Reisz Representation Theorem, there exists functions $g_j \in L^q(D; \mathbb{R})$, such that $G_j(v) = \int_D v(\boldsymbol{x})g_j(\boldsymbol{x})d\mu(\boldsymbol{x})$ for $j = 1, \ldots, n$ and $v \in L^p(D; \mathbb{R})$. By density of $C(D; \mathbb{R})$ in $L^q(D; \mathbb{R})$, we can find functions $\psi_1, \ldots, \psi_n \in C(D; \mathbb{R})$, such that

$$\sup_{j\in\{1,\ldots,n\}} ||\psi_j - g_j||_{L^q(D;\mathbb{R})} \leq \epsilon/n.$$

Then, we define $\tilde{\mathcal{H}}_n : L^p(D; \mathbb{R}) \to C(D; \mathbb{R})$ by

$$\tilde{\mathcal{H}}_n(v) = \sum_{j=1}^{n} \int_D \psi_j(\boldsymbol{y})v(\boldsymbol{y})d\mu(\boldsymbol{y})\varphi_j(\boldsymbol{x})$$

For the universal approximation (density) (Hornik et al., 1989) of neural network, we can find a Multi-layer Feedforward network $h_\eta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ whose activation functions are of the Tauber-Wiener class, such that

$$\sup_{\boldsymbol{x},\boldsymbol{y}\in D} |h_\eta(\boldsymbol{x}, \boldsymbol{y}) - \sum_{j=1}^{n} \psi_j(\boldsymbol{y})\varphi_j(\boldsymbol{x})| \leq \epsilon$$

Let $\mathcal{H}_\eta(\boldsymbol{x}) = \int_D h_\eta(\boldsymbol{x}, \boldsymbol{y}) v(\boldsymbol{y}) d\mu(\boldsymbol{y})$. Then, there exists a constant $C_1 > 0$, such that

$$||\hat{\mathcal{H}}_n(v) - \mathcal{H}(v)||_{L^p(D;\mathbb{R})} \leq C_1(||\hat{\mathcal{H}}_n(v) - \tilde{\mathcal{H}}_n(v)||_{L^p(D;\mathbb{R})} + ||\tilde{\mathcal{H}}_n(v) - \mathcal{H}(v)||_{L^p(D;\mathbb{R})})$$

For the first term, there is a constant $C_2 > 0$, such that

$$\begin{aligned}
||\hat{\mathcal{H}}_n(v) - \tilde{\mathcal{H}}_n(v)||_{L^p(D;\mathbb{R})} &\leq C_2 \sum_{j=1}^{n} || \int_D v(\boldsymbol{y})(g_j(\boldsymbol{y}) - \psi_j(\boldsymbol{y})) d\mu(\boldsymbol{y}) \varphi_j ||_{L^p(D;\mathbb{R})} \\
&\leq C_2 \sum_{j=1}^{n} ||v(\boldsymbol{y})||_{L^p(D;\mathbb{R})} ||g_j(\boldsymbol{y}) - \psi_j(\boldsymbol{y})||_{L^q(D;\mathbb{R})} ||\varphi_j||_{L^p(D;\mathbb{R})} \\
&\leq C_3 \epsilon ||v(\boldsymbol{y})||_{L^p(D;\mathbb{R})}
\end{aligned}$$

for some $C_3 > 0$. And for the second term,

$$\begin{aligned}
||\tilde{\mathcal{H}}_n(v) - \mathcal{H}(v)||_{L^p(D;\mathbb{R})} &= || \int_D v(\boldsymbol{y}) (\sum_{j=1}^{n} \psi_j(\boldsymbol{y}) \varphi_j(\cdot) - h_\eta(\cdot, \boldsymbol{y})) d\mu(\boldsymbol{y}) ||_{L^p(D;\mathbb{R})} \\
&\leq |D| \epsilon ||v||_{L^p(D;\mathbb{R})}
\end{aligned}$$

Therefore, there is a constant $C > 0$, such that

$$\int_{\mathcal{U}} ||\hat{\mathcal{H}}_n(v) - \tilde{\mathcal{H}}_n(v)||_{\mathcal{U}} d\nu(v) \leq \epsilon C \mathbb{E}_{v \sim \nu} ||v||_{\mathcal{U}}$$

Because of the assumption that $\mathbb{E}_{v \sim \nu} ||v||_{\mathcal{U}} < \infty$, and ϵ is arbitary, then

$$||v - \mathcal{H}(v)||_{\mathcal{U}} \leq ||v - \hat{\mathcal{H}}_n(v)||_{\mathcal{U}} + ||\hat{\mathcal{H}}_n(v) - \mathcal{H}(v)||_{\mathcal{U}},$$

the proof is complete. □

Corollary A3. *Define $\mathcal{H}_\eta(v) = \int_D h_\eta(\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x}, a(\boldsymbol{y})) v(\boldsymbol{y}) d\mu(\boldsymbol{y})$, the interpolation operator can also approximate v to any precision ϵ.*

Proof. We use a one-layer neural network $h_\eta : D \times D \to \mathbb{R}$ as an example, which is defined as $h_\eta(\boldsymbol{x}, \boldsymbol{y}, a(\boldsymbol{y}) = \sigma(\sum_{i=1}^{d} w_{x,i} x^{(i)} + w_{y,i} y^{(i)} + b)$. We can rewrite it as

$$h_\eta = \sigma(\sum_{i=1}^{d} w_{x,i}(x^{(i)} - y^{(i)}) + (w_{y,i} + w_{x,i}) y^{(i)} + \sum_{j=1}^{d_a} w_{a,j} a^{(j)}(\boldsymbol{y}) + b),$$

where $w_{a,j} = 0$. □

Corollary A4. *The **Theorem A2** and **Corollary A3** can be extended for $v : D \to \mathbb{R}^{d_v}$, where $d_v > 1$.*

Proof. As $v = (v^{(1)}, v^{(2)}, \ldots, v^{(d_v)})$, for each $v^{(j)}$, a single neural network can be used for approximation. Moreover, in implementation, we make h_η fully-connected, to improve the expressivity. □

Remark. *As $\sum_{\boldsymbol{x}_i \in \boldsymbol{X}} v(\boldsymbol{x}_i) h_\eta(\boldsymbol{x} - \boldsymbol{x}_i, \boldsymbol{x}_i, a(\boldsymbol{x}_i))$ is the unbiased estimation of $\mathbb{E}_{\boldsymbol{y} \sim \mu}(h_\eta(\boldsymbol{x}, \boldsymbol{y}) v(\boldsymbol{y}))$, we use the Equation. (8) for the approximation.*

B EXPERIMENTS

B.1 BENCHMARK METHOD DESCRIPTION

Vision Mixers. We provide a framework for vision mixers as PDE solvers, including VIT, MLP-MIXER, FNET, GFN, FNO, PFNO and our NFS. The intermediate architecture of mixing layers is shown in Fig. B1. The code of our framework will be released soon. And the resampling and back-sampling methods are stacked before 'Equispaced Input' and 'Equispaced Output'. In this way, the description of the Vision Mixers included in our framework can be decribed by different modules, as shown in Table. B1. All the trials on Vision Mixers set *embedding size* as 32, *batch size* as 4, *layer number* of the intermediate equispaced mixing layers as 2. In FNO and PFNO, the truncated $K_{\max}$ is set as 16. The *patch size* of Vision Mixers with patchwise embedding are set as $[4, 2]$ in 1-d PDEs and $[4, 4, 2]$ in 2-d PDEs. The interpolation layers in NFS are composed of one layer of feed-forward network whose perceptron unit is equal to $4\times$ *embedding size* of the model.

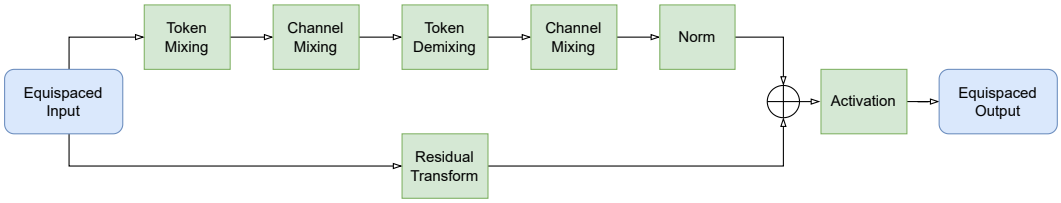


Figure B1: The architecture of Vision Mixers.

Table B1: Description of Vison Mixers in the unifying framework module by module.

Modules	VIT	MLPMIXER	FNET	GFN	FNO	PFNO	NFS
Resampling	Patchwise Embedding	Patchwise Embedding	Patchwise Embedding	Patchwise Embedding	Identity	Patchwise Embedding	Interpolation
Token Mixing	Attention	MLP	Fourier	Fourier	Fourier	Fourier	Fourier
Channel Mixing	Linear	Linear	Linear	Elementwise Product	Low Frequency MatMultiply	Low Frequency MatMultiply	Low Frequency MatMultiply
Token Demixing	Identity	Identity	Identity	Inverse Fourier	Inverse Fourier	Inverse Fourier	Inverse Fourier
Channel Mixing	Identity	Identity	Identity	Linear	Identity	Identity	Identity
Normalization	LayerNorm	LayerNorm	Complex LayerNorm	LayerNorm	Identity	LayerNorm	LayerNorm
Residual	Identity	Identity	Identity	Identity	1x1 Conv	1x1 Conv	1x1 Conv
Activation	Gelu	Gelu	Complex Gelu	Gelu	Gelu	Gelu	Gelu
Back Sampling	Linear+ Rearrange	Linear+ Rearrange	Linear+ Rearrange	Linear+ Rearrange	Identity	Linear+ Rearrange	Interpolation

Graph Spatio-Temporal Models. The evaluated graph spatio-temporal neural networks are based on recurrent neural network for dynamics modeling, where the spatial dependency is modeled by graph neural networks. The spatial and temporal modules for AGCRN, DCRNN and GCGRU are shown in Table. B2. MPPDE used different architecture, with the pushforward trick used for taining, with *rolling* equaling 1 and *time window* equaling to 10 . All the trials on these graph spatio-tempral models set *embedding size* as 64, except MPPDE as 128. *Batch size* is set as 4. When the graph convolution needs multi-hop message-passing, we set the hop as 2. For MPPDE, the layer number of GNNs is 6. The *embedding dimension* in AGCRN is set as 2.

Table B2: Description of different graph spatio-temporal models

Methods	Spatial module	Temporal module
GCGRU Seo et al. (2016)	Cheb Conv Defferrard et al. (2017)	GRU
DCRNN Li et al. (2018)	Diff Conv Atwood & Towsley (2016)	GRU
AGCRN Bai et al. (2020)	Node Similarity Bai et al. (2020)	GRU

B.2 DATA GENERATION

Burgers' Equation. The initial condition $u_0(x)$ is generated according to $u_0 \sim N(0.625(-\Delta + 25I)^{-2})$ with periodic boundary conditions. ν is set as 0.01. $x \in [0, 1]$ and $t \in [0, 1]$. The spatial resolution is 1024, and time resolution is 200. The dataset generation follows FNO's protocol, which can be downloaded from its source code on official Github.

KdV Equation. The equation is written as

$$\partial_t u(x,t) + 3\partial_x u^2(x,t) + \partial_x^3 u(x,t) = 0, \tag{13}$$

where $x \in [0, 1]$. The initial condition $u_0(x)$ is calculated as

$$u(x, 0) = \sum_{i=1}^{K} 0.5 c_i \cos(0.5\sqrt{c_i + b_i} x - a_i)$$

where $c_i \sim N(0, \sigma_i)$, and $a_i, b_i > 0$. The spatial resolution is 1024. The dataset is generated by `scipy` package, with `fftpack.diff` used as pesudo-differential method and `odeint` used as forward Euler method.

Darcy Flow. The equation is written as

$$\begin{aligned} -\nabla(a(\boldsymbol{x})\nabla u(\boldsymbol{x})) &= f(\boldsymbol{x}) \qquad && \boldsymbol{x} \in (0, 1)^2 \\ u(\boldsymbol{x}) &= 0 && \boldsymbol{x} \in \partial[0, 1]^2 \end{aligned} \tag{14}$$

The original resolution is 256×256. $a(\boldsymbol{x})$ is generated by Gaussian random field, and we directly establish the operator to learn the mapping of a to u.

NS Equation. Our generation of NS Equation is based on FNO's Appendix. A.3.3, with the forcing is kept fixed. The original spatial resolution is 128×128, and time resolution is 200.

B.3 Complete Results on Model Comparison

Here we give complete results on the four Equations. Table. B3 give the performance comparison on Darcy flow of both equispaced and non-equispaced scenarios. Table. B4 and B5 gives performance comparison in equispaced scenarios on the other three time-dependent problems. Table. B6 and B7 gives performance comparison in non-equispaced scenarios on the other three time-dependent problems. In all the tasks except `Darcy Flow`, the depth of layer is set as 2, and $k_{\max} = 16$ in both NFS and FNO. However, we find in `Darcy Flow`, $k_{\max}$ should be set much larger, or the loss will not decrease. In the reported results, $k_{\max} = 32, 64, 128$ in `Darcy Flow`.

Table B3: Performance comparison on Darcy Flow.

	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
	`Darcy Flow` ($r = 64$)		`Darcy Flow` ($r = 128$)		`Darcy Flow` ($r = 256$)	
VIT	0.5073±0.0411	0.8468±0.0432	0.9865±0.0002	1.6195±0.0007	1.1078±0.0021	1.8444±0.0023
MLPMIXER	0.4970±0.0021	0.8228±0.0034	0.8909±0.0099	1.4221±0.0118	0.9125±0.0024	1.6459±0.0032
GFN	0.4739±0.0016	0.8345±0.0019	0.8659±0.0046	1.4237±0.0071	0.9618±0.0124	1.6139±0.0128
FNO	0.4289±0.0051	0.7740±0.0046	0.7086±0.0045	0.1324±0.0019	0.9075±0.0051	1.4940±0.0046
NFS	**0.1497**±0.0005	**0.1962**±0.0007	**0.2254**±0.0007	**0.7245**±0.0009	**0.4216**±0.0033	**0.8578**±0.0041
	`Darcy Flow` ($n_s = 1024$)		`Darcy Flow` ($n_s = 4096$)		`Darcy Flow` ($n_s = 16384$)	
DCRNN	1.8146±0.0060	2.6352±0.0029	1.7629±0.0003	2.5760±0.0001	OOM	OOM
AGCRN	1.6938±0.0001	2.4440±0.0001	1.7336±0.0001	2.4167±0.0001	OOM	OOM
GCGRU	1.7633±0.0001	2.5696±0.0001	1.7403±0.0001	2.5363±0.0001	OOM	OOM
MPPDE	0.6673±0.0009	0.9290±0.0012	0.5608±0.0053	0.8424±0.0051	0.6384±0.0005	0.8748±0.0005
NFS	**0.1727**±0.0047	**0.2311**±0.0066	**0.1430**±0.0007	**0.1914**±0.0014	**0.2379**±0.0007	**0.3489**±0.0009

NFS fails to model the non-equispaced Burgers' Equation when n_t is set as 1, in which the performance is far from it can achieve in equispaced scenarios. Such problem will be our future work.

Table B4: Performance comparison with Vision Mixer benchmarks on different equations ($n_t = 1$). Validation loss on `Burgers'` ($n_t = 1$) of VIT, GFN, and FNO does not converge. The results show that the early-stopping occurs in the begining of training.

Vision Mixers	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
	`Burgers'` ($r = 512, n_t' = 10$)		`Burgers'` ($r = 512, n_t' = 40$)		`Burgers'` ($r = 1024, n_t' = 20$)	
VIT	201.6539±0.5284	231.9138±0.8403	183.6696±0.3015	210.6237±0.6767	195.5858±0.7706	224.4712±1.2472
MLPMIXER	201.6547±0.0671	231.9163±0.0263	183.6535±0.0599	210.6160±0.0305	195.5960±0.0240	224.4791±0.0132
GFN	201.6557±0.9513	231.9122±0.9535	183.6674±0.4893	210.6165±0.4831	195.5918±0.0471	224.4736±0.0681
FNO	201.6527±1.1415	231.9119±1.6747	183.6696±0.3015	210.6299±0.4983	195.5902±0.9304	224.4723±0.9230
NFS	**0.1806**±0.0005	**0.2669**±0.0010	**0.3570**±0.0008	**0.5340**±0.0009	**0.4344**±0.0014	**0.6092**±0.0017
	`KdV` ($r = 512, n_t' = 10$)		`KdV` ($r = 512, n_t' = 40$)		`KdV` ($r = 1024, n_t' = 20$)	
VIT	0.2808±0.0006	0.3938±0.0009	0.3428±0.0012	0.6832±0.0016	0.3066±0.0003	0.5461±0.0003
MLPMIXER	0.2732±0.0054	0.4259±0.0088	**0.3336**±0.0045	**0.5923**±0.0081	0.2872±0.0005	0.5235±0.0006
GFN	0.2587±0.0032	0.3490±0.0056	0.3086±0.0223	0.5952±0.0338	0.2011±0.0074	0.3464±0.0063
FNO	0.2619±0.0069	0.3849±0.0107	0.5608±0.0053	0.8424±0.0051	0.3925±0.0079	0.4623±0.0087
NFS	**0.2514**±0.0008	**0.3776**±0.00011	0.4522±0.0013	**0.6290**±0.0022	**0.2254**±0.0007	**0.0745**±0.0010
	`NS` ($r = 64, n_t' = 10$)		`NS` ($r = 64, n_t' = 40$)		`NS` ($r = 128, n_t' = 20$)	
VIT	9.3797±0.0421	12.9291±0.0703	22.8565±0.0935	29.1130±0.1428	15.7398±0.0757	20.6927±0.0664
MLPMIXER	7.5246±0.0080	10.4762±0.0096	15.8632±0.0375	20.1522±0.0604	14.9360±0.0305	19.3268±0.0635
GFN	3.5524±0.0057	4.7071±0.0088	10.2250±0.0331	13.0451±0.0704	6.3976±0.00345	8.2685±0.297
FNO	3.3425±0.0007	5.2566±0.0008	8.9857±0.0010	14.0171±0.0023	4.4627±0.0004	6.3047±0.0004
NFS	**1.7425**±0.0017	**2.2847**±0.0022	**4.7882**±0.0066	**6.1508**±0.0042	**2.6988**±0.0005	**3.5121**±0.0006

Table B5: Performance comparison with Vision Mixer benchmarks on different equations ($n_t = 10$).

Vision Mixers	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
	`Burgers'` ($r = 512, n_t' = 10$)		`Burgers'` ($r = 512, n_t' = 40$)		`Burgers'` ($r = 1024, n_t' = 20$)	
VIT	0.5042±0.0114	0.7667±0.0225	2.4269±0.0288	3.7728±0.0431	1.5327±0.0314	2.4093±0.0408
MLPMIXER	0.1973±0.0070	0.2600±0.0097	0.4210±0.0084	0.5844±0.0101	0.3303±0.0077	0.4473±0.0086
GFN	0.2383±0.0082	0.3066±0.0114	0.4187±0.0079	0.5407±0.0090	0.3500±0.0062	0.4489±0.0081
FNO	0.0978±0.0019	**0.1287**±0.0023	0.1815±0.0009	0.2410±0.0011	**0.1430**±0.0009	**0.1871**±0.0010
NFS	**0.0958**±0.0015	0.1347±0.0022	**0.1708**±0.0006	**0.2351**±0.0009	0.1474±0.0026	0.1957±0.0034
	`KdV` ($r = 512, n_t' = 10$)		`KdV` ($r = 512, n_t' = 40$)		`KdV` ($r = 1024, n_t' = 20$)	
VIT	0.2066±0.0027	0.3525±0.0049	0.2376±0.0022	0.5521±0.0036	0.1897±0.0003	0.3725±0.0009
MLPMIXER	0.2152±0.0023	0.3686±0.0039	**0.2497**±0.0017	0.5400±0.0029	0.2062±0.0007	0.4429±0.0012
GFN	0.1530±0.0004	0.2607±0.0006	0.2691±0.0007	0.5451±0.0014	0.1984±0.0002	0.3869±0.0003
FNO	0.3230±0.0035	1.1105±0.0061	0.9605±0.0024	2.7500±0.0055	0.5929±0.0020	1.6473±0.0033
NFS	**0.0678**±0.0002	**0.1214**±0.0003	0.2709±0.0009	**0.5122**±0.0013	**0.1576**±0.0003	**0.3114**±0.0005
	`NS` ($r = 64, n_t' = 10$)		`NS` ($r = 64, n_t' = 40$)		`NS` ($r = 128, n_t' = 20$)	
VIT	3.9609±0.0101	6.0575±0.0250	12.3433±0.0342	16.5238±0.0415	9.3010±0.0234	14.0027±0.0380
MLPMIXER	3.1530±0.0049	4.4339±0.0067	7.9291±0.0038	10.4149±0.0066	7.7410±0.0037	10.1934±0.0082
GFN	1.7396±0.0016	2.3551±0.0028	5.4464±0.0023	7.2130±0.0032	3.1261±0.0026	4.1691±0.0047
FNO	2.4076±0.0017	3.2861±0.0024	7.6979±0.0035	10.6401±0.0056	3.7001±0.0034	5.0047±0.0072
NFS	**0.8636**±0.0008	**1.2264**±0.0011	**3.1122**±0.0020	**4.1950**±0.0037	**1.8406**±0.0003	**2.5620**±0.0005

Table B6: Performance comparison with graph spatio-temporal benchmarks ($n_t = 1$).

Graph Spatio-Temporal Models	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
	Burgers′ ($n_s = 512, n_t' = 10$)		Burgers′ ($n_s = 256, n_t' = 20$)		Burgers′ ($n_s = 512, n_t' = 40$)	
DCRNN	277.8393±0.0082	346.1716±0.0088	292.1712±0.0280	368.1883±0.0204	298.4096±0.0137	373.0938±0.0186
AGCRN	289.9780±0.0001	360.9834±0.0001	272.6697±0.3404	340.1351±0.5435	305.4976±0.2120	376.0804±0.2385
GCGRU	288.4507±0.0246	361.1175±0.0512	294.9075±0.0005	367.4703±0.0004	291.0365±0.0265	365.1668±0.0827
MPPDE	24.4997±0.0014	34.5123±0.0017	25.4357±0.0002	31.7015±0.0002	25.3311±0.0004	33.7808±0.0005
NFS	**16.1860**±0.0016	**28.1504**±0.0021	**21.1634**±0.0018	**33.8976**±0.0018	**26.0818**±0.0001	**44.7962**±0.0003
	KdV ($n_s = 512, n_t' = 10$)		KdV ($n_s = 256, n_t' = 20$)		KdV ($r = 512, n_t' = 40$)	
DCRNN	1.6855±0.0001	3.0875±0.0001	3.1267±0.0001	4.8662±0.0001	5.7387±0.0001	8.3752±0.0001
AGCRN	4.0753±0.0001	6.8943±0.0001	5.4107±0.0001	9.2333±0.0001	8.4438±0.0001	13.8677±0.0001
GCGRU	1.6554±0.0001	2.6839±0.0001	3.0677±0.0001	4.6557±0.0001	5.8745±0.0001	9.4528±0.0001
MPPDE	1.5452±0.0001	2.6774±0.0001	2.9929±0.0007	5.4582±0.0010	3.0101±0.0001	4.9946±0.0001
NFS	**0.0816**±0.0012	**0.1512**±0.0022	**0.1576**±0.0007	**0.3114**±0.0018	**0.3210**±0.0021	**0.6873**±0.0049
	NS ($n_s = 4096, n_t' = 10$)		NS ($n_s = 1024, n_t' = 20$)		NS ($n_s = 4096, n_t' = 40$)	
DCRNN	30.6756±0.0001	41.7815±0.0001	52.1290±0.0138	69.7019±0.0032	88.3382±0.0864	119.5021±0.0055
AGCRN	OOM	OOM	59.9393±0.0001	79.0434±0.0001	OOM	OOM
GCGRU	28.8537±0.0019	40.1215±0.0008	49.9352±0.0028	67.5623±0.0014	85.9303±0.0731	117.9925±0.0172
MPPDE	8.9810±0.0014	12.1595±0.0022	20.7453±0.0008	32.1098±0.0018	54.2387±0.0006	90.0190±0.0007
NFS	**2.1992**±0.0021	**2.8280**±0.0033	**3.9178**±0.0054	**5.0182**±0.0080	**4.7865**±0.0042	**6.1384**±0.0069

Table B7: Performance comparison with graph spatio-temporal benchmarks ($n_t = 10$).

Graph Spatio-Temporal Models	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
	Burgers′ ($n_s = 512, n_t' = 10$)		Burgers′ ($n_s = 256, n_t' = 20$)		Burgers′ ($n_s = 512, n_t' = 40$)	
DCRNN	2.6122±0.0014	3.8435±0.0019	4.6126±0.0015	6.8853±0.0033	8.5880±0.0020	12.7394±0.0037
AGCRN	4.6667±0.0001	6.2791±0.0001	10.4900±0.0009	13.9810±0.0022	15.6143±0.0002	21.0937±0.0001
GCGRU	1.6643±0.0002	2.5074±0.0003	3.1400±0.0010	4.8008±0.0019	5.7653±0.0021	8.9335±0.0028
MPPDE	1.1271±0.0004	1.8838±0.0007	2.4554±0.0003	4.4315±0.0006	4.1213±0.0006	6.1980±0.0009
NFS	**0.1085**±0.0016	**0.1504**±0.0021	**0.1634**±0.0018	**0.2328**±0.0018	**0.1983**±0.0001	**0.2775**±0.0003
	KdV ($n_s = 512, n_t' = 10$)		KdV ($n_s = 256, n_t' = 20$)		KdV ($r = 512, n_t' = 40$)	
DCRNN	2.3196±0.0001	4.1634±0.0001	3.4503±0.0005	5.7450±0.0003	4.9286±0.0010	8.3912±0.0008
AGCRN	3.9350±0.0001	6.1166±0.0001	5.6631±0.0001	8.1191±0.0001	8.2893±0.0002	11.5684±0.0003
GCGRU	1.6643±0.0001	2.5074±0.0001	3.4205±0.0001	5.6873±0.0001	2.5032±0.0002	5.4515±0.0003
MPPDE	1.4967±0.0003	2.6309±0.0002	2.9708±0.0027	5.3811±0.0050	2.4293±0.0006	4.9310±0.0005
NFS	**0.0816**±0.0012	**0.1512**±0.0022	**0.1576**±0.0007	**0.3114**±0.0018	**0.3210**±0.0021	**0.6873**±0.0049
	NS ($n_s = 4096, n_t' = 10$)		NS ($n_s = 1024, n_t' = 20$)		NS ($n_s = 4096, n_t' = 40$)	
DCRNN	8.7025±0.0003	12.5238±0.0002	27.1069±0.0024	39.1259±0.0031	59.6602±0.0177	88.2946±0.0146
AGCRN	OOM	OOM	42.4197±0.0006	60.5375±0.0008	OOM	OOM
GCGRU	6.3570±0.0001	9.7306±0.0002	21.3537±0.0026	32.9674±0.0033	57.2493±0.0085	84.1847±0.0106
MPPDE	5.4353±0.0041	7.8838±0.0037	17.5902±0.0013	25.9372±0.0016	42.3057±0.0066	76.3374±0.0069
NFS	**0.9335**±0.0011	**1.3254**±0.0012	**1.8239**±0.0012	**2.5291**±0.0008	**3.2768**±0.0026	**4.3988**±0.0009

B.4 MORE VISUALIZATION

Here we provide more visualization results on the three equations. See Fig. B2, Fig. B3 and Fig. B4.

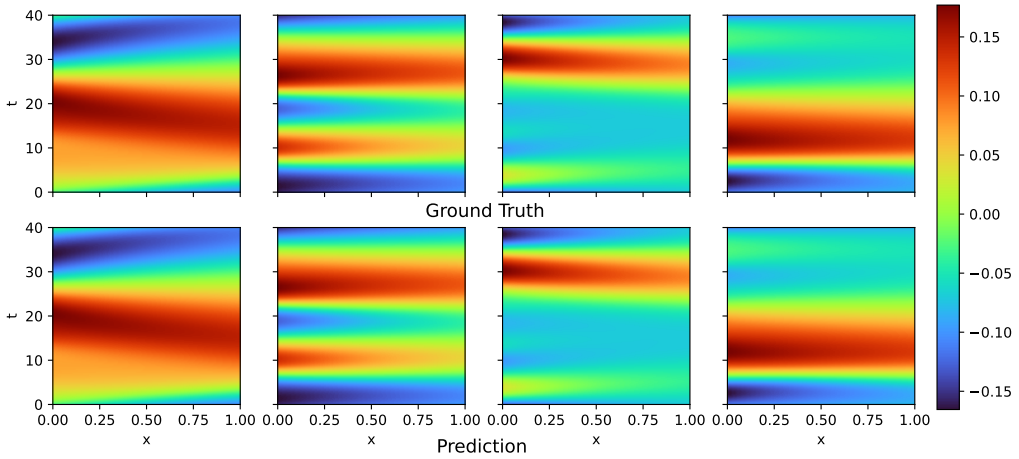


Figure B2: Visualization on equispaced Burgers' equation.

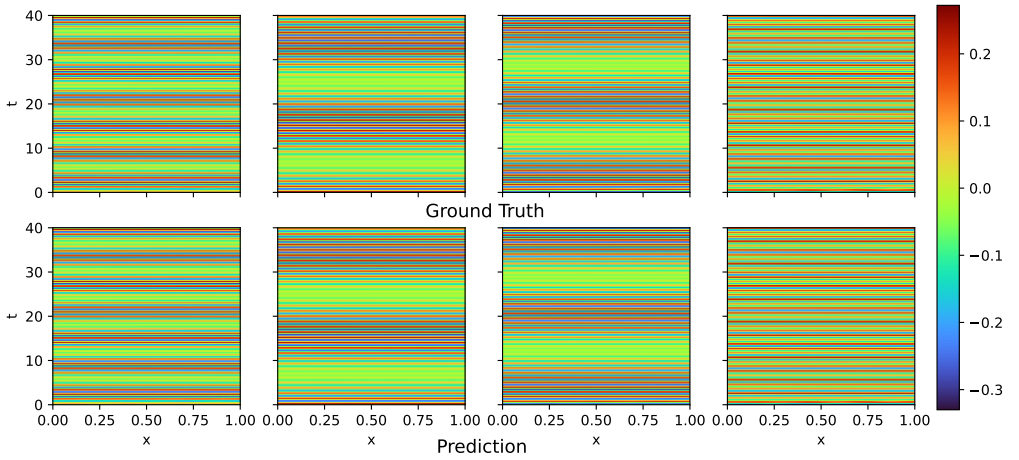


Figure B3: Visualization on equispaced KdV equation.

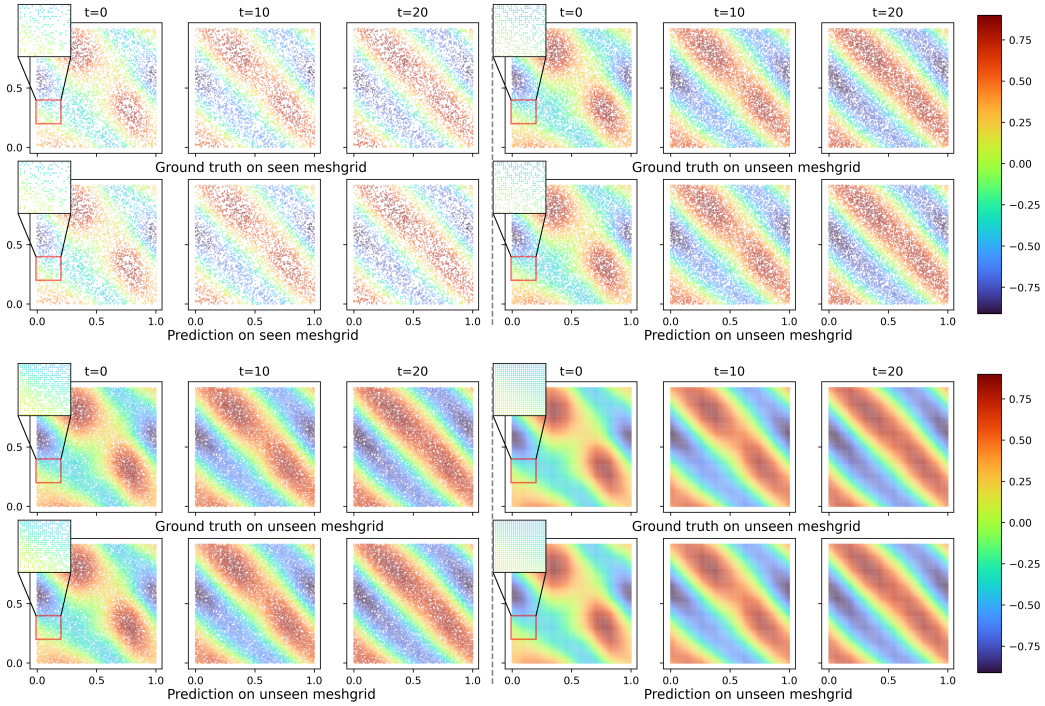


Figure B4: Visualization on non-equispaced NS equation: The training mesh ($n_s = 4096$ in upper-left) is different from the meshes in inference process ($n'_s = 8192$ in upper-right, $n'_s = 12288$ in lower-left and $n's = 16384$ in lower-right).

B.5 MESH-INVARIANT EVALUATION

Table B8: Mesh-invariant performance of NFS on Burgers' and KdV equations $(n_t = 10)$.

	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
	Burgers' $(n_s = 512, n_t = 10, n_t' = 40)$		KdV $(n_s = 512, n_t = 10, n_t' = 40)$	
$\boldsymbol{X}$	0.1983±0.0001	0.2775±0.0002	0.3210±0.0021	0.6873±0.0049
$n_s' = 1.3n_s$	0.2371±0.0034	0.3143±0.0041	0.3769±0.0030	0.7805±0.0077
$n_s' = 1.7n_s$	0.2898±0.0113	0.3742±0.0102	0.4084±0.0072	0.8419±0.0174
$n_s' = 2.0n_s$	0.3052±0.0098	0.4180±0.0100	0.4111±0.0042	0.8471±0.0074

Table B9: Performance of NFS with its variants of NS equations $(n_t = 10)$ on unseen meshes.

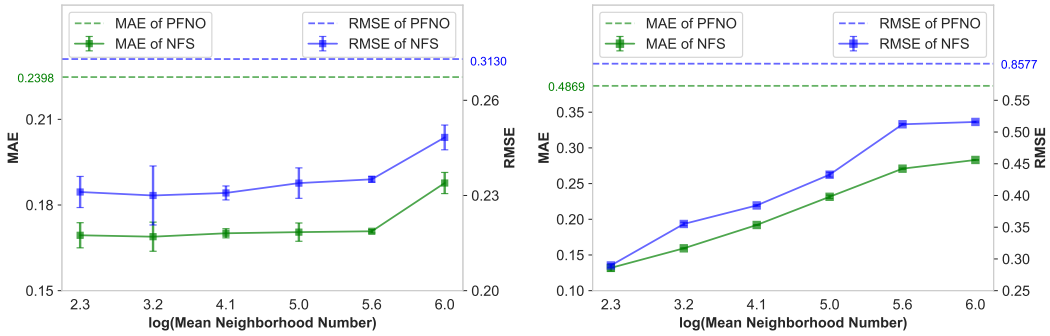
Flex + LN	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
	NS $(n_s = 4096, n_t' = 10)$		NS $(n_s = 1024, n_t' = 20)$		NS $(n_s = 4096, n_t' = 40)$	
$\boldsymbol{X}$	0.9335±0.0011	1.3254±0.0012	1.8239±0.0012	2.5291±0.0008	3.2768±0.0026	4.3988±0.0009
$n_s' = 2n_s$	0.9731±0.0034	1.5042±0.0057	2.3530±0.0051	3.3320±0.0074	3.5439±0.0085	4.7904±0.0168
$n_s' = 3n_s$	1.1071±0.0021	1.5716±0.0038	2.5179±0.0089	3.5477±0.0125	3.6584±0.0180	4.8858±0.0246
$n_s' = 4n_s$	1.1015±0.0000	1.5627±0.0000	2.5919±0.0064	3.6526±0.0071	3.6608±0.0000	4.9521±0.0000
Gaus + LN	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
	NS $(n_s = 4096, n_t' = 10)$		NS $(n_s = 1024, n_t' = 20)$		NS $(n_s = 4096, n_t' = 40)$	
$\boldsymbol{X}$	1.6341±0.0034	2.1992±0.0042	2.1976 ±0.0065	3.0219±0.0090	3.6422±0.0026	5.0097±0.0039
$n_s' = 2n_s$	2.8589±0.0062	4.0562±0.0126	3.7465±0.0041	5.1308±0.0097	3.9092±0.0041	5.2402±0.0075
$n_s' = 3n_s$	3.4513±0.0168	4.5199±0.0377	5.7712±0.0123	5.7137±0.0199	4.2102±0.0082	5.5057±0.0138
$n_s' = 4n_s$	3.4357±0.0000	4.7382±0.0000	5.5990±0.0066	5.5958±0.0049	4.2628±0.0000	5.7679±0.0000
Flex + ~~LN~~	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
	NS $(n_s = 4096, n_t' = 10)$		NS $(n_s = 1024, n_t' = 20)$		NS $(n_s = 4096, n_t' = 40)$	
$\boldsymbol{X}$	1.2138±0.0030	1.7293±0.0047	2.5119±0.0036	3.4923±0.0058	4.2083±0.0037	5.6761±0.0092
$n_s' = 2n_s$	1.4882±0.0146	2.1681±0.0300	7.0203±0.0203	10.6096±0.0345	5.8975±0.0060	8.7704±0.0189
$n_s' = 3n_s$	1.6384±0.0088	2.4130±0.0169	7.9177±0.0059	11.9825±0.0118	6.6622±0.0063	9.5874±0.0131
$n_s' = 4n_s$	1.6975±0.0000	2.5008±0.0000	7.1962±0.0101	10.8860±0.0098	6.6951±0.0000	9.6334±0.0000

The mesh-invariant evaluation on Burgers' and KDV Equations of NFS are given in Table. B8. In Table. B8, when the spatial resolution is just 512, inference performance on unseen meshes deteriorates. This result also validate our conclusion *(2)* in the third paragraph in Sec. 4.

Besides, we give a full evaluation on mesh-invairance of NFS in NS equation, with its variants as a detailed results corresponding to Table. B9.

B.6 NEIGHBORHOOD SIZE'S EFFECTS

The effects of mean neighborhood size on the predictive performance on `Burgers'` ($n_s = 512, n_t = 10, n_t' = 40$) and `KDV` ($n_s = 512, n_t = 10, n_t' = 40$) are shown in Fig. B5.



(a) `Burgers'` ($n_s = 512, n_t' = 40$). (b) `KdV` ($n_s = 512, n_t' = 40$).

Figure B5: The change of MAE and RMSE of NFS with the increase of neighborhood size on `Burgers'` ($n_s = 512, n_t = 10, n_t' = 40$) and `KdV` ($n_s = 512, n_t = 10, n_t' = 40$). PFNO is the baseline.

B.7 INTERPOLATION WITH OTHER VISION MIXERS

We conduct experiments on non-equispaced NS equations with the combination of our interpolation layers and other Vision Mixers to figure out if they can achieve camparable performance.

Table B10: Performance of different Vision Mixers combined with the interpolation layers in non-equispaced scenarios on NS equations ($n_t = 10$).

	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
VIT	NS ($n_s = 4096, n_t' = 10$)		NS ($n_s = 1024, n_t' = 20$)		NS ($n_s = 4096, n_t' = 40$)	
$\boldsymbol{X}$	OOM	OOM	OOM	OOM	OOM	OOM
	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
MLPMIXER	NS ($n_s = 4096, n_t' = 10$)		NS ($n_s = 1024, n_t' = 20$)		NS ($n_s = 4096, n_t' = 40$)	
$\boldsymbol{X}$	6.1854±0.0012	8.1556±0.0018	9.4593±0.0028	12.1316±0.0022	10.1862±0.0045	13.1548±0.0051
$n_s' = 2n_s$	8.1573±0.0126	11.2258±0.0147	12.0706±0.0132	14.9460±0.0159	10.6003±0.0127	13.6872±0.0238
$n_s' = 3n_s$	8.1952±0.0088	11.3840±0.0171	14.9910±0.0094	17.8415±0.0110	10.5633±0.0140	13.6394±0.0147
$n_s' = 4n_s$	8.7773±0.0000	11.3313±0.0000	14.9517±0.0125	17.7857±0.0199	10.5414±0.0000	13.6106±0.0000
	MAE ($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)	MAE($\times 10^{-3}$)	RMSE($\times 10^{-3}$)
GFN	NS ($n_s = 4096, n_t' = 10$)		NS ($n_s = 1024, n_t' = 20$)		NS ($n_s = 4096, n_t' = 40$)	
$\boldsymbol{X}$	12.2373±0.0091	16.2902±0.0133	10.2768±0.0084	13.7852±0.0078	14.7765±0.0055	19.4872±0.0106
$n_s' = 2n_s$	13.7752±0.0164	18.3108±0.0181	17.7216±0.0225	24.1397±0.0371	15.9083±0.0235	21.0041±0.0256
$n_s' = 3n_s$	13.7054±0.0122	18.2192±0.0184	17.8238±0.0112	24.2783±0.0196	15.8986±0.0156	20.9943±0.0158
$n_s' = 4n_s$	13.7140±0.0000	18.2271±0.0000	17.8207±0.0105	24.2833±0.0141	15.8736±0.0000	20.9772±0.0000

B.8 COMPLEXITY COMPARISON

We here first give Table. B11 to show the complexity of time and memory of all the evaluated methods on NS ($r = 64, n_t = 10, n_t' = 40$).

Table B11: comparison on complexity of the evaluated methods

Type	Methods	Time/Epoch	Peak Memory	Parameter Number
Graph Spatio-Temporal Model	GCGRU	$6'18''$	8660MB	74945
	DCRNN	$9'38''$	11120MB	148673
	AGCRN	OOM	OOM	OOM
	MPPDE	$10'54''$	23333MB	622161
Vision Mixer	VIT	$3'14''$	32166MB	773217
	MLPMIXER	$1'12''$	4421MB	79749953
	GFN	$48''$	3296MB	1361729
	FNO	$27''$	3748MB	6299425
	PFNO	$43''$	3380MB	9742145
	NFS	$2'02''$	31938MB	37891937

Table B12: Detailed complexity of NFS

Interpolation on Resampled Points		
Neighbor Searching	Kernel Calculation	Weighted Summation
3522MB	3102MB	6884MB
Interpolation back on Original Points		
Neighbor Searching	Kernel Calculation	Weighted Summation
2506MB	2754MB	6884MB

It demonstrates that our method has comparable efficiency as Vision Mixers. For the graph spatial-temporal models, they suffer from the recurrent network structure, and thus is extremely time-consuming while the parameter number is small, limiting their flexibility.

Time. However, once we compare the used time in PFNO and NFS, we will find that the interpolation layers are considerably time-consuming. Another module that cost time complexity is the normalization layer, as the original FNO does not include Layer-Norm in its architecture, but it is stacked in PFNO. Theoretically, PFNO handles a down-sampled grids in a low resolution, because of the patchwise embedding. However, it takes more time than FNO. Therefore, we conclude that the time complexity brought from Layer-Norm is very significant, but it is affordable because of the performance improvements.

Memory. Besides, the operation of searching for each spatial points' neighborhood and calculating weighted summation in Eq. (9) and Eq. (10) are very memory-consuming. We test it on the same experiment, and give the memory usage of different models in forward process, as shown in Table. B12. The memory cost in backward process is 6902MB.