AttnPINNs: Physics-informed neural networks under the self-attention mechanism for solving PDEs

Jin Song

supervisor： Zhenya Yan

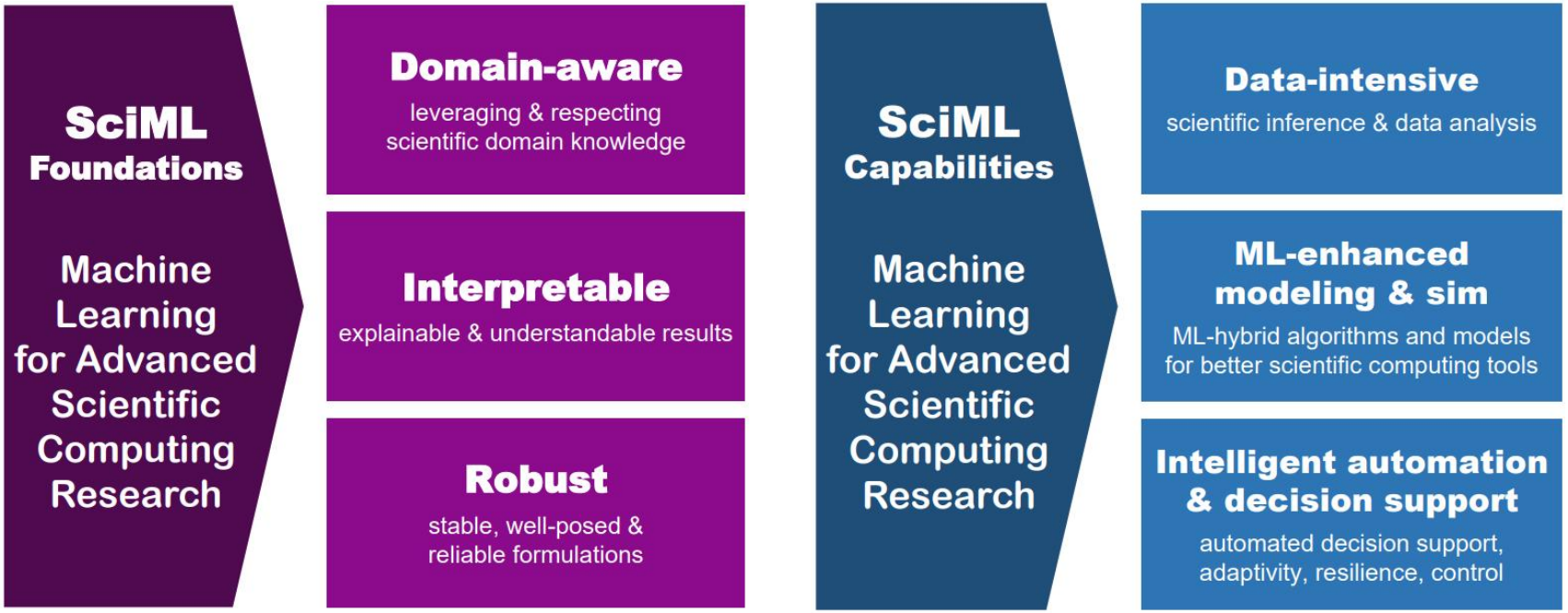
KLMM, Academy of Mathematics and Systems Science,

Chinese Academy of Sciences

10/18/2024

Brown University

Scientific Machine Learning



Numerical methods (physical basis)	Machine learning methods (data driven)
Robust & Interpretable	Generalization & Intelligence
Large-scale limitations & Lack of flexibility	Data dependence & Poor interpretability

Physical knowledge + Machine learning

N. Baker, F. Alexander, et al., Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence, Tech. report, U.S. DOE Office of Science, Washington, DC, 2019.

Outline

1.1 PINNs

Problem setup:

$$\begin{cases} \mathcal{A}[u(\boldsymbol{x})] = 0, \ \boldsymbol{x} \in \Omega, \\ \mathcal{B}[u(\boldsymbol{x})] = 0, \ \boldsymbol{x} \in \partial\Omega, \end{cases}$$

Numerical methods	Machine learning methods
Mesh-based	Mesh-free
Discretization	Automatic differentiation
No generalization	Generalization

Neural network:

$$A_j = \sigma\left(\mathcal{F}_j(A_{j-1})\right) = \sigma(w_j \cdot A_{j-1} + b_j), \quad j = 1, 2, ..., n,$$

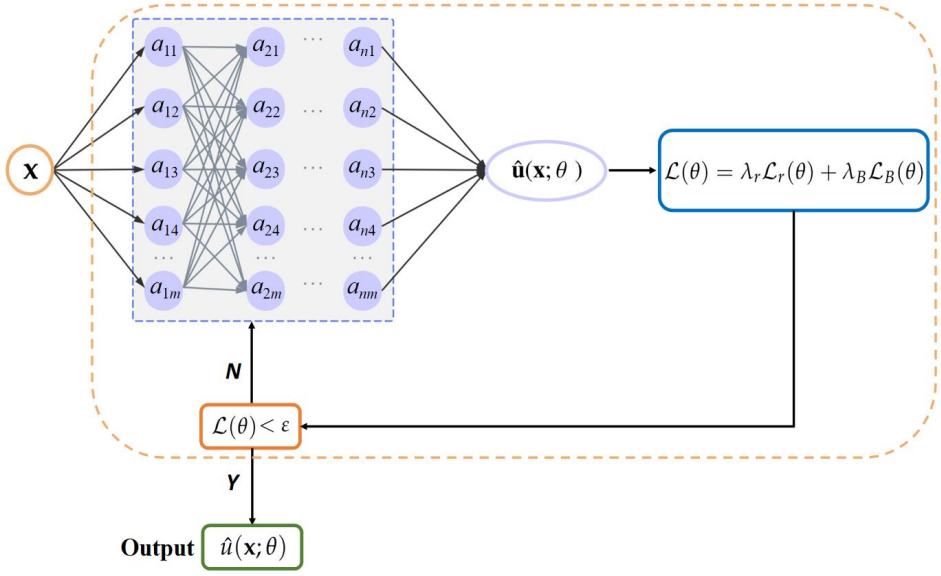
$$u(\boldsymbol{x};\theta) = A_{n+1} = \left(\mathcal{F}_{n+1} \circ \sigma \circ \mathcal{F}_n \circ \cdots \circ \sigma \circ \mathcal{F}_1\right)(\mathbf{x}),$$

Loss function:

$$\mathcal{L}(\theta) = \lambda_r \mathcal{L}_r(\theta) + \lambda_B \mathcal{L}_B(\theta)$$

$$\mathcal{L}_r(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} |\mathcal{A}[u(\boldsymbol{x}_i)]|^2$$

$$\mathcal{L}_B(\theta) = \frac{1}{N_B} \sum_{i=1}^{N_B} |\mathcal{B}[u(\boldsymbol{x}_i)]|^2$$



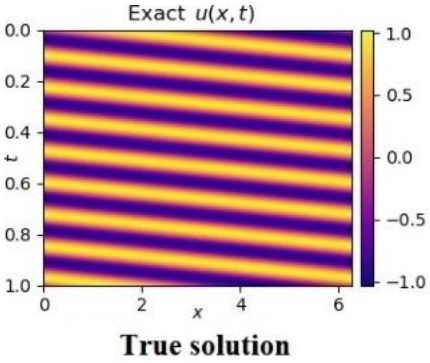
M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys., 378 (2019), pp. 686-707.

Vanilla PINNs

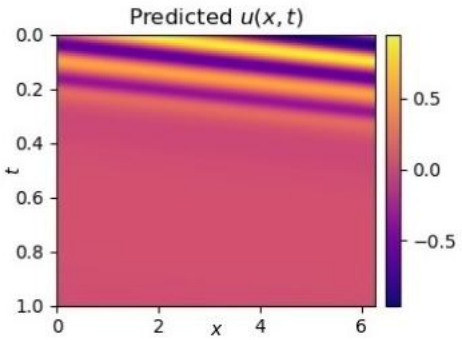
Vanilla PINNs are limited

- **The convection equation**

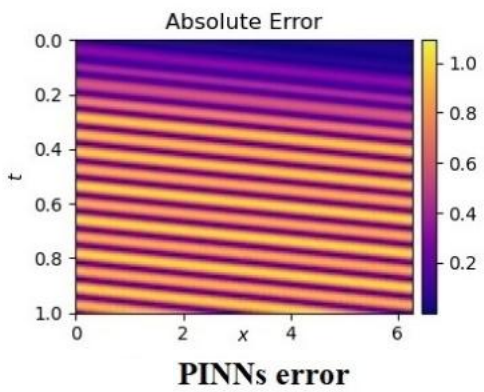
$$\begin{cases} u_t + \beta u_x = 0, \quad x \in [0, 2\pi], \ t \in [0, 1], \\ u(x, 0) = \sin x, \quad x \in [0, 2\pi], \\ u(0, t) = u(2\pi, t), \quad t \in [0, 1], \end{cases}$$



True solution



PINNs prediction



PINNs error

The PINNs is only able to achieve the good solutions for **small values of convection coefficient** or **in a short time**, however it fails when β becomes larger, reaching a relative error of almost 100% for $\beta > 10$.

A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neur. Inf. Process. Syst., 34 (2021), pp. 26548-26560.

Vanilla PINNs

Vanilla PINNs are limited

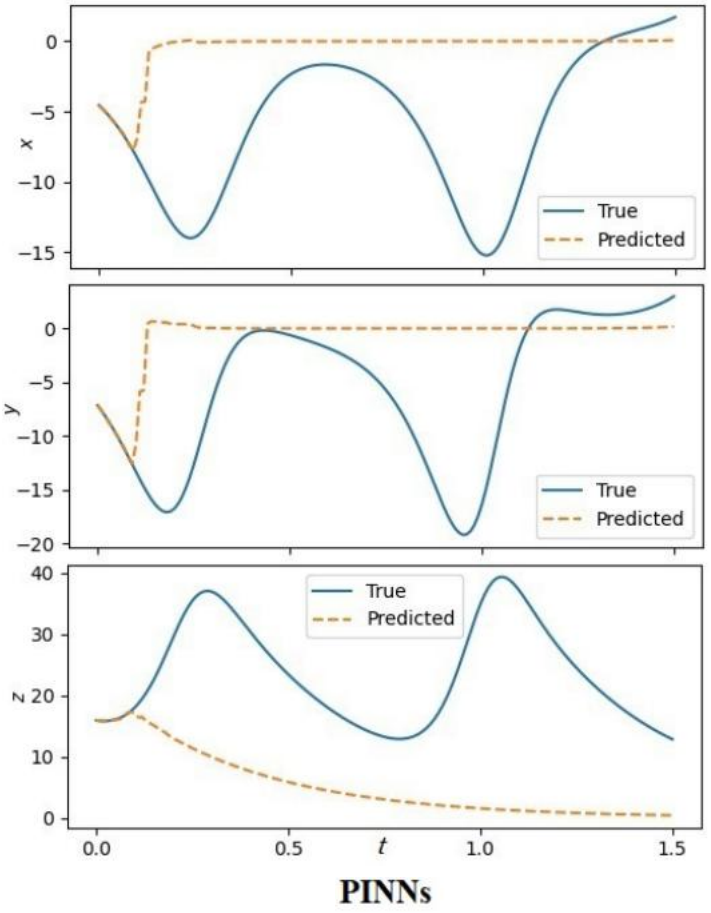
- **The Lorenz system**

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(\rho - z) - y, \\ \dot{z} = xy - \beta x, \end{cases}$$

The dynamics of the Lorenz system demonstrates the chaotic behavior for parameters $\sigma = 10$, $\rho = 28$, $\beta = 8/3$.

Also we consider exactly imposing initial conditions as follows by changing the final output of the network $\vec{u} = [x, y, z]$

$$\hat{\vec{u}} = t \cdot \vec{u} + \vec{u}(0)$$



1.2 PINNs-Based methods

Some improvements to PINNs

(I) Modify the loss function.

- Soft Attention (McClenny, 2020)
- Causal training (Wang, 2022)
- RBA PINNs (Anagnostopoulos, 2024)
- NTK PINNs (Wang, 2022)
- Gradient-enhanced PINNs (Yu, 2022)

(II) Adaptive sampling strategies and domain decomposition.

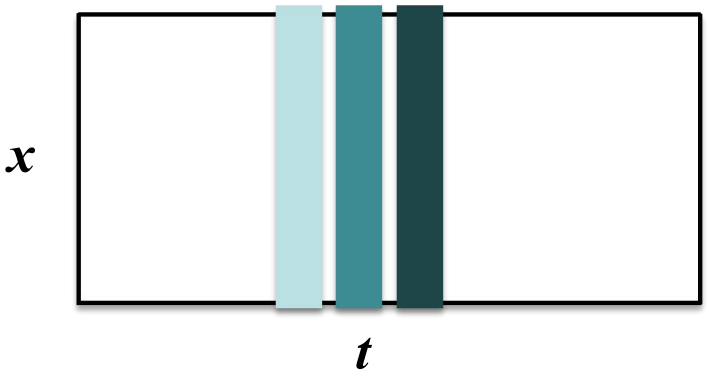
- cPINNs (Jagtap, 2020)
- PhyGeoNet (Gao, 2021)
- RoPINNs (Wu, 2024)
- XPINNs (Jagtap, 2020)
- FI-PINNs (Zhou, 2023)

(III) Advanced network architecture.

- DeepONet (Lu, 2021)
- PINNsFormer (Zhao, 2023)
- FLS (Wong, 2024)
- PhyGeoNet (Gao, 2021)
- RPNNs (Patsatzis, 2023)

Dynamical systems

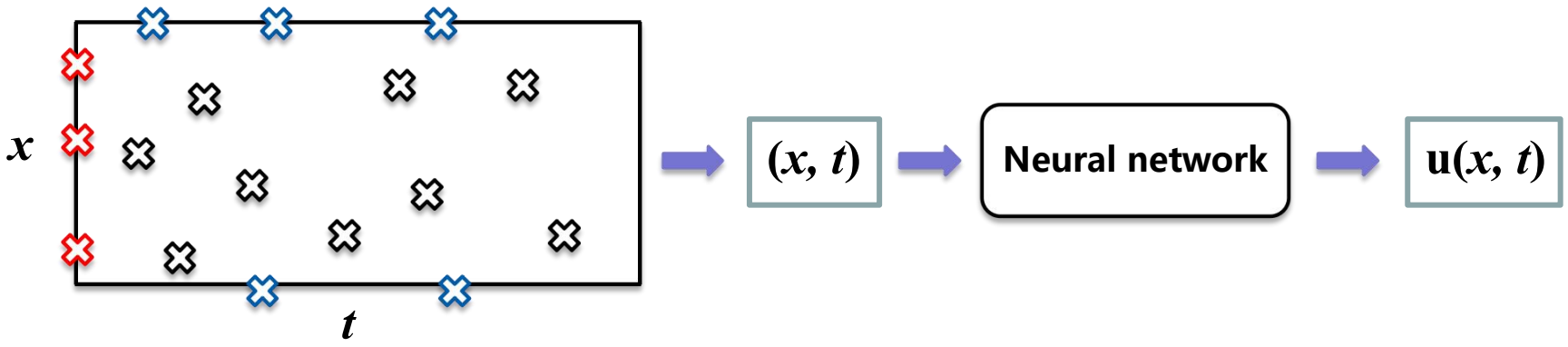
$$u_t - \mathcal{N}[x; u(x,t)] = 0$$



Numerical methods (Euler)

$$u_{n+1} = u_n + \Delta t \mathcal{N}(u_n)$$

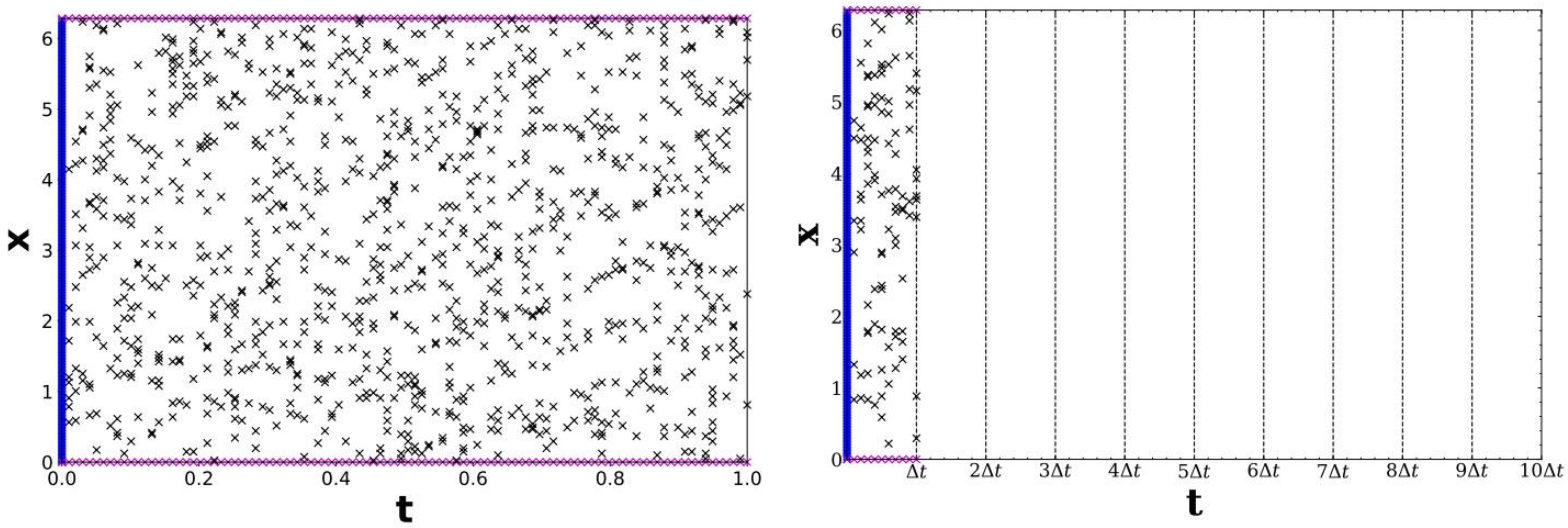
PINNs



Most of PINNs methodologies typically **combine spatial information *x* and temporal information *t* as network inputs**, overlooking **temporal dependencies** across previous or subsequent time steps

Seq2seq learning

Sequence-to-sequence learning task (*Krishnapriyan, 2021*)



(a) *Regular PINN training*

(b) *Sequence-to-sequence learning (model trained every Δt)*

In contrast to regular PINN training, the solution in seq2seq learning is predicted for **only one Δt step at a time**. Then, the **predicted solution at $t = \Delta t$ is used as the initial condition** for the next segment.

A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neur. Inf. Process. Syst., 34 (2021), pp. 26548-26560.

Causal training

Respecting Causality (*Wang, 2024*)

$$\mathcal{L}_r(\boldsymbol{\theta}) = \frac{1}{N_t}\sum_{i=1}^{N_t} w_i \mathcal{L}_r(t_i, \boldsymbol{\theta}).$$

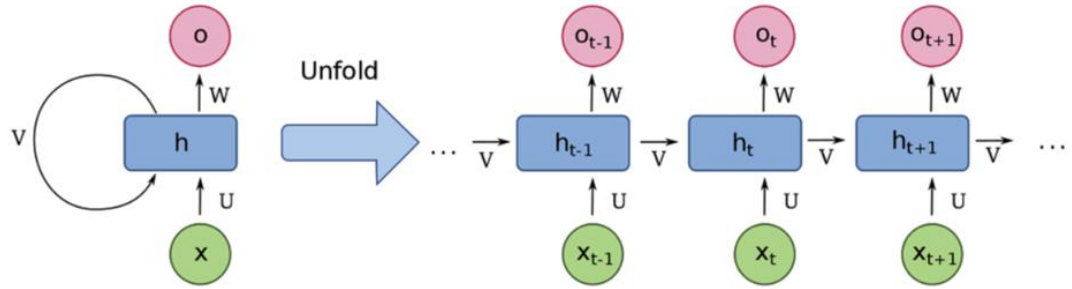
$$w_i = \exp\left(-\epsilon \sum_{k=1}^{i-1} \mathcal{L}_r(t_k, \boldsymbol{\theta})\right), \text{ for } i = 2, 3, \ldots, N_t,$$

The authors recognize that the weights ω_i should be large – and therefore allow the minimization of $\mathcal{L}_r(t_i, \boldsymbol{\theta})$ – only if all residuals $\{\mathcal{L}_r(t_k, \boldsymbol{\theta})\}_{k=1}^{i}$ before t_i are minimized properly.

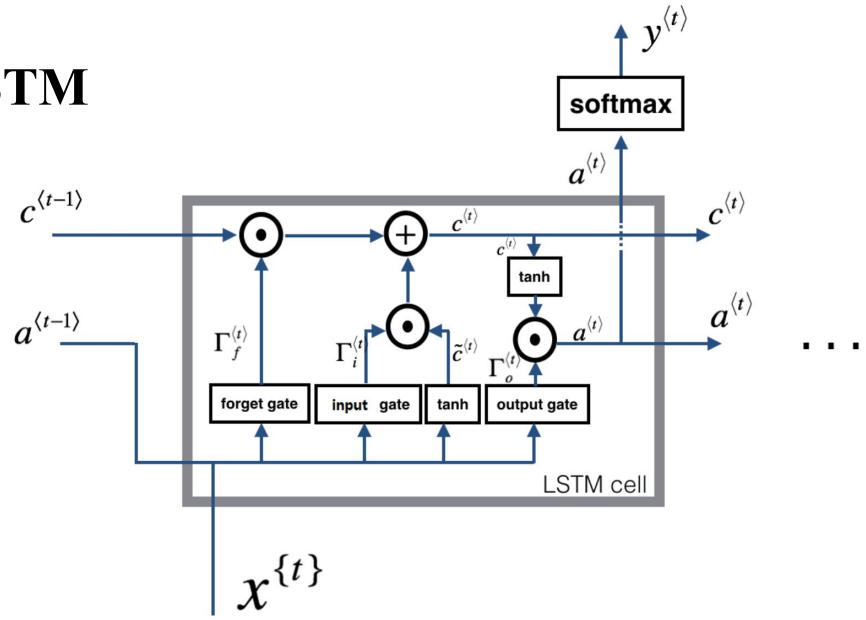
S. Wang, S. Sankaran, and P. Perdikaris, Respecting causality for training physics-informed neural networks, Comput. Methods Appl. Mech. Engrg., 421 (2024), 116813.

Seq2seq learning

RNN



LSTM



$$\Gamma_f^{\langle t \rangle} = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$$

$$\Gamma_i^{\langle t \rangle} = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$

$$\tilde{c}^{\langle t \rangle} = \tanh(W_C[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_C)$$

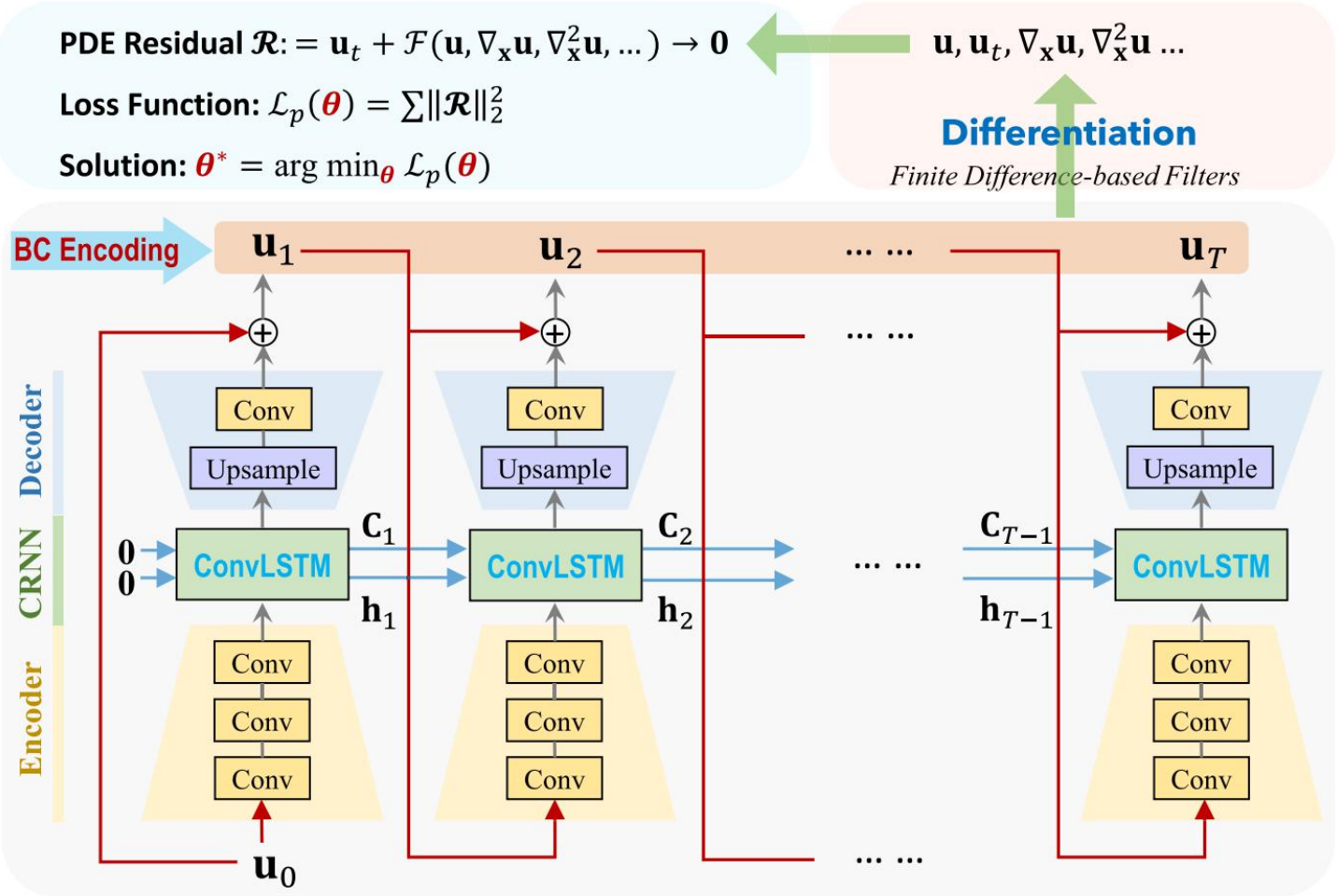
$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle}$$

$$\Gamma_o^{\langle t \rangle} = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$$

$$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} \circ \tanh(c^{\langle t \rangle})$$

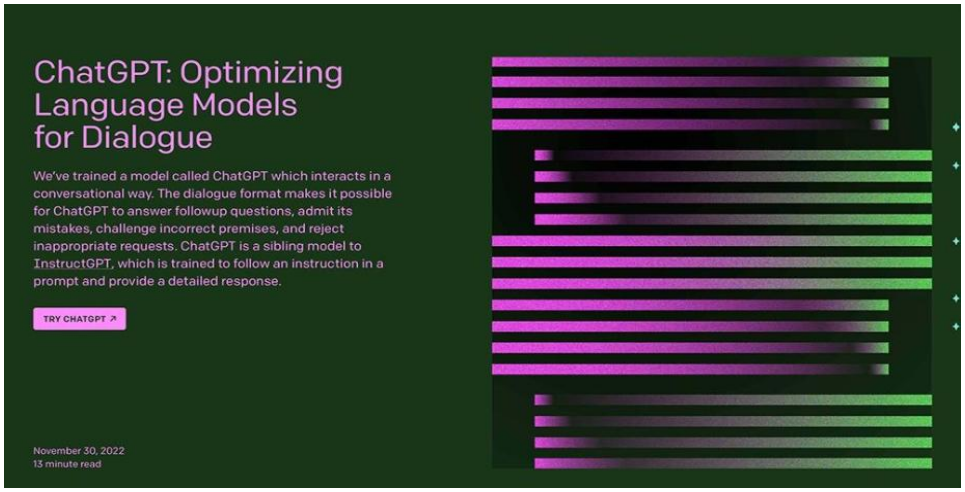
Seq2seq learning

Physics-informed convolutional-recurrent learning (PhyCRNet)

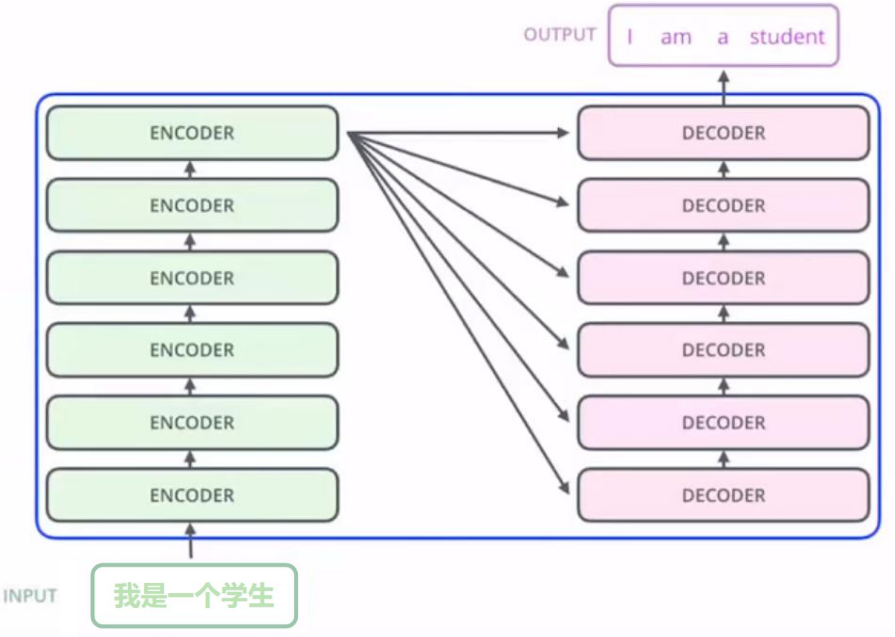


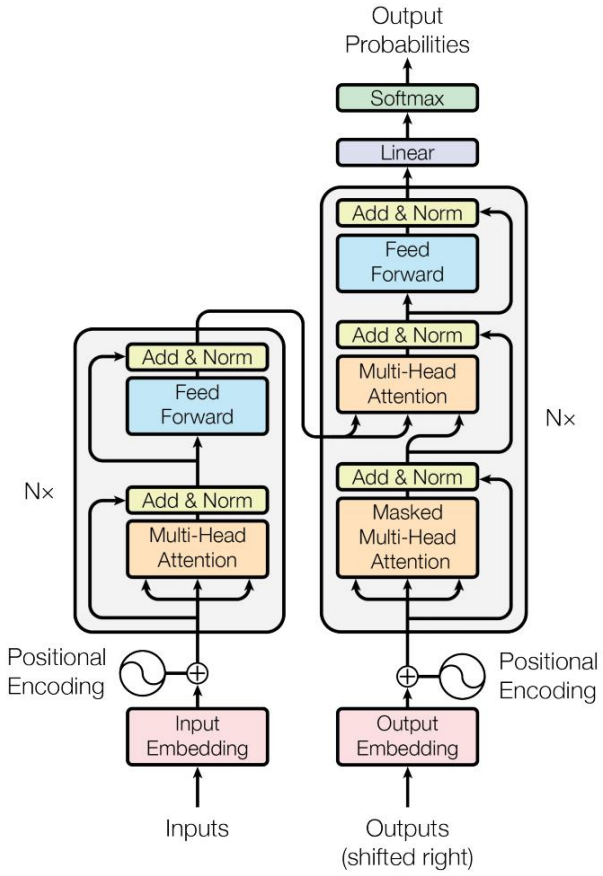
P. Ren, C, Rao, Y. Liu, J. X. Wang, and H. Sun, H, PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs, Computer Methods in Applied Mechanics and Engineering, 389, (2022) 114399.

Transformer



GPT: Generative Pre-trained Transformer





Transformer is a deep neural network architecture to solve the machine translation problem in Natural Language Processing.

A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. U. KAISER, AND I. POLOSUKHIN, Attention is all you need, Adv. Neur. Inf. Process. Syst., 30 (2017).

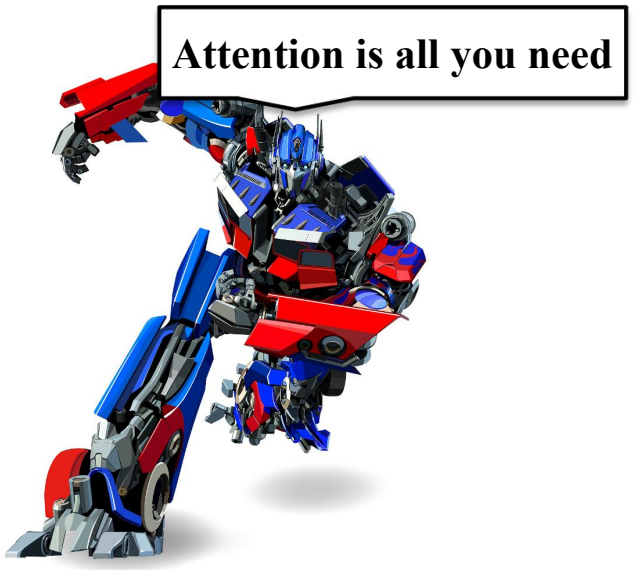
$$T: \quad V \quad \rightarrow \quad V, \quad V = \mathbb{R}^{N \times D}$$

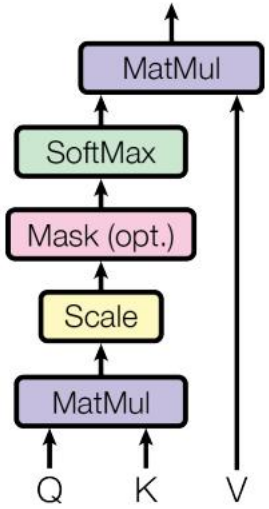
$(u_1, u_2, \ldots, u_N)$ $(v_1, v_2, \ldots, v_N)$

Encoding Decoding

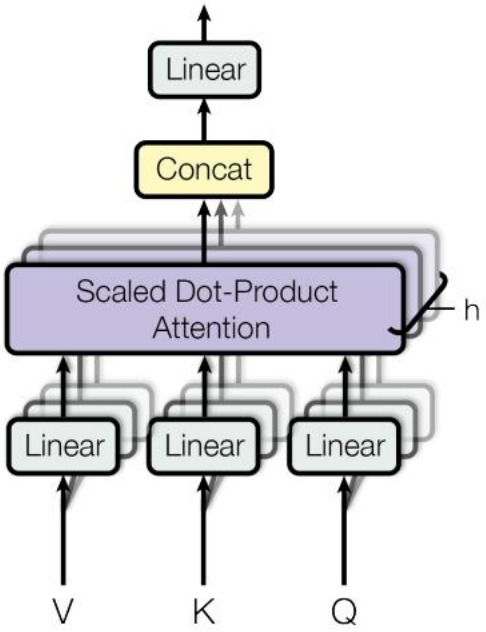
Sentence **Sentence**

Sentence in one language, **embedded into high dimensional spaces**, "translated" to another language's embedding after stacking multiple layers of the same module.





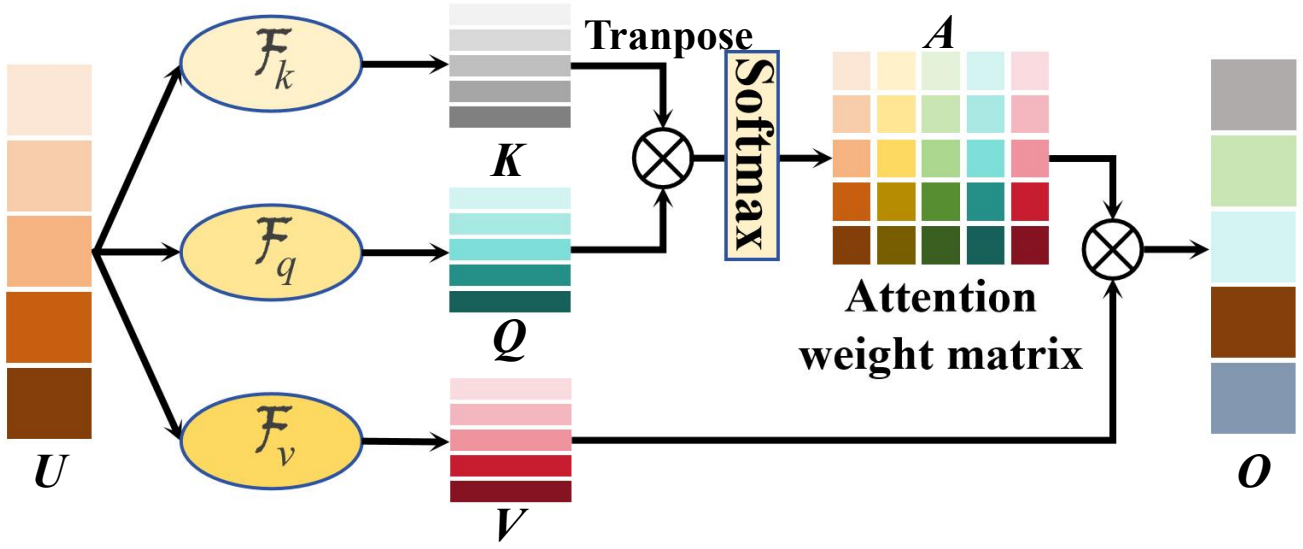
Scaled Dot-Product Attention



Multi-Head Attention

1.3 Self-Attention mechanism

Scaled dot-product attention



For a given input sequence $U = (u_1, u_2, \ldots, u_N)^\top \in \mathbb{R}^{N \times D_u}$, self-attention computes the output sequence O from U as follows

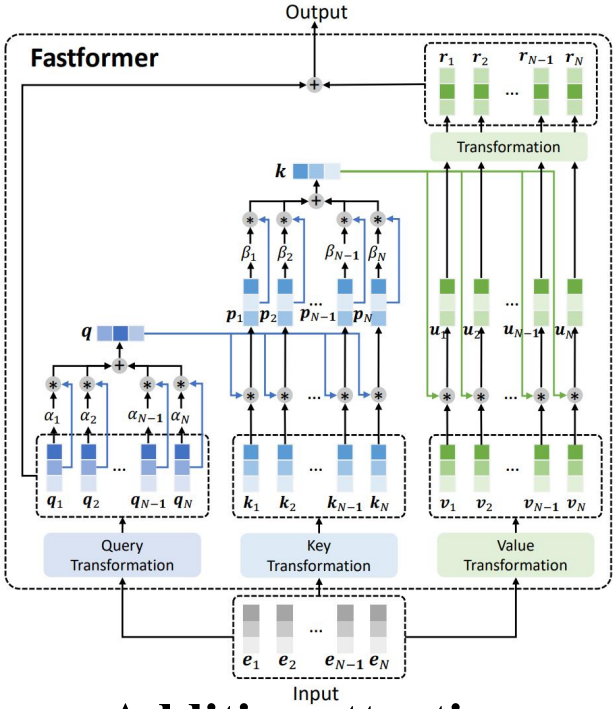
$$K = \mathcal{F}_k(U), Q = \mathcal{F}_q(U), V = \mathcal{F}_v(U),$$

$$O = \text{softmax}(QK^\top / \sqrt{D_k})V,$$

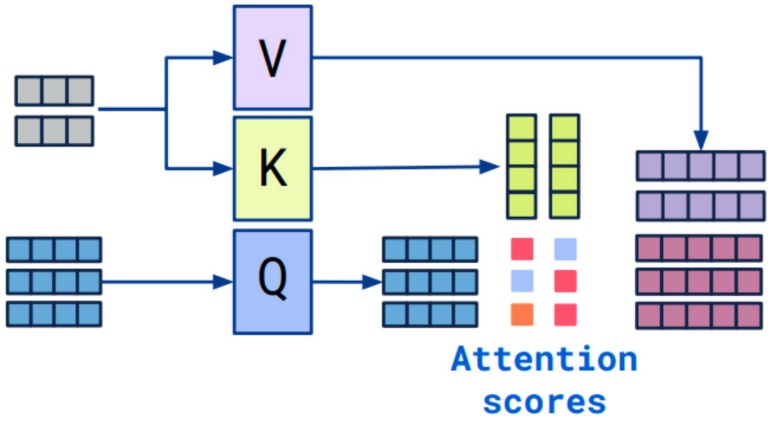
where $K = (k_1, k_2, \ldots, k_N)^\top \in \mathbb{R}^{N \times D_k}$, $Q = (q_1, q_2, \ldots, q_N)^\top \in \mathbb{R}^{N \times D_k}$, $V = (v_1, v_2, \ldots, v_N)^\top \in \mathbb{R}^{N \times D_v}$ are the key, query, and value vectors

A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. U. KAISER, AND I. POLOSUKHIN, Attention is all you need, Adv. Neur. Inf. Process. Syst., 30 (2017).

1.3 Self-Attention mechanism



Additive attention



Cross-attention

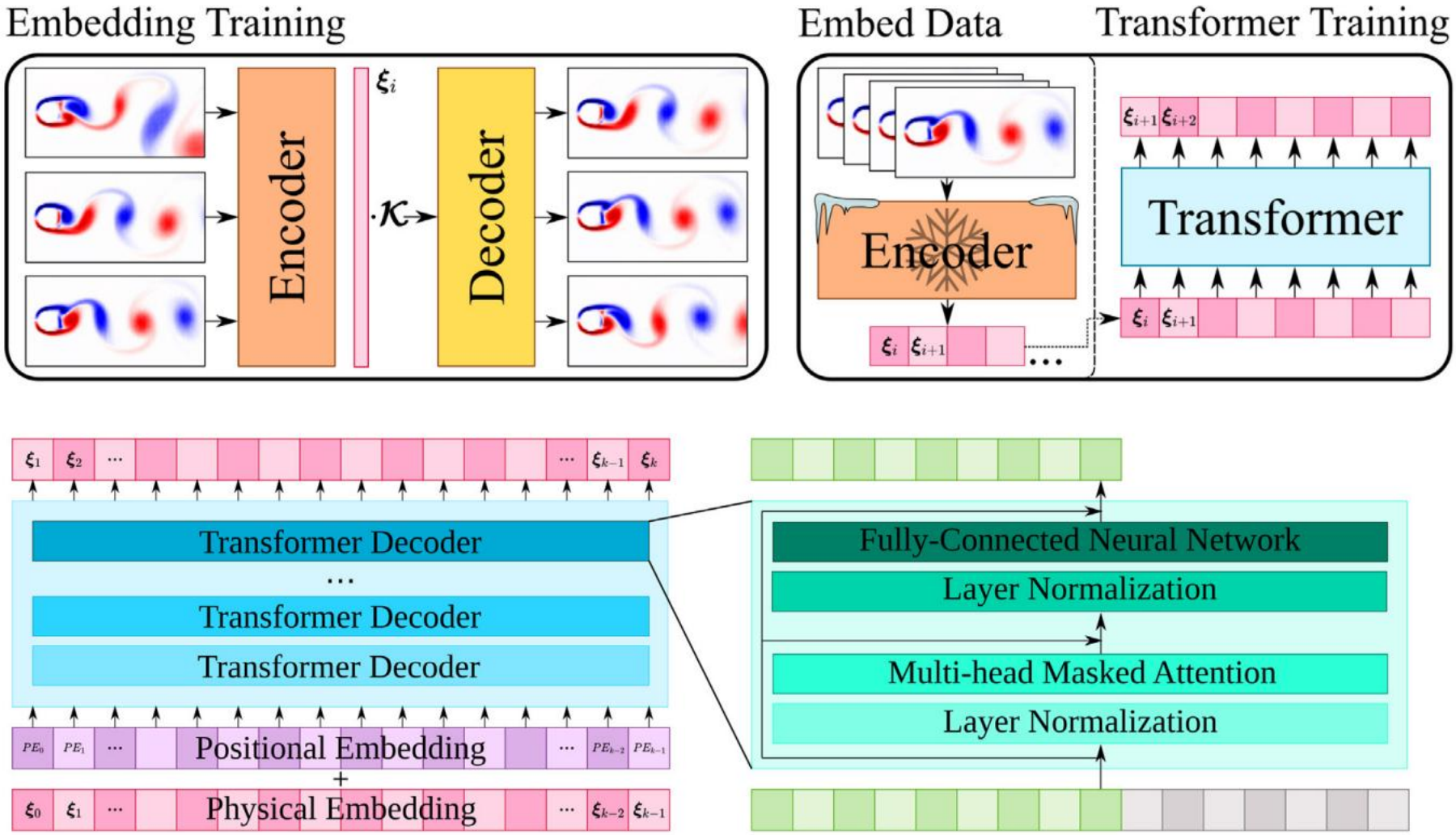
Definition 1 (Fourier Attention) *A Fourier attention is a multi-head attention that does nonparametric regression using the generalized Fourier nonparametric regression estimator $f_{N,R}$. The output $\hat{\boldsymbol{h}}_i$ of the Fourier attention is then computed as*

$$\hat{\boldsymbol{h}}_i := f_{N,R}(\boldsymbol{q}_i) = \frac{\sum_{i=1}^{N} \mathbf{v}_i \prod_{j=1}^{D} \phi\left(\frac{\sin(R(q_{ij}-k_{ij}))}{R(q_{ij}-k_{ij})}\right)}{\sum_{i=1}^{N} \prod_{j=1}^{D} \phi\left(\frac{\sin(R(q_{ij}-k_{ij}))}{R(q_{ij}-k_{ij})}\right)} \qquad \forall\, i \in [N].$$

[1]Fastformer: Additive attention can be all you need, arXiv preprint arXiv:2108.09084, 2021.
[2]Ccnet: Criss-cross attention for semantic segmentation, CVPR, (2019), pp. 603-612.
[3]Fourierformer: Transformer meets generalized fourier integral theorem, Adv. Neur. Inf. Process. Syst., 35 (2022), pp. 29319-29335.

1.3.1 Transformers for PDEs

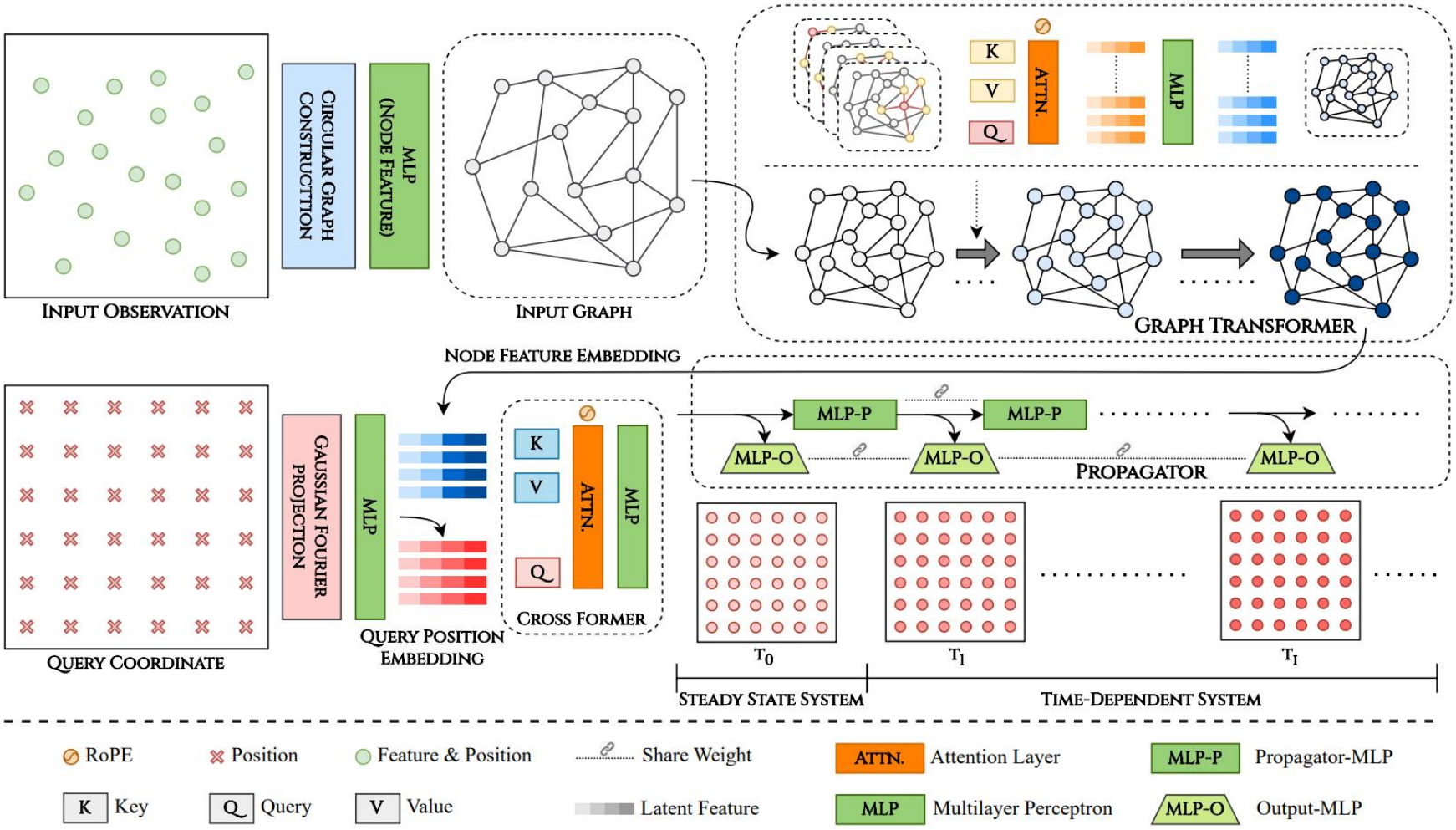
Transformers for modeling physical systems



N. GENEVA, AND N. ZABARAS, Transformers for modeling physical systems, Neural Netw., 146 (2022), pp. 272–289.

1.3.1 Transformers for PDEs

HAMLET: Graph Transformer Neural Operator for PDEs



A. BRYUTKIN, J. HUANG, Z. DENG, G. YANG, C. B. SCHONLIEB, AND A. AVILES-RIVERO, Hamlet: Graph transformer neural operator for partial differential equations, arXiv preprint arXiv:2402.03541, 2024.

1.3.1 Transformers for PDEs

PINNsFormer: A transformer-based framework for physics-informed neural networks

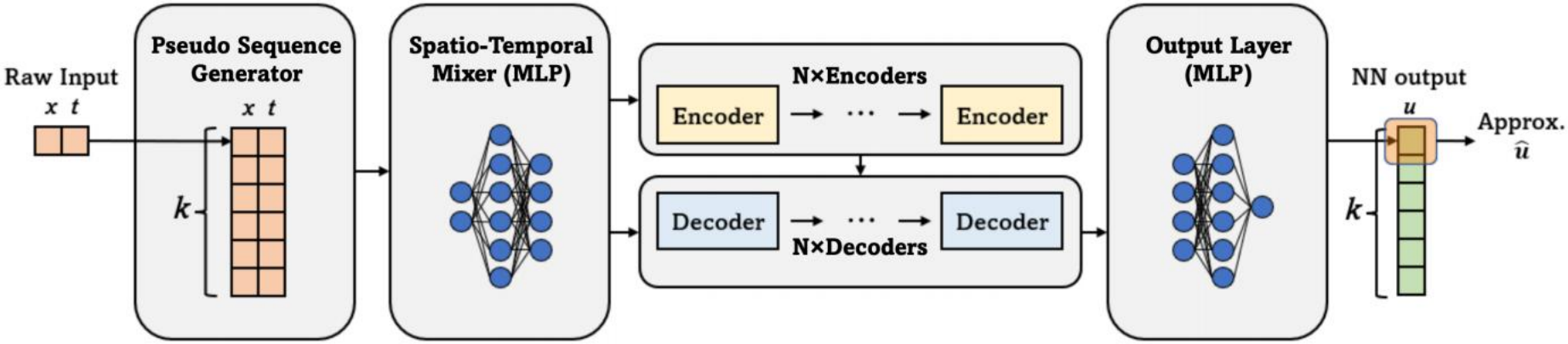


Figure 1: Architecture of proposed PINNsFormer

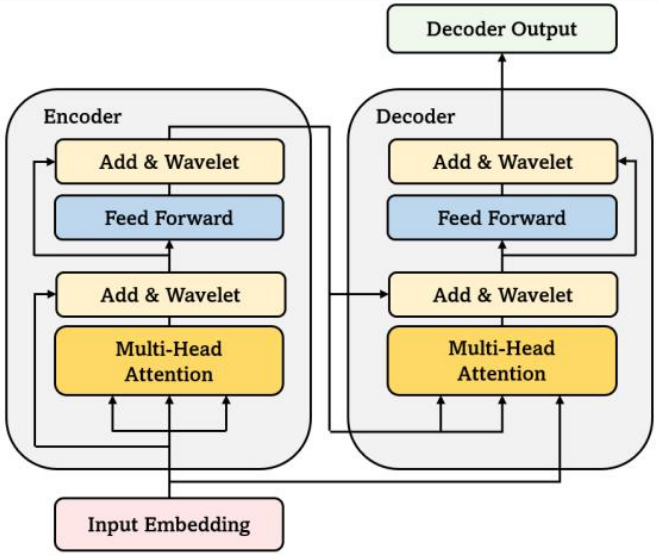


Figure 2: The architecture of PINNsFormer's Encoder-Decoder Layers.

Z. ZHAO, X. DING, AND B. A. PRAKASH, PINNsFormer: A transformer-based framework for physics-informed neural networks, arXiv preprint arXiv:2307.11833, 2023.

2.1 The framework of AttnPINNs

AttnPINNs consists of five components of its architecture: **sequence operator, pre-training PINNs, dimension lifting mapping, self-attention layer, and output layer**.

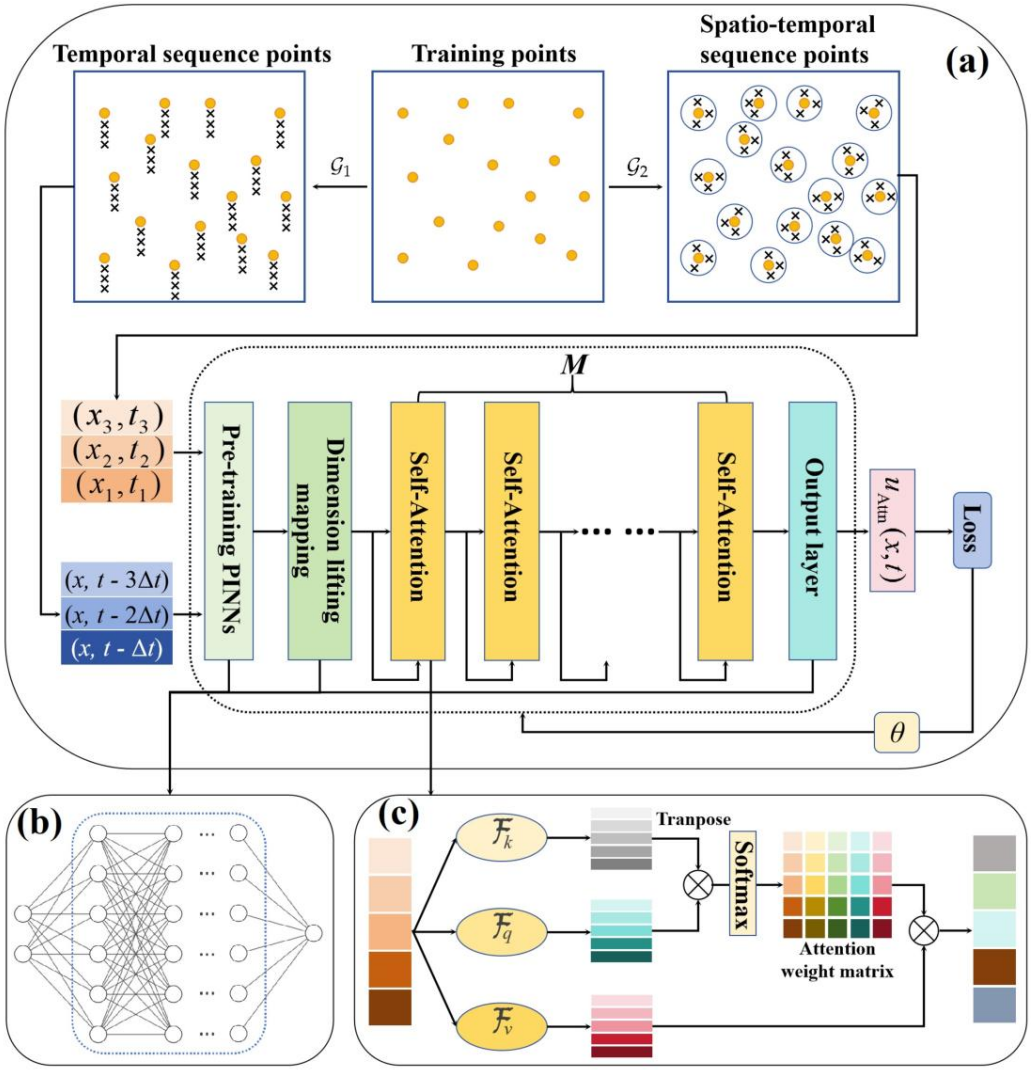


Figure: (a) The architectural overview of AttnPINNs. (b) The diagram of a fully connected pre-training PINN. (c) The schematic of dot-product attention mechanism.

2.1 The framework of AttnPINNs

Component 1: **Sequence operator.**—Firstly, recognizing the inherent capability of self-attention to discern sequence dependencies, we define a sequence operator $\mathcal{G}$. This operator is tasked with transforming inputs from spatio-temporal points into sequential formats

$$\mathcal{G} : (\boldsymbol{x}, t) \in \mathbb{R}^{d+1} \to \mathcal{G}[(x, t)] \in \mathbb{R}^{N\times(d+1)}.$$

Here we present two schemes for the sequence operator $\mathcal{G}$.

$$\mathcal{G}_1 : (\boldsymbol{x}, t) \to \mathcal{G}[(x, t)] = [(\boldsymbol{x}, t - N\Delta t), (\boldsymbol{x}, t - (N-1)\Delta t), \ldots, (\boldsymbol{x}, t - \Delta t)]^\top,$$

$$\mathcal{G}_2 : (\boldsymbol{x}, t) \to \mathcal{G}[(x, t)] = [(\boldsymbol{x}_N, t_N), (\boldsymbol{x}_{N-1}, t_{N-1}), \ldots, (\boldsymbol{x}_1, t_1)]^\top,$$
$$(\boldsymbol{x}_i, t_i) \in \left\{(X, T) \in \mathbb{R}^{d+1} | \|X - \boldsymbol{x}\|_2^2 + |T - t|^2 < \alpha\right\}, i = 1, 2, \ldots, N,$$

where Δt and α are hyperparameters, inherently determining how far the associated position is from the original point. Besides, the definition of $\mathcal{G}_2$ shows that we take into account not only temporal dependencies but also spatial dependencies. And it is obvious that the range of operator $\mathcal{G}_1$ is contained in the range of $\mathcal{G}_2$ when hyperparameters Δt and α satisfy certain conditions.

2.1 The framework of AttnPINNs

Component 2: Pre-training PINNs.—Training a PINNs over the entire region results in a good approximation of its output $u_p(x, t)$ around the initial value of PDEs. Then the temporal sequences $G[(x, t)]$ are submitted as input to the pre-training PINNs. And the output sequences, denoted as $u_p\{G[(x, t)]\} \in R^{N \times D_u}$ are obtained.

Component 3: Dimension lifting mapping.—In cases of low-dimensional problems, directly presenting the output sequences $u_p\{G[(x, t)]\}$ to self-attention blocks may not adequately capture the intricate relationships between different features. Therefore, dimension lifting is crucial to enrich the information content within each vector. To achieve this, we adopt a straightforward linear transformation, expressed as follows

$$\hat{u}_p = u_p\{\mathcal{G}[(\boldsymbol{x}, t)]\}W_L \in \mathbb{R}^{N \times D_m},$$

where $W_L \in R^{D_u \times D_m}$ represents a linear mapping that can be achieved through a single-layer network.

We denote $\hat{u}_p = \left(\hat{u}_1^{(0)}, \hat{u}_2^{(0)}, \ldots, \hat{u}_N^{(0)}\right)^\top$.

2.1 The framework of AttnPINNs

Component 4: Self-attention layer.—Self-attention layer consists of a number of **identical self-attention blocks** with residual connection. We denote the operation of each layer as follows

$$U^{(i+1)} = U^{(i)} + \mathrm{Attn}(U^{(i)}), \quad i = 0, 1, \ldots, M-1,$$

$$U^{(i)} = \left(\hat{u}_1^{(i)}, \hat{u}_2^{(i)}, \ldots, \hat{u}_N^{(i)}\right)^\top \in \mathbb{R}^{N \times D_m} \text{ and } U^{(0)} = \hat{u}_p.$$

where $\mathrm{Attn}(U) = \mathrm{softmax}(QK^{\mathrm{T}}/\sqrt{D_k})V$ with $D_k = D_v = D_m$, and M represents the number of self-attention blocks.

Component 5: Output layer.—Finally, we construct an output layer to map $U^{(M)} \in R^{N \times D_m}$ from the high dimension to the original dimension $P(U^{(M)}) \in R^{N \times D_u}$, where the operator P can be directly represented as a fully-connected neural network. Thus, the value of solution at point (x, t) can be obtained by taking the last element in the sequence

$$u_{\mathrm{Attn}}(\boldsymbol{x}, t) = \mathcal{P}(U^{(M)})[N, \cdot].$$

2.1.1 Algorithm framework

Algorithm 1 The framework of *AttnPINNs*.

Require: Maximum iteration number K; training data points $\mathcal{D}$; sequence operator $\mathcal{G}$; pre-training PINNs $u_p(\boldsymbol{x}, t)$, the number of self-attention blocks M, output layer $\mathcal{P}$.

Ensure: Output $u_{\text{Attn}}(\boldsymbol{x}, t, \theta^*)$.

Randomly initialize the parameters $\theta_0 = \{W_L, \theta_{\text{Attn}}, \theta_P\}$ s.t. they satisfy the normal distribution, where W_L is the dimension lifting matrix, θ_{Attn} is the learnable parameters of self-attention block, and θ_P is the learnable parameters of output layer $\mathcal{P}$.

Obtain the training sequence points $\mathcal{G}(\mathcal{D})$ by sequence operator $\mathcal{G}$.

Calculate the $u_p[\mathcal{G}(\mathcal{D})]$ by the pre-training PINNs.

for $s = 0 : K$ **do**

Obtain the higher-dimensional data $U^{(0)} = u_p[\mathcal{G}(\mathcal{D})]W_L$ by the dimension lifting matrix W_L.

$U^{(i+1)} = U^{(i)} + \text{Attn}(U^{(i)}), i = 0, 1, \ldots, M-1$, where $\text{Attn}(\cdot)$ is the operation of self-attention block.

$u_{\text{Attn}} = \mathcal{P}(U^{(M)})[N, \cdot]$.

Calculate the loss function $\mathcal{L}_{\text{Attn}}(\theta_s)$ according to the output u_{Attn}.

Update θ_s to θ_{s+1} with optimizer (Adam, L-BFGS, etc) to the minimize loss function $\mathcal{L}_{\text{Attn}}$.

end for

$\theta^* = \theta_K$ and output $u_{\text{Attn}}(\boldsymbol{x}, t, \theta^*)$.

We still use the loss function of PINNs as one of the AttnPINNs, that is,

$$\mathcal{L}_{\text{Attn}}(\theta) = \lambda_r \|\mathcal{A}[u_{\text{Attn}}(\boldsymbol{x}, t; \theta)]\|^2_{L^2(\Omega)} + \lambda_I \|\mathcal{I}[u_{\text{Attn}}(\boldsymbol{x}, 0; \theta)]\|^2_{L^2(\Omega_0)} + \lambda_B \|\mathcal{B}[u_{\text{Attn}}(\boldsymbol{x}, t; \theta)]\|^2_{L^2(\partial\Omega_0)}.$$

2.2 Convergence analysis

Problem setup:

$$\begin{cases} u_t - \mathcal{N}[x; u(x,t)] = 0, \quad x \in \Omega_0,\ t \in [0,T], \\ \mathcal{I}[x; u(x,0)] = 0, \quad x \in \Omega_0, \\ \mathcal{B}[x; u(x,t)] = 0, \quad x \in \partial\Omega_0,\ t \in [0,T], \end{cases} \tag{1}$$

we denote $\Omega = \Omega_0 \times [0,T] \subseteq \mathbb{R}^{d+1}$ and $\mathcal{A} = \partial_t - \mathcal{N}$.

We present the **convergence analysis** of the AttnPINNs to guarantee its effectiveness. Here we prove it when $D_u = 1$ and do not consider the dimension lifting mapping. Firstly, the following assumption is required.

Assumption 1 *For any $\varepsilon_p > 0$, there exists $\delta > 0$, such that for any $x \in \Omega_0$, when $|t| < \delta$, we have $|u_p(x,t) - u(x,t)| < \varepsilon_p$.*

Assumption 1 means that we have successfully approximated the ICs data by PINNs. In practice, it is easy to implement since we can **take the weight of the ICs loss term sufficiently large** when training PINNs.

$$\mathcal{L}(\theta) = \lambda_r \|\mathcal{A}[u_p(x,t;\theta)]\|^2_{L^2(\Omega)} + \lambda_I \|\mathcal{I}[u_p(x,0;\theta)]\|^2_{L^2(\Omega_0)} + \lambda_B \|\mathcal{B}[u_p(x,t;\theta)]\|^2_{L^2(\partial\Omega_0)}.$$

2.2 Convergence analysis

The ability of self-attention:

In the realm of dynamical systems, self-attention closely resembles **numerical time-integration methods.** Besides, compared to traditional numerical methods, self-attention provides a significantly greater capacity for learning the time-integration technique. A self-attention block with a residual connection can approximate an **explicit time-integrator** within arbitrarily small error bounds. Going further, the following lemma states that the same holds for nonlinear systems of the form Eq. (1).

Lemma 1 *Consider the dynamical system of the form,* $u_t = \mathcal{N}[\boldsymbol{x}; u(\boldsymbol{x}, t)]$, *that is, Eq. (1), where* $\mathcal{N}$ *is a linear or nonlinear continuous operator. Let the function* $\mathcal{A}_\theta(U) = U[N] + \mathrm{Attn}_\theta(U)[N]$ *be a self-attention block with a residual connection, of context length* N *and containing learnable parameters* $\theta \in \mathbb{R}^{D_\theta}$, *where* $U[N]$ *and* $\mathrm{Attn}_\theta(U)[N]$ *represent the* N*-th element of the vector. Denote* $u_i = u(\boldsymbol{x}, t - i\Delta t)$, $U = (u_N, u_{N-1}, \ldots, u_1)$ *and let* $u_0 = \mathcal{M}(U)$ *be the* N*-th order explicit Adams-bashforth method time-integrator. Then for any* ε, *there exists* $\theta^* \in \mathbb{R}^{D_\theta}$, *such that* $\|\mathcal{M} - \mathcal{A}\|_\infty \leq \mathcal{O}(\varepsilon)$.

N. GENEVA, AND N. ZABARAS, Transformers for modeling physical systems, Neural Netw., 146 (2022), pp. 272–289.

2.2 Convergence analysis

Proof:

N-th order explicit Adams-bashforth method

$$u_0 = \mathcal{M}(U) = u_1 + \Delta t \sum_{j=1}^{N} b_j \mathcal{N}(u_{N-j+1}),$$

According to the scaled dot-product self-attention calculation (6), we denote $\hat{u}_0 = \mathcal{A}_\theta(U)$. Then we have

$$\hat{u}_0 = \mathcal{A}_\theta(U) = u_1 + \sum_{j=1}^{N} \alpha_{Nj} v_j, \qquad \alpha_{Nj} = \frac{\exp(q_N^\top k_j / \sqrt{D_k})}{\sum_{l=1}^{N} \exp(q_N^\top k_l / \sqrt{D_k})},$$

where $k_j = \mathcal{F}_j(u_{N-j+1})$, $q_j = \mathcal{F}_q(u_{N-j+1})$ and $v_j = \mathcal{F}_v(u_{N-j+1})$.

By choosing appropriate key, query, and value, we have

$$\hat{u}_0 = \mathcal{A}_\theta(U) = u_1 + \sum_{j=1}^{N} \alpha_{Nj} v_j = u_1 + \Delta t \sum_{j=1}^{N} b_j \mathcal{N}(u_{N-j+1}) + \mathcal{O}(\varepsilon) = u_0 + \mathcal{O}(\varepsilon).$$

Thus it is proved that $|u_0 - \hat{u}_0| \leq \mathcal{O}(\varepsilon)$. Note that the choice of θ^* is independent of the function u and point (x, t), thus $\|\mathcal{M} - \mathcal{A}\|_\infty \leq \mathcal{O}(\varepsilon)$. □

2.2 Convergence analysis

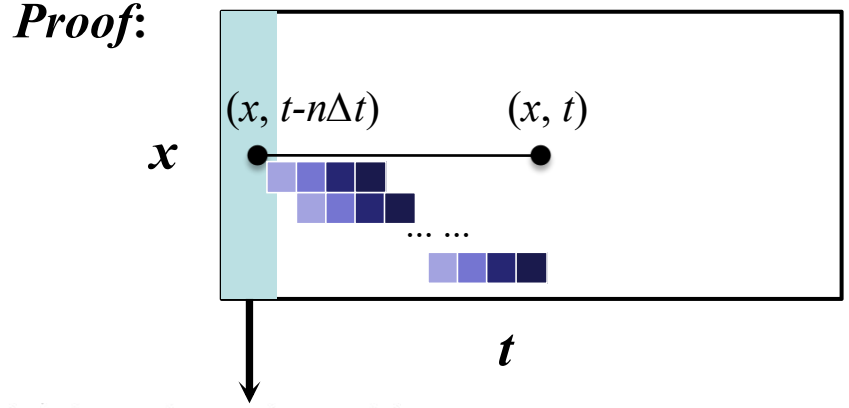
Convergence theorem:

According to *Assumption 1* and *Lemma 1*, we can prove the following convergence theorem for AttnPINNs.

Theorem 1 *Assume the domain Ω is bounded and let $u_{\text{Attn}}(\boldsymbol{x}, t; \theta)$ be the output of AttnPINNs for Eq. (1) with context length N. Suppose that Assumption 1 is satisfied, then for given Δt, and given tolerances $\varepsilon, \varepsilon_p > 0$, there exist $\theta^* \in \mathbb{R}^{D_\theta}$ and $M > 0$ (the number of self-attention blocks), such that*

$$\|u_{\text{Attn}}(\boldsymbol{x}, t; \theta^*) - u(\boldsymbol{x}, t)\|^2_{L^2(\Omega)} \leq \mathcal{O}(\Delta t^N) + \mathcal{O}(\varepsilon) + \mathcal{O}(\varepsilon_p). \tag{21}$$

***Proof*:**



For any $(x, t) \in \Omega$, owning to the *Assumption 1*, we know that for any $\varepsilon_p > 0$, $\exists P \geq 0$, such that when $n \geq P$,

$$|u_p(\boldsymbol{x}, t - n\Delta t) - u(\boldsymbol{x}, t - n\Delta t)| < \varepsilon_p.$$

We denote $u_n^{(0)} = u_p(x, t - n\Delta t)$ and $u_n = u(x, t - n\Delta t)$. Then

$$u_p\{\mathcal{G}_1[(\boldsymbol{x}, t - n\Delta t)]\} = \left(u^{(0)}_{n+N}, u^{(0)}_{n+(N-1)}, \ldots, u^{(0)}_{n+1}\right)^\top$$

***Proof*:**

In terms of *Lemma 1*, there exists $\theta^* \in R^{D_\theta}$, such that the output of the last position after a self-attention block with residual connection is

$$u_n^{(1)} = u_{n+1}^{(0)} + \sum_{j=1}^{N} \alpha_{Nj} v_j = u_{n+1}^{(0)} + \Delta t \sum_{j=1}^{N} b_j \mathcal{N}(u_{n+N-j+1}^{(0)}) + \mathcal{O}(\varepsilon),$$

where $u_n^{(i)}$ is the value at $(x, t - n\Delta t)$ after i self-attention blocks with residual connection. Also because $|u_n^{(0)} - u_n| < \varepsilon_p$ for any $n \geq P$. Thus, it yields that

$$|u_{P-1}^{(1)} - u_{P-1}| \leq \mathcal{O}(\Delta t^N) + \mathcal{O}(\varepsilon) + O(\varepsilon_p)$$

Therefore, according to the relation

$$u_n^{(m)} = u_{n+1}^{(m-1)} + \Delta t \sum_{j=1}^{N} b_j \mathcal{N}(u_{n+N-j+1}^{(m-1)}) + \mathcal{O}(\varepsilon), \quad m \in \mathbb{N}^+,$$

we can determine P, such that after P self-attention blocks with residual connection, it is true that

$$|u_n^{(P)} - u_n| \leq \mathcal{O}(P\Delta t^N) + \mathcal{O}(P\varepsilon) + \mathcal{O}(\varepsilon_p),$$

when $n \geq 0$.

***Proof*:**

Therefore, we have $|u_0^{(P)} - u_0| \leq \mathcal{O}(P\Delta t^N) + \mathcal{O}(P\varepsilon) + \mathcal{O}(\varepsilon_p)$.

That means that for any $(x, t) \in \Omega$, it is true that

$$|u_{\text{Attn}}(\boldsymbol{x}, t; \theta^*) - u(\boldsymbol{x}, t)| \leq \mathcal{O}(P\Delta t^N) + \mathcal{O}(P\varepsilon) + \mathcal{O}(\varepsilon_p).$$

Furthermore, for any $(x, t) \in \Omega$, an upper bound on P can be found, and we denote it as $\boldsymbol{M}$. Therefore, we have

$$\|u_{\text{Attn}}(\boldsymbol{x}, t; \theta^*) - u(\boldsymbol{x}, t)\|^2_{L^2(\Omega)} = \int_\Omega |u_{\text{Attn}}(\boldsymbol{x}, t; \theta^*) - u(\boldsymbol{x}, t)|^2 dt d\boldsymbol{x} \leq \mathcal{O}(\Delta t^N) + \mathcal{O}(\varepsilon) + \mathcal{O}(\varepsilon_p).$$

This completes the proof of the Theorem.

3.1 Main results

We verify the effectiveness of the AttnPINNs method by experimenting with a wide range of PDEs, especially in the **multi-scale, high-frequency, and chaotic problems,** including the chaotic Lorenz system, the convection equation, and the 1D-reaction equation (referred to as "PINNs failure modes" [66]), the Allen–Cahn equation, the Burgers equation, the rogue wave dynamics of NLS equation (whose solutions exhibit "sharper" transitions), and the Navier-Stokes equations in the turbulent regime.

Moreover, we demonstrate the advantages of our AttnPINNs method in terms of accuracy and efficiency compared to the vanilla PINNs and other advanced DNN architectures, including QRes, First-Layer Sine (FLS), and PINNsFormer.

Table 1: The hyperparameters of AttnPINNs method for different physical models.

Models	N	Δt	Pre-training PINNs		D_m	$\mathcal{F}_k, \mathcal{F}_q, \mathcal{F}_v$		Head	M	N_r	N_I	N_b
			layers	size		layers	size					
Lorenz	10	1e-3	6	512	3	2	20	3	2	150	/	/
Convection	8	1e-4	3	256	120	3	256	6	6	5000	60	160
1D-Reaction	10	1e-5	2	64	120	3	64	6	6	5000	60	160
Allen–Cahn	5	1e-4	4	128	32	3	256	8	6	2000	800	800
Burgers	5	1e-4	4	20	64	3	64	1	6	5000	100	200
NLS	3	1e-5	4	64	32	3	64	1	3	10000	400	800
Navier-Stokes	3	1e-3	3	256	32	3	256	4	2	30000	2000	2000

[1] *PINNs, J. Comput. Phys., 378 (2019), pp. 686-707.* [2] *QRes, in Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), pp. 675–683. SIAM, 2021.* [3] *FLS, IEEE Trans. Artif. Intell., 5 (2024), pp. 985-1000.* [4] *PINNsFormer, arXiv preprint arXiv:2307.11833, 2023.*

3.1 Main results

As shown in Table, most of the examples fail with the vanilla PINNs. The order of magnitude of the loss shows that they mostly fall into a local minimum. In contrast, the AttnPINNs can overcome this challenge to some extent. Specifically, compared to the best results between the vanilla PINNs, QRes, FLS, and PINNsFormer, the AttnPINNs can achieve up to **90%** improvement in some examples. In some examples where some advanced methods perform well, our AttnPINNs method is also able to further reduce the error. On the other hand, according to the loss and MaxIN, the AttnPINNs can achieve a smaller loss with a shorter number of iteration steps.

Model		PINNs	QRes	FLS	PINNsFormer	AttnPINNs	Promotion
Lorenz	MaxIN	4050	500	2000	3400	9700	/
	Loss	1.20e-3	1576	**5.46e-4**	1.82e-1	6.09e-2	/
	rMAE	8.56e-1	7.01e-1	6.71e-1	**6.04e-2**	**4.84e-3**	91.99%
	rMSE	9.13e-1	7.64e-1	6.13e-1	**5.87e-2**	**4.89e-3**	91.67%
Convection	MaxIN	8000	11000	11000	11000	4700	/
	Loss	2.13e-2	1.36e-2	1.28e-2	**1.76e-4**	**6.2e-5**	/
	rMAE	8.28e-1	7.25e-1	7.11e-1	**2.10e-2**	**1.46e-2**	30.48%
	rMSE	8.80e-1	7.98e-1	7.84e-1	**2.48e-2**	**1.77e-2**	28.63%
1D-Reaction	MaxIN	600	600	600	5100	2000	/
	Loss	1.99e-1	1.99e-1	1.99e-1	**6.1e-5**	**3.0e-6**	/
	rMAE	9.71e-1	9.83e-1	9.79e-1	**2.79e-2**	**9.98e-3**	64.23%
	rMSE	9.70e-1	9.83e-1	9.78e-1	**4.17e-2**	**1.95e-2**	53.24%

3.1 Main results

Model		PINNs	QRes	FLS	PINNsFormer	AttnPINNs	Promotion
Allen–Cahn	MaxIN	2700	3300	2700	2500	600	/
	Loss	1.27e-2	8.87e-3	6.55e-3	**6.22e-3**	**3.40e-5**	/
	rMAE	4.51e-1	3.75e-1	3.51e-1	**3.35e-1**	**1.59e-2**	95.25%
	rMSE	7.32e-1	6.06e-1	5.67e-1	**5.39e-1**	**4.37e-2**	91.89%
Burgers	MaxIN	12000	10000	12000	9200	750	/
	Loss	**2.6e-5**	3.9e-5	2.1e-5	3.8e-5	**1.2e-5**	/
	rMAE	**1.99e-3**	5.90e-3	2.91e-3	9.55e-3	**1.65e-3**	17.09%
	rMSE	**1.15e-2**	4.92e-2	1.86e-2	7.25e-2	**6.55e-3**	43.04%
NLS	MaxIN	23000	14000	18000	14500	14000	/
	Loss	**3.2e-5**	2.52e-2	5.7e-5	3.05e-2	4.3e-5	/
	rMAE	**2.33e-3**	2.17e-1	2.42e-3	2.83e-1	**1.96e-3**	15.88%
	rMSE	**3.27e-3**	3.22e-1	3.36e-3	3.79e-1	**2.64e-3**	19.27%
Navier-Stokes	MaxIN	10000	10000	10000	60000	58000	/
	Loss	66.47	69.48	66.47	**1.3**	**8.84e-1**	/
	rMAE	9.99e-1	9.99e-1	9.99e-1	**4.31e-1**	**3.16e-2**	92.67%
	rMSE	9.99e-1	9.99e-1	9.99e-1	**5.29e-1**	**4.41e-2**	91.66%

3.2 Numerical experiments

Example 1: The Lorenz system.

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(\rho - z) - y, \\ \dot{z} = xy - \beta x, \end{cases}$$

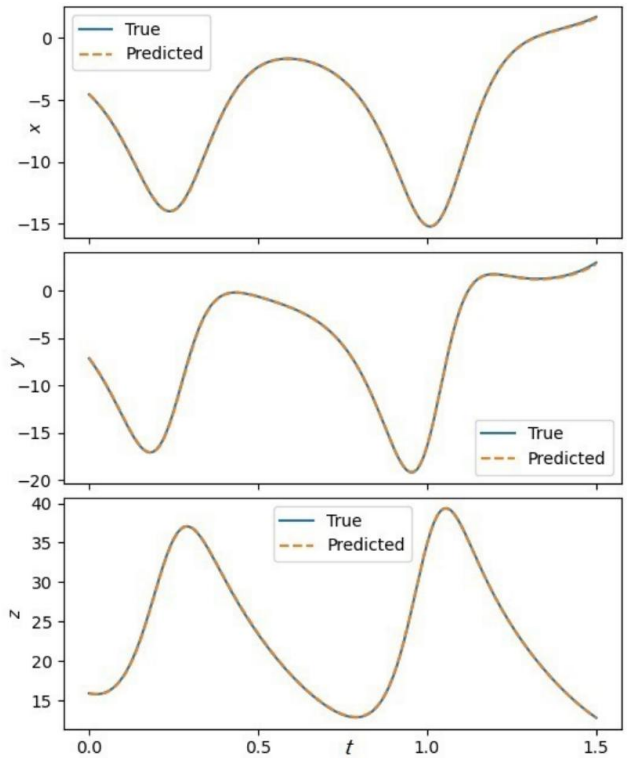
The dynamics of the Lorenz system demonstrates the chaotic behavior for parameters $\sigma = 10$, $\rho = 28$, $\beta = 8/3$. Here we take the initial data $\vec{u}(0) = [x(0), y(0), z(0)]$ = [-4.55, -7.13, 15.92]. We aim to construct a neural network $\vec{u}_{Attn} = [x_{Attn}, y_{Attn}, z_{Attn}]$ to approximate the solution of system. Since the Lorenz system exhibits the strong sensitivity to the initial conditions, we consider exactly imposing initial conditions as follows by changing the final output of the network

$$\hat{\vec{u}}_{\text{Attn}} = t \cdot \vec{u}_{\text{Attn}} + \vec{u}(0).$$

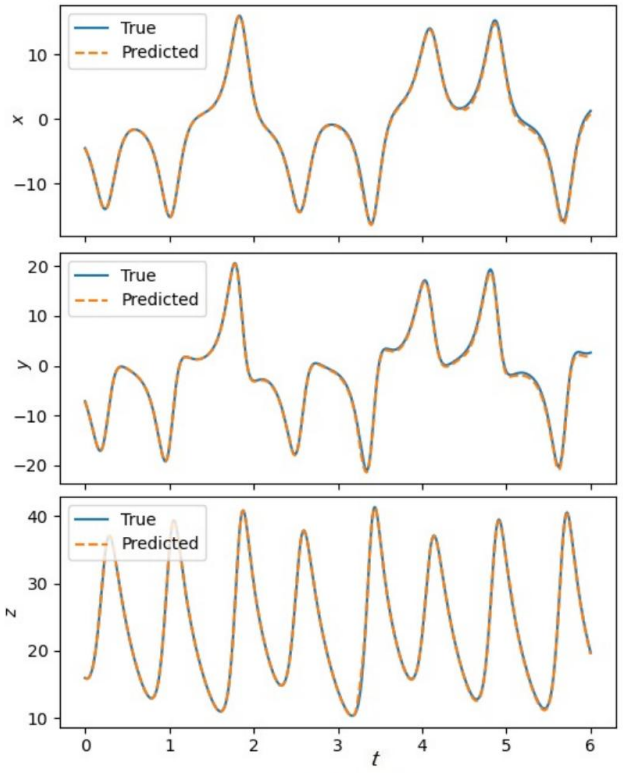
Besides, given that the loss function of the Lorenz system drops from a large values around 1000, we first use 10000 steps of Adam optimization to speed up the training.

3.2 Numerical experiments

In order to predict the behavior of the solution for a long time, we split up the temporal domain $[0, T]$ into many sub-domains $[0, \Delta T]$, $[\Delta T, 2\Delta T]$, . . . , $[T - \Delta T, T]$, and train networks to learn the solution in each sub-domain. Especially, we use the output of the j-th subnetwork at $t = j\Delta T$ as the initial condition for the $(j + 1)$-th subnetwork. Therefore, the predicted value of the former network will greatly influence the later results.



(a) AttnPINNs



(d) AttnPINNs (long)

3.2 Numerical experiments

Example 2: The convection equation.

$$\begin{cases} u_t + \beta u_x = 0, \quad x \in [0, 2\pi], \ t \in [0, 1], \\ u(x, 0) = \sin x, \quad x \in [0, 2\pi], \\ u(0, t) = u(2\pi, t), \quad t \in [0, 1], \end{cases}$$

where β is the convection coefficient. The equation has a simple analytical solution for periodic boundary condition as follows

$$u(x,t) = \mathcal{F}^{-1}[\mathcal{F}[\sin(x)]e^{-i\beta kt}],$$

where F and F^{-1} are the direct and inverse Fourier transforms, respectively, and k represents the frequency in the Fourier domain. The PINNs is only able to achieve the good solutions for small values of convection coefficient, however it fails when β becomes larger, reaching a relative error of almost 100% for $\beta > 10$. Here we set the larger convection coefficient $\beta = 50$.

3.2 Numerical experiments

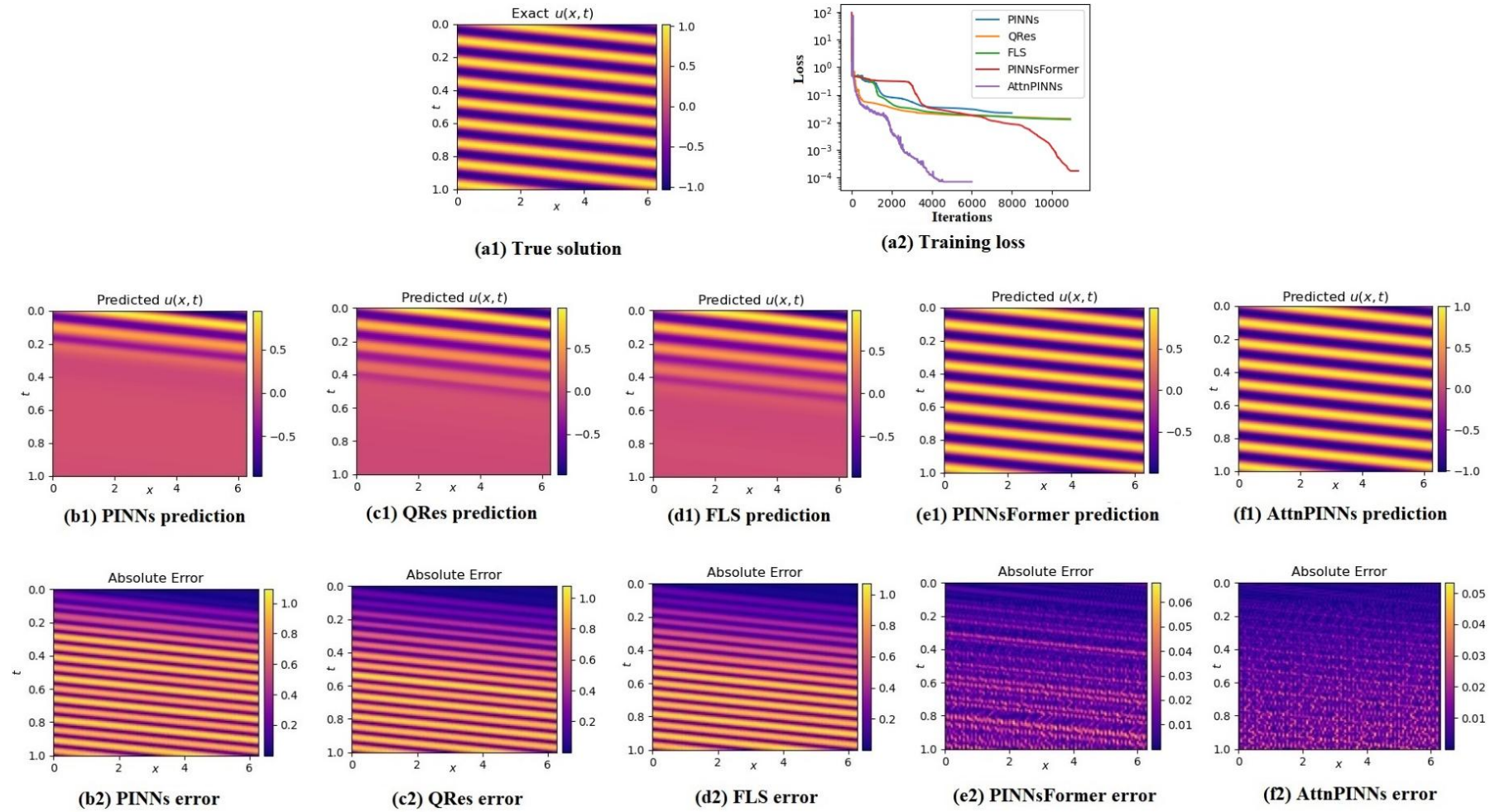


Figure: Convection equation: Exact, predicted solutions, training loss, and absolute errors for PINNs, QRes, FLS, PINNsFormer, and AttnPINNs.

4.1 Algorithm analysis

4.1.1 Effect of hyperparameters *N* and *M*.

First, we discuss the impact of two important parameters, sequence length N and the number of self attention blocks M on the performance of the algorithm. According to *Theorem 1*, we know that the sequence length N, which is the context window of self-attention block, has a great impact on the convergence error. In turn, the number of required self attention blocks is affected by N.

Intuitively, larger values of N and M tend to result in smaller errors. However, experiments show that this is not always the case. We show the effect of the hyperparameters N and M on the rMAE and rMSE by solving convection equation as an example. In general, the larger the value of N and M, the smaller the relative error is. And the smaller N tends to require more self-attention blocks for the algorithm to converge. However, more self-attention blocks are not always better. A larger M means a deeper network, which leads to optimization difficulties, and thus to getting stuck in local minima.

4.1 Algorithm analysis

For example, the rMAE and rMSE are 7.91e-2 and 1.22e-1 for $N = 7$ and $M = 8$, while for $N = 7$ and $M = 7$, the relative errors are smaller, 2.46e-2 and 2.86e-2, respectively. Furthermore, for fixed $M = 6$, we gradually increase the value of N from 1 to 8, and find that the error drops to the order of 1e-2 when $N \geq 4$, but is not monotonic with N. Therefore, it is necessary to find the appropriate N and M.

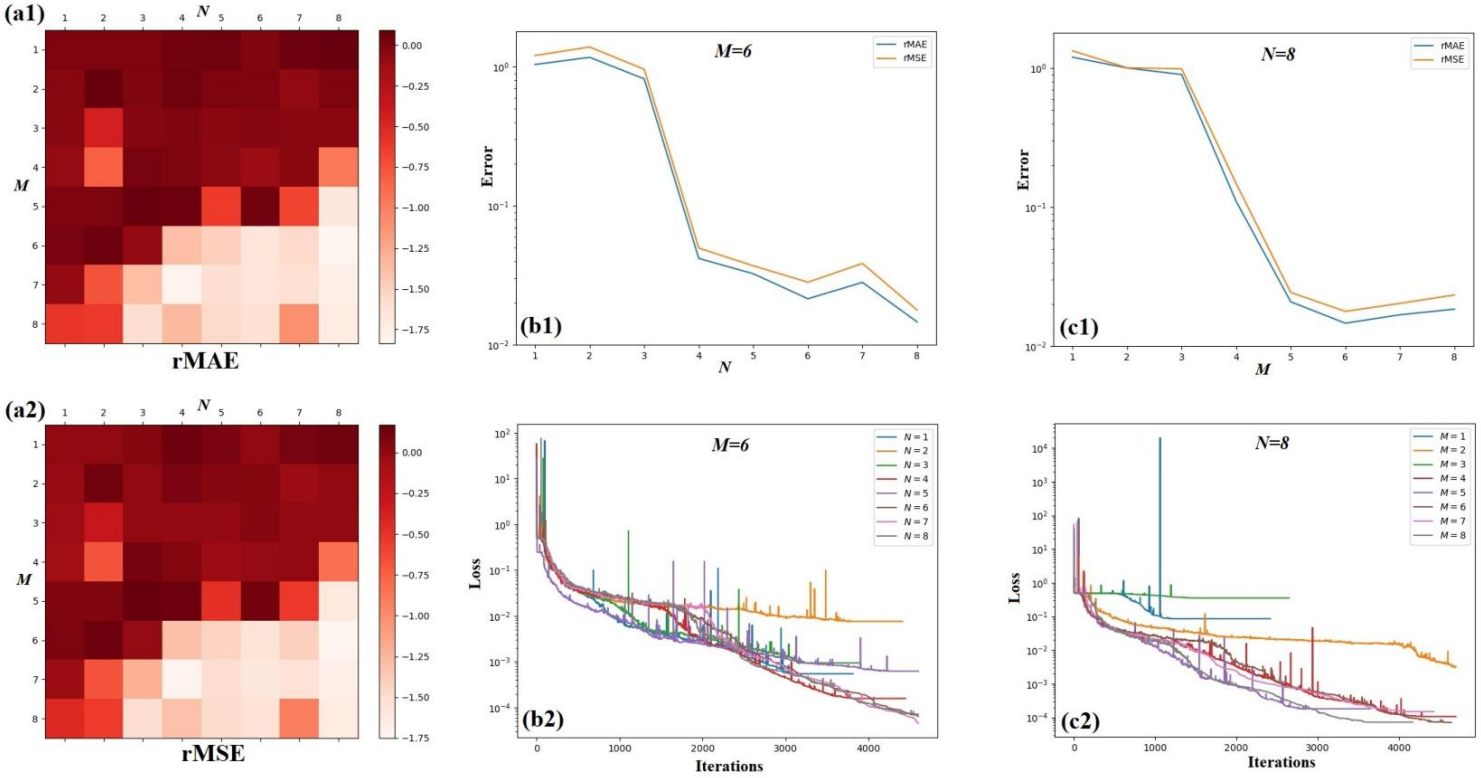


Figure: (a1, a2) The logarithm of rMAE and rMSE for different N and M. (b1, b2) The relative error vs. N and loss plots for fixed $M = 6$. (c1, c2) The relative error vs. M and loss plots for fixed $N = 8$.

4.1 Algorithm analysis

4.1.2 Training cost.

An additional factor to take into account when considering AttnPINNs is its computational and memory cost during training. The memory overhead of AttnPINNs mainly lies in the sequence generation and the self-attention block. Compared to the traditional methods, the number of training points of AttnPINNs is $N(N_r + N_I + N_b)$. And the self-attention block contains three neural networks F_k, F_q and F_v. Fortunately, the M self-attention blocks that are identical.

To show clarity, the hyperparameters and training cost for different methods are exhibited in Table. And the comparison relies on solving the Allen–Cahn equation. It can be seen that our approach requires fewer parameters to train than PINNsFormer. And compared to the PINNs and FLS, the number of training parameters is about 2.14x, which is acceptable. In addition, for the PINNsFormer, the key, query and value vectors are the input of the previous step. In fact, we can also fix F_k and F_q to be the identity map to further reduce the cost in practice.

Method	Hyperparameters	Value	Training parameters	Training points	Training time (sec/epoch)	MaxIN
PINNs/FLS	hidden layers	4	181k	3600	0.024	2700
	hidden size	300				
QRes	hidden layers	4	543k	3600	0.051	3300
	hidden size	300				
PINNsFormer	N	5	395k	18000	0.143	2500
	Δt	1e-4				
	encoder layers	1				
	decoder layers	1				
	embedding size	32				
	head	8				
	hidden size	512				
AttnPINNs	N	5	388k	18000	0.556	600
	Δt	1e-4				
	pre-training PINNs	128×4				
	D_m	32				
	$\mathcal{F}_k, \mathcal{F}_q, \mathcal{F}_v$	256×3				
	head	8				
	M	6				

Table: The hyperparameters and training cost for different methods. And the comparison relies on solving the Allen–Cahn equation.

Time cost

And for time cost, the AttnPINNs method is more expensive than other methods. For example, we observe an approximate 3.88x rise in time cost to train an epoch than the PINNsFormer. However, considering that the number of training steps for the AttnPINNs is much less than other methods, the time cost is acceptable or even lower. For example, Fig. displays the plot of training loss vs. time for solving the Allen–Cahn equation. It can be seen that for the same training time, the loss for the AttnPINNs is smaller. Furthermore, the time overhead mainly lies in optimizing the deep self-attention layer. Based on the results in the previous subsection, we could reduce the number of self-attention blocks M to speed up the training while achieving similar performance.

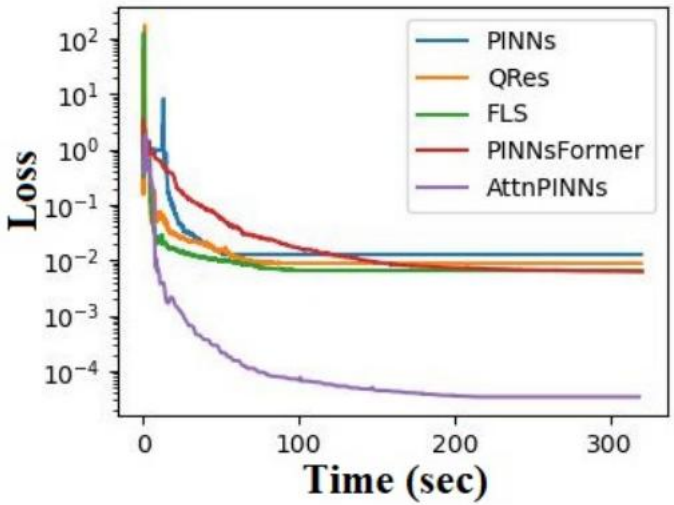


Figure: The profile of training loss vs. time (sec) for solving the Allen–Cahn equation.

4.2 Summary

- We propose the advanced network framework, referred to as **AttnPINNs**, to capture temporal dependencies between solution sequences, facilitating the generation of high-precision global solutions for PDEs.
- Self-attention provides a significantly great capacity for learning the time-integration technique. We prove that a self-attention block with a residual connection can **approximate an explicit time-integrator** within arbitrarily small error bounds. With the aid of it, we provide a rigorous proof of the convergence of our AttnPINNs algorithm.
- We demonstrate the AttnPINNs by a wide range of PDEs, whose solutions tend to have **abrupt changes or exhibit the multi-scale, high-frequency, and chaotic behaviors**. And the numerical results exhibit that the AttnPINNs can effectively capture the solution structures, and outperforms most of the other strategies.

4.3 Future work

- Incorporate variant learning schemes, such as advanced adaptive weights and optimization strategies.

- redesigning an attention mechanism algorithm suitable for high-dimensional scenarios.

Thank you !

songjin21@mails.ucas.ac.cn