HAMLET: Graph Transformer Neural Operator for Partial Differential Equations

Andrey Bryutkin [*1] **Jiahao Huang** [*2] **Zhongying Deng** [3] **Guang Yang** [2] **Carola-Bibiane Schönlieb** [3]
Angelica Aviles-Rivero [3]

Abstract

We present a novel graph transformer framework, HAMLET, designed to address the challenges in solving partial differential equations (PDEs) using neural networks. The framework uses graph transformers with modular input encoders to directly incorporate differential equation information into the solution process. This modularity enhances parameter correspondence control, making HAMLET adaptable to PDEs of arbitrary geometries and varied input formats. Notably, HAMLET scales effectively with increasing data complexity and noise, showcasing its robustness. HAMLET is not just tailored to a single type of physical simulation, but can be applied across various domains. Moreover, it boosts model resilience and performance, especially in scenarios with limited data. We demonstrate, through extensive experiments, that our framework is capable of outperforming current techniques for PDEs.

1. Introduction

Deep learning has revolutionised all domains, including the solution of partial differential equations (PDEs). Traditional numerical methods for solving PDEs can be computationally expensive, especially for high-dimensional problems or those with complex geometries. In recent years, a number of approaches have been developed to leverage the power of deep neural networks for PDEs, including Physics-Informed Neural Networks (PINNs) e.g. (Raissi et al., 2019a; Karniadakis et al., 2021) and Neural Operators e.g. (Li et al., 2022b; Lu et al., 2021). These methods use neural networks to learn the solution to a PDE directly from the data, without the need for discretisation or meshing. Deep learning for PDEs is largely changing many fields of science and engineering, from fluid dynamics and electromagnetics to finance and healthcare (Liu & Wang, 2019; Kissas et al., 2020; Sahli Costabal et al., 2020; Sun et al., 2020; Raissi et al., 2019b).

Although PINNs and their variants have shown success in solving PDEs, they still have limitations. These include limited generalisability across multiple PDE instances, a lack of discretisation invariance, and an inability to generalise beyond a specific resolution/geometry observed during training (Wang et al., 2021a; Fuks & Tchelepi, 2020). A body of researchers has investigated alternatives to mitigate such limitations, where principles driven by neural operators are considered. These neural operators directly approximate the differential operator that underlies the PDE and have yielded promising outcomes in recent years, especially for multi-instance learning. However, significant research has yet to be conducted in this area.

Contributions. We present a novel framework called grap**H** tr**A**nsfor**M**er neura**L** op**E**ra**T**or – HAMLET, which aims to address the challenges in solving PDEs using neural networks. HAMLET stands out as the first framework of its kind, which employs graph transformers with modular input encoders. Choosing a graph neural network as a foundational architecture enables the capture of complex interactions and dynamics more effectively than standard transformers due to its ability to model intricate spatial relationships and nonlinear interactions efficiently. This capability is especially crucial in handling rapid changes and high-frequency information at initial time steps, as demonstrated by our experimental results. HAMLET has adaptability to irregular meshes, allowing it to solve discretisation-invariant PDEs. Furthermore, HAMLET showcases exceptional performance even when faced with limited data, it is meticulously designed to address the current challenges in the field, underscoring a significant advancement over existing approaches. Notably, we emphasise:

- We introduce *the first graph transformer architecture* (HAMLET) for PDEs, in which we highlight:
  - Our HAMLET framework can effectively handle PDEs

[*]Equal contribution [1]Department of Mathematics, MIT, USA [2]Bioengineering Department and Imperial-X, National Heart and Lung Institute & Cardiovascular Research Centre, Imperial College London, UK. [3]Department of Applied Mathematics and Theoretical Physics, University of Cambridge, UK. Correspondence to: Andrey Bryutkin <bryutkin@mit.edu>, Jiahao Huang <j.huang21@imperial.ac.uk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

with arbitrary geometries and diverse input formats, including point non-uniform meshes and design parameters.
- HAMLET provides a strong graph perspective, a feature that significantly boosts the resilience and performance of the model, especially when dealing with limited data availability.
- HAMLET learns solution operators on a flexible grid, which enhances the computational efficiency of transformers and the flexibility of graphs for problem solving.

• We extensively evaluate HAMLET through experiments on various graphs, which enhances its inference characteristics and reduces overfitting. In addition, we compare our proposed technique with existing approaches and validate it on multiple datasets.

2. Related Work

• **Learned PDEs**. New methods have been proposed in various fields, such as fluid dynamics for solving PDEs (Brunton et al., 2020; Stachenfeld et al., 2022). These advances have led to the extension of architectures that describe continuous-time solutions and multiparticle dynamics (Iakovlev et al., 2020; Wang et al., 2020). With knowledge of the underlying PDEs, physics-informed models can be used to retrieve single-instance solutions in both unsupervised and semi-supervised settings (Raissi et al., 2019a; Zhu et al., 2019; Karniadakis et al., 2021). The standard strategy for solving such tasks is to encode the data spatially and then use various schemes to evolve over time. Deep learning architectures employ models such as convolutional layers (Ronneberger et al., 2015; Zhu & Zabaras, 2018; Bhatnagar et al., 2019), symbolic neural networks (Kaiser et al., 2018; Long et al., 2019), residual networks (He et al., 2016), and more advanced physics-informed networks (Jagtap & Karniadakis, 2021), as well as methods known from finite element methods (FEM) such as Galerkin and Ritz (Sirignano & Spiliopoulos, 2018; E & Yu, 2018).

• **Neural Operators**. After the proposal of DeepONet (Lu et al., 2021) as a general operator learning framework, further research was carried out to learn the underlying solution operator using model reduction techniques (Bhattacharya et al., 2021). Neural operators, such as graph neural networks (Li et al., 2020c;b) or convolutional layers in the Fourier space (Li et al., 2020a), are models that describe functions between infinite-dimensional spaces. The Fourier Neural Operator (FNO) and its variants, such as incremental, factorised, and adaptive FNO or FNO+ (Zhao et al., 2022; Tran et al., 2021; Guibas et al., 2022; Li et al., 2022b), have demonstrated remarkable results in terms of speed and error, with their main advantage being discretisation invariance. Most recently, new adaptive neural operator models have been proposed that use transformations (Tripura & Chakraborty, 2022; Gupta et al., 2021), novel message-passing algorithms on the graph (Brandstetter et al., 2022c; Boussif et al., 2022; Ong et al., 2022), attention mechanisms (Li et al., 2023b; Kissas et al., 2022), or different function bases (Fanaskov & Oseledets, 2022). To improve performance, methods with increasing physical inductive bias (Li et al., 2021; Wang et al., 2021b), data augmentation (Brandstetter et al., 2022b), and symmetry-related approaches (Brandstetter et al., 2022a) have been proposed.

• **Graph Neural Networks in Physical Systems**. GNNs have demonstrated their advantage in analysing complex dynamical systems by describing interactions on irregular grids (Li & Farimani, 2022; Battaglia et al., 2016; 2018; Sanchez-Gonzalez et al., 2018; Alet et al., 2019). Message-passing algorithms in GNNs have been used to pass messages between small groups of moving or interacting objects (Gilmer et al., 2017). New research has been conducted on solving differential equations with GNNs, but spatial information about the location of nodes is needed and there is increased computational complexity (Li et al., 2020b;c). GNNs have been used to solve complex initial and boundary value problems and to model continuous-time propagation (Lötzsch et al., 2022; Horie & Mitsume, 2022; Pilva & Zareei, 2022; Iakovlev et al., 2020). All-in-one models have also been proposed by describing PDEs as neural operators (Brandstetter et al., 2022c; Li et al., 2020b).

• **Transformers in Physical Systems**. The attention mechanism has been successful in describing dynamics and patterns in physical systems (Geneva & Zabaras, 2022; Han et al., 2022). Attention layers capture structures and patterns in the spatial domain of PDEs (Shao et al., 2022; Cao, 2021b), while temporal evolution is modelled using attention (Geneva & Zabaras, 2022; Song et al., 2022). Our model uses attention-based layers for spatial encoding and a recurrent multilayer perceptron (MLP) for time marching in latent space. Graph neural networks have also been studied using attention (Min et al., 2022; Yun et al., 2019; Dwivedi & Bresson, 2020), and architectures are used to model the solution operator for PDEs (Bo et al., 2023; Li et al., 2023b; Cao, 2021b; Kissas et al., 2022). However, to the best of our knowledge, graph transformers and their capabilities have not yet been explored for PDEs, opening the door to a new research line. Janny et al. (Janny et al., 2023) introduce a mesh-based transformer (EAGLE) optimising for large distances through pregenerated coarse meshes, combining GNN and self-attention for dynamic data. HAMLET contrasts with its graph transformer encoder, CrossFormer for parameter integration, and MLP-based time propagation, which favours recurrent over autoregressive model updates. Steeven et al. (Steeven et al., 2024), parallel in aim, differs in topology and integration methods, using a GNN backbone and recurrent updates with continuous integration, unlike

HAMLETś single CrossFormer approach.

3. Methodology: HAMLET

Problem statement. We consider parametric partial differential equations defined on a domain $\mathcal{D} \subset \mathbb{R}^n$. We denote the domain by $\mathcal{D}_\theta$, where $\theta \in \mathcal{P}$ and $\mathcal{P} \subset \mathbb{R}^p$ is the parameter space. Thus, the domain $\mathcal{D}_\theta$ is parameterised by θ, which is sampled over the domain according to the distribution ν. The general form of our problems reads:

$$P : \mathcal{P} \times \mathcal{D} \times \mathcal{V} \times \mathbb{R}^m \times \ldots \times \mathbb{R}^m \to \mathbb{R}^\ell, \quad \mathcal{D} \subset \mathbb{R}^n, \mathcal{V} \subset \mathbb{R}^m,$$
$$P\left(\theta, x, u, \partial_{x_1} u, \ldots \partial_{x_n} u, \ldots, \partial_{x_1}^{\alpha_1} \cdots \partial_{x_n}^{\alpha_n} u\right) = 0. \tag{1}$$

The fixed map P is solved by a vector-valued function $u : \mathcal{D} \to \mathcal{V}$, which is not known a priori. Here α is a multiindex defined as $\alpha = (\alpha_1, \ldots \alpha_n)$, with $\alpha_i \in \mathbb{N}_0, |\alpha| = \sum_{i=1}^n \alpha_i$. In our case, the first entry represents the time; the domain reduces to $\mathcal{T} \subset \mathbb{R}_{\geq 0}$ and $\mathcal{D}_\theta \subset \mathbb{R}^{n-1}$, if the equation depends on additional space variables. For a well-defined setting, for most of the problems, at least one of the following conditions should be fulfilled:

$$u(x) = u_0(x), \quad x \in \mathcal{D}_\theta \times \{T_0\}, \tag{2a}$$
$$u(x) = u_b(x), \quad x \in \partial\mathcal{D}_\theta \times \mathcal{T}, \tag{2b}$$

where $T_0 \in \mathcal{T}$. We call u_0 the initial condition and u_b the boundary condition. Assume $\mathcal{P}, \mathcal{D}, \mathcal{V}$ to be Banach spaces with a given norm. We assume that there exists an analytic solution operator: $\mathcal{S} : \mathcal{P} \times \mathcal{D} \times \mathbb{R}^m \times \ldots \times \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathcal{V}$, $(\theta, x, \partial_{x_1} u, \ldots \partial_{x_n} u, \ldots, \partial_{x_1}^{\alpha_1} \cdots \partial_{x_n}^{\alpha_n} u, u_b, u_0) \mapsto u$. We then seek to design an architecture to function as an approximate solution operator, denoted by $\tilde{\mathcal{S}}_\mu : (\theta, u_0, u_b) \mapsto u$, where $\mu \in \mathbb{R}^p$ represents the parameters of the neural network. We can assume that the dataset $\{\theta^{(n)}, u^{(n)}\}_{n=1}^N$ is provided, where $\theta^{(n)} = \{\theta_j^{(n)}\}_{j=1}^L$ is the collection of input parameters describing the system's state on a L-discretised domain. We use the notation $\theta^{(n)} = \theta(x^{(n)})$ and $u^{(n)} = u(x^{(n)})$ for brevity. Furthermore, we assume that our operator $\tilde{\mathcal{S}}_\mu(\theta^{(n)}) = u^{(n)}$ may be corrupted with noise. We underline that in this work, we consider both type of problems: stationary and time-dependent. Refer to Figure 1 for an overview of our HAMLET model.

3.1. Graph Construction for HAMLET

Input Format for a Graph Space. The input of HAMLET is defined on a general graph $\mathcal{G}$ composed of a set of nodes V and edges E. Each node in the graph also has predefined node features that are specific to the PDE problem being considered. The model can accept various input parameter formats. In many cases, the input is given by a specific mesh design that discretises the spatial domain $\mathcal{D}_\theta$ into L nodes, which we call an L *point discretisation* of the domain $\mathcal{D}_\theta$.

Graph Construction. We define a graph $\mathcal{G} = (V, E)$ with a set of nodes $V = \{h_i\}_{i=1}^L$ and undirected edges $E = \{e_{ij}\}$. The node feature vectors $h_i \in \mathbb{R}^d$ represent the attributes of each node, where d is the dimension of the feature vector. Edge features are properties of connections between nodes, which in our case are used to generate messages in the transformer architecture. To define the edges on the graph, we use a truncation function, which creates the edges by assigning each node i its neighbourhood $\mathcal{N}_i$.

How can we generate the edges between the nodes for the connectivity of the graph? To generate the edges between the nodes of the graph, we treat the L-point discretisation $\{x_i\}_{i=1}^L \subset \mathcal{D}_\theta$ as the nodes and use a truncation function s to map the nodes to subsets of $\mathcal{D}$ denoted $\mathcal{B}(\mathcal{D}_\theta)$. We assign and concatenate the parameter value θ_i and the corresponding position x_i to each node, where the node feature can be defined as $h_i = \theta_i \| x_i$, ($\|$ is concatenation). Connectivity between nodes is defined by the intersection of a node's image under s with the set of all nodes V. These subsets of nodes $\mathcal{B}(\mathcal{D}_\theta)$ are called the neighbourhoods of the nodes. In our case, the nodes are connected by an edge if the Euclidean distance between them is less than a predefined radius r, and the neighbours are distributed within a circular area. If $s(x) = \mathcal{D}_\theta$, the graph is fully connected. Our choice of using a connectivity ball generator as s leads to a sparse graph, which reduces computational cost.

3.2. Graph Transformer for HAMLET

Graph Transformer Block. Graph transformers (Dwivedi & Bresson, 2020) are neural networks that process graph-structured data, allowing them to learn robust node representations and representations of the entire graph. The formulation of the graph-based multi-head self-attention mechanism enables efficient message passing on a graph.

$$\hat{h}_i^\ell = O_h^\ell \|_{k=1}^H \left(\frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^\ell \right),$$
$$\text{where, } w_{ij}^{k,\ell} = \text{softmax}_j \left(\frac{Q^{k,\ell} h_i^\ell \cdot K^{k,\ell} h_j^\ell}{\sqrt{d_k}} \right), \tag{3}$$

and $Q^{k,\ell}, K^{k,\ell}, V^{k,\ell} \in \mathbb{R}^{d_k \times d}$, $O_h^\ell \in \mathbb{R}^{d \times d}$ are fully-connected layers at k^{th} attention head in the l^{th} transformer block. $\|$ denotes concatenation. We integrate position information using Rotary Position Embedding (RoPE) (Su et al., 2021), even though the node features contain positional information unlike vector-based multi-head self-attention. For brevity, the RoPE operator is included into $Q^{k,\ell} := R_\Theta \bar{Q}^{k,\ell}$ and $K^{k,\ell} := R_\Theta \bar{K}^{k,\ell}$.

In the message-passing algorithm framework, the attention layer acts as a message that aggregates data from neighbouring nodes. Normalisation is crucial before summing to ensure alignment with neural operator classes. The feed-

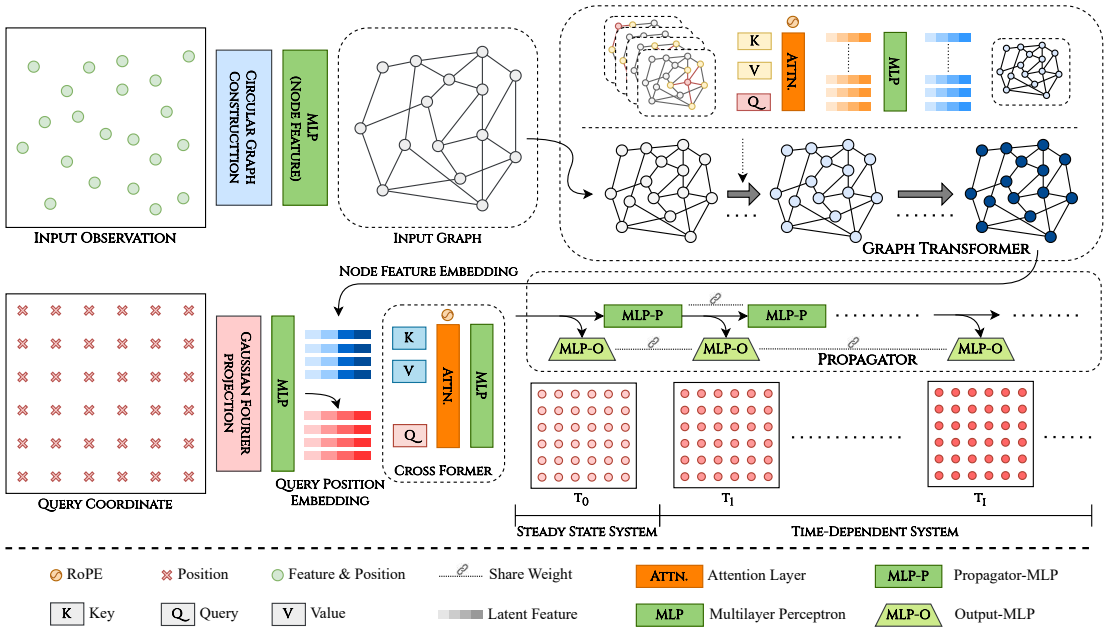

Figure 1. Architectural overview of HAMLET. The diagram depicts the transformation of input observations into graph representations, followed by node feature embedding using a multilayer perceptron (MLP) to prepare key, value and query vectors for the graph transformer. The transformer captures data structure and node relationships, which are then refined by the cross-former with query positions. Finally, shared MLPs propagate the embeddings to model system behaviour over time.

forward neural network takes node features from the latent vector after the multi-head self-attention layer:

$$\begin{aligned} \hat{h}_i^\ell &= \mathrm{Attn}(h_i^\ell), & \hat{\hat{h}}_i^\ell &= \mathrm{Norm}(h_i^\ell + \hat{h}_i^\ell), \\ \hat{\hat{h}}_i^\ell &= W_2^\ell \cdot \mathrm{ReLU}(W_1^\ell \hat{\hat{h}}_i^\ell), & h_i^{\ell+1} &= \mathrm{Norm}(\hat{\hat{h}}_i^\ell + \hat{\hat{h}}_i^\ell), \end{aligned} \tag{4}$$

where $W_1^\ell, \in \mathbb{R}^{2d\times d}$ and $W_2^\ell, \in \mathbb{R}^{d\times 2d}$ are fully-connected layers. $\hat{h}_i^\ell$, $\hat{\hat{h}}_i^\ell$ and $\hat{\hat{h}}_i^\ell$ denote intermediate representations. Norm can be either layer normalisation (Ba et al., 2016) or batch normalisation (Ioffe & Szegedy, 2015). The attention layer acts as the message m_i.

HAMLET corresponds to the class of Neural Operators. The operator definition (see Appendix B) captures the local and global behaviour of latent phase features, simulating the behaviour of a classical analytical operator. The kernel integral operator $\mathcal{K}$ is formally written as an integral over the domain $\mathcal{D}_\theta$, utilising a learnable kernel $\kappa^{(l)} \in C(\mathcal{D}_\theta \times \mathcal{D}_\theta; \mathbb{R}^{d_{l+1}\times d_l})$ and a Borel measure γ on $\mathcal{D}_\theta$. Specifically, $\mathcal{K}$ is defined as $(\mathcal{K}v_l)(x) = \int_{\mathcal{D}_\theta} \kappa^{(l)}(x,y)v_l(y)d\gamma(y)$ for all $x \in \mathcal{D}_\theta$. On the graph, we are dealing with a discretised version of $\mathcal{K}$, where the kernel is based on the message-passing algorithm framework (Gilmer et al., 2017), employing average aggregation to update node features h_j and obtain the operator solution $u(x_j)$, which reads:

$$u(x_j) = \frac{1}{|\mathcal{N}_j|} \sum_{y\in\mathcal{N}_j} \kappa(x_j, y)\, v(y), \quad j = 1, \ldots, J. \tag{5}$$

This representation corresponds to the Monte-Carlo approximation of the integral. The multi-head attention layer in the proposed transformer resembles the integration kernel defined in (5) on the graph. We next demonstrate that the neural operator framework is a continuum generalisation of the graph transformer architecture.

Proposition 3.1. *The residual block of the graph transformer layer, as proposed above, can be seen as a special case of the integration kernel of the neural operator.*

Proof. Let us rewrite the concatenation as a full vector considering that different heads act on different parameters. This yields the following.

$$\hat{h}_i^\ell = \frac{O_h^\ell}{|\mathcal{N}_i|} \sum_{j\in\mathcal{N}_i} \begin{bmatrix} w_{ij}^{1,\ell} V^{1,\ell} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_{ij}^{H,\ell} V^{H,\ell} \end{bmatrix} \cdot \begin{bmatrix} h_j^\ell \\ \vdots \\ h_j^\ell \end{bmatrix} \tag{6}$$

Considering that the components are distinct entries in the diagonal matrix, we can rewrite this as: $\hat{h}_i^\ell = \frac{O_h^\ell}{|\mathcal{N}_i|} \sum_{j\in\mathcal{N}_i} \left(w_{ij}^{1,\ell} V^{1,\ell} \oplus \cdots \oplus w_{ij}^{H,\ell} V^{H,\ell} \right) h_j^\ell$. We then absorb O_h^ℓ into a block diagonal matrix to obtain:

$\hat{h}_i^\ell = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\left(w_{ij}^{1,\ell}\tilde{V}^{1,\ell}\oplus\cdots\oplus w_{ij}^{H,\ell}\tilde{V}^{H,\ell}\right)h_j^\ell$. We use ReLU as the activation function σ. Denote $f_j^{n,l,i} := \exp\left(\frac{Q^{n,\ell}h_i^\ell\cdot K^{n,\ell}h_j^\ell}{\sqrt{d_n}}\right)$, then we can write:

$$\begin{aligned} h_i^{\ell+1} &= \sigma\Bigg(W_1^\ell h_i^\ell + \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\Bigg(\frac{\exp\left(\frac{Q^{1,\ell}h_i^\ell\cdot K^{1,\ell}h_j^\ell}{\sqrt{d_1}}\right)}{\sum_{k\in\mathcal{N}_j}f_j^{1,l,k}}\tilde{\tilde{V}}^{1,\ell} \\ &\qquad \oplus\cdots\oplus \frac{\exp\left(\frac{Q^{H,\ell}h_i^\ell\cdot K^{H,\ell}h_j^\ell}{\sqrt{d_H}}\right)}{\sum_{k\in\mathcal{N}_j}f_j^{H,l,k}}\tilde{\tilde{V}}^{H,\ell}\Bigg)h_j^\ell\Bigg) \\ &= \sigma\left(W_1^\ell h_i^\ell + \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\kappa h_j^\ell\right) \end{aligned} \tag{7}$$

Defining κ as the kernel and using the node features h_j and h_i as inputs, we obtain: $\mathcal{K} = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\kappa(h_i,h_j)h_j$, which can be considered as the Monte-Carlo approximation of the integral $u(x) = \int_{s(x)}\kappa(x,y)v(y)\mathrm{d}y, \quad \forall x\in\mathcal{D}_\theta$. Thus, the graph transformer is also applied to the class of neural operators. □

3.3. Unboxing the HAMLET Architecture

HAMLET features three key modules: input parameter encoder (EncI), query and information fusion encoder (EncQ), and a decoder (Dec) for steady-state or time-dependent system output.

☞ HAMLET **Encoder.** As mentioned in Sec. 3.1, the input graph $\mathcal{G}_{\text{in}}$ is constructed with input parameters and the corresponding positions as node features, using circular truncation for edge connectivity. $\mathcal{D}_{\text{in}}$ denote the spatial grid of the input parameters. In EncI, the node features $\{h_i\}_{i=1}^L$ are embedded into latent space $\{\tilde{h}_i\}_{i=1}^L$ through a fully connected layer FC_{in}, and the graph is updated as $\tilde{\mathcal{G}}_{\text{in}}$. The data tuple $\{\tilde{\mathcal{G}}_{\text{in}},\mathcal{D}_{\text{in}}\}$ is then fed into a stack of graph transformer blocks $\{\text{GT}_k\}_{k=1}^{N_{\text{GT}}}$ discussed in Sec. 3.2, which can be expressed as follows:

$$\begin{aligned} \tilde{\mathcal{G}}_{\text{in}} &= \text{FC}_{\text{in}}(\mathcal{G}_{\text{in}}), \\ \mathcal{G}_k &= \text{GT}_k(\mathcal{G}_{k-1},\mathcal{D}_{\text{in}}), \quad k=1,\ldots,N_{\text{GT}}, \end{aligned} \tag{8}$$

where $\mathcal{G}_0 = \tilde{\mathcal{G}}_{\text{in}}$. $\mathcal{G}_{\text{EncI}} = \mathcal{G}_{N_{\text{GT}}}$ is the output of EncI, with the node features $\{h_{\text{EncI},i}\}_{i=0}^L$, denoted as $\mathcal{H}_{\text{EncI}}$ for brevity.

The aim of EncQ is to encode the query location and integrate the query location embedding and the input parameter features. Specifically, the query spatial grid $\mathcal{D}_{\text{qry}}$ (L'-point discretised) is embedded by a Gaussian Fourier projection GF_{qry} (Tancik et al., 2020) and then fed to a MLP_{qry} for the embedding of the query position $\{h_{\text{EncQ},i'}\}_{i'=0}^{L'}$, denoted as $\mathcal{H}_{\text{EncQ}}$ for brevity. Subsequently, a cross-attention-based transformer block, namely, CrossFormer (Li et al., 2023b), is applied to integrate the $\mathcal{H}_{\text{EncI}}$ and $\mathcal{H}_{\text{EncQ}}$, which can be expressed:

$$\mathcal{H}_{\text{EncQ}} = \text{MLP}_{\text{qry}}(\text{GF}_{\text{qry}}(\mathcal{D}_{\text{qry}})), \tag{9a}$$
$$\mathcal{H}_{\text{Enc}} = \text{CF}(\mathcal{H}_{\text{EncI}},\mathcal{H}_{\text{EncQ}}), \tag{9b}$$

where CF denotes the CrossFormer and $\mathcal{H}_{\text{Enc}}$ are integrated latent features for the subsequent decoder.

Similarly to graph transformer blocks, the CrossFormer adopts a standard "Attn-Norm-MLP-Norm" with residual connection, where "Attn" is Galerkin-type cross-attention (Cao, 2021a; Li et al., 2023b), allowing for arbitrary query position. By introducing three sets of basis functions $\{k_l(\cdot),v_l(\cdot),q_l(\cdot)\}_{l=1}^d$, the cross-attention CAttn reads:

$$\begin{aligned} \text{CAttn}_j(h_{\text{EncQ},i'}) = \sum_{l=1}^d\left(\frac{\sum_{i=1}^L k_l(h_{\text{EncI},i})v_j(h_{\text{EncI},i})}{L}\right) \\ \times q_l(h_{\text{EncQ},i'}). \end{aligned} \tag{10}$$

☞ HAMLET **Decoder.** As discussed above, encoders lift the representation into a higher dimensional space, increasing the degrees of freedom and allowing for the interpretation of input parameters and query positions. The main task of the decoder is to project the latent feature back into the observable space. For a steady-state system, such as Darcy Flow, an MLP_{out} is utilised to project the latent feature to observable space

$$\hat{u} = \text{MLP}_{\text{out}}(\mathcal{H}_{\text{Enc}}\|\mathcal{D}_{\text{qry}}). \tag{11}$$

However, for a time-dependent system, directly tracking all problem parameters and learning the whole temporal sequence raise complexity and lead to limited performance, especially problems with long temporal sequences, e.g. Diffusion Reaction. Consequently, inspired by (Li et al., 2023b), a recurrent-style propagator is adopted for time-dependent system decoding. The decision to utilise an RNN on top of the encoder features was driven by the requirement to capture temporal dependencies and propagate information across time steps, which is critical for the accurate simulation of PDEs over time. The latent state's role as input to the RNN over time is to maintain a continuous and coherent evolution of the system state, ensuring that predictions at each time step are informed by the accumulated knowledge of prior states. The encoder output is used as an initial state $\mathcal{H}_{\text{Dec}}^0 = \mathcal{H}_{\text{Enc}}$, and fed into the propagator-MLP MLP_{prop} (with a residual connection) for recurrent running. After reaching a specific recurrent step, output-MLP MLP_{out} is used to project the latent feature to an observable space. The propagation in latent space from state t to $t+\triangle t$, and the corresponding output projection, can be expressed as

follows:

$$\mathcal{H}_{\text{Dec}}^{t+\triangle t} = \text{MLP}_{\text{prop}}(\mathcal{H}_{\text{Dec}}^{t} \| \mathcal{D}_{\text{qry}}) + \mathcal{H}_{\text{Dec}}^{t}, \tag{12a}$$

$$\hat{u}^{t+\triangle t} = \text{MLP}_{\text{out}}(\mathcal{H}_{\text{Dec}}^{t+\triangle t} \| \mathcal{D}_{\text{qry}}). \tag{12b}$$

The outputs of MLP_{out} are collected during the recurrent running $\hat{u}|_{t\in[\triangle t,T]} = [\hat{u}^{\triangle t}, \hat{u}^{2\triangle t}, \ldots, \hat{u}^{T}]$.

HAMLET Loss Function & Discretisation Invariance. We aim to learn the mapping between two infinite-dimensional spaces using a finite collection of dataset pairs. Our objective is to optimise the parameters $\mu = \{\mu_{\text{EncI}}, \mu_{\text{EncQ}}, \mu_{\text{Dec}}\}$ to obtain the best possible approximation $\tilde{\mathcal{S}}_{\mu_{opt}} \sim \mathcal{S}$, which represents a specific configuration of the operator map. We can solve this problem by minimising the empirical risk, considering an entire sampling set of N independent and identically distributed (i.i.d.) datasets, which is given by:

$$\begin{aligned} \min_{\mu\in\mathbb{R}^p} \mathcal{L}_{\text{data}}\ (u, \mathcal{S}_\mu(\theta)) &= \min_{\mu\in\mathbb{R}^p} \mathbb{E}_{\theta\sim\nu} \mathcal{L}(u, \mathcal{S}_\mu(\theta)) \\ &\approx \min_{\mu\in\mathbb{R}^p} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(u, \mathcal{S}_\mu(\theta)), \end{aligned} \tag{13}$$

where $\mathcal{L}_{\text{data}}$ is the empirical loss of distance function $\mathcal{L}$, which is either the MSE loss $\mathcal{L} = (u - \mathcal{S}_\mu(\theta))^2$ or the relative L_2 Norm Loss $\mathcal{L} = \frac{\|u-\mathcal{S}_\mu(\theta)\|_2}{\|u\|_2}$. The selection of loss function is based on the benchmark protocol; refer to Appendix D.

The data pairs $\{\theta^{(n)}, u^{(n)}\}_{n=1}^{N}$ are the discretisations of our continuous space. We assume that we only have access to the point-wise evaluation of finite points. Let $\mathcal{D}^{(n)} = \{x_i^{(n)}\}_{i=1}^{L_n} \subset \mathcal{D}$ be a L_n-point discretisation of the domain $\mathcal{D}$. Despite being trained on a L_n-point discretisation, the operator $\tilde{S}_\mu$ should be able to provide $u(x)$ for any arbitrary $x \in \mathcal{D}$ given an input $\theta \in \mathcal{P}$. In particular, the discretised architecture maps into the space $\mathcal{V}$ and not into a discretisation thereof. For additional details, see Appendix B.

4. Experimental Results

In this section, we detail all the experiments carried out to validate our proposed HAMLET technique.

4.1. Dataset & Implementation Details

Dataset and PDEs. We mainly utilised the datasets from PDEBench (Takamoto et al., 2022) – a wider range public benchmark for PDE-based simulation tasks, selecting Darcy Flow, Shallow Water, and Diffusion Reaction showcasing the stationary and time-dependent problems on a uniform grid. We also utilised the Airfoil dataset from (Pfaff et al., 2021), which models aerodynamics around an airfoil wing's cross-section, for experiments on irregular grids. Refer to Appendix C for details on the dataset and setting used.

Darcy Flow. We conducted experiments on the steady-state solution of 2D Darcy Flow over the uniform square. The setting used is detailed in Appendix C.

Shallow Water. We performed the experiments on the time-dependent system of 2D Shallow Water equations, providing a model for analysing problems related to free-surface flows. Refer to Appendix C for details on our setting.

Diffusion Reaction. The time-dependent system of 2D Diffusion-Reaction problems were included in the experiment. Our setting is detailed in Appendix C.

Airfoil. For non-uniform grid, we incorporated the airfoil problem based on 2D time-dependent compressible flow utilising datasets from (Pfaff et al., 2021). Refer to Appendix C for details on our setting.

Evaluation Protocol & Implementation Details.

We performed comparative experiments between our technique and several leading methods, including U-Net (Ronneberger et al., 2015)[1], FNO (Li et al., 2020a), PINN (Raissi et al., 2019a), DeepONet (Lu et al., 2021), Geo-FNO (Li et al., 2022a), MAgNet (Boussif et al., 2022), GNOT (Hao et al., 2023), EAGLE (Janny et al., 2023) and OFormer (Li et al., 2023b). For Darcy Flow, Shallow Water and Diffusion Reaction datasets, we report that the normalised root mean squared error (nRMSE) provides scale-independent information, following the PDEBench protocol (Takamoto et al., 2022). In addition, we report the results of RMSE in the Appendix A. For Airfoil, we report the results using the relative L_2 error following the settings from (Li et al., 2023b). All experiments were performed on a single NVIDIA A100 GPU with 80GB of memory, running under the same conditions for a fair comparison. For dynamical PDE settings, we trained HAMLET directly by unrolling for 90 timestamps. In contrast, UNet, FNO, GeoFNO, and MAgNet were trained using an auto-regressive approach. DeepONet was trained directly for the full timestamps, while OFormer was also trained by unrolling for the full timestamps. More details about the model architecture, training setting, and hyper-parameter can be found in Appendix D.

4.2. Results & Discussion

Numerical Comparison. We carried out a thorough comparison of our framework against leading deep learning techniques for solving PDEs. The results are summarised in Table 1[2]. Further supporting results are available in Appendix A. Our evaluation of the HAMLET approach in various datasets and parameter settings has demonstrated its robust performance. Notably, within the Darcy Flow dataset, HAMLET exhibited a commendable adaptability to changing

[1] Using the PDEBench implementation (Takamoto et al., 2022)

[2] Note, that MAgNet was also trained on Shallow Water and Diffusion Reaction PDEs but the training failed to converge. Consequently, the results for MAgNet are not reported in Table 1

Table 1. Numerical comparison of our approaches vs. existing techniques. The values reflect the nRMSE. The best-performing results are highlighted in green.

Dataset Setting		Normalised RMSE (nRMSE)							
Dataset	Param.	GNOT	U-Net	FNO	DeepONet	OFormer	GeoFNO	MAgNet	HAMLET
Darcy Flow	$\beta = 0.01$	–	1.10E+00	2.50E+00	3.25E-01	3.04E-01	1.03E+00	7.71E-02	**3.11E-01**
Darcy Flow	$\beta = 0.1$	–	1.80E-01	2.20E-01	1.67E-01	1.15E-01	3.13E-01	8.10E-02	**7.65E-02**
Darcy Flow	$\beta = 1.0$	–	3.30E-02	6.40E-02	5.12E-02	2.05E-02	6.34E-02	1.03E-01	**1.40E-02**
Darcy Flow	$\beta = 10.0$	–	8.20E-03	1.20E-02	3.97E-02	6.34E-03	2.51E-02	1.62E-01	**4.77E-03**
Darcy Flow	$\beta = 100.0$	–	4.40E-03	6.40E-03	3.64E-02	4.19E-03	2.04E-02	1.95E-01	**3.46E-03**
Shallow Water	–	4.16E-03	8.30E-02	4.40E-03	2.35E-03	2.90E-03	6.70E-03	–	**2.04E-03**
Diffusion Reaction	–	8.22E-01	8.40E-01	1.20E-01	8.42E-01	3.28E+00	7.72E+00	–	**9.02E-02**

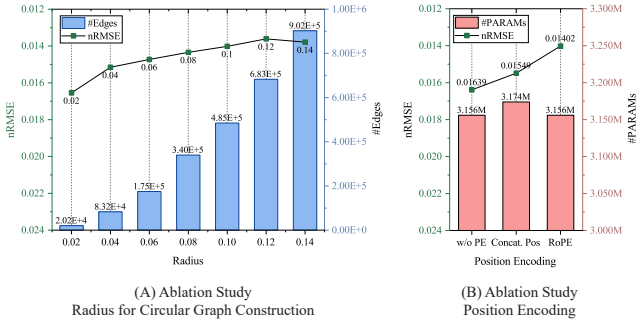

Figure 2. Performance impacts of graph construction and position encoding on model accuracy. (A) The trade-off between circular graph radius and nRMSE alongside edge count. (B) Comparison of nRMSE across position encoding methods.

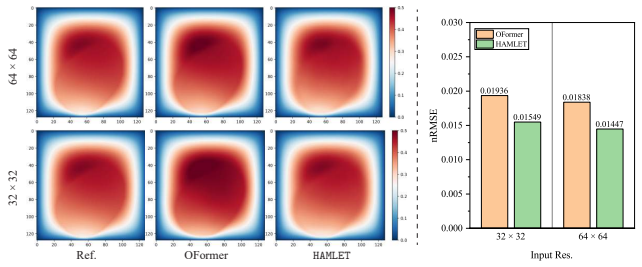

Figure 3. Comparison of model predictions (128 × 128) against a reference for two input resolutions (32 × 32 and 64 × 64). Left: Heatmap comparisons of the reference and predicted outcomes by OFormer and HAMLET models. Right: Corresponding nRMSE values, demonstrating the superior accuracy of HAMLET.

parameters settings of β. The decrease in nRMSE as β increased from 0.01 to 100.0 highlights HAMLET's capability to scale effectively with increased complexity or noise within the data. This trend suggests that HAMLET is not only robust, but also potentially more adept at handling complex simulations where other models may falter.

The generalisability of HAMLET was further underscored by its performance on the Shallow Water and Diffusion

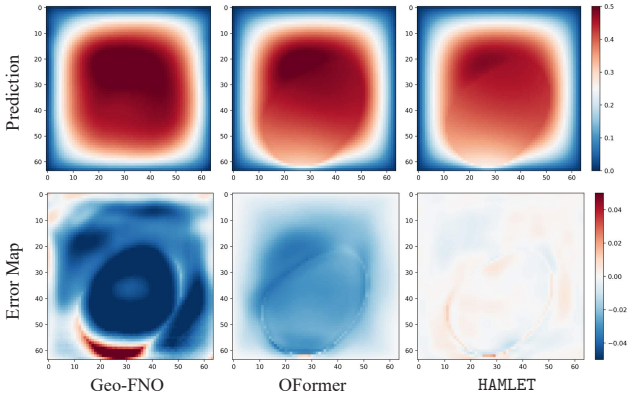

Figure 4. Predictions and corresponding error maps for Geo-FNO, OFormer, and HAMLET models using Darcy Flow ($\beta = 1.0$).

Reaction datasets. Its ability to maintain low nRMSE values across these diverse datasets suggests that HAMLET is not just tailored to a single type of physical simulation, but can be applied across various domains. This adaptability is critical for real-world applications, where conditions can be unpredictable and models must maintain a high degree of accuracy. Furthermore, there is a clear trend that as β increases, the nRMSE decreases for HAMLET, suggesting that it scales well with complexity or noise within the data.

In addition, we conducted an in-depth comparison of a dynamic PDE modelling as predicted by OFormer and HAMLET against the reference solution. The results, illustrated in Figure 5, underscore the accuracy of HAMLETś predictions, which closely align with the reference, highlighting its effectiveness in dynamic PDE modelling. Our additional investigations reinforce HAMLETś state-of-the-art performance. Additional numerical experiments are also available under Appendix A.

Does the Graph Radius Affect Performance? In Figure 2-(A), the ablation study, using Darcy Flow, on the radius for the construction of the circular graph

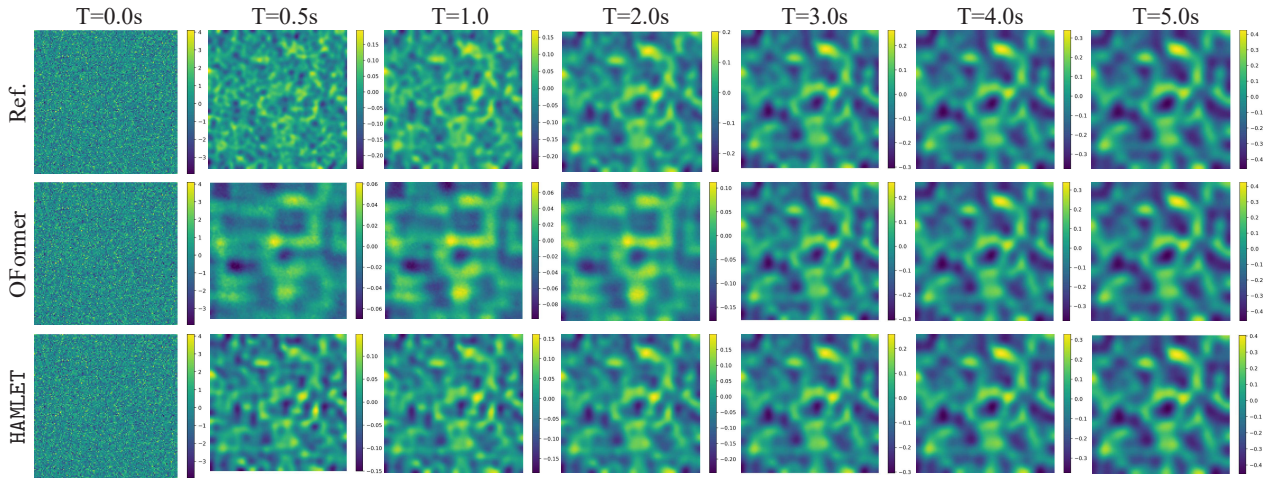

Figure 5. Time evolution (activator) of the diffusion-reaction process as predicted by OFormer and HAMLET compared to the reference solution. HAMLET's predictions closely match the reference, showcasing its efficacy in dynamic PDE modelling.

reveals a clear inverse relationship between the radius and the nRMSE. As the graph radius increases, there is an improvement in the nRMSE, indicating enhanced model performance. This trend continues until the radius reaches approximately 0.1 to 0.14, at which point the performance gains begin to plateau. This plateau suggests a convergence in performance improvement despite the graph becoming more complex with a greater number of edges.

Performance Across Resolutions. The visual representations in Figure 3 provide a comparison between the reference results and those predicted by OFormer (a non-graph transformer) and HAMLET models at different input resolutions. Qualitatively, the HAMLET produces heat maps that more closely resemble the reference, especially at the finer 64×64 resolution. This observation is quantitatively supported by the nRMSE values, where HAMLET consistently outperforms OFormer. At 32×32 resolution, HAMLET demonstrates a lower nRMSE to OFormer. The trend of superior performance continues at 64×64 resolution. These results suggest that HAMLET is better at capturing the intricacies of the dataset at different resolutions, which could be attributed to its graph-based perspective that may be more adept at encoding spatial relationships within the data.

HAMLET – Graph or non-Graph, that is the question. We further support our method through Figure 4. This shows that a comparison with Geo-FNO underscores the effectiveness of HAMLET which is also graph-based but exhibits the lowest error among models. This not only highlights the utility of graph-based approaches in capturing the complexity of the data, but also shows the specific strengths of HAMLET in minimising prediction errors. Moreover, Table 4 presents a concise quantitative analysis of the performance of HAMLET and OFormer on the Airfoil dataset, characterised by its non-uniform grid structure. The superior performance of HAMLET in this context not only confirms the adaptability of graph-based approaches to nonuniform grids, but also underscores the potential advantages of such methods over their transformer counterparts in capturing complex spatial relationships inherent in computational fluid dynamics datasets such as the Airfoil dataset.

Table 2. Ablation studies for graph constructions on Darcy Flow 2D ($\beta = 1.0$) in 64×64 grid.

	KNN				Circular
	$k = 21$	$k = 51$	$k = 101$	$k = 151$	
nRMSE	0.02018	0.01996	0.02054	0.02094	**0.01402**

Table 3. Quantitative Results for models trained on different dataset sizes.

Darcy Flow ($\beta = 1.0$), nRMSE				
#Training Data	9K	5K	2K	1K
OFormer	2.048E-02	2.093E-02	2.674E-02	3.321E-02
HAMLET	1.402E-02	1.642E-02	2.211E-02	2.779E-02
Shallow Water, nRMSE				
#Training Data	900	500	200	100
OFormer	2.900E-03	1.190E-02	2.310E-02	2.910E-02
HAMLET	2.044E-03	2.320E-03	3.255E-03	4.746E-03

Can Position Encoding Transform Model Performance? The results of Figure 2-(B) demonstrate the impact of different position encoding strategies on model performance. The Rotary Position Embedding (RoPE) outperforms other methods, indicating that incorporating relational information between positions can significantly enhance the model's understanding of the spatial structure within the data. Di-

Table 4. Quantitative Results on Airfoil Dataset

Airfoil, Relative L_2 Norm	
OFormer	3.486E-02
GNOT	4.310E-02
EAGLE	1.192E-01
HAMLET	3.030E-02

rectly concatenating the coordinates to the input features offers a performance benefit over not using any position encoding, but it is still inferior to the relational encoding offered by RoPE. This suggests that while the model can derive some spatial context from direct coordinate information, the rotational invariance and relative positional information encoded by RoPE provide a more nuanced and effective representation. This emphasises the value of sophisticated position encoding mechanisms of our graph framework.

Graph Construction Ablation. Our ablation study for graph construction on the Darcy flow 2D dataset (β=1.0), indicates a clear advantage of our circular strategy over the traditional k-nearest neighbours (KNN) method. By defining neighbours within a variable radius, our approach dynamically captures the spatial context of the dataset, resulting in a significant performance leap with the lowest nRMSE compared to KNN's. This single-radius, context-aware strategy outperforms the fixed-edge KNN method, confirming the importance of spatial relationships in accurately modelling complex physical phenomena.

Data Size Performance Impact. We examine the effect of dataset size on model performance. From Table 3, we can observe that both OFormer and our proposed HAMLET model exhibit a decline in performance as the amount of training data decreases. This trend is evident in both the Darcy Flow and Shallow Water datasets, with nRMSE values rising as the dataset size reduces from 9K to 1K and from 900 to 100 respectively. Despite this common trend, HAMLET consistently outperforms OFormer in all datasets, demonstrating its superior ability to generalise from limited data. Particularly in the most constrained scenarios—1K for Darcy Flow and 100 for Shallow Water— HAMLET demonstrates greater resilience to data scarcity. HAMLET demonstrates improved performance under data-scarce conditions because it utilises a graph-based approach, which inherently requires fewer data points to model complex relationships effectively compared to traditional methods. Our graph perspective allows the neural operator to infer and generalise from less training data by leveraging the intrinsic geometrical and topological information contained within the graphs, which represent physical domains or conditions in PDEs. This approach enables more efficient learning, particularly advantageous when dealing with limited datasets commonly encountered in PDE scenarios.

5. Conclusion

Our HAMLET model introduces an advancement in the neural network-based resolution of PDEs, offering a flexible and robust solution adaptable to various geometries and conditions. HAMLET uses of graph transformers and modular input encoders establishes new benchmarks in the field, particularly in complex scenarios with limited data availability. A limitation is the graph construction time, which is a common aspect in graph-based approaches. However, we view this not as a major constraint but as an inherent step that enables our model's robust performance. The construction of the graph is a crucial phase in which HAMLET captures the complex dependencies within the data. We believe that the benefits gained in accuracy and adaptability to diverse PDEs far outweigh the computational time required during this stage. Our extensive testing confirms that HAMLET not only meets but often exceeds the performance of existing techniques. Future work includes integration of Lie-symmetry preservation and augmentation. But also includes extending HAMLET to handle higher-dimensional PDEs such as 3D problems. This will involve a dedicated exploration to refine the model architecture and compare it with existing approaches, similar to the methodology used by Li et al. for large-scale 3D PDEs (Li et al., 2023a).

Acknowledgments

AB was supported in part by the Akamai Presidential Fellowship and the Hans-Messer Foundation. JH and GY were supported in part by the ERC IMI (101005122), the H2020 (952172), the MRC (MC/ PC/21013), the Royal Society (IEC NSFC211235), the NVIDIA Academic Hardware Grant Program, the SABER project supported by Boehringer Ingelheim Ltd, Wellcome Leap Dynamic Resilience, and the UKRI Future Leaders Fellowship (MR/V023799/1). CBS acknowledges support from the Philip Leverhulme Prize, the Royal Society Wolfson Fellowship, the EPSRC advanced career fellowship EP/V029428/1, EPSRC grants EP/S026045/1 and EP/T003553/1, EP/N014588/1, EP/T017961/1, the Wellcome Innovator Awards 215733/Z/19/Z and 221633/Z/20/Z, CCMI and the Alan Turing Institute. AAR gratefully acknowledges funding from the Cambridge Centre for Data-Driven Discovery and Accelerate Programme for Scientific Discovery, made possible by a donation from Schmidt Futures, ESPRC Digital Core Capability Award, and CMIH and CCIMI, University of Cambridge.

Impact Statement

The impact of our work is observed primarily in the domain of solving partial differential equations (PDEs) using neural networks. Although our work does not

directly address specific ethical concerns, such as those related to large language models or image generation, advanced models like HAMLET can influence a wide range of applications, from climate modelling to biomedical simulations. Therefore, we encourage continued reflection on how advancements in machine learning may shape and be shaped by societal and ethical considerations.

References

Alet, F., Jeewajee, A. K., Villalonga, M. B., Rodriguez, A., Lozano-Perez, T., and Kaelbling, L. Graph element networks: adaptive, structured computation and memory. In *International Conference on Machine Learning*, pp. 212–222. PMLR, 2019. 2

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. In *Advances in neural information processing systems*, pp. 198–206, 2016. 4

Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016. 2

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 2

Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., and Kaushik, S. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64:525–545, 2019. 2

Bhattacharya, K., Hosseini, B., Kovachki, N. B., and Stuart, A. M. Model reduction and neural networks for parametric pdes. *The SMAI journal of computational mathematics*, 7:121–157, 2021. 2

Bo, D., Shi, C., Wang, L., and Liao, R. Specformer: Spectral graph neural networks meet transformers. *arXiv preprint arXiv:2303.01028*, 2023. 2

Boussif, O., Bengio, Y., Benabbou, L., and Assouline, D. Magnet: Mesh agnostic neural pde solver, 2022. 2, 6

Brandstetter, J., Berg, R. v. d., Welling, M., and Gupta, J. K. Clifford neural layers for pde modeling. *arXiv preprint arXiv:2209.04934*, 2022a. 2

Brandstetter, J., Welling, M., and Worrall, D. E. Lie point symmetry data augmentation for neural pde solvers. In *International Conference on Machine Learning*, pp. 2241–2256. PMLR, 2022b. 2

Brandstetter, J., Worrall, D., and Welling, M. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022c. 2

Brunton, S. L., Noack, B. R., and Koumoutsakos, P. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52:477–508, 2020. 2

Cao, S. Choose a transformer: Fourier or galerkin. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24924–24940. Curran Associates, Inc., 2021a. 5

Cao, S. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34: 24924–24940, 2021b. 2

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020. 2, 3

E, W. and Yu, B. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6 (1):1–12, 2018. 2

Fanaskov, V. and Oseledets, I. Spectral neural operators. *arXiv preprint arXiv:2205.10573*, 2022. 2

Fuks, O. and Tchelepi, H. A. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1), 2020. 1

Geneva, N. and Zabaras, N. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, 2022. 2

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017. 2, 4

Guibas, J., Mardani, M., Li, Z., Tao, A., Anandkumar, A., and Catanzaro, B. Adaptive fourier neural operators: Efficient token mixers for transformers, 2022. 2

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. *arXiv preprint arXiv:2109.13459*, 2021. 2

Han, X., Gao, H., Pfaff, T., Wang, J.-X., and Liu, L.-P. Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113*, 2022. 2

Hao, Z., Wang, Z., Su, H., Ying, C., Dong, Y., Liu, S., Cheng, Z., Song, J., and Zhu, J. Gnot: A general neural operator transformer for operator learning, 2023. 6

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 2

Horie, M. and Mitsume, N. Physics-embedded neural networks: Graph neural pde solvers with mixed boundary conditions. *Advances in Neural Information Processing Systems*, 35:23218–23229, 2022. 2

Iakovlev, V., Heinonen, M., and Lähdesmäki, H. Learning continuous-time pdes from sparse data with graph neural networks. *arXiv preprint arXiv:2006.08956*, 2020. 2

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4

Jagtap, A. D. and Karniadakis, G. E. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. In *AAAI Spring Symposium: MLPS*, pp. 2002–2041, 2021. 2

Janny, S., Béneteau, A., Nadri, M., Digne, J., Thome, N., and Wolf, C. Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers, 2023. 2, 6

Kaiser, E., Kutz, J. N., and Brunton, S. L. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474, 2018. 2

Karniadakis, G., Kevrekidis, Y., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, pp. 1–19, 2021. doi: 10.1038/s42254-021-00314-5. 1, 2

Kissas, G., Yang, Y., Hwuang, E., Witschey, W. R., Detre, J. A., and Perdikaris, P. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358:112623, 2020. 1

Kissas, G., Seidman, J. H., Guilhoto, L. F., Preciado, V. M., Pappas, G. J., and Perdikaris, P. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022. 2

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces, 2021. 15

Li, Z. and Farimani, A. B. Graph neural network-accelerated lagrangian fluid simulation. *Computers & Graphics*, 103: 201–211, 2022. 2

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a. 2, 6

Li, Z., Kovachki, N., 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*, 2020b. 2

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020c. 2

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021. 2

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for pdes on general geometries. *arXiv preprint arXiv:2207.05209*, 2022a. 6

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier Neural Operator with Learned Deformations for PDEs on General Geometries, 2022b. 1, 2

Li, Z., Kovachki, N. B., Choy, C., Li, B., Kossaifi, J., Otta, S. P., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., and Anandkumar, A. Geometry-informed neural operator for large-scale 3d pdes, 2023a. 9

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023b. 2, 5, 6, 17

Liu, D. and Wang, Y. Multi-fidelity physics-constrained neural network and its application in materials modeling. *Journal of Mechanical Design*, 141(12), 2019. 1

Long, Z., Lu, Y., and Dong, B. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019. 2

Lötzsch, W., Ohler, S., and Otterbach, J. S. Learning the solution operator of boundary value problems using graph neural networks. *arXiv preprint arXiv:2206.14092*, 2022. 2

Lu, L., Jin, P., and Karniadakis, G. E. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, March 2021. arXiv:1910.03193 [cs, stat]. 1, 2, 6

Min, E., Chen, R., Bian, Y., Xu, T., Zhao, K., Huang, W., Zhao, P., Huang, J., Ananiadou, S., and Rong, Y. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022. 2

Ong, Y. Z., Shen, Z., and Yang, H. Iae-net: Integral autoencoders for discretization-invariant learning. *arXiv preprint arXiv:2203.05142*, 2022. 2

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. 6, 17

Pilva, P. and Zareei, A. Learning time-dependent pde solver using message passing graph neural networks. *arXiv preprint arXiv:2204.07651*, 2022. 2

Raissi, M., Perdikaris, P., and Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019a. ISSN 00219991. doi: 10.1016/j.jcp.2018.10.045. 1, 2, 6

Raissi, M., Wang, Z., Triantafyllou, M. S., and Karniadakis, G. E. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137, 2019b. 1

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015. 2, 6

Sahli Costabal, F., Yang, Y., Perdikaris, P., Hurtado, D. E., and Kuhl, E. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020. 1

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pp. 4470–4479. PMLR, 2018. 2

Shao, Y., Change, L. C., and Bo, D. Sit: Simulation transformer for particle-based physics simulation. *International Conference on Learning Representations*, 2022. 2

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018. 2

Song, W., Zhang, M., Wallwork, J. G., Gao, J., Tian, Z., Sun, F., Piggott, M., Chen, J., Shi, Z., Chen, X., et al. M2n: mesh movement networks for pde solvers. *Advances in Neural Information Processing Systems*, 35:7199–7210, 2022. 2

Stachenfeld, K., Fielding, D. B., Kochkov, D., Cranmer, M., Pfaff, T., Godwin, J., Cui, C., Ho, S., Battaglia, P., and Sanchez-Gonzalez, A. Learned simulators for turbulence. In *International Conference on Learning Representations*, 2022. 2

Steeven, J., Madiha, N., Julie, D., and Christian, W. Space and time continuous physics simulation from partial observations, 2024. 2

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv e-prints*, art. arXiv:2104.09864, April 2021. doi: 10.48550/arXiv.2104.09864. 3

Sun, L., Gao, H., Pan, S., and Wang, J.-X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020. 1

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., and Niepert, M. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35: 1596–1611, 2022. 6, 16, 17

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547. Curran Associates, Inc., 2020. 5

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021. 2

Tripura, T. and Chakraborty, S. Wavelet neural operator: A neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*, 2022. 2

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1457–1466, 2020. 2

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a. 1

Wang, S., Wang, H., and Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40): eabi8605, 2021b. 2

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019. 2

Zhao, J., George, R. J., Li, Z., and Anandkumar, A. Incremental spectral learning in fourier neural operator. *arXiv preprint arXiv:2211.15188*, 2022. 2

Zhu, J., Zou, Y., and Karniadakis, G. E. Physics-informed autoencoder for solving nonlinear partial differential equations. *Proceedings of the National Academy of Sciences*, 116(45):22252–22263, 2019. 2

Zhu, Y. and Zabaras, N. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018. 2

Appendix

This document builds upon the practical aspects and visual results detailed in the main paper, providing additional insights into our methodology and experimental results.

A. Further Numerical & Visual Results.

In this section, we extend the numerical and visual comparison of Table 1 and Figure 5 of the main paper. The RMSE results presented in Table 5 illustrate the robustness of the HAMLET approach across various datasets and complexity levels, consistently showcasing competitive or superior performance compared to existing models. Particularly in the DarcyFlow datasets with extreme β values and the physically intricate Shallow Water and Diffusion Reaction datasets, HAMLET's graph-based methodology excels in capturing complex spatial relationships, often outperforming existing models including a nongraph transformer-based technique.[3]

Table 5. Numerical comparison of our approaches vs. existing techniques. The values reflect the RMSE. The best-performing results are highlighted in green.

Dataset Setting		RMSE						
Dataset	Param.	U-Net	FNO	DeepONet	OFormer	GeoFNO	MAgNet	HAMLET
DarcyFlow	β = 0.01	4.00E-03	8.00E-03	3.31E-03	**2.21E-03**	2.70E-03	8.07E-03	2.45E-03
DarcyFlow	β = 0.1	4.80E-03	6.20E-03	4.88E-03	2.55E-03	4.15E-03	1.05E-02	**2.60E-03**
DarcyFlow	β = 1.0	6.40E-03	1.20E-02	9.65E-03	3.00E-03	6.20E-03	2.90E-02	**2.74E-03**
DarcyFlow	β = 10.0	1.40E-02	2.10E-02	6.79E-02	7.32E-03	2.08E-02	2.18E-01	**5.51E-03**
DarcyFlow	β = 100.0	7.30E-02	1.10E-01	6.21E-01	4.91E-02	1.65E-01	2.11E+00	**3.37E-02**
Shallow Water	–	8.60E-02	4.50E-03	2.44E-03	3.10E-03	7.30E-03	–	**2.13E-03**
Diffusion Reaction	–	6.10E-02	8.10E-03	6.46E-02	1.26E-02	5.20E-02	–	**5.48E-03**

Figure 6 presents a comparative visualisation of the time evolution predictions for the Shallow Water problem made by OFormer, a transformer-based model, and HAMLET our graph-based transformer approach, against the reference solution. Across the sequence from T=0s to T=1s, HAMLET's predictions are more closely aligned with the reference, particularly at later time steps where the complexity of the system behaviour increases. This visual alignment is quantitatively supported by our previous results, which showed HAMLETachieving lower error metrics compared to OFormer. The comparison emphasises HAMLET's superior performance in capturing the dynamic nature of the system, likely due to its graph-based structure, which enables a more nuanced interpretation of spatial-temporal data.

Figure 7 presents complementary results to Figure 5, offering a comparative visualisation of the time evolution predictions (inhibitor) for the Diffusion Reaction process made by OFormer and our HAMLET in comparison to the reference solution. We want to note that the unique characteristics of the Diffusion Reaction system, particularly the non-linear reaction terms, lead to complex patterns from initial conditions. HAMLET architecture is better suited for capturing the intricate dynamics, due to our graph perspective, of this system early on due to its ability to model complex spatial relationships and non-linear interactions more effectively than OFormer, especially under conditions of rapid change and high-frequency information present at the initial time steps. This advantage is particularly evident in the early stages, where the intricate interplay of diffusion and reaction is most challenging to model accurately. Therefore, we intentionally chose to showcase these results to highlight the advantages of our approach.

B. Discretisation Invariance & Further Definitions.

[3]The use of GeoFNO for the Darcy Flow dataset does not imply variation in geometry within the Darcy equation itself, as the Darcy Flow problem typically involves a fixed geometry. Instead, the application of GeoFNO likely aims to leverage its learning capacity on complex geometries and could be used to assess the robustness of the model even when the underlying PDE does not exhibit geometric variability. The model's application in this context might be more about demonstrating its potential in handling geometric complexities should they arise in other PDEs, rather than indicating a varying geometry within the Darcy Flow problem itself.

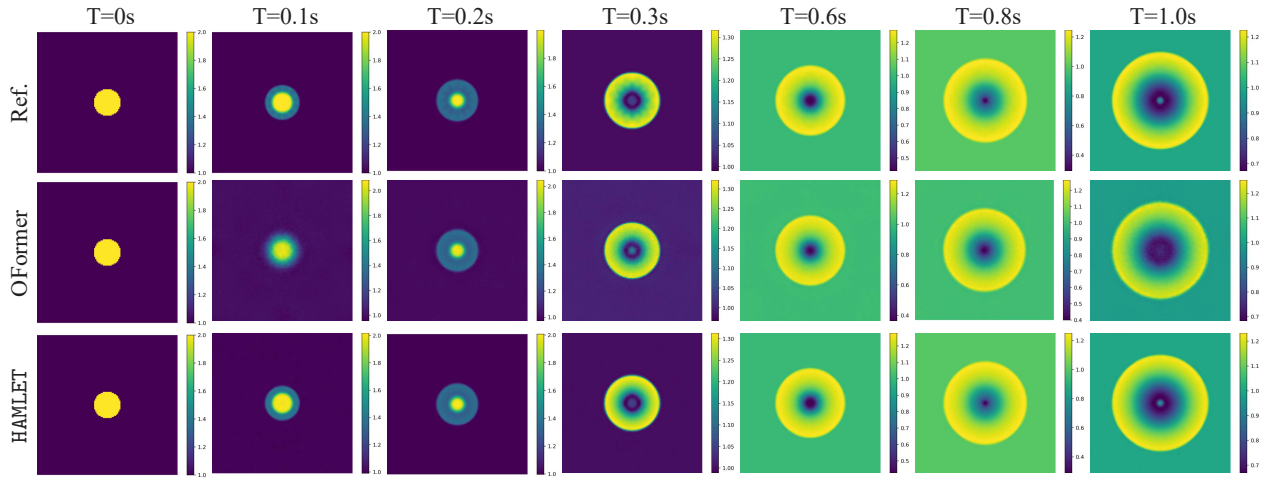

Figure 6. Time-lapse predictions of a dynamic process by OFormer and HAMLET compared with the reference solution using Shallow Water 2D. HAMLET's graph-based approach yields closer alignment to the reference.

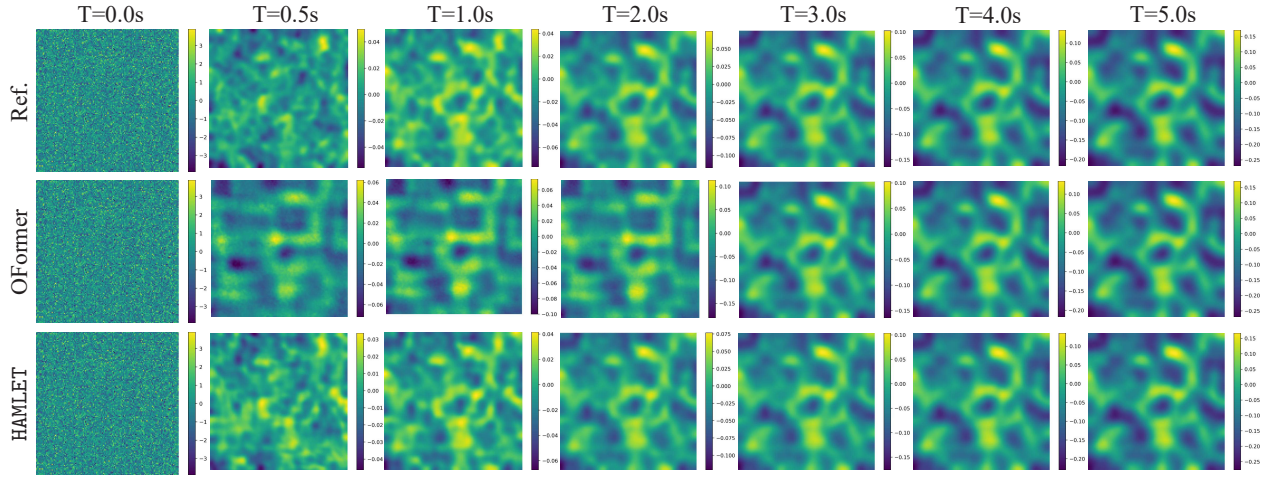

Figure 7. Time evolution (inhibitor) of the diffusion-reaction process as predicted by OFormer and HAMLET compared with the reference solution. HAMLET's predictions closely align with the reference, indicating its strong performance in modelling dynamic PDE systems.

● **Discretisation Invariance.** Suppose $\mathcal{P}$ is a Banach space of $\mathbb{R}^s$-valued functions on the domain $\mathcal{D} \subset \mathbb{R}^n$. Let $\mathcal{S} : \mathcal{D} \to \mathcal{V}$ be an operator, $\mathcal{D}_L$ be an L-point discretisation of $\mathcal{D}$, and $\tilde{\mathcal{S}}_\mu$ the approximate solution operator. For any compact $K \subset \mathcal{P}$, one can define the maximum offset of the operators following the ideas of Kovachki et al. (Kovachki et al., 2021):

$$R_K\left(\mathcal{S}, \tilde{\mathcal{S}}_\mu, \mathcal{D}_L\right) = \sup_{\theta \in K} \left\| \hat{\mathcal{S}}\left(\mathcal{D}_L, \theta|_{\mathcal{D}}\right) - \mathcal{S}(\theta) \right\|_{\mathcal{V}} \tag{14}$$

Given a set of discretisations $\{\mathcal{D}^{(i)}\}_{i=1}^{\infty}$ with the condition $\mathcal{D}^{(i)} \subset \mathcal{D}^{(j)}$ for $j > i$ of the domain $\mathcal{D} \subset \mathbb{R}^n$ such that $\mathcal{D} \subset \cup_{i=1}^{\infty} \mathcal{D}^{(i)}$, we say $\tilde{\mathcal{S}}_\mu$ is discretisation-invariant if the offset tends to zero when evaluating and training the neural operator $\tilde{\mathcal{S}}^{(i)}$ on other N_i discretisations, for any $\theta \in \mathcal{P}$ and any compact set $K \subset \mathcal{P}$:

$$\lim_{N \to \infty} R_K\left(\mathcal{S}(\cdot, \theta), \tilde{\mathcal{S}}^{(N)}(\cdot, \cdot, \theta), \mathcal{D}^{(N)}\right) = 0 \tag{15}$$

To show that a neural network architecture is numerically discretisation-invariant, we must show that the approximation error is approximately constant as we refine the discretisation.

● **Neural Operator.**

Definition B.1 (Neural operator $\tilde{\mathcal{S}}_\mu$)**.** Define the neural operator

$$\tilde{\mathcal{S}}_\mu := \mathcal{Q} \circ (W_L + \mathcal{K}_L + b_L) \circ \cdots \circ \sigma (W_1 + \mathcal{K}_1 + b_1) \circ \mathcal{P} \tag{16}$$

where $\mathcal{P} : \mathbb{R}^{d_a} \to \mathbb{R}^{d_1}$, $\mathcal{Q} : \mathbb{R}^{d_L} \to \mathbb{R}^{d_u}$ are the pointwise neural networks that encode the lower dimension function into higher dimensional space and vice versa. The model stack L layers of $\sigma (W_l + \mathcal{K}_l + b_l)$ where $W_l \in \mathbb{R}^{d_{l+1} \times d_l}$ are pointwise linear operators (matrices), $\mathcal{K}_l : \{\mathcal{D} \to \mathbb{R}^{d_l}\} \to \{\mathcal{U} \to \mathbb{R}^{d_{l+1}}\}$ are integral kernel operators, $b_l : D \to \mathbb{R}^{d_{l+1}}$ are the bias terms made of a linear layer, and σ are fixed activation functions. The parameters μ consist of all the parameters in $\mathcal{P}, \mathcal{Q}, W_l, \mathcal{K}_l, b_l$

C. Further Details on the Datasets & PDEs.

● **Darcy Flow.** The 2D Darcy Flow equation is defined by:

$$\begin{aligned} -\nabla(a(x,y)\nabla u(x,y)) &= f(x,y), \qquad (x,y) \in \Omega, \\ u(x,y) &= 0, \qquad\quad (x,y) \in \partial\Omega, \end{aligned} \tag{17}$$

where $a(x,y)$ and $u(x,y)$ are the diffusion coefficient and the solution respectively, which are defined on a square domain $\Omega = (0,1)^2$. The force term $f(x)$ is set to a constant value β, which affects the scale of the solution $u(x)$.

We conducted experiments on the steady-state solution of 2D Darcy Flow over the uniform square. The objective solution operator $\mathcal{S}$ is defined as follows:

$$\mathcal{S} : a \mapsto u, \quad (x,y) \in \Omega, \tag{18}$$

where $a(x,y)$ and $u(x,y)$ are the diffusion coefficient and the solution respectively, which are defined on a square domain $\Omega = (0,1)^2$. Following the PDEBench (Takamoto et al., 2022) protocol, five subsets with constant value force term $\beta = 0.01, 0.1, 1.0, 10.0, 100.0$ were used in the experiment section, each of which contains 9000/1000 samples for training/testing sets. All data samples are officially discretised and $\times 2$ subsampled in a spatial resolution of 64×64.

● **Shallow-Water.** The 2D Shallow Water equation is written as follows:

$$\begin{aligned} \partial_t h + \partial_x hu + \partial_y hv &= 0, \\ \partial_t hu + \partial_x \left(u^2 h + \frac{1}{2} g_r h^2 \right) &= -g_r h \partial_x b, \\ \partial_t hv + \partial_y \left(v^2 h + \frac{1}{2} g_r h^2 \right) &= -g_r h \partial_y b, \end{aligned} \tag{19}$$

where $u := u(x,y,t)$ and $v := v(x,y,t)$ describes the velocities in horizontal and vertical direction and $h := h(x,y,t)$ describes the water depth. $b := b(x,y)$ and g_r denote spatially variable bathsymetry and gravitational acceleration, respectively. This specific dataset describes a 2D radial dam break scenario on a square domain $\Omega = [-2.5, 2.5]^2$ and time range $t \in [0,1]$. with the following initialisation setting:

$$h(t=0,x,y) = \begin{cases} 2.0, & \text{for } r < \sqrt{x^2+y^2} \\ 1.0, & \text{for } r \geq \sqrt{x^2+y^2} \end{cases} \tag{20}$$

where the radius $r \sim \mathcal{D}(0.3, 0.7)$.

We conducted experiments on the time-dependent system of 2D Shallow Water equations, providing an appropriate model for analysing problems related to free-surface flows. The objective solution operator $\mathcal{S}$ is defined as follows:

$$\mathcal{S} : h|_{t\in[0,t']} \mapsto h|_{t\in(t',T]}, \quad (x,y) \in \Omega, \tag{21}$$

where $t' = 0.009s$ and $T = 1.000s$, $\Omega = [-2.5, 2.5]^2$. $h := h(x,y,t)$ describes the depth of the water.

Each sample in the dataset is officially discretised into a spatial resolution of 128×128 and a time resolution of 101 respectively, where the first 10 time points were used as input and the rest of 91 as target. Following the PDEBench (Takamoto et al., 2022) protocol, the dataset contain 900/100 for training/testing.

● **Diffusion Reaction.** The Diffusion Reaction equations are rewritten as follows:

$$\begin{aligned} \partial_t u &= D_u \partial_{xx} u + D_u \partial_{yy} u + R_u, \\ \partial_t v &= D_v \partial_{xx} v + D_v \partial_{yy} v + R_v, \end{aligned} \tag{22}$$

where the activator $u = u(x, y, t)$ and the inhibitor $v = v(x, y, t)$ are two non-linearly coupled variables. The diffusion coefficients for the activator and inhibitor are $D_u = 1 \times 10^{-3}$ and $D_v = 5 \times 10^{-3}$, respectively. The activator and inhibitor reaction functions are defined as follows:

$$R_u(u, v) = u - u^3 - k - v, \qquad R_v(u, v) = u - v, \tag{23}$$

where $k = 5 \times 10^{-3}$. The simulation domain includes $\Omega = [-1, 1]^2$ and $t \in [0, 5]$.

The objective solution operator $\mathcal{S}$ is defined as follows:

$$\mathcal{S} : \{u, v\}|_{t\in[0,t']} \mapsto \{u, v\}|_{t\in(t',T]}, \quad (x, y) \in \Omega, \tag{24}$$

where $t' = 0.045s$, $T = 5.000s$ and $\Omega = [-1, 1]^2$. $u := u(x, y, t)$ and $v := v(x, y, t)$ is the activator and inhibitor. Similarly to the Shallow Water dataset, each sample is officially discretised and subsampled at a spatial resolution of 128×128 and a time resolution of 101 (10 for input and 91 for target). Following the PDEBench (Takamoto et al., 2022) protocol, the dataset contain 900/100 for training/testing.

● **Airfoil.** The 2D time-dependent compressible flow for Airfoil problem is written as follows:

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho \mathbf{u}) &= f_1, \\ \partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + p\mathbb{I}) &= \mathbf{f}_2, \\ \partial_t (\rho E) + \nabla \cdot (\rho E \mathbf{u} + p\mathbf{u}) &= f_3, \end{aligned} \tag{25}$$

where $\mathbf{u} := \mathbf{u}(x, t)$, $\rho := \rho(x, t)$ and $p := p(x, t)$ are the velocity field, density and pressure respectively. E denotes the total energy per unit mass. and f_1, $\mathbf{f}_2$, f_3 are generic source terms. The simulation domain includes $x \in \Omega$, $t \in [0, 4.800]$. The dataset we use is pregenerated from (Pfaff et al., 2021) [4] and preprocessed by (Li et al., 2023b) [5].

The objective solution operator $\mathcal{S}$ is defined as follows:

$$\mathcal{S} : \{\mathbf{u}, \rho, p\}|_{t\in[0,t']} \mapsto \{\mathbf{u}, \rho, p\}|_{t\in(t',T]}, \quad (x, y) \in \Omega, \tag{26}$$

where $t' = 0.576s$ and $T = 4.800s$. $\mathbf{u} := \mathbf{u}(x, y, t)$ is the velocity field, $\rho := \rho(x, y, t)$ is the density, and $p := p(x, y, t)$ is the pressure. Following the settings from (Pfaff et al., 2021), 1000/100 samples are included for training and testing, respectively. Each data sample is discretised into 5,233 irregular grid, and 101 time points, which are further temporally undersampled to 26 time points (4 for input and 22 for target).

D. Further Details on Model Architecture & Implementations

Table 6 provides a detailed breakdown of the implementation specifics of our model trained on various datasets, including Darcy Flow 2D, Shallow Water 2D, Diffusion Reaction 2D, and Airfoil. The table also outlines the dataset settings, indicating the amount of training and testing data used, spatial and temporal resolution, and specific graph-related parameters such as radius and node feature channels, highlighting the bespoke nature of the model configuration. The dataset-specific hyperparameters follow the PDEBench (Takamoto et al., 2022) setting, while model-specific hyperparameters follow the default setting of baseline methods suggested by the code repositories or their papers.

We have presented results of our HAMLET on a wide range of datasets, some of which are larger and more complex, e.g., larger spatial and temporal resolution on Diffusion Reaction 2D. On these datasets, there are no scalability issues found in our HAMLET, implying that our method does not suffer from the scalability issue. To indicate computational complexity, we use the inference time[6].We directly train HAMLET unrolling for 90 timestamps.

[4] https://github.com/deepmind/deepmind-research/tree/master/meshgraphnets

[5] https://github.com/BaratiLab/OFormer

[6] Inference time is measured as an average of 50 runs, with a batch size of 1, on an NVIDIA RTX3090. We use inference time to indicate computational complexity, since FLOPs are not constant for our proposed model, as they partly depend on the number of edges.

Table 6. The implementation details of HAMLET

Dataset	Darcy Flow 2D	Shallow Water 2D	Diffusion Reaction 2D	Airfoil
Learning Rate (LR) Parameter				
Initial LR	0.0001	0.0001	0.0001	0.0001
Schedule	OneCycleLR	OneCycleLR	OneCycleLR	OneCycleLR
"div_factor"	20	20	20	20
"pct_start"	0.05	0.05	0.05	0.05
"final_div_factorr"	1000	1000	1000	1000
Optimisation Parameter				
Optimiser	Adam	Adam	Adam	Adam
Data Loss Type	MSE	MSE	MSE	Reletive L2 Norm
Data Loss Weight	1.0	1.0	1.0	1.0
Encoder (Input) Architecture				
Hidden Dim (EncI)	96	192	192	192
#Blocks – Graph Transformer	10	6	6	6
#Head	8	4	4	4
Encoder (Query) Architecture				
Hidden Dim (EncQ)	96	192	192	192
##Linear Layer – MLP	2	2	2	2
Decoder Architecture				
Hidden Dim (Dec)	256	512	512	512
#Linear Layer – Output-MLP	2	3	3	3
#Linear Layer – Propagator-MLP	N.A.	3	3	3
#Head - Cross Attention	8	4	4	4
Dataset Setting (Benchmark Setting)				
#Training Data	9000	900	900	1000
#Testing Data	1000	100	100	100
Spatial Res.	64×64	128×128	128×128	N.A.
Temporal Res.	N.A.	101	101	26
Length – Input Seq.	N.A.	10	10	4
Length – Output Seq.	N.A.	91	91	22
Graph - Radius	0.1	0.1	0.08	0.001
Graph - Node Feature Chnl.	3	12	22	18
Number of Parameters	3.16M	4.29M	4.30M	5.16M
Inference Time	0.306s	0.313s	0.378s	0.094s