2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Approximation of Koopman Operator using Spiking Neural Networks

Arun M. George*, Dighanchal Banerjee*, Titas Bera, Sounak Dey

Abstract— **To analyse, control, and predict the behaviour of the states of a non-linear dynamical system using measurement functions in Hilbert space are a widely used method known as Koopman operator theory. Traditional approaches leverage matrix-based methods or artificial neural networks (specifically encoder-decoder architectures) to approximate Koopman operator. However, such methods necessitate significant power and computational resources, hence may not be suitable for applications that require real-time on-board processing, such as sensor fusion in robotics/autonomous vehicles or adaptive control for drone stabilization. The recent development of brain-inspired spiking neural networks and neuromorphic computing platforms could provide an effective solution, as these offer extremely low-energy computation and real-time responses. In this paper, we introduce the implementation of a Spiking Neural Network (SNN), that can efficiently approximate Koopman operator. Our model, when tested over four systems, demonstrated significant computational savings - up to 4× fewer addition operations and 43× fewer multiplication operations, while using only 20% of the input data compared to its ANN counterpart.**

I. INTRODUCTION

The Koopman operator theory has recently gained traction as a powerful alternative approach for analysing a non-linear dynamical system using the evolution of measurement function. In a seminal work published by B. Koopman in 1931 [1], demonstrated that a non-linear dynamical system can be effectively represented by an infinite-dimensional linear operator acting on a Hilbert space encompassing all possible combination of the system's state. Since then, the method has been used to project the behaviour of non-linear fluid flow [2], specifically quantify transport or mixing in turbulent flows, nonlinear state reconstruction and data fusion, among many others [3].

A fundamental challenge in Koopman operator theory lies in its inherent infinite dimensionality. This stems from the requirement to encompass the space of all possible measurement functions, which necessitates an infinite number of degrees of freedom to fully represent. To overcome this hurdle, the focus of research efforts is on developing finite-dimensional, matrix-based approximations of the Koopman operator. One prominent technique for Koopman operator approximation is Dynamic Mode Decomposition (DMD) [4]. This method aims to identify a low-rank linear model that best approximates the Koopman operator's behaviour in projecting spatial measurements forward in time. However, the reliance solely on linear measurements in DMD can limit its effectiveness in capturing the complexities of inherently nonlinear systems.

While augmenting the measurement space with nonlinear functions has the potential to enrich the resulting model, this approach presents a new challenge. The introduction of non-linearities may compromise a crucial property – closure under the Koopman operator. This lack of closure can hinder the interpretability and effectiveness of the approximated model.

Recently, Artificial Neural Network (ANN) based work has emerged to discover and represent eigenfunctions from data. Normally, Encoder-Decoder Networks [5] are used for this purpose, but these approaches are both compute and power intensive and their memory footprint are quite high - thereby making them unfit for cases such as sensor data fusion for autonomous vehicles or robots, where compute, memory and energy resources are scarce. Moreover, there are cases such as stable and agile manoeuvring of drones in a gust where quick adaptation and understanding of the change in system with a varying degree of environmental disturbances is a must-have requirement and corresponding computation must be done in-situ in the drone.

Off late, Spiking Neural Networks (SNN) has emerged as a new generation of AI paradigm that mimic functionalities of mammalian brain neural networks, process sparse asynchronous events, and are good at learning temporal patterns [6] [7]. When run on neuromorphic computing platforms, that follow the non von Neumann architecture, SNNs offer high energy efficiency [8] [9] which is crucial in control engineering. SNNs can integrate with reinforcement learning mechanisms [10], allowing the system to adapt based on environmental feedback. By integrating environmental dynamics into the reward function, SNNs can learn robust controllers efficiently. Due to these capabilities, our hypothesis is that SNNs can be a good candidate to approximate Koopman operators more efficiently thereby enhancing Koopman-based control systems. Moreover, SNNs offer very low energy consumption when they run on compatible neuromorphic hardware, enabling their implementation in real-world control systems.

In this paper, we designed and implemented SNNs that can approximate Koopman operator and have validated our network performance on different systems. To the best of our knowledge, ours is the first effort to use SNN for approximation of Koopman operator. We found that our model is computationally efficient, and it achieves at par performance in approximating non linear systems compared to state of the art. Notable features of our model are as follows:

*These authors contributed equally to this work
All authors are with TCS Research, Kolkata, India (arunm.george, dighanchal.b, sounak.d, titas.bera)@tcs.com

- Our SNN models can be used for systems with both discrete and continuous eigen spectrum. Moreover, it can approximate Koopman operator with different initial condition of the system.
- It can effectively approximate the Koopman operator even with minimal length of data; it is observed to work with 20% of the original data length. This faster inference even with very small lengths of data potentially saves a lot of time and computational resources.
- Our model is observed to perform up to $4\times$ less addition operations and $43\times$ less multiplication compared to its ANN counterpart.
- And finally, this model is estimated to consume $\sim 10^{-2}$ μJ per inference when run on Intel Loihi neuromorphic board.

The structure of the paper is as follows: Section II provides a review of works related to the approximation techniques of the Koopman Operator. Section III elucidates the theory of the Koopman operator and its approximation using ANN while Section IV gives a concise overview of SNN. In Section V, we delve into the proposed methodology and architecture. Section VI describes the datasets, the experiments conducted, the results obtained, and their analysis. Finally, we conclude in Section VII with a mention of potential future work.

II. Related Works

The Koopman operator can be approximated using several methods. Kernel-Based Approximations is a data-driven approach for estimating stochastic differential equations on reproducing kernel Hilbert spaces (RKHS), while Finite Section Theory achieves same by applying the concept of infinite-dimensional operators on those equations [11]. Dynamic Mode Decomposition (DMD), Extended Dynamic Mode Decomposition (EDMD) and other similar variants are data-driven methods used to approximate the leading eigenvalues, eigenfunctions, and modes of the Koopman operator, [4], [12], [13]. Deep neural networks are also used for learning Koopman eigenfunctions using autoencoder networks [5], [14].

However, there are several gaps in the approximation of the Koopman operator. Problems like Finite-Dimensional Limitations arise as we only have access to finite-dimensional data, thus it is important to study approximation properties of finite-dimensional numerical algorithms. Secondly, the composition operators under study are rarely compact or self-adjoint, posing Operator Properties challenges. Third, Infinite-Dimensional Challenges occur as finite-dimensional approximation of the Koopman operator involves identifying a subspace spanned by a subset of eigenfunctions [15]. Other issues include the non-obvious linearization of non-linear DEs by the Koopman function for closed-loop systems, the compute-heavy and high latency nature of approximating Koopman via ANN, and the computational heaviness of solving n-th order differential equations for real-world systems due to non-linearity.

III. Koopman Operator and its approximation using ANN

A. Koopman Operator Theory

Koopman Operator Theory states that non-linear dynamical systems can be represented as a linear operator in Hilbert space of measurement functions of its state. This linear operator should be able to predict the measurement function in time and tends to be infinite dimensional in nature. One major difference between classical linearization techniques and the Koopman operator method is that while the former is local (i.e. behaves linearly around a fixed set of points), the later tends to be global. For any given continuous dynamical system of the form in Eqn 1, we can describe it as an equivalent discrete-time system in form of a Flow Map, $\mathbf{F_t}$ as shown in Eqn 2,

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)) \tag{1}$$

$$\mathbf{F_t} = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{f}(\mathbf{x}(\tau))d\tau \tag{2}$$

Koopman Operator κ for this system can be defined as shown below:

$$\mathcal{K}_t\mathbf{g} = \mathbf{g} \circ \mathbf{F_t} \Rightarrow \mathcal{K}_t\mathbf{g}(\mathbf{x}(t)) = \mathbf{g}(\mathbf{x}(t+1)) \tag{3}$$

where $\mathbf{g}$ is the Hilbert space measurement function, known as observables. Simplest possible observable function is identity function under some chosen Hilbert space basis, but this becomes more complex with time and associated advancing dynamics. Moreover, Koopman observables are of infinite dimensions and hence computing it and designing a control system based on it becomes intractable. Due to these reasons, we need to find an appropriate finite representation for key measurement functions. One possibility is to try finding a relevant subset of Koopman eigenfunctions, $\phi(\mathbf{x})$ as they can evolve the dynamics in a linear fashion.

B. Koopman Operator approximation using ANN

Traditional approaches for learning Koopman Operator, like DMD, tries to capture the best-fit linear model in a linear observable space. As such, it often fails to capture the desirable non-linear features of the system. ANNs can be used to capture finite-length approximations of non-linear Koopman observable functions in a data-driven manner. Encoder-Decoder Networks are normally used to learn these representations [5], as detailed below:

1) **Encoder**: Encoder block is a multi-layered neural transformation which learns a lifted space representation or as a compressed latent space representation with Reservoir Computers (using RNN) or with Deep Autoencoders respectively. The combined transform performs the change of coordinates as required by a Koopman observable.
2) **Linear Transformations**: Domain knowledge can be incorporated into the ANNs via imposing various constraints on the loss function of the Encoder-Decoder network. For approximating Koopman Operator, the most important constraint is to restrict the latent space to

evolve in a linear fashion (after application of a learned linear transformation). This linear transformation performs the role of the Koopman matrix, which would be the projection of the required Koopman operator in the earlier encoded subspace.

3) **Decoder**: Decoder block performs the inverse transformation of mapping the elements of the Koopman observable space to the original coordinates. It can also be a multi-layered network which can be structured symmetrically to the encoder block.

Even though ANNs are competent in learning accurate representations for Koopman approximation, they also have some major disadvantages which limits their practical uses. Deep ANNs require large amounts of data to learn from during the training time and that might not be possible in certain scenarios like fluid flow dynamics. The computational and power requirements of these networks are often very large, thereby limiting their applicability in situations where power, memory and compute are scarce. For instance, many of the robotic applications require large amount of portability and as such their control systems are limited by power and compute. Learning such data and compute efficient representations for approximating Koopman operator, while leveraging the power of deep networks still remains an open problem.

IV. Spiking Neural Networks (SNN)

SNNs are a new generation of neural network that mimic mammalian brain functionality very closely [6]. Here the behaviour of individual neurons is simulated via a discrete chain of spikes (aka events) over a time span against a given external stimulus. Unlike traditional ANNs that operate based on continuous firing rates, the spiking neurons do not transmit information at each propagation cycle, but they transmit information only when a membrane potential - an intrinsic quality of the neuron related to its membrane electrical charge - reaches a specific threshold value. When the membrane potential reaches the threshold, the neuron fires a voltage surge and a spike is emitted that travels to subsequent neurons in the network and they, in turn, increase or decrease their potentials in response to this signal. This event based asynchronous processing of input is the key feature of SNN and this in turn results into reduction of processing and computation power.

Amongst several computational models of neuron that are used in SNN, the most used and prominent one is the Leaky Integrate-and-Fire (LIF) model. In this model, the momentary activation level (modeled as differential Eqn. 4) is normally considered to be the neuron's state, with incoming spikes pushing this value higher or lower, until the state eventually reaches threshold (or decays down), and finally the neuron fires. After firing, the state variable is reset to a lower value called resting potential. In the above Eqn. 4, V denotes the neuron's membrane potential, which fluctuates based on the input stimuli. When V surpasses a certain threshold, V_{thresh}, a spike s is emitted by the neuron. The resting membrane potential is represented by V_{rest}, while I signifies the cumulative input current to the neuron from all its synapses. The total resistance that I encounters during synaptic transmission is denoted by R. In the absence of any input, V undergoes an exponential decay with time constant τ_m.

$$\begin{aligned} \tau_m \frac{dV}{dt} &= (V_{rest} - V) + IR \\ s &= \begin{cases} 1, & V \geq V_{thresh} \\ 0, & V < V_{thresh} \end{cases} \end{aligned} \tag{4}$$

We've employed the Euler method to discretize Eqn. 4 for the purpose of simulation. The final vectorized form of the equations for a layer of LIF neurons is given by Eqn. 5.

$$\begin{aligned} \mathbf{u}_t &= \mathbf{v}_{t-1} + \mathbf{W}^T \mathbf{s}_t^{in} \\ \mathbf{s}_t^{out} &= \mathscr{H}(\mathbf{u}_t - \mathbf{v}^{thresh}) \\ \mathbf{v}_t &= \alpha \mathbf{u}_t - (\mathbf{v}^{thresh} \odot \mathbf{s}_t^{out}) \end{aligned} \tag{5}$$

At time t, $\mathbf{v}_t$ denotes the membrane potential vectors of the LIF neurons, and $\mathbf{u}_t$ signifies an intermediate potential vector. In Eqn. 4, the incoming input term IR is represented as the dot product of the synaptic weights $\mathbf{W}$ and the incoming input spike vector $\mathbf{s}_t^{in}$. The output spiking activity of the LIF neurons is represented by $\mathbf{s}_t^{out}$, which is computed as the Heaviside Step function ($\mathscr{H}$) of the difference between $\mathbf{u}_t$ and the threshold. The membrane potential at time t decays with a constant decay factor of α, approximated by $e^{\frac{-1}{\tau_m}}$ and $\odot$ symbolizes the Hadamard product. Our discretized LIF equations employ a reset by subtraction method rather than making the membrane potential reset to zero. This reduces some of the information loss that can occur in the vanilla spiking networks by retaining the residual membrane potential as shown in [16].

V. Methodology

We propose an efficient SNN based method for Koopman operator approximation. Our methodology is inspired by the deep neural network architecture mentioned in [5]. The proposed system identifies sparse, binary representations of Koopman eigenfunctions from a given sequence of states of a dynamical system. Spiking Neural Networks have same expressive power as the ANNs [17] and as such can be leveraged to learn complicated non-linear transformations such as Koopman eigenfunctions. The proposed methodology for approximating the Koopman operator using SNN is depicted in Fig. 1. The detailed description about each component of the architecture is given below.

A. Spiking Encoder Block (ϕ_s)

Given a time-varying state vector $\mathbf{x}_t$ with t ranging from $1:T$, we learn a spiking encoder block $\phi_{\mathbf{s}}$, which tries to find the finite approximations of suitable subsets of Koopman eigenfunctions of the system. Instead of simply trying to find a finite dimensional approximation of any measurable function $\mathbf{g}$, we focus on Koopman eigenfunctions ϕ, as they ensure the closure property within the observable space. Along with reduced computations, a spike-based encoder can also impart robustness to the system [18].

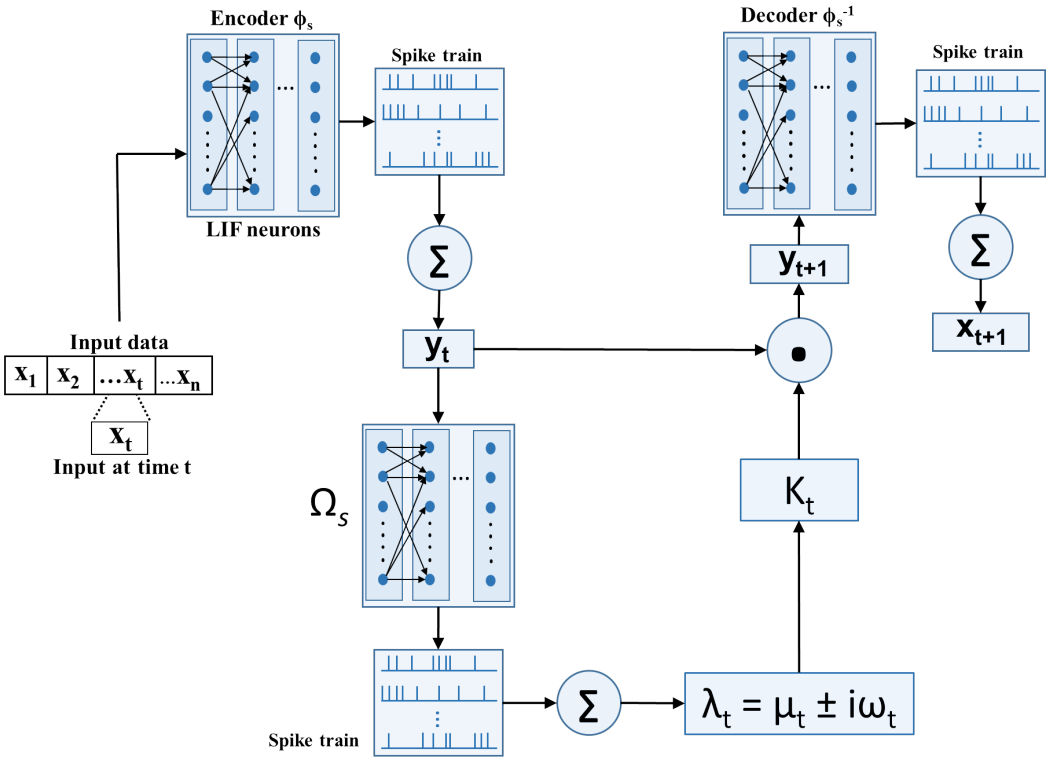

Fig. 1: Architecture diagram of the proposed system for Koopman approximation. The network comprises of an Encoder block, Koopman Block and a Decoder Block. The Koopman Block, Ω_s continuously predicts the complex eigen values based on which the time-varying Koopman Operator Matrix, $\mathbf{K}_t$ is computed.

Encoder block comprises of stacked LIF-neuron layers, with a Direct spike encoding as the first layer, which converts the real valued states ($\mathbf{x_t}$) into spikes, by applying them to the membrane potential of the LIF neurons. The corresponding spike-activity dictates the spike domain representation of the states. Due to the stateful nature, LIF neurons are able to encode the continuous sequential values into sparse spike-train representations retaining only the relevant non-redundant information. The subsequent LIF-neuron layers extract the latent eigenfunction representations in the spike domain. The output spike trains of the encoder block are integrated & converted into real-valued domain (represented as $\mathbf{y}_t$) and then are forwarded to the next block.

B. Spiking Koopman Block (Ω_s)

For capturing the continuous spectrum of eigenvalues, we use another spiking network, Ω_s. This layer continuously estimates the varying eigenvalues of the system from $\mathbf{y}_t$. For enforcing symmetry, we extract the radius of the latent states ($\|\mathbf{y}_t\|_2^2$) and use this as a substitute for estimating the eigenvalues. The Koopman Operator Matrix which linearly evolves the dynamics in the latent space, can then be continuously constructed as a parameterization of these estimated eigenvalues. For each of the complex pair of eigenvalues at time t, $\lambda_t = \mu_t \pm i\omega_t$, the corresponding estimated Koopman Operator Matrix, $\mathbf{K_t}$ will be having a Jordan block of the form:

$$exp(\mu_t \Delta t) \begin{bmatrix} cos(\omega_t \Delta t) & -sin(\omega_t \Delta t) \\ sin(\omega_t \Delta t) & cos(\omega_t \Delta t) \end{bmatrix} \tag{6}$$

C. Spiking Decoder Block (ϕ_s^{-1})

Finally, we also use another spiking network called Decoder Block, ϕ_s^{-1} for performing the inverse transformation from the latent space $\mathbf{y}_t$ to the native state space $\mathbf{x}_t$ of the system. Structure of the decoder block is exactly the same as that of the encoder block, but in reverse order. For Instance, for an encoder block with layer structure $l_1 \times l_2 \times \cdots \times l_n$, the decoder network will be having a layer structure like $l_n \times l_{n-1} \times \cdots \times l_1$.

D. Koopman Learning in Spike Domain

Backpropagation is a common method used in training traditional neural networks, and its application in SNNs is an area of active research [19]. The challenge lies in the fact that SNNs involve discrete spike events, which create discontinuities in the error surface, making it difficult to apply standard backpropagation. However, researchers have developed several methods to overcome this, such as surrogate gradient learning [20], where a differentiable function is used to approximate the discontinuous spiking nonlinearity. This allows the network to leverage the power of gradient-based learning algorithms. In our case, we have used *arctan* as a gradient approximation function. Also, we have used

stochastic gradient descent (SGD) as an optimizer with the loss function as mentioned below.

Loss Function: We use a custom loss function to accurately learn the Koopman operator and observables. The loss can be divided into 3 major components and associated regularizations. The loss function can be written as shown below:

$$\mathscr{L} = \beta_1 L_{recon} + \beta_2 L_{pred} + \beta_3 L_{lin} + \beta_4 L_{reg} \tag{7}$$

L_{recon} corresponds to the reconstruction loss of the Spiking Encoder-Decoder network, which is the Mean Squared Error between the input state value $\mathbf{x_t}$ and the reconstructed output from spike decoder, $\phi_s^{-1}(\mathbf{y_t})$. In addition to this, to ensure that the observable space that we learn is indeed a Koopman eigenspace, we enforce a prediction loss component L_{pred} and a linearity loss component L_{lin}. Linearity component ensures that the Koopman Matrix $\mathbf{K_t}$ evolves the dynamics in the latent space in a linear fashion. The prediction loss ensures that future states can be accurately predicted from the evolved latent space for a time window, T_{pred}. In all our experiments, T_{pred} is taken as half of the T. Each of these components are then combined together with corresponding weight factors (β_i) to form the final loss $\mathscr{L}$.

$$\begin{aligned}\mathscr{L} = \beta_1 \|\mathbf{x_t} - \phi_s^{-1}(\phi_s(\mathbf{x_t}))\|_2 + \\ \beta_2 \frac{1}{T_{pred}} \sum_{i=1}^{T_{pred}} \|\mathbf{x}_{t+i} - \phi_s^{-1}(\mathbf{K}^i \phi_s(\mathbf{x}_t))\|_2 + \\ \beta_3 \frac{1}{T-1} \sum_{i=1}^{T-1} \|\phi_s(\mathbf{x}_{t+i}) - \mathbf{K}^i \phi_s(\mathbf{x}_t)\|_2 + \beta_4 L_{reg}\end{aligned} \tag{8}$$

In the above equation, $\mathbf{K}^i$ refer to the combined Koopman Operator Matrix, which can be defined as $\mathbf{K}^i = \prod_{j=1}^{i} \mathbf{K}_j$. To improve the performance, we additionally penalize the contributor of maximum error during the reconstruction and next-timestep prediction. To balance out and prevent overfitting, we also use an L_2 regularization on the parameters of the Encoder-Decoder network ($\mathbf{W}$). These two regularizations constitute the L_{reg} term and it can be written as follows:

$$\begin{aligned}L_{reg} = \|\mathbf{x}_t - \phi_s^{-1}(\phi_s(\mathbf{x}_t))\|_\infty + \\ \|\mathbf{x}_{t+1} - \phi_s^{-1}(\mathbf{K}\phi_s(\mathbf{x}_t))\|_\infty + \|\mathbf{W}\|_2^2\end{aligned} \tag{9}$$

VI. Results and Discussion

To support our assertions, we conducted experiments using four distinct datasets - two with discrete eigen spectrum and other two with continuous eigen spectrum.

1) Datasets with Discreet Eigen Spectrum: To create such datasets, we have considered first a simple nonlinear system, followed by a spacecraft rendezvous problem.

Case I - Discrete Spectrum: This is a simple nonlinear system with a single fixed point and a discrete eigenvalue spectrum as represented by Eqn. 10.

$$\begin{aligned}\dot{x_1} &= c_1 x_1 \\ \dot{x_2} &= c_2(x_2 - x_1^2)\end{aligned} \tag{10}$$

The dataset for the discrete spectrum is generated using random initial conditions, where x_1 and x_2 are in the range $[-0.5, 0.5]$ and $c_1 = -0.05$ and $c_2 = -1$. This specific region of the phase space is adequate to encapsulate the dynamics. The Koopman embedding is used to identify nonlinear coordinates that flatten this inertial manifold, providing a globally linear representation of the dynamics.

Case II- Spacecraft Rendezvous Problem: As another instance of system with discrete eigenstructure, we examine the problem of Low-Earth-Orbit rendezvous between two spacecrafts, considered as a 'target' and a 'chaser', in a circular orbit. The relative dynamics between the two spacecraft can be applied using Euler-Lagrange formulation to the Lagrangian of the overall system $\mathbf{L} = \mathbf{T} - \mathbf{U}$ where $\mathbf{T}$ is the kinetic energy and $\mathbf{U}$ is the potential energy can be expanded as $\mathbf{U} = -\frac{\mu}{a}\sum_{i=0}^{\infty} P_k(\cos\alpha)(\frac{\rho}{a})^k$, where α is the relative angle between the target and the chaser, P_k are the k^{th} order Legendre polynomials, ρ is the relative distance between the two spacecrafts, μ is a gravitational parameter, and a is the radius of the circular orbit. Consideration of the first three potentials in the expansion of $\mathbf{U}$, leads to the linear Hill-Clohessy-Wiltshire (HCW) [21] equations as described in Eq.(11), for a $2-D$ engagement scenario,

$$\begin{aligned}\ddot{x} &= 3n^2 x + 2n\dot{y} \\ \ddot{y} &= -2n\dot{x}\end{aligned} \tag{11}$$

where, $[x, y]^T$ is the relative separation between two spacecraft defined in a moving LVLH (local vertical local horizontal) coordinate system centered around the target spacecraft. In these equations, the term n is defined as $\sqrt{\frac{\mu_g}{a^3}}$, where a is the length of the semi-major axis $(= 6793137)$, indicative of a low earth orbit), and μ_g is a constant, valued at 3.986×10^{14}. When the magnitude of the relative position and velocity is substantial, a linear approximation of the HCW equations cannot fully describe the system and consideration of higher orders (greater than 3) in $\mathbf{U}$ are necessary leading to a non-linear system of equations.

To test the efficacy of our Koopman approximation method under various initialization conditions, we simulated the above equations perturbed with a 4^{th} order gravitational potential term, under random initial positions and having random initial velocities. We generated 5000 such random trajectories for training the Koopman operator network where initial relative coordinates were sampled uniformly from $[-100, 100]$ and initial relative velocities were sampled from $[-3.0, 3.0]$. To check the generalization with respect to initial conditions, a separate testing dataset of 2000 trajectories was generated from a broader range of initialization values, with positional values sampled from $[-150, 150]$ and velocity values from $[-5.0, 5.0]$.

2) Datasets with Continuous Eigen Spectrum: : To create such datasets, we have considered a nonlinear pendulum system and a high-dimensional unsteady fluid flow around a cylinder as these exhibit continuous spectra of eigen values.

Case I - Non-linear Pendulum: Here we examine a nonlinear pendulum that has a continuous eigenvalue spectrum as shown in the following,

$$\begin{aligned}\dot{x_1} &= x_2 \\ \dot{x_2} &= -\sin(x_1)\end{aligned} \tag{12}$$

The dataset for this pendulum is generated using random

initial conditions, where x_1 in the range $[-3.1, 3.1]$, slightly less than the range $[-\pi, \pi]$, and x_2 varies between $[-2, 2]$.

Case II - High-dimensional nonlinear fluid flow: Next we examined the nonlinear fluid flow that occurs around a circular cylinder as a bluff body [5]. The high-dimensional dynamics of this system evolve on a low-dimensional attractor, represented by a slow manifold as shown in the Eqn. 13.

$$\begin{aligned} \dot{x_1} &= c_1 x_1 - c_2 x_2 + A x_1 x_3 \\ \dot{x_2} &= c_2 x_1 + c_1 x_2 + A x_2 x_3 \\ \dot{x_3} &= -c_3 (x_3 - x_1^2 - x_2^2) \end{aligned} \tag{13}$$

where $\{x_1, x_2, x_3\}$ are the dimensional representation, and $\{c_1, c_2, c_3, A\}$ are constants. The fluid flow problem, when confined to the slow manifold, is derived from random initial conditions x on the bowl with radius r. Here we transform the Cartesian co-ordinates to polar co-ordinates by the substitution $x_1 = r\cos\theta$, $x_2 = r\sin\theta$, and $x_3 = (x_1^2 + x_2^2)$, where r lies in the range [0, 1.1], θ is within $[0, 2\pi]$. This encapsulates all the dynamics on the slow manifold, which are composed of trajectories spiraling towards the limit cycle at $r = 1$. *This is referred as* **Fluid Flow 1** *in the later sections*.

The fluid flow problem, when extended beyond the slow manifold, is derived from random initial conditions x where x_1 lies in the range $[-1.1, 1.1]$, x_2 is within $[-1.1, 1.1]$, and x_3 is within $[0, 2.42]$. These boundaries are selected to encompass the dynamics on the slow manifold covered by the preceding dataset, as well as trajectories that originate off the slow manifold. Any trajectory that expands to $x_3 > 2.5$ is discarded to ensure the domain remains reasonably compact and well-sampled. *This is referred as* **Fluid Flow 2** *in the later sections*. For both the cases we have used c_1 = 0.1, c_2 = 1, c_3 = 10 and A = -0.1.

A. Experimental Setup:

The proposed SNN network was simulated using PyTorch and SNNTorch [22]. The network architecture for each of the datasets is given in Table I. Only the encoder architecture (ϕ_s) is provided as the spiking decoder (ϕ_s^{-1}) is always symmetrical to the encoder structure as described in section V-C. For each of the above mentioned datasets, we tried to come up with the optimal hyper-parameters by means of a randomized grid search. In all our experiments, the decay factor α is set to 0.65 and the optimal v^{thresh} is set to 0.5. The number of epochs and the learning rate was fixed at 100 and 0.001 respectively in all our experiments. The β-values used for learning the Koopman operator is given in the Table II. The discretized time step Δt is kept as 0.02 for all the experiments.

TABLE I: Network architectures of the proposed Encoder block and Koopman block for various datasets. (FC = Fully Connected layer & associated dimensions are in numbers.)

	ϕ_s	Ω_s
Discrete Spectrum	2-FC-160-FC-160-FC- 2	1-FC-90-FC-90-FC-90-FC-1
Non-Linear Pendulum	2-FC-80-FC-80-FC- 2	1-FC-170-FC-2
Fluid Flow 1	3-FC-130-FC-3	1-FC-20-FC-20-FC-2,
Fluid Flow 2	3-FC-105-FC-2	1-FC-300-FC-2

TABLE II: β-values used during the training of the SNN

	Discrete Spectrum	**Non-Linear Pendulum**	**Fluid Flow 1**	**Fluid Flow 2**
β_1	1E-1	1E-3	1E-1	1E-1
β_2	1E-1	1E-3	1E-1	1E-1
β_3	1E0	1E-1	1E0	1E0
β_4	1E-8	1E-9	1E-8	1E-9

B. Results

A comparative analysis of the Approximation performance between ANN and SNN Koopman models was conducted and the results are shown in Table III. The approximation performance is quantified by the Mean Error, combining the first 3 components of the Loss function $\mathscr{L}$ during inference. From the data in the table, it is evident that the ANN model's performance in approximating the Koopman Operator is optimal when dealing with larger data lengths. This suggests that the ANN model's ability to accurately approximate the Koopman Operator is heavily dependent on the size of the data it is processing. In contrast, our SNN model demonstrates a consistent error rate when approximating the Koopman Operator, regardless of the trajectory length. *This indicates that the SNN model's performance is not significantly affected by the size of the data, making it a more reliable choice for diverse data sizes.*

Moreover, the SNN model exhibits a faster learning of dynamics, even when the sample length is as short as 21 or 11. This highlights the SNN model's superior adaptability and efficiency in learning from smaller data sets. *However, the ANN model's performance significantly deteriorates when dealing with smaller data lengths.* This degradation in performance underscores the limitations of the ANN model in handling smaller data sets and its dependency on larger data lengths for optimal performance. It is to be mentioned that we observed one outlier to this in case of the non-linear pendulum dataset, where ANN shows superior performance over SNN irrespective of the length of the data. The outlier behaviour could be a statistical happenstance but nevertheless warrants some further study. For any practical purposes, this behaviour can be overlooked as the difference in test error is only in the order of 1E-4 and can be less significant for many of the control problems.

As mentioned previously, we further test the generalization of our Koopman approximation method under various initialization conditions of the spacecraft rendezvous problem. The performance comparison between the ANN and SNN models is shown in Table IV. *The Test error of SNN is lower than that of the ANN, implying a better generalization in the range beyond the training data distribution*. The mean Koopman operator for the spacecraft rendezvous problem, when estimated using the ANN and the SNN, show similar eigenvalues at $[0.9997 \pm 0.0139i]$ which is consistent with

TABLE III: Performance Comparison between ANN and SNN for Koopman approximation on multiple datasets

		ANN			SNN		
Dataset Name	**Length of data**	**Train Error**	**Val Error**	**Test Error**	**Train Error**	**Val Error**	**Test Error**
Discrete Spectrum	51	1.65E-7	1.87E-7	**1.87E-7**	4.73E-4	4.42E-4	4.97E-4
	41	2.35E-3	2.79E-3	2.99E-3	4.68E-4	4.74E-4	**4.75E-4**
	31	1.24E-3	1.76E-3	1.76E-3	4.57E-4	4.79E-4	**4.98E-4**
	21	6.09E-4	1.07E-3	1.07E-3	5.36E-4	9.90E-4	**9.42E-4**
	11	2.86E-4	7.26E-4	8.11E-4	2.05E-4	4.09E-4	**4.11E-4**
Non-Linear Pendulum	51	8.30E-6	1.11E-5	**1.30E-5**	5.85E-4	5.69E-4	5.34E-4
	41	4.39E-4	7.02E-4	7.19E-4	5.63E-4	7.23E-4	7.74E-4
	31	2.07E-4	4.09E-4	3.84E-4	6.12E-4	6.34E-4	6.57E-4
	21	9.13E-5	2.30E-4	2.66E-4	6.71E-4	6.89E-4	7.14E-4
	11	3.87E-5	1.74E-4	1.72E-4	7.18E-4	7.39E-4	7.56E-4
Fluid Flow1	101	7.00E-6	7.61E-6	**7.53E-6**	7.73E-4	7.84E-4	7.95E-4
	81	7.19E-3	1.11E-2	1.09E-2	3.49E-4	3.64E-4	**3.53E-4**
	61	7.37E-3	1.10E-2	1.10E-2	3.65E-4	3.63E-4	**3.66E-4**
	41	5.43E-3	1.07E-2	1.07E-2	1.57E-4	1.62E-4	**1.58E-4**
	21	1.08E-3	4.25E-3	4.28E-3	3.39E-4	3.60E-4	**3.44E-4**
Fluid Flow2	121	5.22E-7	6.12E-7	**6.27E-7**	4.27E-4	4.22E-4	4.29E-4
	96	3.66E-3	5.78E-3	5.91E-3	3.73E-4	3.57E-4	**3.80E-4**
	72	2.28E-3	5.51E-3	5.46E-3	3.79E-4	3.63E-4	**3.72E-4**
	48	2.37E-3	5.24E-3	5.27E-3	3.61E-4	3.62E-4	**3.60E-4**
	24	2.39E-4	1.96E-3	2.01E-3	3.18E-4	3.05E-4	**3.07E-4**

the observations that an inclusion of the higher order gravitational potential terms leads to a divergent relative separation trajectory for the uncontrolled system. However, SNN based approach operates on lesser number of past training window data, when compared to ANN, making it more adaptive to the system changes.

TABLE IV: A comparison between ANN and SNN approaches for spacecraft rendezvous dataset

	Train Error	**Val Error**	**Test Error**
ANN	8.63E-4	8.80E-4	8.94E-4
SNN	6.05E-4	6.05E-4	**5.95E-4**

We also have evaluated the predictive performance of our Koopman approximation method using the Fluid Flow 1 dataset. We attempted to evolve the states of a random trajectory solely from its initial states using the SNN. The original trajectory and the prediction comparisons between ANN and SNN are depicted in Fig. 2a and Fig. 2b, respectively. As illustrated in the figures, *the difference in prediction quality between ANN and SNN approximations is minimal, and in both cases, the predicted trajectory successfully mirrors the original dynamics*. We also attempted to visualise the latent space representations learned by both networks. Fig. 2c shows the latent space representation generated by the ANN for the aforementioned Fluid Flow trajectory. It learns an observable space that closely resembles the original trajectory. The actual latent representations from the SNN, which are spike trains, cannot be visualised in the same manner as that of ANN. Therefore, the visualisation we present is the one that has been decoded following the integration operation. The SNN representations exhibit a stark contrast (refer to Fig. 2d), they do not maintain any structure of the original trajectory. This is an anticipated outcome as we are transitioning states from the real-valued domain to a discrete and discontinuous spike domain. Despite this, it can still

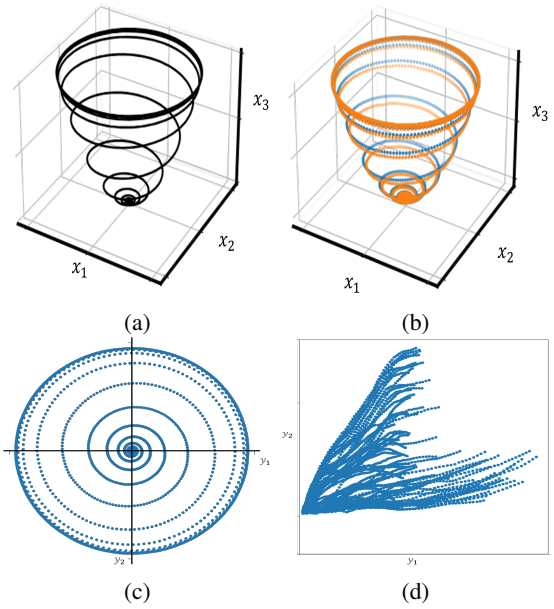

Fig. 2: Visualisation of Koopman Approximation System: (a) A sample trajectory of the 3-dimensional Fluid-Flow data; (b) The predicted trajectory estimated by both ANN (blue curve) and SNN (orange curve) for the sample trajectory data; (c) ANN representation of the latent space; (d) SNN representation of the latent space

predict the original dynamics consistently with minimal error degradation.

The suggested approach delivers at par accuracy with existing methods, but the true advantage lies in the reduced computational effort it requires. As indicated in Table V, *across all datasets, the average number of additions during inference is 2-4× lower than that in an ANN*. However, the key distinction becomes apparent when comparing the

average number of multiplications performed by both the ANN and SNN models. *For the SNN model, this number is significantly lower (6-43×) than that of the ANN*, which is a result of the inherent sparsity of the SNN. In our comparison, network architecture of the SNN is strictly kept same as that of the ANN. All the traditional compression techniques like quantization, pruning, distillation etc. which reduces the number of operations for ANN, can also be applied on top of our network for further reduction in computations.

TABLE V: Comparison of computation effort per inference between ANN and SNN

	# of Addition		# of Multiplication		Estimate on Loihi for SNN	
Dataset	**ANN**	**SNN**	**ANN**	**SNN**	**SOP/ts**	**Energy (μJ)**
Discrete Spectrum	4.3M	1.2M	4.3M	0.1M	96	0.036
Non-linear Pendulum	0.7M	0.3M	0.7M	51K	74	0.021
Fluid Flow 1	0.2M	0.1M	0.3M	33K	57	0.012
Fluid Flow 2	0.2M	95K	0.2M	30K	51	0.024

In SNNs, the total number of operations is typically quantified by a metric known as Synaptic Operations (SOP) [9]. SOP is defined as the average number of spikes per timestep, taking into account all training and inference activities. As can be observed from the sixth column of Table V, the number of SOPs per timestep per inference is quite small, attributable to the optimised spiking activity within the SNN layers. The power requirements of an SNN are directly proportional to the total number of SOPs performed during its operation. The final column of the Table V provides an estimate of the energy consumption when the system is implemented on Intel's neuromorphic chip, Loihi [9]. The estimated energy consumption per inference, measured in microJoules (μJ), has been computed and is sufficiently low to align with the power budget of edge devices.

VII. CONCLUSIONS AND FUTURE WORKS

Real time approximation of Koopman operator in-situ at edge devices such as drones, satellites etc are crucial in order to adapt and understand the behaviour of underlying non-linear dynamical system. In this paper, we have shown for the first time that how SNN can be efficiently used to address this problem. Based on the promising results of our simulation-based experiments, we plan to test the effectiveness of this method on other varieties of complex non-linear systems to further substantiate our assertions. Additionally, this paper suggests a potential power advantage which we aim to validate [23] through testing in neuromorphic hardware such as BrainChip Akida [24] and Intel Loihi [9].

REFERENCES

[1] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[2] H. Arbabi, M. Korda, and I. Mezić, "A data-driven koopman model predictive control framework for nonlinear partial differential equations," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6409–6414.

[3] W. A. Manzoor, S. Rawashdeh, and A. Mohammadi, "Vehicular applications of koopman operator theory—a survey," *IEEE Access*, vol. 11, pp. 25 917–25 931, 2023.

[4] K. K. Chen, J. H. Tu, and C. W. Rowley, "Variants of dynamic mode decomposition: boundary condition, koopman, and fourier analyses," *Journal of nonlinear science*, vol. 22, pp. 887–915, 2012.

[5] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, p. 4950, 2018.

[6] F. Ponulak and A. Kasinski, "Introduction to spiking neural networks: Information processing, learning and applications." *Acta neurobiologiae experimentalis*, vol. 71, no. 4, pp. 409–433, 2011.

[7] S. Dey, D. Banerjee, A. M. George, A. Mukherjee, and A. Pal, "Efficient time series classification using spiking reservoir," in *International Joint Conference on Neural Networks (IJCNN) (Padua: IEEE)*. IEEE, 2022, pp. 1–8.

[8] D. Banerjee, S. Dey, and A. Pal, "An snn based ecg classifier for wearable edge devices," in *NeurIPS 2022 workshop on learning from time series for health*, 2022.

[9] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.

[10] M. Akl, D. Ergene, F. Walter, and A. Knoll, "Toward robust and scalable deep spiking reinforcement learning," *Frontiers in Neurorobotics*, vol. 16, p. 1075647, 2023.

[11] I. Mezić, "On numerical approximations of the koopman operator," *Mathematics*, vol. 10, no. 7, p. 1180, 2022.

[12] S. L. Brunton, "Notes on koopman operator theory," *Universität von Washington, Department of Mechanical Engineering, Zugriff*, vol. 30, 2019.

[13] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, pp. 1307–1346, 2015.

[14] V. Zinage and E. Bakolas, "Neural koopman lyapunov control," *Neurocomputing*, vol. 527, pp. 174–183, 2023.

[15] A. Salova, J. Emenheiser, A. Rupe, J. P. Crutchfield, and R. M. D'Souza, "Koopman operator and its approximations for systems with symmetries," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 9, 2019.

[16] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 294078, 2017.

[17] S.-Q. Zhang and Z.-H. Zhou, "Theoretically provable spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 19 345–19 356, 2022.

[18] S. Sharmin, N. Rathi, P. Panda, and K. Roy, "Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*. Springer, 2020, pp. 399–414.

[19] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 228000, 2016.

[20] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[21] M. Bando and A. Ichikawa, "In-plane motion control of hill–clohessy–wiltshire equations by single input," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 5, pp. 1512–1522, 2013.

[22] J. K. Eshraghian, M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023.

[23] C. Kadway, S. Dey, A. Mukherjee, A. Pal, and G. Bézard, "Low power & low latency cloud cover detection in small satellites using on-board neuromorphic processors," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–8.

[24] "Brainchip unveils the akidatm development environment." *https://www.brainchipinc.com/news-media/press-releases/detail/61/brainchip-unveils-the-akida-development-environment*, 2019.