A Mathematical Explanation of Transformers for Large Language Models and GPTs *

Xue-Cheng Tai†, Hao Liu‡, Lingfeng Li§, and Raymond H. Chan¶

Abstract. The Transformer architecture has revolutionized the field of sequence modeling and underpins the recent breakthroughs in large language models (LLMs). However, a comprehensive mathematical theory that explains its structure and operations remains elusive. In this work, we propose a novel continuous framework that rigorously interprets the Transformer as a discretization of a structured integro-differential equation. Within this formulation, the self-attention mechanism emerges naturally as a non-local integral operator, and layer normalization is characterized as a projection to a time-dependent constraint. This operator-theoretic and variational perspective offers a unified and interpretable foundation for understanding the architecture's core components, including attention, feedforward layers, and normalization. Our approach extends beyond previous theoretical analyses by embedding the entire Transformer operation in continuous domains for both token indices and feature dimensions. This leads to a principled and flexible framework that not only deepens theoretical insight but also offers new directions for architecture design, analysis, and control-based interpretations. This new interpretation provides a step toward bridging the gap between deep learning architectures and continuous mathematical modeling, and contributes a foundational perspective to the ongoing development of interpretable and theoretically grounded neural network models.

Key words. Transformer, attention, operator splitting, integro-differential equation

MSC codes. 68U10, 94A08

1. Introduction.

Deep neural networks (DNNs) have revolutionized numerous fields, including natural language processing [15, 55], computer vision [23], scientific computing [31, 36, 26], and medical diagnostics [35, 22]. Among these architectures, Transformers [46] have recently emerged as particularly powerful tools, underpinning remarkable successes in large language models (LLMs) such as GPT-3 and GPT-4 [46]. Besides language processing [59], the Transformer and its variants have also been applied in applications, including image processing [8, 41], graph processing [56], operator learning [6, 3, 28, 54].

Recently, a series of works have been conducted to study the theoretical foundation or interpretability of Transformers. Approximation and generalization error of Transformers were studied in [10, 44, 19]. It was shown in [19, 40] that Transformers are adaptive to data's low-dimensional structures. For interpretability, in [24], it was shown that Transformers are cubic or high-order splines. It was shown in [21] that the vision Transformer can learn spatial structures of the dataset. The Transformer was interpreted as an ODE solver for multi-particle dynamical systems in [9, 11, 33]. Meng et al. [**?**] showed that Transformer is a step for solving

*Submitted to the editors DATE.

Funding:

†Norwegian Research Centre, Bergen, Norway (xtai@norceresearch.no, xuechengtai@gmail.com).

‡Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (haoliu@hkbu.edu.hk).

§Hong Kong Centre for Cerebro-Cardiovascular Health Engineering, Shatin, Hong Kong (lfli@hkcoche.org.)

¶School of Data Science and Department of Operations and Risk Management, Lingnan University, Tuen Mun, Hong Kong (raymond.chan@ln.edu.hk).

This manuscript is for review purposes only.

a nonlocal variational model. A general introduction to Transformers with mathematical descriptions can be found in [45].

Compared to Transformers, more studies can be found for deep neural networks (DNNs). A widely accepted interpretation views DNN as nonlinear high-dimensional function approximators [53, 30, 58, 4]. Under this view, the layers of a DNN perform successive nonlinear transformations that approximate complex mappings from input to output data. This approximation framework has been instrumental in explaining the expressive power of DNNs. Another promising direction for explaining and designing neural network architectures involves interpreting them as discretizations of continuous-time dynamical systems governed by differential or integral equations [5, 49, 2, 48]. E et al. show that DNNs can be viewed as continuous dynamical systems in [48] and special discretization of continuous problems in [49]. In [5], Chen et al. pointed out connections between residual networks and ordinary differential equations, and proposed Neural ODE. Benning et al. investigated the relations between DNNs and optimal control problems [2]. In this class of approach, neural network training is seen as an optimal control problem constrained by a continuous evolution equation. Related to the above-mentioned approach, we especially want to mention some recent studies that have proposed a general framework for designing neural network architectures inspired by continuous-time dynamical systems [42, 27]. These methods leverage operator-splitting techniques to construct neural networks, which are so-called continuous neural networks, as discretizations of continuous differential equations. The continuous learning problem is an optimal control problem with this "neural network" as a constraint. In this framework, the evolution of hidden states in a network is governed by differential equations (continuous neural network). The goal of this continuous learning problem is to find optimal controls—represented by learnable parameters—within a discretized dynamical process.

The use of operator-splitting allows these complex dynamics to be decomposed into simpler substeps, each of which corresponds to a specific type of operation (e.g., convolution, nonlinearity, normalization). These substeps are then unrolled into individual layers of a neural network. This interpretation not only clarifies the structure and purpose of different architectural components but also facilitates the incorporation of prior knowledge about the underlying data dynamics or physics.

Building on this theory, we have recently provided a rigorous mathematical explanation of the well-known UNet architecture [43]. We demonstrated that UNet can be viewed as a specific discretization of the following simple differential equation, where each encoder and decoder stage corresponds to a particular substep in the operator-splitting scheme:

$$\begin{cases} \frac{\partial u(\mathbf{x},t)}{\partial t} = W(\mathbf{x},t) * u(\mathbf{x},t) + d(t) - \ln \frac{u(\mathbf{x},t)}{1-u(\mathbf{x},t)} + \partial I_\Sigma(u), \ (\mathbf{x},t) \in \Omega \times (0,T], \\ u(\mathbf{x},0) = f(\mathbf{x}), \ \mathbf{x} \in \Omega, \end{cases} \tag{1.1}$$

where $*$ denotes convolution, I_Σ is the indicator function of $\Sigma = \{u : u(\mathbf{x}) \geq 0 \text{ for } \mathbf{x} \in \Omega\}$, f is a function defined in a domain Ω and T is the final time chosen by the user for time variable t. The learnable variables $\theta = \{W(\mathbf{x},t), d(t)\}$ are regarded as control variables which will be learned in the training process. The mapping $\mathcal{N}_\theta(f) \mapsto u(\mathbf{x},T)$ turns out to be the UNet in the continuous setting. This perspective not only elucidates the structure of the UNet architecture but also provides a foundation for systematic modifications and enhancements.

A range of alternative interpretations and mathematically inspired designs of deep neural networks can be found in the literature, including [39, 2, 38, 17, 20, 25, 29, 16, 52, ?]. For example, neural networks have been interpreted as optimal control problems in [2, 38]. Gregor and LeCun [16] were among the first to connect deep learning with sparse coding through learned iterative shrinkage algorithms. Ruthotto and Haber [39] proposed neural networks derived from partial differential equations (PDEs), while He and Xu [20] introduced MgNet, which employs multigrid principles for feature extraction. Hagemann et al. [18] formulated stochastic normalizing flows as Markov chains to tackle inverse problems. Long et al. [29] built PDENet based on finite difference schemes to recover PDE models from data. Yang et al. [52] constructed ADMM-CSNet by unrolling the ADMM optimization algorithm into a network architecture. Wang et al. [47] developed a diffusion-residual network inspired by convection-diffusion ODEs for classification tasks. Most recently, Martin et al. [34] introduced a plug-and-play architecture using flow matching for image restoration. Liu et al. [?] proposed a novel network regularizer inspired by the reverse process of PDE-based evolution models. Networks regularized by prior information are proposed in [?, ?, ?]. These pioneering efforts have provided essential insights and have directly inspired our recent work [42, 43] and the present study.

In this work, we will develop a mathematical theoretical framework to explain the Transformer architecture of [46], which has achieved remarkable success in large language models (LLMs) and other sequence modeling tasks. We show that the Transformer can be interpreted as a discretization of the following continuous integro-differential equation:

$$
\begin{cases}
u_t = \underbrace{\Big\langle \gamma(\mathbf{x},\cdot,t;u), V(\cdot,\mathbf{y},t;u) \Big\rangle_{\Omega_x}}_{\text{I: attention}} + \underbrace{\partial I_{S_1(\sigma_1(t),\sigma_2(t))}(u)}_{\text{II: layer normalization}} \\
\qquad\qquad + \underbrace{\sum_{j=1}^{J} \left(\Big\langle W_j(\cdot,\mathbf{y},t) u(\mathbf{x},\cdot,t) \Big\rangle_{\Omega_y} + b_j(\mathbf{x},t) \right) + \partial I_{S_2}(u)}_{\text{III: fully connected network}} \quad \text{for } t \in (0,T], \\
u(\mathbf{x},\mathbf{y},0) = f(\mathbf{x},\mathbf{y}),
\end{cases}
\tag{1.2}
$$

for $(\mathbf{x},\mathbf{y}) \in \Omega_x \times \Omega_y$, where $\mathbf{x}$ denotes the token index, $\mathbf{y}$ the entry of token vectors, Ω_x and Ω_y are continuous domains, and $T > 0$ is a fixed terminal time. This formulation provides a rigorous mathematical interpretation of the Transformer and its key components, such as self-attention and feedforward layers. The definitions of attention, layer normalization, and the fully connected layer, along with detailed explanations of the domains Ω_x, Ω_y, and the control variables, are given in Section 2. Equation (1.2) highlights the non-local nature of attention and reveals the Transformer as a structured operator derived from a discretized variational principle. We note that several previous works have also established connections between Transformer-based models and multi-particle dynamical systems [9, 11, 33]. These studies interpret the original or modified Transformer models as discretizations of interacting particle systems in a time-continuous setting, treating interactions across spatial variables as particle interactions. In contrast, our formulation (1.2) is fundamentally different: we show

that the Transformer arises as a discretization of a continuous integro-differential equation, offering a more unified operator-theoretic and variational perspective.

This perspective offers several advantages. First, it provides a unifying mathematical framework that connects diverse architectures, such as CNNs, UNets, and Transformers, under the common lens of differential and integral equations. This unification improves our understanding of the design principles underlying modern deep networks and provides insights for cross-architectural studies.

Second, by casting neural network architectures as time-stepping schemes of dynamical systems, our framework allows the systematic exploration of new architectures using well-established tools from numerical analysis. For instance, stability, convergence, and approximation properties of the underlying continuous models can inform the selection of network structures and hyperparameters, leading to more robust and interpretable models.

Third, this approach creates a principled pathway for embedding domain-specific knowledge, such as physical laws, geometric structures, or conservation principles, directly into the design of neural architectures, leading to architectures tailored for specific scientific or engineering tasks.

In summary, our formulation not only advances the theoretical understanding of deep neural networks but also provides actionable tools for the principled design of next-generation models. It bridges the gap between continuous mathematical modeling and discrete network implementation, establishing a foundation for informed, explainable, and application-aware neural network development.

This paper is organized as follows: We introduce the integral equation for Transformer with single-head attention in continuous setting in Section 2. An operator-splitting based algorithm is proposed in Section 3 to solve the integral equation. We show in Section 4 that the discretized algorithm exactly recovers the Transformer encoder in [46], and also discuss how to adapt the proposed algorithm to recover the architecture of ViT [8]. In Section 5, we extend the integral equation and proposed algorithm to Transformer with multi-head attention. Extension to convolutional Transformer and its relation to CvT [50] are discussed in Section 6. This paper is concluded in Section 7.

In the rest of this paper, we use bold lowercase letters to denote vectors, normal letters to denote scalars and functions, and calligraphic letters to denote operators.

2. The Continuous Model.

In the following, we explain the Transformer in the seminal work in [46] in the continuous setting and show later that proper discretizations of the proposed model will recover the Transformer exactly.

2.1. The continuous setting for Transformers.

In this section, we propose a control problem in a continuous setting. The continuous Transformer is a time-dependent integral equation. Later, we will show that after discretizations by operator splitting for the time variable t and spatial discretizations for the $\mathbf{x}, \mathbf{y}$ variables, the resulting splitting method recovers the Transformer architecture. The Transformer proposed in [46] consists of three components: attention, layer normalization, and feedforward networks. In the derivation of our control problem, we show that each component has a corresponding operation in the integral equation. We first focus on single-head attention. Our framework can be easily extended to the multi-head case, which will be detailed in Section 5.

To make the explanation clearer and simpler, our notations and setup will follow [46], but in a continuous counterpart. Let $u(\mathbf{x}, \mathbf{y}, t)$ be a function of $\mathbf{x}, \mathbf{y}$ and time t. In applications of LLMs, $\mathbf{x} \in \Omega_x$ refers to the index of tokens, $\mathbf{y} \in \Omega_y$ refers to entries of token vectors. Please refer to Appendix A for explanations of the physical meanings of $\mathbf{x}, \mathbf{y}$, and u for applications of Transformers to LLM.

As stated in [46], attention is all you need for the Transformer. In fact, we show below that the attention layer has a very simple mathematical explanation in the continuous setting. It consists of integral transformations on the input functions to extract features with three different kernels. We use features extracted from the first two kernels to generate the attention score through a softmax operator, and then multiply the attention score by the feature extracted from the third kernel to compute the output. All three kernels in the integral transformation will be parameterized as learnable variables, see (2.11).

2.2. Integral Transformations for Attention and Feature Extraction. Let the functions $W^Q(\mathbf{y}, \widetilde{\mathbf{y}}, t)$, $W^K(\mathbf{y}, \widetilde{\mathbf{y}}, t)$, and $W^V(\mathbf{y}, \widetilde{\mathbf{y}}, t)$ be three kernels defined on $\Omega_y \times \Omega_y \times [0, T]$, and they will be learned from data as shown in LLM applications. See Appendix A for explanations of the physical meanings of $\mathbf{x}, \mathbf{y}, u(\mathbf{x}, \mathbf{y}, t)$. For a given function $u(\mathbf{x}, \mathbf{y}, t)$, we define integral transformations:

$$Q(\mathbf{x}, \mathbf{y}, t; u) = \Big\langle W^Q(\cdot, \mathbf{y}, t), u(\mathbf{x}, \cdot, t) \Big\rangle_{\Omega_y} = \int_{\Omega_y} W^Q(\xi, \mathbf{y}, t) u(\mathbf{x}, \xi, t) d\xi, \tag{2.1}$$

$$K(\mathbf{x}, \mathbf{y}, t; u) = \Big\langle W^K(\cdot, \mathbf{y}, t), u(\mathbf{x}, \cdot, t) \Big\rangle_{\Omega_y} = \int_{\Omega_y} W^K(\xi, \mathbf{y}, t) u(\mathbf{x}, \xi, t) d\xi, \tag{2.2}$$

$$V(\mathbf{x}, \mathbf{y}, t; u) = \Big\langle W^V(\cdot, \mathbf{y}, t), u(\mathbf{x}, \cdot, t) \Big\rangle_{\Omega_y} = \int_{\Omega_y} W^V(\xi, \mathbf{y}, t) u(\mathbf{x}, \xi, t) d\xi. \tag{2.3}$$

Here and later, we use $\langle \cdot, \cdot \rangle_{\Omega_y}$ to denote L^2 inner product on a given domain Ω_y. Its induced norm will be denoted by $\| \cdot \|_{\Omega_y}$. Functions Q and K will be used to generate attention scores, and V is used to extract features from u. When all these functions are discretized over a pixel-type grid over $\Omega_x \times \Omega_y$, the discrete counterparts of these functions are represented by matrices and the discrete counterparts of the integral transformations are just standard matrix multiplications, see Section 3.4, especially formula (3.20). We compute the attention score by

$$\gamma(\mathbf{x}, \widetilde{\mathbf{x}}, t; u) = \mathrm{Softmax}_2 \left(\frac{1}{\sqrt{|\Omega_y|}} \Big\langle Q(\mathbf{x}, \cdot, t; u),\ K(\widetilde{\mathbf{x}}, \cdot, t; u) \Big\rangle_{\Omega_y} \right), \tag{2.4}$$

where $\gamma(\mathbf{x}, \widetilde{\mathbf{x}}, t)$ is defined on $\Omega_x \times \Omega_x \times [0, T]$, $\mathrm{Softmax}_2$ represents the softmax function along the second dimension:

$$\mathrm{Softmax}_2(a(\mathbf{x}, \widetilde{\mathbf{x}}, t)) = \frac{\exp(a(\mathbf{x}, \widetilde{\mathbf{x}}, t))}{\int_{\Omega_x} \exp(a(\mathbf{x}, \eta, t)) d\eta}.$$

The output of the attention layer is then obtained by taking the inner product of the attention score γ and the feature V: $\Big\langle \gamma(\mathbf{x}, \cdot, t; u), V(\cdot, \mathbf{y}, t; u) \Big\rangle_{\Omega_x}$. We emphasize that the first two integral transformations in (2.1)–(2.2) are used to generate the attention score $\gamma(\mathbf{x}, \widetilde{\mathbf{x}}, t; u)$ and the last integral transformation (2.3) will be used for feature extractions. All three integral kernels will be learned during training, see explanations later in this section.

2.3. Mathematical formulations for the Layer Normalization. The layer normalization [1] in the Transformer [46] and also many other neural networks [60, 57, 51, 7] can be mathematically described as a projection of a function to a set with given mean value σ_1 and variance σ_2^2. Define sets

$$S_1(\sigma_1,\sigma_2)=\Big\{u:\frac{1}{|\Omega_y|}\int_{\Omega_y}u(\mathbf{x},\xi,t)d\xi=\sigma_1,\\ \frac{1}{|\Omega_y|}\int_{\Omega_y}(u(\mathbf{x},\xi,t)-\sigma_1)^2d\xi=\sigma_2^2\Big\}, \tag{2.5}$$

$$S_2=\{u:u\geq 0\}, \tag{2.6}$$

and their corresponding indicator functions

$$I_{S_1(\sigma_1,\sigma_2)}(u)=\begin{cases}0 & \text{if } u\in S_1,\\ +\infty & \text{otherwise,}\end{cases}\quad I_{S_2}(u)=\begin{cases}0 & \text{if } u\in S_2,\\ +\infty & \text{otherwise.}\end{cases} \tag{2.7}$$

Later, we shall show that layer normalization is just a projection of a function into the set $S_1(\sigma_1,\sigma_2)$, see Section 3.3.2 and the ReLU activation function is just a projection of a function to the set S_2, see Section 3.3.3.

2.4. The continuous Transformer. In addition to the control variables $W^Q(\mathbf{y},\widetilde{\mathbf{y}},t)$, $W^K(\mathbf{y},\widetilde{\mathbf{y}},t)$, $W^V(\mathbf{y},\widetilde{\mathbf{y}},t)$, let us also introduce control variables $\{W_j(\widetilde{\mathbf{y}},\mathbf{y},t)\}_{j=1}^J$ on $\Omega_y\times\Omega_y\times[0,T]$, and $b(\mathbf{x},t), b_j(\mathbf{x},t)$ on $\Omega_x\times[0,T]$. We call the following continuous integral equation a **Continuous Transformer**:

$$\begin{cases} u_t=\underbrace{\Big\langle\gamma(\mathbf{x},\cdot,t;u),\ V(\cdot,\mathbf{y},t;u)\Big\rangle_{\Omega_x}}_{\text{I: attention}}+\underbrace{\partial I_{S_1(\sigma_1(t),\sigma_2(t))}(u)}_{\text{II: layer normalization}}\\ \qquad\underbrace{+\sum_{j=1}^J\left(\Big\langle W_j(\cdot,\mathbf{y},t),\ u(\mathbf{x},\cdot,t)\Big\rangle_{\Omega_y}+b_j(\mathbf{x},t)\right)+\partial I_{S_2}(u)}_{\text{III: fully connected network}},\quad t\in(0,T],\\ u(\mathbf{x},\mathbf{y},0)=f(\mathbf{x},\mathbf{y}).\end{cases} \tag{2.8}$$

We shall show in the rest of this paper that the Transformer proposed in [46] is just a discretization of this continuous Transformer. In (2.8), f is some initial state, T is a fixed time chosen by the user. Following conventions in the literature, we denote all the control variables as

$$\theta=\left\{W^Q(\mathbf{x},\mathbf{y},t),W^K(\mathbf{x},\mathbf{y},t),W^V(\mathbf{x},\mathbf{y},t),\{W_j(\mathbf{x},\mathbf{y},t),b_j(\mathbf{x},t)\}_{j=1}^J,\sigma_1(t),\sigma_2(t)\right\}. \tag{2.9}$$

The Continuous Transformer is the mapping:

$$\mathcal{N}_\theta: f\mapsto u(\mathbf{x},\mathbf{y},T). \tag{2.10}$$

In Section 3, we will discretize (2.8) by the operator-splitting method for the time variable and use a uniform grid for the spatial variables $\mathbf{x},\mathbf{y}$. After discretization, the continuous Transformer becomes the Transformer proposed in [46]. In relation to this equivalence, the operations in the right-hand side of (2.8) can be classified into three sets:

- Operations in Set I correspond to attentions in Transformers. It computes the attention score $\gamma(\mathbf{x}, \widetilde{\mathbf{x}}, t; u)$ and applies it to the extracted features in $V(\mathbf{x}, \mathbf{y}, t)$.
- Operations in Set II correspond to layer normalization. For each token location $\mathbf{x}$, it requires function u to have mean σ_1 and variance σ_2^2 along the $\mathbf{y}$ direction. In other words, for each token, the corresponding series is expected to have mean σ_1 and variance σ_2^2.
- Operations in Set III correspond to feedforward networks with J layers. The first component corresponds to linear layers, each of which contains one linear transformation W_j and a bias b_j. The second component corresponds to activation functions. This term projects u to S_1, which can be realized by ReLU functions.

2.5. The continuous control problem for learning. Our objective is to optimize θ so that given an input $f(\mathbf{x}, \mathbf{y})$, the control problem will drive $u(\mathbf{x}, \mathbf{y}, T)$ to a desired state v. Specifically, given a dataset $\{(u_i, v_i)\}_{i=1}^B$, where u_i is the input and v_i is the target state, let $\ell(\cdot, \cdot)$ be a loss function measuring the discrepancy between its arguments. Let $\theta, \mathcal{N}_\theta$ be as defined in (2.9)–(2.10), the training process with the loss function ℓ is in fact a process to optimize θ for the following minimization problem:

$$\min_\theta \frac{1}{B} \sum_{i=1}^{B} \ell(\mathcal{N}_\theta(u_i), v_i). \tag{2.11}$$

This is essentially a PDE-constrained optimization problem of the following form:

$$\min_\theta \frac{1}{B} \sum_{i=1}^{B} \ell(w_i, v_i), \text{ under constraint } w_i = \mathcal{N}_\theta(u_i) \text{ solves the control problem (2.8).} \tag{2.12}$$

3. Discretizations of the Continuous Problem. Following the idea of PottsMGNet [42], in this section, we use some proper operator splitting methods to discretize the time variable for (2.8). For a review about operator splitting methods, please refer to [12, 13, 14]. Specifically, we discretize the temporal domain of (2.8) by sequential splitting, c.f. [14, Sec. 2.2]. The number of sequential splittings corresponds to the number of layers in Transformers. Spatial discretization for $\mathbf{x}$ and $\mathbf{y}$ will be discussed in Section 3.4. The discretization of $\mathbf{x}$ determines the number of tokens in the input, and the discretization of $\mathbf{y}$ determines the size of the embedding vector for each token.

3.1. The overall time discretization. Let $\{t^n\}_{n=0}^{N_t}$ be the set of time grids with time step $\Delta t = T/N_t$. We denote the numerical solution of (2.11) at time t^n by u^n. We will design a numerical scheme to approximate $\mathcal{N}_\theta$. Specifically, for each t^n, we design a discrete propagator $\bar{\mathcal{N}}^n$ propagating u^{n-1} to u^n. Then, problem (2.8) is numerically approximated by

$$u(t^{N_t}) \approx u^{N_t} = \bar{\mathcal{N}}^{N_t} \circ \bar{\mathcal{N}}^{N_t - 1} \circ \cdots \circ \bar{\mathcal{N}}^1(f),$$

see Figure 1(b) for an illustration.

In each $\bar{\mathcal{N}}^n$, time-discretized control variables θ will be used. We denote the control variables used in $\bar{\mathcal{N}}^n$ by θ^n. To show the dependence of $\bar{\mathcal{N}}^n$ on the control variable θ^n, we will

also use $\bar{\mathcal{N}}^n_{\theta^n}$ to denote the propagator $\bar{\mathcal{N}}^n$. From the following subsections, c.f. §3.4, we will see that

$$\theta^n = \{W^{Q,n-1}, W^{K,n-1}, W^{V,n-1}, \{W_j^{n-1}, b_j^{n-1}\}_{j=1}^J, \sigma_1^{n-1}, \sigma_2^{n-1}\}.$$

Denote all of the discretized variables by $\bar{\theta}$, i.e. $\bar{\theta} = \{\theta^n\}_{n=1}^{N_t}$. The whole evolution procedure from f to u^{N_t} by $\bar{\mathcal{N}}_{\bar{\theta}}$, i,e, $\bar{\mathcal{N}}_{\bar{\theta}} = \bar{\mathcal{N}}^{N_t}_{\theta^{N_t}} \circ \bar{\mathcal{N}}^{N_t-1}_{\theta^{N_t-1}} \circ \cdots \circ \bar{\mathcal{N}}^1_{\theta^1}$. Given a dataset $\{(u_i, v_i)\}_{i=1}^B$, then the discrete training problem is to solve the following constrained optimization problem:

$$\min_{\bar{\theta}} \frac{1}{B} \sum_{i=1}^{B} \ell(\bar{\mathcal{N}}_{\bar{\theta}}(u_i), v_i). \tag{3.1}$$

3.2. Time Discretizations by Operator Splitting Method. In the following sections, we illustrate our numerical scheme for propagator $\bar{\mathcal{N}}^1_{\theta^1}$. Propagator $\bar{\mathcal{N}}^n_{\theta^n}$ for $n = 2, ..., N_t$ is exactly the same by replacing θ^1 by θ^n. Our numerical scheme is designed based on operator splitting methods. For simplicity, we take $\Delta t = 1$. It is often used in data analysis and image processing literature and is equivalent to changing the network parameters properly. In the following, with a slight abuse of notation, we denote the time-discretized version of any time-dependent function $a(\cdot, t)$ by $a^n(\cdot) := a(\cdot, t^n)$, where t^n is the n-th time step. Using the Lie scheme [14, Sec. 2.2] as explained in Appendix C, given $u^0 = f$, we compute u^1 by $M = 4 + J$ substeps:

Substep 1: Solve $u^{1/M}$ from

$$u^{1/M} - u^0 = \left\langle \gamma^0(\mathbf{x}, \cdot; u^0),\ V^0(\cdot, \mathbf{y}; u^0) \right\rangle_{\Omega_x}. \tag{3.2}$$

Substep 2: Compute $u^{2/M}$ from

$$u^{2/M} - u^{1/M} = \partial I_{S_1(\sigma_1^0, \sigma_2^0)}(u^{2/M}). \tag{3.3}$$

Substep 2+j : Compute $u^{(2+j)/M}$ sequentially for $j = 1, ..., J$ from

$$u^{(2+j)/M} - u^{(1+j)/M} = \left\langle W_j^0(\cdot, \mathbf{y}),\ u^{(1+j)/M}(\mathbf{x}, \cdot) \right\rangle_{\Omega_y} + b_j^0(\mathbf{x}) + \partial I_{S_2}(u^{(2+j)/M}). \tag{3.4}$$

Substep 3+J: Compute $u^{(3+J)/M}$ from

$$u^{(3+J)/M} = \frac{1}{2}(u^{(2+J)/M} + u^{2/M}) \tag{3.5}$$

Substep 4+J: Compute u^1 from

$$u^1 - u^{(3+J)/M} = \partial I_{S_1(\sigma_1^0, \sigma_2^0)}(u^1) \tag{3.6}$$

Note that in (3.2)–(3.4) and (3.6), the left hand side $u^{(k+1)/M} - u^{k/M} = \frac{u^{(k+1)/M} - u^{k/M}}{\Delta t}$ with $\Delta t = 1$ is the finite difference approximation of u_t.

Substep 1 to Substep $(4+J)$ evolve u^0 to u^1 by sequentially passing it through all operators in (2.8), corresponding to all components in Transformer from input to output. Substep 1 corresponds to the attention layer and the skip connection. Substep 2 corresponds to layer

normalization. Substep 3 to Substep $(4+J)$ correspond to the J layers in the fully connected network. Substep $3+J$ corresponds to the skip connection after the fully connected network. The last substep corresponds to the final layer normalization. From here, we can already see that each substep in the Lie splitting scheme corresponds to one layer in the Transformer. In [46], it used $J=2$ and thus $M=6$. The different layers in the Transformer is just a sequential computation of the substep solutions:

$$\begin{aligned}(3.7)\quad u^0 \quad &\text{(input token)} \mapsto u^{1/6}\text{(attention)} \mapsto u^{2/6}\text{(normalization layer)}\\ &\mapsto u^{3/6} \text{ and } u^{4/6}\text{(fully connected layer with ReLU)}\\ &\mapsto u^{5/6}\text{(skip connection layer)} \mapsto u^1\text{(normalization layer)}.\end{aligned}$$

3.3. Solutions to Each Subproblem. In the following, we supply the details for the solution to each subproblem in the splitting scheme (3.2)–(3.6).

3.3.1. Solution to Subproblem (3.2). We have

$$(3.8)\qquad u^{1/M} = u^0 + \int_{\Omega_x} \mathrm{Softmax}_2\left(\frac{1}{\sqrt{|\Omega_y|}}\left\langle Q^0(\mathbf{x},\cdot;u^0),\ K^0(\eta,\cdot;u^0;)\right\rangle_{\Omega_y}\right) V^0(\eta,\mathbf{y};u^0)d\eta.$$

This step is a continuous version of the attention layer in [46]. It looks very simple and uses two integral transformations to get the attention score through a softmax operator. This score function has values between $[0,1]$ and it is multiplied with the $V^0(\eta,\mathbf{y};u^0)$, which are the extracted features from the input data through another integral transformation.

3.3.2. Solution to Subproblem (3.3) and (3.6). For $u^{2/M}$ and u^1 in problem (3.3) and (3.6), they are in the form of

$$(3.9)\qquad u - v = \partial I_{S_1(\sigma_1,\sigma_2)}(u)$$

for the given function $v(\mathbf{x},\mathbf{y})$ and constants σ_1,σ_2. It is easy to derive that the solution u of (3.9) solves the following problem:

$$(3.10)\qquad u = \operatorname*{argmin}_{\bar{u}\in S_1(\sigma_1,\sigma_2)} \frac{1}{2}\|\bar{u}-v\|^2_{\Omega_y}.$$

This shows that u is a projection of v to the set $S_1(\sigma_1,\sigma_2)$. The following theorem gives a closed-form solution formula for u:

Theorem 3.1. *The solution to problem (3.10) is given as:*

$$(3.11)\qquad u(\mathbf{x},\mathbf{y}) = \frac{v(\mathbf{x},\mathbf{y})-\alpha(\mathbf{x})}{\sqrt{\beta(\mathbf{x})}}\sigma_2 + \sigma_1,$$

$$(3.12)\qquad \textit{with } \alpha(\mathbf{x};v) = \frac{1}{|\Omega_y|}\int_{\Omega_y} v(\mathbf{x},\xi)d\xi, \quad \beta(\mathbf{x};v) = \frac{1}{|\Omega_y|}\int_{\Omega_y}(v(\mathbf{x},\xi)-\alpha(\mathbf{x}))^2 d\xi.$$

Theorem 3.1 is proved in Appendix B. We remark that the expression (3.12) recovers the normalization layer in the continuous setting. According to Theorem 3.1, we see that the solutions $u^{2/M}$ and $u^{(4+J)/M}$ are explicitly given as:

$$u^{2/M}(\mathbf{x},\mathbf{y}) = \frac{u^{1/M}(\mathbf{x},\mathbf{y}) - \alpha(\mathbf{x}; u^{1/M})}{\sqrt{\beta(\mathbf{x}; u^{1/M})}}\sigma_2^0 + \sigma_1^0, \tag{3.13}$$

$$u^{1}(\mathbf{x},\mathbf{y}) = \frac{u^{(3+J)/M}(\mathbf{x},\mathbf{y}) - \alpha(\mathbf{x}; u^{(3+J)/M})}{\sqrt{\beta(\mathbf{x}; u^{(3+J)/M})}}\sigma_2^0 + \sigma_1^0. \tag{3.14}$$

3.3.3. Solution to Subproblem (3.4). Problem (3.4) is a semi-implicit equation for $u^{(2+j)/M}$. It can be solved exactly by the following sequential splitting

$$\begin{cases} \bar{u}^{(2+j)/M} = u^{(1+j)/M} + \left\langle W_j^0(\cdot,\mathbf{y}),\ u^{(1+j)/M}(\mathbf{x},\cdot)\right\rangle_{\Omega_y} + b_j^0(\mathbf{x}), \\ u^{(2+j)/M} - \bar{u}^{(2+j)/M} = \partial I_{S_2}(u^{(2+j)/M}). \end{cases} \tag{3.15}$$

The first equation in (3.15) is linear in $\bar{u}^{(2+j)/M}$, corresponding to a linear layer in a feedforward network in continuous settings. The second equation solves

$$u^{(2+j)/M} = \operatorname*{argmin}_{v\in S_1} \int_{\Omega_x}\int_{\Omega_y} |v - \bar{u}^{(2+j)/M}|^2 d\mathbf{y}d\mathbf{x}. \tag{3.16}$$

It is easy to see that its solution can be computed point-wisely:

$$u^{(2+j)/M} = \max\{\bar{u}^{(2+j)/M}, 0\} = \mathrm{ReLU}(\bar{u}^{(2+j)/M}), \tag{3.17}$$

which corresponds ReLU activation. As a result, Substep $2+j$ for $j = 1, ..., J$ is a feedforward network layer activated by ReLU.

3.4. Spatial Discretizations. In this subsection, we discretize the solution discussed in Section 3.3 spatially. We will show that after spatial discretizations, the splitting scheme (3.2)–(3.6) exactly recovers the Transformer architecture in [46].

Suppose $\Omega_x = [0, L_x],\ \Omega_y = [0, L_y]$. We uniformly discretize Ω_x, Ω_y by N_x and N_y grids in the $\mathbf{x}$ and $\mathbf{y}$ coordinates, respectively. For simplicity, we suppose $L_x = N_x, L_y = N_y$, leading to discretization steps $\Delta x = L_x/N_x = 1,\ \Delta y = L_y/N_y = 1$, and the grids by $\mathbf{x}_k = k, \mathbf{y}_k = k$. We use $\mathbf{1}_m$ to denote an m-dimensional vector with elements 1. For a matrix $\mathbf{W} \in \mathbb{R}^{N_x\times N_y}$ and a vector $\boldsymbol{b} \in \mathbb{R}^{N_x}$, the matrix $\mathbf{z} = (\mathbf{W} + \boldsymbol{b}) \in \mathbb{R}^{N_x\times N_y}$ is computed so that

$$\mathbf{z}_{k,l} = W_{k,l} + b_k.$$

This is an easy notation to add a vector to a matrix in the discrete setting. With the spatial discretizations above, we define the discrete integrals (sum) as

$$\int_{\bar{\Omega}_x} f d\mathbf{x} = \sum_{k=1}^{N_x} f_k, \quad \int_{\bar{\Omega}_x}\int_{\bar{\Omega}_y} f(\mathbf{x},\mathbf{y}) d\mathbf{x}d\mathbf{y} = \sum_{k=1}^{N_x}\sum_{l=1}^{N_y} f_{k,l}. \tag{3.18}$$

For any functions $v(\mathbf{x}), w(\mathbf{x})$ defined on Ω_x, denote their discrete counterpart by $\mathbf{v}, \mathbf{w}$. We denote the discretized inner product between $\mathbf{v}$ and $\mathbf{w}$ as

$$\left\langle \mathbf{v}, \mathbf{w} \right\rangle_{\bar{\Omega}_x} = \mathbf{v}^\top \mathbf{w} = \sum_{k=1}^{N_x} v_k w_k. \tag{3.19}$$

The discretized integrals and inner products $\langle \cdot, \cdot \rangle_{\bar{\Omega}_y}$ and $\langle \cdot, \cdot \rangle_{\bar{\Omega}_x \times \bar{\Omega}_y}$ are defined similarly.

3.4.1. Computing the Discrete Solutions $u^{1/M}$ in (3.8). Before we discretize the formula (3.8), we first discretize some related quantities and operations. To compute the attention scores, we discretize the integral transformations as follows:

$$\mathbf{Q}^0(\mathbf{u}^0) = \mathbf{u}^0 \mathbf{W}^{Q,0}, \quad \mathbf{K}^0(\mathbf{u}^0) = \mathbf{u}^0 \mathbf{W}^{K,0}, \quad \mathbf{V}^0(\mathbf{u}^0) = \mathbf{u}^0 \mathbf{W}^{V,0}. \tag{3.20}$$

Here, $\mathbf{u}^0$ is in $\mathbb{R}^{N_x \times N_y}$ and $\mathbf{W}^{Q,0}, \mathbf{V}^{Q,0}, \mathbf{K}^{Q,0}$ are in $\mathbb{R}^{N_y \times N_y}$, and the expressions such as $\mathbf{u}^0 \mathbf{W}^{Q,0}$ represent standard matrix multiplications that yield new matrices in $\mathbb{R}^{N_x \times N_y}$. We emphasize that the multiplication used here is the conventional matrix product. For $\mathbf{A} = \{a_{k,l}\}_{k,l} \in \mathbb{R}^{N_x \times N_x}$, the discretized version of $\mathrm{Softmax}_2(\mathbf{A})$ is given as

$$\left(\mathrm{Softmax}_{2,\mathrm{dis}}(\mathbf{A})\right)_{k,l} = \frac{\exp(a_{k,l})}{\sum_{l=1}^{N_x} \exp(a_{k,l})}. \tag{3.21}$$

Then the updating formula (3.8) for $u^{1/M}$ in the discrete setting is

$$\mathbf{u}^{1/M} = \mathbf{u}^0 + \mathrm{Softmax}_{2,\mathrm{dis}}\left(\mathbf{Q}^0(\mathbf{u}^0)(\mathbf{K}^0(\mathbf{u}^0))^\top\right)\mathbf{V}^0(\mathbf{u}^0). \tag{3.22}$$

We emphasis again that this is exactly the attention layer in [46]. This is just a compact mathematical expression for it. All the matrices $\mathbf{W}^{Q,0}, \mathbf{V}^{Q,0}, \mathbf{K}^{Q,0}$ will be learned in the training process.

3.4.2. Computing the Discrete Solutions $u^{2/M}$ and $u^{(4+J)/M}$ in (3.9). In Theorem 3.1, for any $\mathbf{v} = \{v_{k,l}\}_{k,l} \in \mathbb{R}^{N_x \times N_y}$, we discretize $\alpha(\mathbf{x})$ and $\beta(\mathbf{x})$ as

$$\alpha_k(\mathbf{v}) = \frac{1}{L_y}\sum_{l=1}^{N_y} v_{k,l}, \;\; \beta_k(\mathbf{v}) = \frac{1}{L_y}\sum_{l}(v_{k,l} - \alpha_k(\mathbf{v}))^2 \tag{3.23}$$

for $k = 1, ..., N_x$. Then we compute the discrete solutions $u^{2/M}$ and $u^{(4+J)/M}$ as

$$u_{k,l}^{2/M} = \sigma_2^0 \frac{u_{k,l}^{1/M} - \alpha_k(\mathbf{u}^{1/M})}{\sqrt{\beta_k(\mathbf{u}^{1/M})}} + \sigma_1^0, \tag{3.24}$$

$$u_{k,l}^{n+(4+J)/M} = \sigma_2^0 \frac{u_{k,l}^{(3+J)/M} - \alpha_k(\mathbf{u}^{(3+J)/M})}{\sqrt{\beta_k(\mathbf{u}^{(3+J)/M})}} + \sigma_1^0, \tag{3.25}$$

for $k = 1, ..., N_x$ and $l = 1, ..., N_y$.

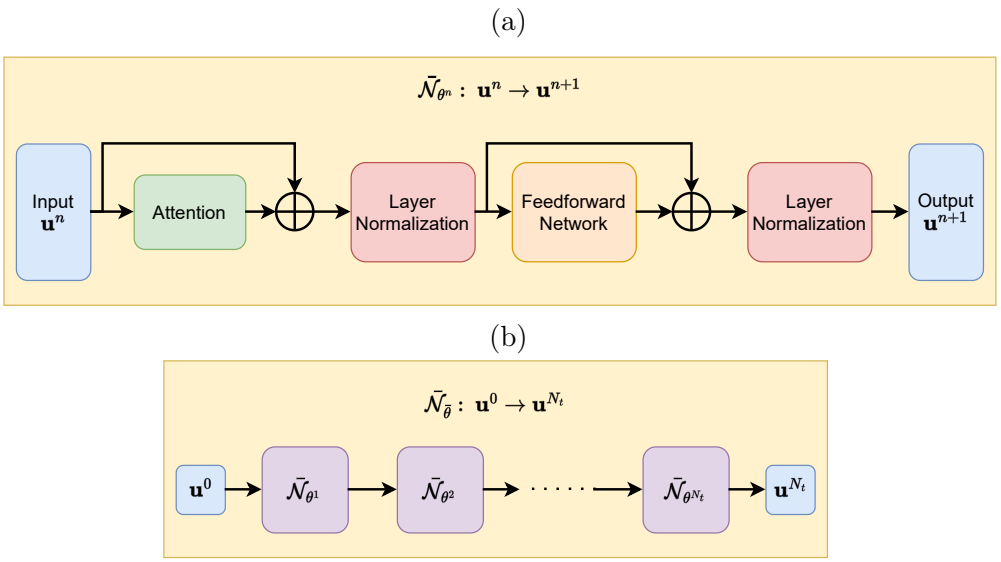

Figure 1. *(a) Illustration of the discretized operator-splitting scheme (3.2)–(3.6). (b) Illustration of the whole evolution.*

3.4.3. Computing the Discrete Solutions $u^{(3+J)/M}$ in (3.5). For $u^{(3+J)/M}$, its discrete analogue is computed as

$$\mathbf{u}^{(3+J)/M} = \frac{1}{2}(\mathbf{u}^{(2+J)/M} + \mathbf{u}^{2/M}). \tag{3.26}$$

3.4.4. Computing the Discrete Solutions $u^{(2+j)/M}$ in (3.15). In (3.15), we first compute

$$\bar{\mathbf{u}}^{(2+j)/M} = \mathbf{u}^{(1+j)/M} + \mathbf{u}^{(1+j)/M}\mathbf{W}_j^0 + \boldsymbol{b}_j^0, \tag{3.27}$$

and then update

$$\mathbf{u}^{(2+j)/M} = \mathrm{ReLU}(\bar{\mathbf{u}}^{(2+j)/M}). \tag{3.28}$$

The structure of the discretized scheme (3.2)–(3.6) is illustrated in Figure 1(a), and it exactly recovers the well-known Transformer architecture introduced in [46], see the next section for detailed explanations.

4. Mathematical Explanations of Transformers.

4.1. Connections to Transformer Encoder in [46]. Transformer architecture was first proposed in [46] for natural language processing. After time and spatial discretization, our splitting scheme recovers the well-known Transformer with single-head attention in [46]. We supply the details of the explanations in the following. Let us first show that one iteration $\mathbf{u}^0 \to \mathbf{u}^1$ realizes one Transformer block:

- Input $\mathbf{u}^0$: With our discretization, $\mathbf{u}^0 \in \mathbb{R}^{N_x \times N_y}$. It is an input with N_x tokens. Each token is represented as an integer-vector with N_y numbers.

- Scaled dot-product attention: Equation (3.22) is a single head attention with a skip connection. The second term in the right-hand side of (3.22) is a single-head attention with query matrix $\mathbf{Q}^0$, key matrix $\mathbf{K}^0$ and value matrix $\mathbf{V}^0$, whose weight matrices are $\mathbf{W}^{Q,0}, \mathbf{W}^{K,0}$ and $\mathbf{W}^{V,0}$, respectively. It recovers the single-head attention in [46]. The first term in the right-hand side is a skip connection, adding together the input and output of attention.
- Layer normalization: Equation (3.24)–(3.25) perform layer normalization. For each token, (3.24)–(3.25) normalize the input along the embedding direction to have mean σ_1 and variance σ_2^2.
- Position-wise feedforward network: Equation (3.27)–(3.28) realizes a feedforward network with J layers activated by ReLU. In particular, (3.27) can be rewritten as

$$\bar{\mathbf{u}}^{(2+j)/M} = \mathbf{u}^{(1+j)/M}(\mathbf{I} + \mathbf{W}_j^0) + \boldsymbol{b}_j^0,$$

  where $\mathbf{I}$ denotes the identity matrix. It is a linear layer with weight matrix $(\mathbf{I} + \mathbf{W}_j^0)$ and bias $\boldsymbol{b}_j^0$. Set $J = 2$. Equation (3.27)–(3.28) recovers the feedforward network in [46].
- Skip connection: Equation (3.26) is a relaxation step in the splitting scheme. It realizes the skip connection by averaging the input and output of the feedforward network.

The discussion above shows that one time step of the splitting scheme supplied in Section 3.2 realizes the Transformer of [46]. In Section 3.2, the time domain is discretized into N_t time steps. Thus, the whole operator-splitting scheme is equivalent to compositions of N_t Transformer blocks. Since the learnable variables $\theta^n = \{\mathbf{W}^{Q,n}, \mathbf{W}^{K,n}, \mathbf{W}^{V,n}, \{\mathbf{W}_j^n, \boldsymbol{b}_j^n\}_{j=1}^J\}$ depends on time, they are different for different time steps. Setting $N_t = 6$ recovers the architecture in [46]. Furthermore, solving (3.1) for the control variables is equivalent to training the network.

4.2. Connections to ViT in [8]. Based on the Transformer encoder in [46], vision Transformer (ViT) was proposed in [8] for image classification. In ViT, each image is cropped into patches, each of which is taken as a token. ViT has a similar architecture as the Transformer encoder in [46], except that it has an additional input embedding layer with learnable parameters, and a linear layer at the end for output.

In order to use (3.2)–(3.6) to provide a mathematical explanation of ViT, we need to incorporate the additional embedding layer and last linear layer with our algorithm. It can be easily done by introducing data-driven data pre-processing and post-processing steps. For the additional input embedding, we design a pre-processing step to generate the initial condition f. Consider the discrete setting. Denote $\mathbf{R} \in \mathbb{R}^{(N_x-1)\times D}$ as the collection of $(N_x - 1)$ patches of an image, where each row of $\mathbf{R}$ corresponds to one flattened patch of dimension D. Let $\mathbf{E} \in \mathbb{R}^{D\times N_y}$ be an embedding matrix and $\mathbf{v}_0 \in \mathbb{R}^{1\times N_y}$ be a vector where both are learnable. The vector $\mathbf{v}_0$ represents the extra learnable embedding. We generate $\mathbf{u}_0$ as

$$\mathbf{u}_0 = F(\mathbf{R}) = \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{R}\mathbf{E} \end{bmatrix}. \tag{4.1}$$

The last linear layer in ViT can be taken as a data post-processing. For any $\mathbf{u} \in \mathbb{R}^{N_x\times N_y}$ and

weight matrix $\mathbf{B} \in \mathbb{R}^{N_y \times d}$. Define

$$G(\mathbf{B}; \mathbf{u}) = \begin{bmatrix} 1 & \mathbf{0}_{1\times(N_x-1)} \end{bmatrix} \mathbf{u}\mathbf{B}, \tag{4.2}$$

where $\mathbf{0}_{1\times(N_x-1)}$ is a zero vector of size $(N_x - 1)$. Let $\bar{\theta}, \bar{\mathcal{N}}$ be defined as in Section 3.1. We denote

$$\widetilde{\theta} = \{\bar{\theta}, \mathbf{v}_0, \mathbf{E}, \mathbf{B}\}$$

as the collection of all learnable parameters. With the data pre- and post-processing, we define the whole operation

$$\bar{\mathcal{N}}_{\widetilde{\theta}}^{\mathrm{ViT}} = G \circ \bar{\mathcal{N}} \circ F. \tag{4.3}$$

One can readily check that $\bar{\mathcal{N}}_{\widetilde{\theta}}^{\mathrm{ViT}}$ recovers the ViT in [8]. Given a dataset $\{(\mathbf{u}_i, \mathbf{v}_i)\}_{i=1}^B$, we learn all parameters by solving

$$\min_{\widetilde{\theta}} \frac{1}{B} \sum_{i=1}^{B} \ell(\bar{\mathcal{N}}_{\widetilde{\theta}}^{\mathrm{ViT}}(\mathbf{u}_i), \mathbf{v}_i), \tag{4.4}$$

which is equivalent to training a ViT.

5. Multi-Head Attention. As discussed in Section 4, the proposed splitting scheme for problem (2.8) is the Transformer architecture with one-head attention. In this section, we extend (2.8) and splitting scheme (3.2)–(3.6) so that it realizes multi-head attention.

5.1. Control Problem with Continuous Head Dimension. We first consider a continuous-head setting. After discretizations, we will show that it recovers multi-head attention. In the continuous-head setting, we take head as an additional dimension. To accommodate the new dimension, we update the definition of W^Q, W^K, W^V to $W^Q(\mathbf{y}, \widetilde{\mathbf{y}}, h, t), W^K(\mathbf{y}, \widetilde{\mathbf{y}}, h, t), W^V(\mathbf{y}, \widetilde{\mathbf{y}}, h, t)$ on $\Omega_y \times \Omega_y \times \Omega_h \times [0, +\infty)$, where h is the variable for the head dimension with domain Ω_h. Similar to (2.1)–(2.3), we define

$$\begin{aligned} Q(\mathbf{x}, \mathbf{y}, h, t; u) &= \left\langle W^Q(\cdot, \mathbf{y}, h, t),\ u(\mathbf{x}, \cdot, t) \right\rangle_{\Omega_y}, \\ K(\mathbf{x}, \mathbf{y}, h, t; u) &= \left\langle W^K(\cdot, \mathbf{y}, h, t),\ u(\mathbf{x}, \cdot, t) \right\rangle_{\Omega_y}, \\ V(\mathbf{x}, \mathbf{y}, h, t; u) &= \left\langle W^V(\cdot, \mathbf{y}, h, t),\ u(\mathbf{x}, \cdot, t) \right\rangle_{\Omega_y}. \end{aligned} \tag{5.1}$$

We update $\gamma(\mathbf{x}, \widetilde{\mathbf{x}}, t; u)$ in (2.4) to

$$\gamma(\mathbf{x}, \widetilde{\mathbf{x}}, h, t; u) = \mathrm{Softmax}_2\left(\frac{1}{\sqrt{|\Omega_y|}} \left\langle Q(\mathbf{x}, \cdot, h, t; u), K(\widetilde{\mathbf{x}}, \cdot, h, t; u) \right\rangle_{\Omega_y} \right). \tag{5.2}$$

The control problem (2.8) is updated to

$$
\begin{cases}
u_t = \underbrace{\int_{\Omega_h} \Big\langle \gamma(\mathbf{x},\cdot,h,t;u),\ V(\cdot,\mathbf{y},h,t;u) \Big\rangle_{\Omega_x} dh}_{\text{I-a: attention}} + \underbrace{\partial I_{S_1(\sigma_1,\sigma_2)}(u)}_{\text{II: layer normalization}} \\
\qquad \underbrace{+ \sum_{j=1}^{J} \left(\Big\langle W_j(\cdot,\mathbf{y},t), u(\mathbf{x},\cdot,t) \Big\rangle_{\Omega_y} + b_j(\mathbf{x},t) \right) + \partial I_{S_2}(u)}_{\text{III: fully connected network}} \ \text{ for } t \in (0,T], \\
u(\mathbf{x},\mathbf{y},0) = f(\mathbf{x},\mathbf{y}).
\end{cases} \tag{5.3}
$$

Compared to (2.8), problem (5.3) replace I by I-a, which introduces an additional integral with respect to h.

5.2. Time Discretizations Through Operator-Splitting Method. The time discretizations and operator-splitting scheme for problem (5.3) are the same as those for problem (2.8) except that we update Substep 1 to

$$
u^{1/M} - u^0 = \int_{\Omega_h} \Big\langle \gamma^0(\mathbf{x},\cdot,h;u^0), V^0(\cdot,\mathbf{y},h;u^0) \Big\rangle_{\Omega_x} dh, \tag{5.4}
$$

whose solution is given as

$$
u^{1/M} = \tag{5.5}
$$

$$
u^0 + \int_{\Omega_h} \int_{\Omega_x} \mathrm{Softmax}_2 \left(\frac{1}{\sqrt{|\Omega_y|}} \Big\langle Q^0(\mathbf{x},\cdot,h;u^0), K^0(\eta,\cdot,h;u^0;) \Big\rangle_{\Omega_y} \right) V^0(\eta,\mathbf{y},h;u^0) d\eta dh.
$$

5.3. Discrete Solutions of $u^{1/M}$ in (5.5). Suppose $\Omega_h = [0, L_h]$. In addition to the spatial discretization discussed in Section 3.4, we discretize the head domain into N_h grids. For simplicity, we suppose $L_h = N_h$.

For function $W^{K,0}(\mathbf{y},\widetilde{\mathbf{y}},h)$, we denote $W^{Q,0}_{k,l,m} = W^{Q,0}(k,l,m), \mathbf{W}^{Q,0}_m = \{W^{Q,0}_{k,l,m}\}_{k,l} \in \mathbb{R}^{N_y \times N_y}$ and $\mathbf{W}^{Q,0} = \{\mathbf{W}^{Q,0}_m\}_m \in \mathbb{R}^{N_y \times N_y \times N_h}$. We define $\mathbf{W}^{K,0}, \mathbf{W}^{V,0}$ similarly. We denote the discretized integral transformation in the updated attention score as

$$
\mathbf{Q}^0_m(\mathbf{u}^0) = \mathbf{u}^0 \mathbf{W}^{Q,0}_m,\ \mathbf{K}^0_m(\mathbf{u}^0) = \mathbf{u}^0 \mathbf{W}^{K,0}_m,\ \mathbf{V}^n_m(\mathbf{u}^0) = \mathbf{u}^0 \mathbf{W}^{V,0}_m,
$$

and denote $\mathbf{Q}^0(\mathbf{u}) = \{\mathbf{Q}^0_m(\mathbf{u}^0)\}_m$, $\mathbf{K}^0(\mathbf{u}^0) = \{\mathbf{K}^0_m(\mathbf{u}^0)\}_m$, $\mathbf{V}^0(\mathbf{u}^0) = \{\mathbf{V}^0_m(\mathbf{u}^0)\}_m \in \mathbb{R}^{N_x \times N_y \times N_h}$. The inner product $\Big\langle Q^0(\mathbf{x},\cdot,h;u^0), K^0(\widetilde{\mathbf{x}},\cdot,h;u^0) \Big\rangle_{\Omega_y}$ is discretized as

$$
\Big\langle \mathbf{Q}^0_m(\mathbf{u}^0), \mathbf{K}^0_m(\mathbf{u}^0) \Big\rangle_{\bar{\Omega}_y} = \mathbf{Q}^0_m(\mathbf{u}^0)(\mathbf{K}^0_m(\mathbf{u}^0))^\top. \tag{5.6}
$$

Then the updating formula (5.5) for $u^{1/M}$ is discretized as

$$
\mathbf{u}^{1/M} = \mathbf{u}^0 + \sum_{m=1}^{N_h} \mathrm{Softmax}_{2,\mathrm{dis}} \left(\frac{1}{\sqrt{L_x}} \mathbf{Q}^0_m(\mathbf{u}^0)(\mathbf{K}^0_m(\mathbf{u}^0))^\top \right) \mathbf{V}^0_m(\mathbf{u}^0). \tag{5.7}
$$

5.4. Connections to Transformers. Similar to the discussion in Section 4, with (5.7), the operator-splitting scheme for the control problem (5.3) realizes N_t encoder time steps in [46] with multi-head attention. To show this, we only need to demonstrate the equivalence between (5.7) and the multi-head attention.

The second term in the right-hand side of (5.7) sums over the embedding dimension and head dimension. It is a multi-head attention with query matrix $\mathbf{Q}^0$, key matrix $\mathbf{K}^0$ and value matrix $\mathbf{V}^0$, whose weight matrices are $\mathbf{W}^{Q,0}$, $\mathbf{W}^{K,0}$ and $\mathbf{W}^{V,0}$, respectively.

6. Convolution Transformers for Video and Image Problems. Image and video data inherently possess rich local spatial structures, which have been effectively exploited by convolutional neural networks (CNNs) [37]. Convolutions serve as a powerful mechanism for capturing spatially localized patterns, making them especially well-suited for data defined on Cartesian grids.

In contrast, Transformer architectures were originally developed for language processing, where the data—represented by functions such as $u(\mathbf{x}, \mathbf{y}, t)$—often lack explicit spatial structure, as in the case of sentences or token sequences. For such unstructured data, attention-based mechanisms leveraging general integral transforms offer a flexible and effective modeling tool.

However, when applying Transformers to structured data such as images, videos, or other signals on regular grids, it has been shown that convolutional operations are significantly more efficient for feature extraction. This is due to their locality, weight sharing, and computational efficiency. Notably, convolutions can be interpreted as a special class of integral transformations with translation-invariant kernels and localized support.

Thanks to the integral formulation of the Q, K, and V operators in our framework (Sections 2–3), it becomes straightforward to specialize these operators to convolutions. This allows our model to seamlessly incorporate convolutional structures, thereby leveraging domain-specific inductive biases and enhancing computational performance when applied to grid-structured data.

6.1. Convolution Transformers. We discuss the case for grayscale images. The framework can be extended to color images and videos by introducing additional variables to u.

We consider functions of interest $u(x, \mathbf{y}, t)$ where $x \in \Omega_x \subset \mathbb{R}$ is the index of tokens (image patches), $\mathbf{y} \in \Omega_y \subset \mathbb{R}^2$ are variables for spatial domain (indices of pixel location), and t is the time. For each token and any fixed time t, u is a two-dimensional image.

In this setting, we define $W^Q(\mathbf{y}, t), W^K(\mathbf{y}, t), W^V(\mathbf{y}, t) \in \Omega_y \times [0, +\infty)$ as convolution kernels. Operations Q, K, V are defined by covolution as

$$
\begin{aligned}
Q(x, \mathbf{y}, t; u) &= W^Q(\cdot, t) * u(x, \cdot, t) = \int_{\Omega_y} W^Q(\boldsymbol{\xi}, t) u(x, \mathbf{y} - \boldsymbol{\xi}, t) d\boldsymbol{\xi}, \\
K(x, \mathbf{y}, t; u) &= W^K(\cdot, t) * u(x, \cdot, t) = \int_{\Omega_y} W^K(\boldsymbol{\xi}, t) u(x, \mathbf{y} - \boldsymbol{\xi}, t) d\boldsymbol{\xi}, \\
V(x, \mathbf{y}, t; u) &= W^V(\cdot, t) * u(x, \cdot, t) = \int_{\Omega_y} W^V(\boldsymbol{\xi}, t) u(xb, \mathbf{y} - \boldsymbol{\xi}, t) d\boldsymbol{\xi}.
\end{aligned} \tag{6.1}
$$

The score function is computed as

$$\gamma(x,\widetilde{x},t;u)=\mathrm{Softmax}_2\left(\frac{1}{\sqrt{|\Omega_y|}}\int_{\Omega_y}Q(x,\boldsymbol{\xi},t;u)K(\widetilde{x},\boldsymbol{\xi},t;u)d\boldsymbol{\xi}\right), \tag{6.2}$$

and the output of the attention layer is then given by: $\left\langle \gamma(x,\cdot,t;u), V(\cdot,\mathbf{y},t;u)\right\rangle_{\Omega_x}$.

With the modification above, scheme (3.2)–(3.4) gives rise to convolutional Transformer. Solvers discussed in Section 3.3 can still be used to solve subproblems in the new scheme.

6.2. Connections to CvT in [50]. Convolutional vision Transformer (CvT) [50] incorporates Vit with convolutional neural networks (CNN), which utilizes both image local spatial structures (by CNN) and global contexture (by Transformer). CvT consists of two components, convolutional token embedding and convolutional Transformer block. The convolution Transformer discussed in Section 6.1 realizes a one-head CvT with one stage and without convolutional token embedding.

In fact, the convolutional token embedding is an additional convolution layer. Our operator-splitting scheme can be easily revised to accomodate this layer. Let $W^C(\mathbf{x},\mathbf{y},t)$ be a convolution kernel and $b^C(\mathbf{x},t)$ be a bias term. We consider the following control problem

$$\begin{cases} u_t = \underbrace{\left\langle \gamma(\mathbf{x},\cdot,t;u),\ V(\cdot,\mathbf{y},t;u)\right\rangle_{\Omega_x}}_{\text{I: attention}} + \underbrace{\partial I_{S_1(\sigma_1(t),\sigma_2(t))}(u)}_{\text{II: layer norml.}} + \underbrace{W^C(\cdot,t)*u(\mathbf{x},\cdot,t)+b^C(\mathbf{x},t)}_{\text{IV: conv. token embedding}} \\ \qquad \underbrace{+\sum_{j=1}^{J}\left(\left\langle W_j(\cdot,\mathbf{y},t),\ u(\mathbf{x},\cdot,t)\right\rangle_{\Omega_y}+b_j(\mathbf{x},t)\right)+\partial I_{S_2}(u)}_{\text{III: fully connected network}}, \quad t\in(0,T], \\ u(\mathbf{x},\mathbf{y},0)=f(\mathbf{x},\mathbf{y}), \end{cases} \tag{6.3}$$

where the score function γ is computed as discussed in Section 6.1. For the operator-splitting scheme (3.2)–(3.4), we add two additional substeps before (3.2). Given u^0, we compute u^1 by $M=6+J$ substeps:

Substep 1: Solve $\widetilde{u}^{1/M}$ from

$$u^{1/M}-u^0=W^{C,0}(\cdot)*u^0(\mathbf{x},\cdot)+b^{C,0}(\mathbf{x}). \tag{6.4}$$

Substep 2: Compute $u^{2/M}$ from

$$u^{2/M}-u^{1/M}=\partial I_{S_1(\sigma_1^0,\sigma_2^0)}(u^{2/M}). \tag{6.5}$$

The remaining intermediate variables $u^{3/M},...,u^1$ can be computed via (3.2)–(3.4), respectively. In spatial discretization, N_x corresponds to the number of tokens, i.e., the number of channels in CvT. Equipped with data pre- and post-processing as discussed in Section 4.2, the discretized new operator splitting scheme recovers a one-head one-stage CvT.

For a K-stage CvT, the number of tokens changes from stage to stage. This property can be realized by incorporating the new operator-splitting scheme with a hybrid splitting scheme [42]. Specifically, we need to define K convolution kernels and a bias for convolutional token embedding. Based on the multigrid idea, these kernels are discretized at different scales, corresponding to various numbers of tokens.

7. Conclusion. In this paper, we have introduced a novel operator-theoretic framework that interprets the Transformer architecture as a discretized operator-splitting scheme for a continuous integro-differential equation. Within this formulation, core components of the Transformer—such as self-attention, layer normalization, and feedforward networks—are rigorously derived as substeps in a structured variational process and operator-splitting techniques. This approach establishes a unified mathematical foundation that applies broadly to Transformer-based models, including the original Transformer [46], Vision Transformer (ViT) [8], and Convolutional vision Transformer (CvT) [50]. By bridging deep learning architectures with continuous mathematical modeling, our framework advances theoretical understanding and offers a principled pathway for the design, analysis, and control of future neural architectures. We believe this perspective will stimulate further developments in both the theoretical analysis and practical refinement of attention-based models.

Appendix.

Appendix A. Explanation of the variables for Transformers. The input to the Transformer model is usually a sequence. In the context of language processing, the input is usually sentences. Transformers do not process raw text directly. Instead, they break sentences into tokens. A token can be a word (e.g., "hello"), a subword (e.g., "multi" + "grid" = "multigrid"), or even a single character. Then, each different token is transformed into unique embedding vectors through an embedding layer. The way text is tokenized affects how efficiently a model understands language.

For example, we have an input sentence "Life is like a boat", and we treat each word as a token. The embedding layer would transform each token into a 512-dimensional vector. Then, the input sentence is represented by a matrix U of size 5×512:

$$\underbrace{\text{Life is like a boat}}_{\text{Input sentence}} \to \underbrace{[\text{"Life"},\text{"is"},\text{"like"},\text{"a"},\text{"boat"}]}_{\text{Tokens}} \to U = \underbrace{\begin{bmatrix} \text{Embedding("Life")} \\ \text{Embedding("is")} \\ \text{Embedding("like")} \\ \text{Embedding("a")} \\ \text{Embedding("boat")} \end{bmatrix}}_{\text{Embedded matrix}}.$$

In general, a sentence of N_x tokens with N_y-dimensional embedding would be transformed into a matrix U of size $N_x \times N_y$. This matrix U can be viewed as the discretization of a continuous function $u(\mathbf{x}, \mathbf{y}, t) : \Omega_x \times \Omega_y \to \mathbb{R} \times [0, T)$, where Ω_x and Ω_y are continuous domains, on a structured grid of size $N_x \times N_y$. Here, the variable $\mathbf{x}$ indicates the position of tokens, $\mathbf{y}$ indicates the entry of token vectors, and t indicates the time variable. The embedded matrix U is fed into the Transformer model and updated by attention layers.

Appendix B. Proof of Theorem 3.1.

Proof of Theorem 3.1. Problem (3.10) can be rewritten as

$$\begin{cases} u(\mathbf{x},\mathbf{y}) = \operatorname{argmin}_{\bar{u}} \frac{1}{2}\|\bar{u} - v\|^2_{\Omega_y}, \\ \frac{1}{|\Omega_y|}\int_{\Omega_y} u(\mathbf{x},\xi)d\xi = \sigma_1, \\ \frac{1}{|\Omega_y|}\int_{\Omega_y} (u(\mathbf{x},\xi)-\sigma_1)^2 d\xi = \sigma_2^2. \end{cases} \tag{B.1}$$

Introducing Lagrange multipliers λ_1, λ_2, the minimizers for the above problem satisfy:

$$\begin{cases} (u-v) + \frac{\lambda_1}{|\Omega_y|} + \frac{2\lambda_2}{|\Omega_y|}(u-\sigma_1) = 0, \\ \frac{1}{|\Omega_y|}\int_{\Omega_y} u(\mathbf{x},\mathbf{y})d\mathbf{y} = \sigma_1, \\ \frac{1}{|\Omega_y|}\int_{\Omega_y} (u(\mathbf{x},\mathbf{y})-\sigma_1)^2 d\mathbf{y} = \sigma_2^2. \end{cases} \tag{B.2}$$

Integrate the first equation over $\mathbf{y}$ gives rise to

$$0 = \int_{\Omega_y} u(\mathbf{x},\xi)d\xi - \int_{\Omega_y} v(\mathbf{x},\xi)d\xi + \lambda_1 = |\Omega_y|\sigma_1 - \int_{\Omega_y} v(\mathbf{x},\xi)d\xi + \lambda_1, \tag{B.3}$$

implying that

$$\lambda_1 = \int_{\Omega_y} v(\mathbf{x},\xi)d\xi - |\Omega_y|\sigma_1. \tag{B.4}$$

Multiply the first equation in (B.2) by $(u-\sigma_1)$ and then integrate over $\mathbf{y}$ gives rise to

$$\int_{\Omega_y} (u(\mathbf{x},\xi) - v(\mathbf{x},\xi))(u(\mathbf{x},\xi)-\sigma_1)d\xi + 2\lambda_2\sigma_2^2 = 0, \tag{B.5}$$

from which we deduce

$$\int_{\Omega_y} (u^2(\mathbf{x},\xi) - u(\mathbf{x},\xi)\sigma_1)d\xi - \int_{\Omega_y} v(\mathbf{x},\xi)(u(\mathbf{x},\xi)-\sigma_1)d\xi + 2\lambda_2\sigma_2^2 = 0. \tag{B.6}$$

Note that

$$\begin{aligned} \int_{\Omega_y} (u^2(\mathbf{x},\xi) - u(\mathbf{x},\xi)\sigma_1)d\xi &= \int_{\Omega_y} (u^2(\mathbf{x},\xi) - 2u(\mathbf{x},\xi)\sigma_1 + \sigma_1^2)d\xi \\ &= \int_{\Omega_y} (u(\mathbf{x},\xi)-\sigma_1)^2 d\xi = |\Omega_y|\sigma_2^2. \end{aligned} \tag{B.7}$$

Substituting (B.7) into (B.6) gives

$$\lambda_2 = -\frac{|\Omega_y|}{2} + \frac{\int_{\Omega_y} v(\mathbf{x},\xi)(u(\mathbf{x},\xi)-\sigma_1)d\xi}{2\sigma_2^2}. \tag{B.8}$$

Substituting (B.4) and (B.8) into the first equation of (B.2), we get

$$0 = (u-v) + \frac{\lambda_1}{|\Omega_y|} + \frac{2\lambda_2}{|\Omega_y|}(u-\sigma_1)$$

$$
\begin{aligned}
=&(u-v)+\frac{1}{|\Omega_y|}\int_{\Omega_y} v(\mathbf{x},\xi)d\xi-\sigma_1+\left(-1+\frac{\int_{\Omega_y} v(\mathbf{x},\xi)(u(\mathbf{x},\xi)-\sigma_1)d\xi}{|\Omega_y|\sigma_2^2}\right)u\\
&-\left(-1+\frac{\int_{\Omega_y} v(\mathbf{x},\xi)(u(\mathbf{x},\xi)-\sigma_1)d\xi}{|\Omega_y|\sigma_2^2}\right)\sigma_1\\
=&\frac{\int_{\Omega_y} v(\mathbf{x},\xi)(u(\mathbf{x},\xi)-\sigma_1)d\xi}{|\Omega_y|\sigma_2^2}(u-\sigma_1)-v+\frac{1}{|\Omega_y|}\int_{\Omega_y} v(\mathbf{x},\xi)d\xi. \qquad \text{(B.9)}
\end{aligned}
$$

We have

$$
\frac{\int_{\Omega_y} v(\mathbf{x},\xi)(u(\mathbf{x},\xi)-\sigma_1)d\xi}{|\Omega_y|\sigma_2^2}(u-\sigma_1)=v-\frac{1}{|\Omega_y|}\int_{\Omega_y} v(\mathbf{x},\xi)d\xi. \qquad \text{(B.10)}
$$

Define $\alpha(\mathbf{x}), \beta(\mathbf{x})$ as in Theorem 3.1. We claim that the solution to (B.10) is given as

$$
u=\frac{v-\alpha}{\sqrt{\beta}}\sigma_2+\sigma_1. \qquad \text{(B.11)}
$$

To verify it, note that (B.11) can be rewritten as

$$
\frac{u-\sigma_1}{\sigma_2}=\frac{v-\alpha}{\sqrt{\beta}}. \qquad \text{(B.12)}
$$

Utilizing (B.12), we deduce

$$
\begin{aligned}
&\frac{\int_{\Omega_y} v(\mathbf{x},\xi)(u(\mathbf{x},\xi)-\sigma_1)d\xi}{|\Omega_y|\sigma_2^2}(u-\sigma_1)\\
=&\frac{1}{|\Omega_y|}\left(\int_{\Omega_y} v(\mathbf{x},\xi)\frac{(u(\mathbf{x},\xi)-\sigma_1)}{\sigma_2}d\xi\right)\frac{u-\sigma_1}{\sigma_2}\\
=&\frac{1}{|\Omega_y|}\left(\int_{\Omega_y} v(\mathbf{x},\xi)\frac{(v(\mathbf{x},\xi)-\sigma_1)}{\sqrt{\beta}}d\xi\right)\frac{v-\sigma_1}{\sqrt{\beta}}\\
=&\frac{1}{|\Omega_y|\beta}\left(\int_{\Omega_y}[v^2(\mathbf{x},\xi)-2\sigma_1 v(\mathbf{x},\xi)+\sigma_1^2]d\xi\right)(v-\sigma_1)\\
=&\frac{1}{|\Omega_y|\beta}\left(\int_{\Omega_y}(v(\mathbf{x},\xi)-\sigma_1)^2 d\xi\right)(v-\sigma_1)\\
=&\frac{1}{|\Omega_y|\beta}(|\Omega_y|\beta)(v-\sigma_1)\\
=&v-\frac{1}{|\Omega_y|}\int_{\Omega_y} v(\mathbf{x},\xi)d\xi, \qquad \text{(B.13)}
\end{aligned}
$$

where the second equality follows from (B.12). The derivation above verifies that (B.11) is the solution to (B.10). ∎

Appendix C. An Introduction to Operator Splitting Schemes.

Operator-splitting methods decompose a complicated time-evolution problem into several substeps so that each substep can be solved efficiently. Consider the evolution equation

$$\begin{cases} u_t + \sum_{k=1}^{K} A_k(t;u) = 0 \text{ in } \Omega \times (0,T], \\ u(0) = u_0, \end{cases} \tag{C.1}$$

where $A_k(t;u)$'s are operators applied on u. Denote $t^n = n\Delta t$. Given u^n, there are three types of splitting strategies to compute u^{n+1}.

Sequential Lie scheme [14, Sec. 2.2] decomposes this computation into K sequential substeps. Given u^n, one computes u^{n+1} as follows:
For $k = 1, ..., K$, solve

$$\begin{cases} u_t + A_k(t;u) = 0 \text{ in } \Omega \times (t^n, t^{n+1}], \\ u(t^n) = u^{n+(k-1)/K}, \end{cases} \tag{C.2}$$

and set $u^{n+k/K} = u(t^{n+1})$.

Parallel splitting [32] decomposes the computation into K parallel substeps:
For $k = 1, ..., K$, compute

$$\begin{cases} v_t + KA_k(t;v) = 0 \text{ in } \Omega \times (t^n, t^{n+1}], \\ v(t^n) = u^n, \end{cases} \tag{C.3}$$

and set $v_k = u(t^{n+1})$. Then compute

$$u^{n+1} = \frac{1}{K}\sum_{k=1}^{K} v_k.$$

Hybrid splitting [42] is a combination of sequential and parallel splitting, which decomposes the computation into sequential substeps, each of which contains several parallel substeps.

Substep (C.2) and (C.3) can be solved by one-step explicit or implicit scheme. For example, (C.2) can be solved by

$$\frac{u^{n+k/K} - u^{n+(k-1)/K}}{\Delta t} + A_k(u^{n+k/K}) = 0 \quad \text{or} \quad \frac{u^{n+k/K} - u^{n+(k-1)/K}}{\Delta t} + A_k(u^{n+(k-1)/K}) = 0.$$

Our scheme in Section 3.2 is a sequential splitting scheme, in which (3.2) uses explicit scheme, and (3.3) and (3.6) use implicit scheme. Substep (3.4) is semi-implicit, which is slightly different from others. In fact, (3.4) is a composition of two substeps, as it is solved by another sequential splitting (3.15). In (3.15), the first substep is explicit and the second one is implicit.

It can be shown that with proper conditions, all three strategies are first-order accurate schemes for the evolution equation.

REFERENCES

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, arXiv preprint arXiv:1607.06450, (2016).

[2] M. Benning, E. Celledoni, M. J. Ehrhardt, B. Owren, and C.-B. Schönlieb, *Deep learning as optimal control problems: Models and numerical methods*, Journal of Computational Dynamics, 6 (2019), pp. 171–198.

[3] A. Bryutkin, J. Huang, Z. Deng, G. Yang, C.-B. Schönlieb, and A. Aviles-Rivero, *Hamlet: graph transformer neural operator for partial differential equations*, in Proceedings of the 41st International Conference on Machine Learning, 2024, pp. 4624–4641.

[4] M. Chen, H. Jiang, W. Liao, and T. Zhao, *Nonparametric regression on low-dimensional manifolds using deep relu networks: Function approximation and statistical recovery*, Information and Inference: A Journal of the IMA, 11 (2022), pp. 1203–1253.

[5] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, *Neural ordinary differential equations*, Advances in neural information processing systems, 31 (2018).

[6] C.-W. Cheng, J. Huang, Y. Zhang, G. Yang, C.-B. Schönlieb, and A. I. Aviles-Rivero, *Mamba neural operator: Who wins? transformers vs. state-space models for pdes*, arXiv preprint arXiv:2410.02113, (2024).

[7] Y. Cui, Y. Xu, R. Peng, and D. Wu, *Layer normalization for tsk fuzzy system optimization in regression problems*, IEEE Transactions on Fuzzy Systems, 31 (2022), pp. 254–264.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., *An image is worth 16x16 words: Transformers for image recognition at scale*, in International Conference on Learning Representations, 2021.

[9] S. Dutta, T. Gautam, S. Chakrabarti, and T. Chakraborty, *Redesigning the transformer architecture with insights from multi-particle dynamical systems*, Advances in Neural Information Processing Systems, 34 (2021), pp. 5531–5544.

[10] T. Furuya, M. V. de Hoop, and G. Peyré, *Transformers are universal in-context learners*, arXiv preprint arXiv:2408.01367, (2024).

[11] B. Geshkovski, C. Letrouit, Y. Polyanskiy, and P. Rigollet, *A mathematical perspective on transformers*, arXiv preprint arXiv:2312.10794, (2023).

[12] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*, vol. 9, Society for Industrial Mathematics, 1989.

[13] R. Glowinski, S. J. Osher, and W. Yin, *Splitting methods in communication, imaging, science, and engineering*, Springer, 2017.

[14] R. Glowinski, T.-W. Pan, and X.-C. Tai, *Some facts about operator-splitting and alternating direction methods*, in Splitting Methods in Communication, Imaging, Science, and Engineering, Springer, 2016, pp. 19–94.

[15] A. Graves, A.-r. Mohamed, and G. Hinton, *Speech recognition with deep recurrent neural networks*, in 2013 IEEE international conference on acoustics, speech and signal processing, Ieee, 2013, pp. 6645–6649.

[16] K. Gregor and Y. LeCun, *Learning fast approximations of sparse coding*, in Proceedings of the 27th international conference on international conference on machine learning, 2010, pp. 399–406.

[17] E. Haber, L. Ruthotto, E. Holtham, and S.-H. Jun, *Learning across scales—multiscale methods for convolution neural networks*, in Proceedings of the AAAI conference on artificial intelligence, vol. 32, 2018.

[18] P. Hagemann, J. Hertrich, and G. Steidl, *Stochastic normalizing flows for inverse problems: A markov chains viewpoint*, SIAM/ASA Journal on Uncertainty Quantification, 10 (2022), pp. 1162–1190.

[19] A. Havrilla and W. Liao, *Understanding scaling laws with statistical and approximation theory for transformer neural networks on intrinsically low-dimensional data*, Advances in Neural Information Processing Systems, 37 (2024), pp. 42162–42210.

[20] J. He and J. Xu, *Mgnet: A unified framework of multigrid and convolutional neural network*, Science china mathematics, 62 (2019), pp. 1331–1354.

[21] S. Jelassi, M. Sander, and Y. Li, *Vision transformers provably learn spatial structure*, Advances in Neural Information Processing Systems, 35 (2022), pp. 37822–37836.

[22] F. Jiang, Y. Jiang, H. Zhi, Y. Dong, H. Li, S. Ma, Y. Wang, Q. Dong, H. Shen, and Y. Wang,

Artificial intelligence in healthcare: past, present and future, Stroke and vascular neurology, 2 (2017).
[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 25 (2012).
[24] Z. Lai, L.-H. Lim, and Y. Liu, *Attention is a smoothed cubic spline*, arXiv preprint arXiv:2408.09624, (2024).
[25] Y. Lan, Z. Li, J. Sun, and Y. Xiang, *Dosnet as a non-black-box pde solver: When deep learning meets operator splitting*, Journal of Computational Physics, 491 (2023), p. 112343.
[26] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, *Fourier neural operator for parametric partial differential equations*, arXiv preprint arXiv:2010.08895, (2020).
[27] H. Liu, J. Liu, R. H. Chan, and X.-C. Tai, *Double-well net for image segmentation*, Multiscale Modeling & Simulation, 22 (2024), pp. 1449–1477.
[28] Y. Liu, Z. Zhang, and H. Schaeffer, *Prose: Predicting multiple operators and symbolic expressions using multimodal transformers*, Neural Networks, 180 (2024), p. 106707.
[29] Z. Long, Y. Lu, X. Ma, and B. Dong, *Pde-net: Learning pdes from data*, in International conference on machine learning, PMLR, 2018, pp. 3208–3216.
[30] J. Lu, Z. Shen, H. Yang, and S. Zhang, *Deep network approximation for smooth functions*, SIAM Journal on Mathematical Analysis, 53 (2021), pp. 5465–5506.
[31] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, *Learning nonlinear operators via deeponet based on the universal approximation theorem of operators*, Nature machine intelligence, 3 (2021), pp. 218–229.
[32] T. Lu, P. Neittaanmaki, and X.-C. Tai, *A parallel splitting-up method for partial differential equations and its applications to navier-stokes equations*, ESAIM: Mathematical Modelling and Numerical Analysis, 26 (1992), pp. 673–708.
[33] Y. Lu, Z. Li, D. He, Z. Sun, B. Dong, T. Qin, L. Wang, and T.-y. Liu, *Understanding and improving transformer from a multi-particle dynamic system point of view.*, in ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations.
[34] S. Martin, A. Gagneux, P. Hagemann, and G. Steidl, *Pnp-flow: Plug-and-play image restoration with flow matching*, arXiv preprint arXiv:2410.02423, (2024).
[35] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, *Deep learning for healthcare: review, opportunities and challenges*, Briefings in bioinformatics, 19 (2018), pp. 1236–1246.
[36] 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*, Journal of Computational physics, 378 (2019), pp. 686–707.
[37] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, Springer, 2015, pp. 234–241.
[38] D. Ruiz-Balet and E. Zuazua, *Neural ode control for classification, approximation, and transport*, SIAM Review, 65 (2023), pp. 735–773.
[39] L. Ruthotto and E. Haber, *Deep neural networks motivated by partial differential equations*, Journal of Mathematical Imaging and Vision, 62 (2020), pp. 352–364.
[40] Z. Shen, A. Havrilla, R. Lai, A. Cloninger, and W. Liao, *Transformers for learning on noisy and task-level manifolds: Approximation and generalization insights*, arXiv preprint arXiv:2505.03205, (2025).
[41] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, *Segmenter: Transformer for semantic segmentation*, in Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 7262–7272.
[42] X.-C. Tai, H. Liu, and R. Chan, *Pottsmgnet: A mathematical explanation of encoder-decoder based neural networks*, SIAM Journal on Imaging Sciences, 17 (2024), pp. 540–594.
[43] X.-C. Tai, H. Liu, R. H. Chan, and L. Li, *A mathematical explanation of unet*, Mathematical Foundations of Computing, (2024).
[44] S. Takakura and T. Suzuki, *Approximation and estimation ability of transformers for sequence-to-sequence functions with infinite dimensional input*, in International Conference on Machine Learning,

PMLR, 2023, pp. 33416–33447.
[45] R. E. Turner, *An introduction to transformers*, arXiv preprint arXiv:2304.10557, (2023).
[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, Advances in neural information processing systems, 30 (2017).
[47] T. Wang, Z. Dou, C. Bao, and Z. Shi, *Diffusion mechanism in residual neural network: theory and applications*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 46 (2023), pp. 667–680.
[48] E. Weinan, *A proposal on machine learning via dynamical systems*, Communications in Mathematics and Statistics, 5 (2017), pp. 1–11.
[49] E. Weinan, C. Ma, and L. Wu, *Machine learning from a continuous viewpoint, i*, Science China Mathematics, 63 (2020), pp. 2233–2266.
[50] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, *CvT: Introducing convolutions to vision transformers*, in Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 22–31.
[51] J. Xu, X. Sun, Z. Zhang, G. Zhao, and J. Lin, *Understanding and improving layer normalization*, Advances in neural information processing systems, 32 (2019).
[52] Y. Yang, J. Sun, H. Li, and Z. Xu, *Admm-csnet: A deep learning approach for image compressive sensing*, IEEE transactions on pattern analysis and machine intelligence, 42 (2018), pp. 521–538.
[53] D. Yarotsky, *Error bounds for approximations with deep relu networks*, Neural networks, 94 (2017), pp. 103–114.
[54] Z. Ye, X. Huang, L. Chen, H. Liu, Z. Wang, and B. Dong, *Pdeformer: Towards a foundation model for one-dimensional partial differential equations*, arXiv preprint arXiv:2402.12652, (2024).
[55] T. Young, D. Hazarika, S. Poria, and E. Cambria, *Recent trends in deep learning based natural language processing*, ieee Computational intelligenCe magazine, 13 (2018), pp. 55–75.
[56] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, *Graph transformer networks*, Advances in neural information processing systems, 32 (2019).
[57] B. Zhang and R. Sennrich, *A lightweight recurrent network for sequence modeling*, in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 1538–1548.
[58] D.-X. Zhou, *Universality of deep convolutional neural networks*, Applied and computational harmonic analysis, 48 (2020), pp. 787–794.
[59] J. Zhu, X. Chen, K. He, Y. LeCun, and Z. Liu, *Transformers without normalization*, arXiv preprint arXiv:2503.10622, (2025).
[60] A. Ziaee and E. Çano, *Batch layer normalization a new normalization layer for cnns and rnns*, in Proceedings of the 6th International Conference on Advances in Artificial Intelligence, 2022, pp. 40–49.