END-TO-END LEARNING FOR STATE ESTIMATION IN NONLINEAR DYNAMICAL SYSTEMS

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF SCIENCE
IN
APPLIED MATHEMATICS

Submitted by

ADITYA KUMAR ABROL

(2K23/MSCMAT/03)

Under the supervision of

Dr. Nilam

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

MAY, 2025

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Aditya Kumar Abrol, Roll No- 2K23/MSCMAT/03 student of M.Sc Applied Mathematics,hereby declare that the project Dissertation titled "End-to-End Learning for State Estimation in Nonlinear Dynamical Systems" which is submitted by me to the Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Sciences, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi Aditya Kumar Abrol

Date:

This is to certify that the student has incorporated all the corrections suggested by the examiners in the dissertation and the statement made by the candidate is correct to the best of my knowledge.

Signature of Supervisor **Signature of External Examiner**

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "End-to-End Learning for State Estimation in Nonlinear Dynamical Systems" which is submitted by Aditya Kumar Abrol, Roll No– 2K23/MSCMAT/03, Department of Applied Mathematics ,Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Science, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi — Dr. Nilam

Date: — **SUPERVISOR**

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I wish to express my sincerest gratitude to Dr Nilam for her continuous guidance and mentorship that she provided me during the project. She showed me the path to achieve my targets by explaining all the tasks to be done and explained to me the importance of this project as well as its industrial relevance. She was always ready to help and clear my doubts regarding any hurdles in this project. Without her constant support and motivation, this project would not have been successful.

Place: Delhi Aditya Kumar Abrol

Date:

Abstract

This paper explores an end-to-end learning framework for state estimation that leverages deep learning techniques to directly infer latent system states from observational data. State estimation in nonlinear dynamical systems is a critical task across numerous scientific and engineering domains. Reliable state estimation in nonlinear dynamical systems is critical for defense applications such as missile guidance, UAV navigation, and real-time control of hypersonic vehicles—where classical filters often fall short due to nonlinearity, unmodeled dynamics, and sensor noise. This paper presents an end-to-end learning framework that combines Physics-Informed Neural Networks (PINNs) and Deep Koopman Operators for accurate and robust state estimation without relying on explicit system models. PINNs incorporate known physical laws—such as conservation of momentum and energy—into the loss function, while Koopman-based models learn interpretable linear embeddings of nonlinear systems. It demonstrate our approach on simulated high-speed flight dynamics and benchmark systems like the Lorenz attractor and Van der Pol oscillator, under both ideal and noisy conditions. Results show that our method outperforms traditional approaches like the Extended Kalman Filter in both accuracy and stability, highlighting its potential for deployment in mission- critical DRDO systems requiring real-time, data-driven control and decision-making. Experimental results demonstrate that our hybrid learning approach achieves higher accuracy and greater robustness compared to classical filtering techniques. These results indicate strong potential for real-time deployment in sensor fusion, UAV navigation, and adaptive control of nonlinear systems.

Contents

List of Tables

List of Figures

List of Symbols

x_t	Hidden (latent) state at time t
y_t	Observation or measurement at time t
$w_t^{(i)}$	Weight of the i^{th} particle at time t
$N_{\mathbf{eff}}$	Effective sample size in Particle Filter
h_t	Hidden state in LSTM at time t
c_t	Cell state in LSTM
K	Koopman operator matrix
$\psi(x)$	Observable function in Koopman theory
$\mathcal{L}$	Total loss in Physics-Informed Neural Networks (PINNs)
$\mathcal{L}_{\mathbf{physics}}$	Physics-based loss component in PINNs
$\hat{x}_{t\|t}$	Updated state estimate from Kalman filter
K_t	Kalman gain

Chapter 1

INTRODUCTION

1.1 General Definitions

A non-linear dynamical system is a system where the output is not directly proportional to the input given to the system. These systems often exhibit complex behaviors like chaos, bifurcations, and self-organization. They can appear chaotic, unpredictable, or counterintuitive. But, their behaviour is not random. They are an active area of research in physical sciences. State estimation in nonlinear dynamical systems involves estimating the system's state (variables) using measurements and a model of the system's dynamics, which can be challenging due to the non-Gaussian nature of the posterior state distributions. State estimation is crucial in control and decision-making because it allows for the accurate monitoring and prediction of system behaviour, enabling informed decisions and proactive control actions, especially in complex systems where not all variables are directly measurable. Nonlinear dynamical systems are ubiquitous across various scientific and engineering domains, describing the evolution of state variables where the relationships are inherently non-proportional [1].

Unlike linear systems, which obey the principles of superposition and homogeneity, nonlinear systems exhibit a rich array of complex behaviors, including multiple equilibrium points, sustained oscillations known as limit cycles, qualitative changes in system dynamics termed bifurcations, and unpredictable, yet deterministic, motion referred to as chaos [2] A nonlinear dynamical system is defined as a mathematical model where the rate of change of the system's state variables is not a linear function of the current state 1. This implies that the system's behavior cannot be predicted by simply scaling or superposing individual responses to different inputs [1]. Real-world systems across various disciplines, including physics, biology, engineering, and economics, frequently exhibit nonlinear behavior [1] End-to-end learning is often implemented using deep learning techniques, particularly neural networks like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). A model is trained to map raw inputs to desired outputs using a large amount of labelled data. The model learns to extract useful features from the data and to use these features to make predictions. It is a key concept in deep learning, streamlines the learning process by training a model to map raw inputs directly to desired outputs, eliminating the need for manual feature engineering and improving overall system performance. It simplifies the learning process by allowing the model to learn all the necessary steps from raw input to final output simultaneously, rather than requiring separate stages for feature extraction and prediction.

Definition of State Estimation in Nonlinear Systems State estimation is the process

of determining the internal states of a nonlinear dynamical system using noisy, indirect, or incomplete measurements. Since many real-world systems (e.g., robotics, weather, and biological systems) exhibit nonlinear dynamics, accurately estimating their states is essential for control, prediction, and decision-making.

1.2 Challenges in Nonlinear State Estimation

1.2.1 Challenges of Traditional Methods for State Estimation

State estimation in nonlinear dynamical systems is complicated by several factors:

- **Nonlinearity:** Traditional linear filters like the Kalman Filter (KF) fail in the presence of strong nonlinear dynamics.
- **Noise and Uncertainty:** Real-world systems are subject to unknown disturbances and non-Gaussian noise.
- **High-Dimensional States:** Estimating states in large-scale systems introduces significant computational challenges.
- **Sparse or Indirect Measurements:** Observations may be noisy, intermittent, or only indirectly related to the true system state.

Common solutions to these challenges include:

- **Extended Kalman Filter (EKF):** Linearizes the system around the current estimate using a first-order Taylor expansion.
- **Unscented Kalman Filter (UKF):** Uses a set of sigma points to better approximate the propagation of uncertainty.
- **Particle Filter (PF):** Employs sampling-based techniques to approximate the posterior distribution in complex settings.
- **Deep Learning Methods:** Models like LSTMs, Koopman Networks, and PINNs learn to estimate states directly from data.

Limitations of Classical Filters

Extended Kalman Filter (EKF): The EKF uses a first-order Taylor series to linearize the nonlinear system about the current estimate. While computationally efficient, it suffers from degraded accuracy in highly nonlinear settings and may diverge if the system is non-differentiable or the initial estimate is poor.

Example: In mobile robotics, the EKF often fails to track sharp maneuvers due to poor linearization.

Unscented Kalman Filter (UKF): The UKF improves on the EKF by using deterministic sampling (sigma points) to better capture the distribution of nonlinear transformations. However, its computational burden grows rapidly with the state dimension, and performance is sensitive to tuning parameters like process and observation noise covariances.

Example: In radar tracking with a high number of state variables, UKF offers better accuracy than EKF but at the cost of real-time feasibility.

Particle Filter (PF): PFs handle highly nonlinear and non-Gaussian systems by approximating the posterior distribution using a set of weighted particles. They are highly flexible but suffer from the curse of dimensionality—requiring a prohibitive number of particles in high-dimensional spaces.

Example: In SLAM (Simultaneous Localization and Mapping) for autonomous vehicles, PFs can become computationally infeasible due to the need for millions of particles to maintain accuracy.

1.3 Advantages of End-to-End Deep Learning Methods

Modern deep learning techniques address many of the above challenges:

- **Model-Free Learning:** Approaches such as LSTMs and Transformers learn to estimate states directly from data without explicit models.
- **Handling High Dimensionality:** Koopman Networks learn linear representations in latent spaces, making them suitable for large systems.
- **Robustness to Noise:** Bayesian Neural Networks (BNNs) and Variational Autoencoders (VAEs) capture non-Gaussian uncertainty in sensor data.
- **Irregular Observations:** Attention mechanisms and Graph Neural Networks (GNNs) manage sparse and asynchronous measurements effectively.
- **Computational Efficiency:** GPU acceleration enables real-time inference, often unachievable with PFs.

Example: Deep learning models enable robust state estimation for autonomous drones operating in contested environments, where GPS signals are jammed or spoofed.

These advantages make deep learning a compelling alternative for state estimation in complex, nonlinear, and adversarial environments.

1.3.1 Background on Extended Kalman Filter (EKF)

The EKF predicts the next state using a nonlinear process model and linearizes the dynamics around the current estimate using the Jacobian matrix. This linearized model is then used to perform the Kalman Filter update steps.

The EKF remains popular in real-time systems like GPS navigation, aerospace, and robotics due to its computational efficiency. However, it assumes Gaussian noise and can become unstable when faced with strong nonlinearities or poor initial estimates.

The need to recompute Jacobians at every time step also makes the EKF sensitive to modeling errors and unsuitable for systems with discontinuous dynamics. In such cases, more robust filters such as the UKF or PF—or learning-based alternatives—are preferred. Unscented Kalman Filter (UKF) The UKF addresses the shortcomings of the EKF by using the Unscented Transform to propagate a set of sigma points through the nonlinear system instead of linearizing it. These sigma points are selected to capture the mean and covariance of the state distribution up to the second order, which leads to better performance in nonlinear settings. Unlike the EKF, the UKF does not require the computation

of Jacobians, making it more suitable for systems with complex or unknown derivatives. It is especially effective in applications such as autonomous vehicles, spacecraft navigation, and nonlinear control systems, where moderate nonlinearity and uncertainty are present. Nevertheless, the UKF still assumes that the noise in the system is Gaussian and that the state distribution is approximately unimodal. It is more computationally demanding than the EKF, particularly in high-dimensional state spaces. When the posterior becomes multimodal or skewed, UKF may produce biased estimates or fail to capture the full uncertainty of the system.

1.3.2 Particle Filter (PF)

The Particle Filter is a non-parametric, fully Bayesian approach to state estimation that approximates the posterior distribution using a set of discrete weighted samples, or particles. Each particle represents a hypothesis of the system state, and their weights are updated according to how well they match the observed data.

PFs are highly flexible and can handle nonlinear, non-Gaussian, and even multimodal distributions. They are often used in applications such as object tracking, fault detection, and SLAM (Simultaneous Localization and Mapping). The particles are propagated using the system's transition model and resampled based on their likelihood to reflect the updated belief about the state.

Despite their versatility, Particle Filters suffer from several practical challenges. A large number of particles is often needed to accurately approximate the posterior, especially in high-dimensional problems, leading to high computational cost. Additionally, particle degeneracy can occur, where after several iterations, most particles carry negligible weight. This is addressed using resampling techniques, which themselves can introduce sample impoverishment if not handled carefully.

1.4 Machine Learning Approaches for Dynamical Systems and Time-Series Modeling

- **Long Short-Term Memory (LSTM) Networks:** Used for modeling and forecasting temporal sequences by capturing long-range dependencies. Suitable for nonlinear and nonstationary time-series data.
- **Transformers for Time-Series:** Attention-based models that allow parallel processing of sequential data and excel at capturing global dependencies, making them effective for complex time-series prediction tasks.
- **Koopman Operator Learning:** A framework to represent nonlinear dynamical systems in a linear, infinite-dimensional function space using data-driven methods. Koopman-based models enable linear prediction of nonlinear systems and can be learned via neural networks.
- **Physics-Informed Neural Networks (PINNs):** Incorporate physical laws described by differential equations directly into the loss function of neural networks. Useful for learning solutions to PDEs and ensuring model predictions remain physically consistent.

1.5 Defense Applications of End-to-End Learning in Nonlinear Dynamics

State estimation is a cornerstone for the functionality of autonomous vehicles and robotics within the defense sector, providing the essential information needed for critical tasks such as navigation, control, and comprehensive situational awareness. End-to-end learning methodologies offer promising avenues for significantly enhancing the performance of these sophisticated systems.

In the realm of Unmanned Aerial Vehicles (UAVs), end-to-end learning techniques are actively being explored to elevate the accuracy and robustness of state estimation, particularly in challenging operational environments such as those where GPS signals are either denied or unreliable, or during the execution of complex flight maneuvers. For example, an innovative online end-to-end learning method has been proposed for proactive state estimation of UAVs. This method integrates refined UAV dynamics, modeled using LSTM-based Recurrent Neural Networks (RNNs), with Kalman filter-based state estimation that incorporates active learning of noise parameters. This integrated approach has demonstrated the capability to achieve high levels of accuracy in state estimation by leveraging real-time flight data. The reported root-mean-square (RMS) state prediction errors for position and orientation using this online end-to-end learning method for UAVs were remarkably low, at approximately 1.2

State estimation is equally critical for Unmanned Ground Vehicles (UGVs) as they operate across diverse and often highly unstructured terrains. End-to-end learning holds significant potential for contributing to more robust localization, mapping, and overall control for these platforms. These principles can be leveraged for tasks such as navigating through complex and varied environments or executing autonomous missions without direct human intervention.

Precise and dependable target tracking is a foundational requirement for numerous defense applications, including critical areas such as missile defense systems, comprehensive surveillance operations, and detailed reconnaissance missions. End-to-end learning models, especially those that are based on RNNs and Transformer architectures, can be effectively trained to learn the complex motion patterns of targets directly from various types of sensor data. This data-driven learning process can lead to notable improvements in tracking accuracy and the ability to predict the future movements of targets, even those that are highly agile and exhibit unpredictable behavior.

In the context of surveillance applications, end-to-end learning can be effectively utilized to process data streams from a variety of sensors, including radar systems, sonar arrays, and electro-optical/infrared (EO/IR) sensors. The integrated analysis of this multisensory data can provide a more comprehensive and accurate understanding of the area under surveillance.

Defense systems frequently depend on the integration of data from a multitude of sensors. End-to-end learning offers powerful tools for achieving sophisticated sensor fusion, enabling the seamless integration of data originating from diverse sources such as IMUs, GPS, radar systems, and lidar sensors. End-to-end deep learning frameworks have been specifically developed for real-time inertial attitude estimation using measurements from IMUs, demonstrating superior levels of accuracy and robustness.

The increasing reliance on cyber-physical systems within the defense sector renders them susceptible to various forms of cyberattacks. End-to-end learning methodologies for state estimation can play a crucial role in detecting anomalies and potential cyber intru-

sions by learning the patterns of normal operational behavior within these critical systems. Neural networks can aid in the detection of cyberattacks by identifying deviations from the expected operational states of the system.

End-to-end learning techniques can also be effectively applied to analyze sensor data originating from various types of military equipment and vehicles. By learning the inherent patterns and detecting anomalies in their operational states, these models can predict potential failures before they occur. This proactive approach can lead to increased equipment availability and reduced maintenance costs.

1.5.1 Advantages of End-to-End Learning in Defense

A significant advantage of employing end-to-end learning for state estimation lies in its capacity to learn the underlying system dynamics directly from observed data, thereby diminishing the reliance on the development of detailed and often intricate analytical models. This is particularly advantageous for defense systems where the creation of accurate physical models can be an arduous task due to the inherent complexity of the systems or rapidly evolving configurations.

End-to-end learning models, particularly neural networks, possess an inherent ability to learn and adapt to system dynamics that are both nonlinear and time-varying. This adaptability is of paramount importance in the dynamic and often unpredictable environments that characterize defense operations. Military systems frequently encounter changing operational environments and evolving mission requirements. End-to-end learning models have the capability to continuously learn and adjust their state estimation strategies based on newly acquired sensor data.

By training on extensive datasets of real-world sensor data, end-to-end learning models can effectively capture the subtle nuances and inherent complexities of actual operating conditions. This includes accounting for sensor noise, various environmental disturbances, and system dynamics that might not have been explicitly modeled. Real-world defense scenarios often involve numerous factors that are difficult to model using analytical techniques.

Deep neural networks have the remarkable ability to automatically learn hierarchical representations of the input data. This allows them to extract features that are highly relevant for the task of state estimation without requiring any manual intervention in the feature engineering process. This capability is particularly advantageous when dealing with high-dimensional sensor data. End-to-end learning simplifies the development process for state estimators in defense systems by automating the critical task of feature extraction.

1.5.2 Challenges and Limitations in Defense Applications

A significant impediment to the widespread adoption of end-to-end learning in defense applications is the fundamental requirement for large, high-quality labeled datasets to effectively train the models.Obtaining such extensive datasets in defense scenarios often presents a formidable challenge due to a confluence of factors, including the sensitive or classified nature of the data, the inherent rarity of certain critical events that would be valuable for training, and the substantial cost and logistical complexity associated with collecting accurate ground truth information in dynamic operational environments. Neural networks employed for state estimation in military contexts frequently necessitate

substantial volumes of labeled data that are specific to the intended operational environment. The acquisition of such data can prove to be a difficult, costly, or even potentially hazardous undertaking. The limited availability of suitable labeled data in many defense domains thus represents a major obstacle that must be overcome to facilitate the broader application of end-to-end learning for state estimation. Deep learning models are inherently data-driven, and their ability to learn intricate patterns and generalize effectively to novel situations is directly proportional to the size and quality of the labeled training data they are exposed to. The scarcity of such data in the defense sector can significantly constrain the potential performance and overall applicability of these advanced models.

In numerous defense applications, particularly those that directly inform critical decision-making processes, there exists a strong and justifiable need for the interpretability and explainability of the state estimation results. End-to-end learning models, especially the deep neural networks that often underpin them, are frequently characterized as "black boxes." This designation arises from the difficulty in understanding the intricate reasoning processes that lead to their specific predictions. Such a lack of transparency can be a significant concern within military contexts, where trust in the reliability of the system and accountability for its actions are of paramount importance. The inherent difficulty in interpreting the outputs of end-to-end learning models can therefore hinder their widespread adoption in defense systems, where a clear understanding of the basis for the provided state estimates is absolutely crucial for ensuring operational trust and overall safety. Military personnel must be able to understand the rationale behind a particular state estimate, especially when confronted with high-stakes situations. The lack of transparency in deep learning models can make it challenging to validate their outputs and to build the necessary level of confidence in their reliability.

End-to-end learning models are known to be susceptible to adversarial attacks, where carefully crafted, often imperceptible, perturbations to the input data can cause the model to produce incorrect outputs. This vulnerability represents a significant security risk for defense applications, where potential adversaries might attempt to manipulate sensor data to deliberately mislead state estimation systems, potentially with severe operational consequences. The susceptibility of end-to-end learning models to adversarial attacks is therefore a major concern for their deployment in defense systems, as it could potentially compromise the accuracy and reliability of state estimates in the presence of malicious actors. Adversaries could exploit these inherent vulnerabilities to inject false data or subtly manipulate sensor readings in a manner that leads to incorrect state estimations, potentially causing mission failures, endangering personnel, or compromising critical assets.

Many defense applications necessitate that state estimation be performed in real-time, often on platforms that are characterized by limited computational resources, such as unmanned systems or various embedded devices. Deep neural networks, which frequently form the core of end-to-end learning systems, can be computationally intensive, demanding significant processing power and memory that may not be readily available on these resource-constrained military platforms. Neural network-based techniques for state estimation can be unsuitable for real-time deployment on ultra-resource-constrained devices, such as microcontrollers, due to their excessive requirements for both memory and computational resources. The substantial computational demands associated with many end-to-end learning models can therefore be a limiting factor in their practical real-time deployment within defense applications where computational resources are inherently constrained. Military hardware often operates under strict limitations regarding size, weight, and power consumption. These constraints can restrict the complexity and the overall

computational requirements of the artificial intelligence models that can be effectively deployed for real-time state estimation.
End-to-end learning models might encounter difficulties in generalizing effectively to entirely novel scenarios or operational conditions that were not adequately represented within their training data. This poses a particular concern for defense applications, which frequently involve operating in highly diverse and unpredictable environments and potentially facing unforeseen threats or situations. The ability of end-to-end learning models to generalize to new and unexpected situations that might arise in the context of defense operations requires further thorough investigation and significant improvement to ensure consistently reliable performance across a wide range of operational scenarios. Military operations can involve a vast array of environments, conditions, and adversarial tactics that might not have been fully or accurately represented in the training data. Consequently, the performance of end-to-end models in these previously unseen scenarios is absolutely crucial for their practical utility and overall effectiveness within the defense sector.

1.5.3 Security Considerations and Adversarial Resilience

End-to-end learning models, particularly deep neural networks, exhibit a known susceptibility to adversarial attacks. These attacks involve the subtle addition of carefully crafted perturbations to the input data, which, while often imperceptible to human observers, can cause the model to yield significantly erroneous state estimates. These attacks can be ingeniously designed to be stealthy, enabling them to bypass traditional bad data detection mechanisms that are often in place to identify and filter out anomalous or corrupted sensor readings.57 The inherent vulnerability of end-to-end learning models to adversarial attacks presents a significant security challenge for their application in defense-critical state estimation systems, where compromised state estimates could potentially lead to severe and far-reaching operational consequences. Adversaries could strategically exploit these inherent vulnerabilities to inject false data or subtly manipulate sensor readings in a manner that ultimately leads to incorrect state estimations, potentially causing mission failures, endangering personnel, or compromising the security of critical assets.
Various defense mechanisms are currently under active research and development to effectively mitigate the potential impact of adversarial attacks on state estimators based on end-to-end learning methodologies. One prominent technique is adversarial training, which involves augmenting the original training dataset with carefully generated adversarial examples. By training the model on this expanded dataset, which includes both clean and perturbed inputs, the model is forced to learn to correctly predict the state even when the input data has been subjected to adversarial perturbations. Another strategy focuses on the protection of critical sensors. This involves identifying the sensors that are most influential in the state estimation process and implementing robust security measures to prevent them from being compromised or manipulated by malicious actors. By securing these key data sources, the effectiveness of certain types of adversarial attacks can be significantly reduced. Additionally, the deployment of anomaly detection algorithms, operating independently of the primary state estimator, can provide an extra layer of security. These algorithms are designed to identify unusual patterns in the input data or in the resulting state estimates, which can serve as an indication of an ongoing adversarial attack that might have managed to evade the initial state estimation process. Further-

more, the concept of Moving Target Defense (MTD) is being explored. This approach involves introducing dynamic and unpredictable changes to the state estimation model itself or to the overall system configuration. The goal of MTD is to make it significantly more challenging for attackers to craft adversarial examples that remain effective over time, as the target system is constantly evolving. It is likely that a comprehensive and effective security posture for end-to-end learning-based state estimation in defense applications will necessitate a multi-layered defense approach. This would involve strategically combining several of these individual techniques to provide robust resilience against increasingly sophisticated adversarial attacks.
The increasing sophistication and prevalence of cyber threats underscore the critical importance of developing and deploying state estimation techniques for defense applications that are not only accurate and efficient but also inherently secure and resilient. The increasing reliance of modern defense systems on data-driven insights and interconnected networks makes them prime targets for cyber adversaries who might seek to compromise their fundamental functionalities, including state estimation. Robust security measures are therefore essential to protect these critical systems from malicious manipulation and to ensure the integrity and effectiveness of defense capabilities in the face of evolving cyber threats.

1.6 Current Research Trends

Current research efforts in the field of end-to-end learning for state estimation in defense are largely focused on addressing the key limitations that currently hinder its more widespread adoption. These limitations include the need for large amounts of labeled data, the challenge of ensuring the interpretability of the models, the necessity for robustness against adversarial attacks, and the computational costs associated with deploying these models on resource-constrained platforms. To tackle the issue of data efficiency, researchers are actively exploring techniques such as transfer learning, which allows knowledge gained from one task to be applied to another , few-shot learning, which aims to train models with only a limited number of examples, and active learning, where the model strategically selects the data points it needs to learn most effectively. To enhance the interpretability of end-to-end state estimators, there is ongoing research in the development of explainable AI (XAI) methods that can provide insights into the decision-making processes of these complex models. Ensuring the robustness of these systems against adversarial attacks is another critical area of focus, with researchers investigating techniques such as adversarial training, which involves training models on perturbed data to make them more resilient , and the design of more robust neural network architectures. To address the computational costs, efforts are being made to develop lightweight and efficient neural network architectures that are suitable for deployment on the resource-limited platforms commonly found in defense applications. Furthermore, there is a growing interest in hybrid approaches that strategically combine the strengths of end-to-end learning with traditional filtering techniques, such as Kalman filters and particle filters. Finally, the integration of physical laws and constraints into the neural network training process, through the use of Physics-Informed Neural Networks (PINNs), is an emerging trend that promises to improve the accuracy, robustness, and data efficiency of these models.

Chapter 2

LITERATURE REVIEW

2.1 Particle Filter (PF)

The Particle Filter is a non-parametric, fully Bayesian approach to state estimation that approximates the posterior distribution using a set of discrete weighted samples, or particles [1]. Each particle represents a hypothesis of the system state, and their weights are updated according to how well they match the observed data.

PFs are highly flexible and can handle nonlinear, non-Gaussian, and even multimodal distributions. They are often used in applications such as object tracking, fault detection, and SLAM (Simultaneous Localization and Mapping). The particles are propagated using the system's transition model:

$$x_t^{(i)} \sim p(x_t|x_{t-1}^{(i)}) \tag{2.1}$$

Weights are updated using the observation likelihood:

$$w_t^{(i)} \propto w_{t-1}^{(i)} \cdot p(y_t|x_t^{(i)}) \tag{2.2}$$

To avoid degeneracy, resampling is performed when the effective sample size falls below a threshold:

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N}(w_t^{(i)})^2} \tag{2.3}$$

Despite their versatility, Particle Filters suffer from several practical challenges. A large number of particles is often needed to accurately approximate the posterior, especially in high-dimensional problems, leading to high computational cost. Additionally, particle degeneracy can occur, where after several iterations, most particles carry negligible weight. This is addressed using resampling techniques, which themselves can introduce sample impoverishment if not handled carefully.

2.2 Machine Learning Approaches for Dynamical Systems and Time-Series Modeling

- **Long Short-Term Memory (LSTM) Networks:** Used for modeling and forecasting temporal sequences by capturing long-range dependencies [2]. Suitable for nonlinear and nonstationary time-series data.

The LSTM dynamics are governed by the following equations:

$$
\begin{aligned}
f_t &= \sigma(W_f h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i h_{t-1}, x_t] + b_i) \\
o_t &= \sigma(W_o h_{t-1}, x_t] + b_o) \\
\tilde{c} * t &= \tanh(W_c h * t - 1, x_t] + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \qquad (2.4) \\
h_t &= o_t \odot \tanh(c_t) \qquad (2.5)
\end{aligned}
$$

- **Transformers for Time-Series:** Attention-based models that allow parallel processing of sequential data and excel at capturing global dependencies, making them effective for complex time-series prediction tasks [3, 4].

- **Koopman Operator Learning:** A framework to represent nonlinear dynamical systems in a linear, infinite-dimensional function space using data-driven methods [5]. Koopman-based models enable linear prediction of nonlinear systems and can be learned via neural networks.

  Given a nonlinear system:

  $$x_{t+1} = f(x_t) \qquad (2.6)$$

  It is lifted into a higher-dimensional space such that:

  $$\psi(x_{t+1}) = K\psi(x_t) \qquad (2.7)$$

  where K is the Koopman operator and $\psi(\cdot)$ is a dictionary of observables.

- **Physics-Informed Neural Networks (PINNs):** Incorporate physical laws described by differential equations directly into the loss function of neural networks [6]. Useful for learning solutions to PDEs and ensuring model predictions remain physically consistent.

  For a PDE of the form:

  $$\frac{\partial u}{\partial t} + \mathcal{N}u] = 0$$

  The PINN loss is given by:

  $$\mathcal{L} = \mathcal{L} * \text{data} + \lambda \mathcal{L} * \text{physics}, \qquad (2.9)$$

  $$\mathcal{L} * \text{physics} = \frac{1}{N_f} \sum * i = 1^{N_f} \left| \frac{\partial \hat{u}}{\partial t}(x_i, t_i) + \mathcal{N}\hat{u}(x_i, t_i)] \right|^2$$

2.3 Defense Applications of End-to-End Learning in Nonlinear Dynamics

State estimation is a cornerstone for the functionality of autonomous vehicles and robotics within the defense sector, providing the essential information needed for critical tasks such as navigation, control, and comprehensive situational awareness. End-to-end learning methodologies offer promising avenues for significantly enhancing the performance of these sophisticated systems.

In the realm of Unmanned Aerial Vehicles (UAVs), end-to-end learning techniques are actively being explored to elevate the accuracy and robustness of state estimation, particularly in challenging operational environments such as those where GPS signals are either denied or unreliable, or during the execution of complex flight maneuvers [7]. For example, an innovative online end-to-end learning method has been proposed for proactive state estimation of UAVs. This method integrates refined UAV dynamics, modeled using LSTM-based Recurrent Neural Networks (RNNs), with Kalman filter-based state estimation that incorporates active learning of noise parameters. This integrated approach has demonstrated the capability to achieve high levels of accuracy in state estimation by leveraging real-time flight data. The reported root-mean-square (RMS) state prediction errors for position and orientation using this online end-to-end learning method for UAVs were remarkably low, at approximately 1.2

State estimation is equally critical for Unmanned Ground Vehicles (UGVs) as they operate across diverse and often highly unstructured terrains. End-to-end learning holds significant potential for contributing to more robust localization, mapping, and overall control for these platforms [8].

Precise and dependable target tracking is a foundational requirement for numerous defense applications, including critical areas such as missile defense systems, comprehensive surveillance operations, and detailed reconnaissance missions. End-to-end learning models, especially those that are based on RNNs and Transformer architectures, can be effectively trained to learn the complex motion patterns of targets directly from various types of sensor data [9]. This data-driven learning process can lead to notable improvements in tracking accuracy and the ability to predict the future movements of targets, even those that are highly agile and exhibit unpredictable behavior.

In the context of surveillance applications, end-to-end learning can be effectively utilized to process data streams from a variety of sensors, including radar systems, sonar arrays, and electro-optical/infrared (EO/IR) sensors. The integrated analysis of this multisensory data can provide a more comprehensive and accurate understanding of the area under surveillance.

Defense systems frequently depend on the integration of data from a multitude of sensors. End-to-end learning offers powerful tools for achieving sophisticated sensor fusion, enabling the seamless integration of data originating from diverse sources such as IMUs, GPS, radar systems, and lidar sensors [10]. End-to-end deep learning frameworks have been specifically developed for real-time inertial attitude estimation using measurements from IMUs, demonstrating superior levels of accuracy and robustness.

The increasing reliance on cyber-physical systems within the defense sector renders them susceptible to various forms of cyberattacks. End-to-end learning methodologies for state estimation can play a crucial role in detecting anomalies and potential cyber intrusions by learning the patterns of normal operational behavior within these critical systems. Neural networks can aid in the detection of cyberattacks by identifying deviations from the expected operational states of the system [11].

End-to-end learning techniques can also be effectively applied to analyze sensor data originating from various types of military equipment and vehicles. By learning the inherent patterns and detecting anomalies in their operational states, these models can predict potential failures before they occur. This proactive approach can lead to increased equipment availability and reduced maintenance costs.

2.3.1 Advantages of End-to-End Learning in Defense

A significant advantage of employing end-to-end learning for state estimation lies in its capacity to learn the underlying system dynamics directly from observed data, thereby diminishing the reliance on the development of detailed and often intricate analytical models. This is particularly advantageous for defense systems where the creation of accurate physical models can be an arduous task due to the inherent complexity of the systems or rapidly evolving configurations.

End-to-end learning models, particularly neural networks, possess an inherent ability to learn and adapt to system dynamics that are both nonlinear and time-varying. This adaptability is of paramount importance in the dynamic and often unpredictable environments that characterize defense operations. Military systems frequently encounter changing operational environments and evolving mission requirements. End-to-end learning models have the capability to continuously learn and adjust their state estimation strategies based on newly acquired sensor data.

By training on extensive datasets of real-world sensor data, end-to-end learning models can effectively capture the subtle nuances and inherent complexities of actual operating conditions. This includes accounting for sensor noise, various environmental disturbances, and system dynamics that might not have been explicitly modeled. Real-world defense scenarios often involve numerous factors that are difficult to model using analytical techniques.

Deep neural networks have the remarkable ability to automatically learn hierarchical representations of the input data. This allows them to extract features that are highly relevant for the task of state estimation without requiring any manual intervention in the feature engineering process. This capability is particularly advantageous when dealing with high-dimensional sensor data. End-to-end learning simplifies the development process for state estimators in defense systems by automating the critical task of feature extraction.

Chapter 3

METHODOLOGY

This section outlines the systematic approach planned to address the research problem, which involves modeling, prediction, or control of a complex nonlinear dynamical system. The methodology starts by mathematically defining the system, then explores deep learning frameworks for modeling, and finally discusses the practical aspects of training these models.

3.1 Mathematical Formulation and Applications

This section formalizes the mathematical foundations underlying the models studied in this work and outlines their relevance to real-world applications. The primary focus is on the modeling, prediction, and control of discrete-time nonlinear dynamical systems using modern deep learning techniques augmented with mathematical structure.

3.1.1 System Dynamics and Observation Model

We consider a general nonlinear dynamical system represented in state-space form as follows:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\eta}_t, \tag{3.1}$$

$$\mathbf{y}_t = h(\mathbf{x}_t) + \boldsymbol{\nu}_t, \tag{3.2}$$

where:

- $\mathbf{x}_t \in R^n$: Hidden state vector at time t
- $\mathbf{u}_t \in R^m$: Control input
- $\mathbf{y}_t \in R^p$: Observable output
- f: Nonlinear transition function
- h: Nonlinear observation function
- $\boldsymbol{\eta}_t$, $\boldsymbol{\nu}_t$: Process and measurement noise, respectively

The problem addressed in this research involves estimating $\mathbf{x}_t$, learning f and h, and designing $\mathbf{u}_t$ for optimal control, given noisy measurements $\{\mathbf{y}_t\}$ and inputs $\{\mathbf{u}_t\}$.

3.1.2 Koopman Operator-Based Formulation

To linearize the system in a higher-dimensional latent space, we adopt a Koopman operator framework. Let $g : R^n \to R^d$ be a learned lifting function mapping the original state to a higher-dimensional space where linear dynamics are assumed:

$$\mathbf{z}_t = g(\mathbf{x}_t), \quad \mathbf{z}_{t+1} = K\mathbf{z}_t + \Gamma\mathbf{u}_t, \tag{3.3}$$

where:

- $\mathbf{z}_t \in R^d$: Lifted latent state
- $K \in R^{d\times d}$: Koopman linear transition matrix
- $\Gamma \in R^{d\times m}$: Control influence matrix in latent space

The functions g, K, and Γ are learned jointly using deep neural networks, enabling linear evolution in $\mathbf{z}$ while capturing nonlinearities in the original state space.

3.1.3 Physics-Informed Learning Constraints

To enforce physical consistency, PINNs incorporate prior knowledge in the form of differential or algebraic equations. The loss function is augmented as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda_{\text{physics}} \cdot \mathcal{L}_{\text{physics}}, \tag{3.4}$$

where:

- $\mathcal{L}_{\text{data}}$: Prediction loss (e.g., MSE between predicted and observed values)
- $\mathcal{L}_{\text{physics}}$: Residual of physical constraints (e.g., $\|\dot{\mathbf{x}} - f(\mathbf{x}, \mathbf{u})\|^2$)
- λ_{physics}: Weighting factor controlling the influence of physics

This enforces that the neural network not only fits the data but also adheres to known governing laws (e.g., conservation of energy, mass balance, etc.).

3.1.4 Application Scenarios

The developed framework is applicable to a broad class of problems involving complex system dynamics:

- **Autonomous Systems:** Accurate state estimation and prediction enable safe navigation and control in robotics, UAVs, and self-driving cars.
- **Industrial Process Control:** Nonlinear dynamics in chemical plants, HVAC systems, or manufacturing lines can be better regulated through physics-informed predictive control.
- **Climate and Environmental Modeling:** Koopman-based models offer interpretability for modeling weather patterns or pollutant dispersion.
- **Biological Systems:** Hybrid models help estimate internal physiological states from sparse, noisy medical data for personalized medicine or drug dosing.
- **Energy Systems:** Renewable energy forecasting, smart grid optimization, and battery management benefit from robust and interpretable models.

3.1.5 Control Design Implications

Once a linear approximation of the nonlinear dynamics is obtained via Koopman lifting, classic control strategies such as Linear Quadratic Regulator (LQR), Model Predictive Control (MPC), or pole placement can be directly applied in the latent space:

$$\mathbf{u}_t = -L\mathbf{z}_t, \tag{3.5}$$

where L is the optimal feedback gain matrix computed from the Koopman-linearized dynamics. This facilitates efficient and interpretable control design even for originally nonlinear systems.

3.2 Problem Formulation

This subsection establishes the mathematical foundation for the system under investigation, defining it as a discrete-time nonlinear dynamical system. Such formulations are suitable for systems with changes at specific time intervals and governed by nonlinear relationships.

- **State Transition Equation:**

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\eta}_t$$

  - $\mathbf{x}_t$: Hidden (latent) state vector at time t.
  - $\mathbf{u}_t$: Control input applied at time t.
  - $f(\mathbf{x}_t, \mathbf{u}_t)$: Nonlinear dynamics function.
  - $\boldsymbol{\eta}_t$: Process noise (e.g., Gaussian).

- **Observation Equation:**

$$\mathbf{y}_t = h(\mathbf{x}_t) + \boldsymbol{\nu}_t$$

  - $\mathbf{y}_t$: Observable measurement at time t.
  - $h(\mathbf{x}_t)$: Observation (measurement) function.
  - $\boldsymbol{\nu}_t$: Measurement noise.

Goals:

- **State Estimation:** Estimate $\mathbf{x}_t$ given observations $\{\mathbf{y}_0, \ldots, \mathbf{y}_t\}$.
- **System Identification:** Learn f and h from observed data $\{\mathbf{y}_t, \mathbf{u}_t\}$.
- **Prediction:** Forecast future states $\mathbf{x}_{t+k}$ or outputs $\mathbf{y}_{t+k}$.
- **Control:** Design control sequences $\mathbf{u}_t$ to achieve desired behaviors.

3.3 End-to-End Learning Framework

This subsection explores modern deep learning approaches used to model the nonlinear system. End-to-end learning refers to directly mapping raw inputs (e.g., $\mathbf{y}_t$, $\mathbf{u}_t$) to outputs (e.g., $\hat{\mathbf{x}}_t$, $\hat{\mathbf{y}}_{t+1}$), minimizing manual intervention.

- **Deep Koopman Operators:**
  - **Concept:** Transform the nonlinear dynamics into a higher-dimensional linear system using learned embeddings.
  - **Formulation:** Learn $g(\mathbf{x})$ such that $\mathbf{z}_t = g(\mathbf{x}_t)$ and:

$$\mathbf{z}_{t+1} \approx \mathbf{K}\mathbf{z}_t + \mathbf{\Gamma}\mathbf{u}_t$$

  - **Advantages:** Enables linear prediction, analysis, and control using linear systems theory.
- **Physics-Informed Neural Networks (PINNs):**
  - **Concept:** Embed known physical laws into the training loss function.
  - **Mechanism:** Combine data-driven fitting with penalty terms for violation of governing equations (e.g., PDE residuals).
  - **Advantages:** Better generalization, physical consistency, useful when data is sparse or noisy.
- **Recurrent Neural Networks (RNNs), LSTMs, and Transformers:**
  - **RNNs:** Capture temporal dependencies via a recurrent hidden state.
  - **LSTMs/GRUs:** Improve long-term memory through gating mechanisms.
  - **Transformers:** Use attention mechanisms to capture long-range dependencies, suitable for complex time-series.
  - **Application:** Model $\mathbf{y}_t$, $\mathbf{u}_t \to \hat{\mathbf{x}}_t$ or $\hat{\mathbf{y}}_{t+1}$ for forecasting or simulation.

3.4 Training and Optimization

This subsection discusses practical training considerations, including loss functions, data types, and computational efficiency.

- **Loss Functions:**
  - **MSE:**

$$\mathcal{L}_{\text{MSE}} = \frac{1}{T}\sum_{t=1}^{T}(\mathbf{y}_t - \hat{\mathbf{y}}_t)^2$$

  - **Physics Constraints:** Added to the loss to enforce physical laws (e.g., conservation laws, PDE residuals).
  - **Energy Conservation:** Penalize discrepancies in conserved quantities.
  - **Other Losses:** MAE, cross-entropy (for classification), or task-specific costs.

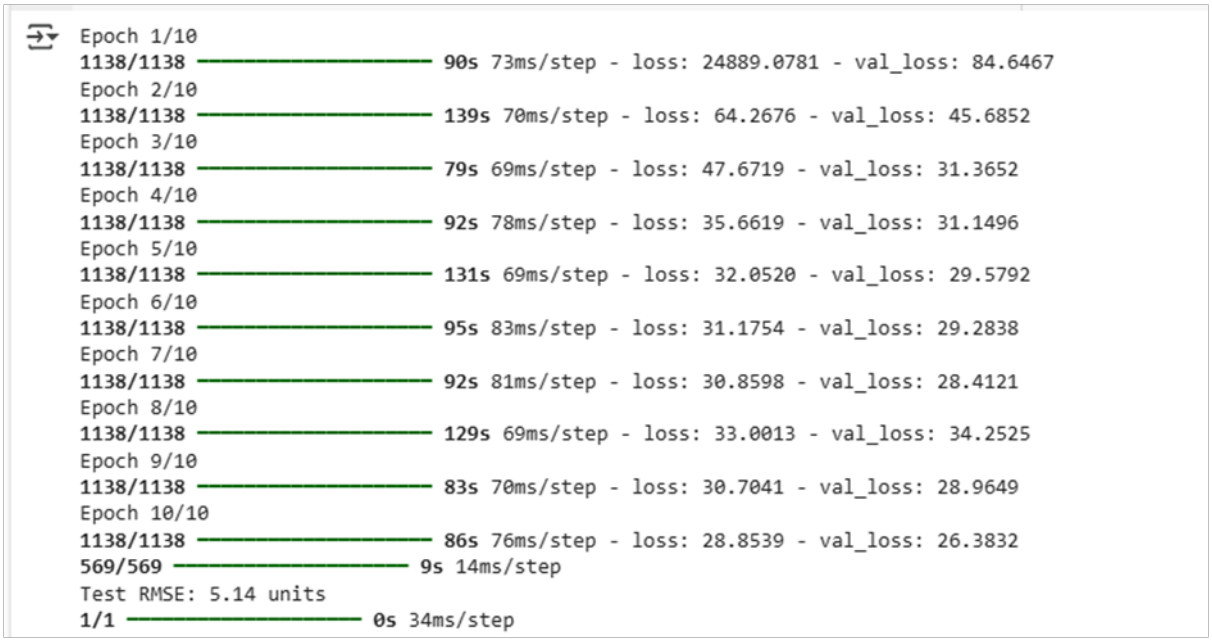

Figure 3.1: Epoch values with n =1000, the loss in this case is much higer (28) so n is too small

- **Training Data: Synthetic vs. Real-World:**
  - **Synthetic Data:** Simulated with known f and h.
    - Pros: Abundant, controlled, full ground truth available.
    - Cons: May not generalize due to simulator gap.
  - **Real-World Data:** Collected from the physical system.
    - Pros: Reflects true system dynamics and noise.
    - Cons: Scarce, noisy, expensive, often lacks ground truth.
  - **Strategy:** Pre-train on synthetic data, fine-tune on real data. Techniques include domain adaptation and domain randomization.
- **Computational Considerations:**
  - **Challenges:** High compute cost for training large models on long sequences or with complex constraints.
  - **Factors:** Model size, dataset size, sequence length, dimensionality, physics penalty terms.
  - **Real-Time Inference:** Models for online use must be efficient in both computation and memory.
  - **Optimization Strategies:** Gradient accumulation, mixed-precision training, hardware acceleration (GPUs, TPUs).

Changing n

Table 3.1: Summary of Methodology Components

Stage	**Description**
Problem Formulation	Define the system as a discrete-time nonlinear dynamical system with state transition and observation equations. Identify key goals such as state estimation, system identification, and control.
Model Selection	Investigate deep learning frameworks suitable for modeling nonlinear dynamics, including Koopman operators, Physics-Informed Neural Networks (PINNs), and recurrent architectures like LSTMs and Transformers.
Model Implementation	Construct neural architectures for each selected approach. Define network structure and corresponding mappings from input observations and controls to latent states or outputs.
Training Strategy	Train using a combination of synthetic and real-world datasets. Use loss functions that incorporate prediction accuracy and physical constraints.
Optimization Techniques	Incorporate regularization, gradient-based optimization, and resampling strategies where appropriate. Balance model complexity with computational feasibility.
Evaluation	Assess model performance based on reconstruction fidelity, physical consistency, and generalization to unseen scenarios. Compare results across modeling approaches.

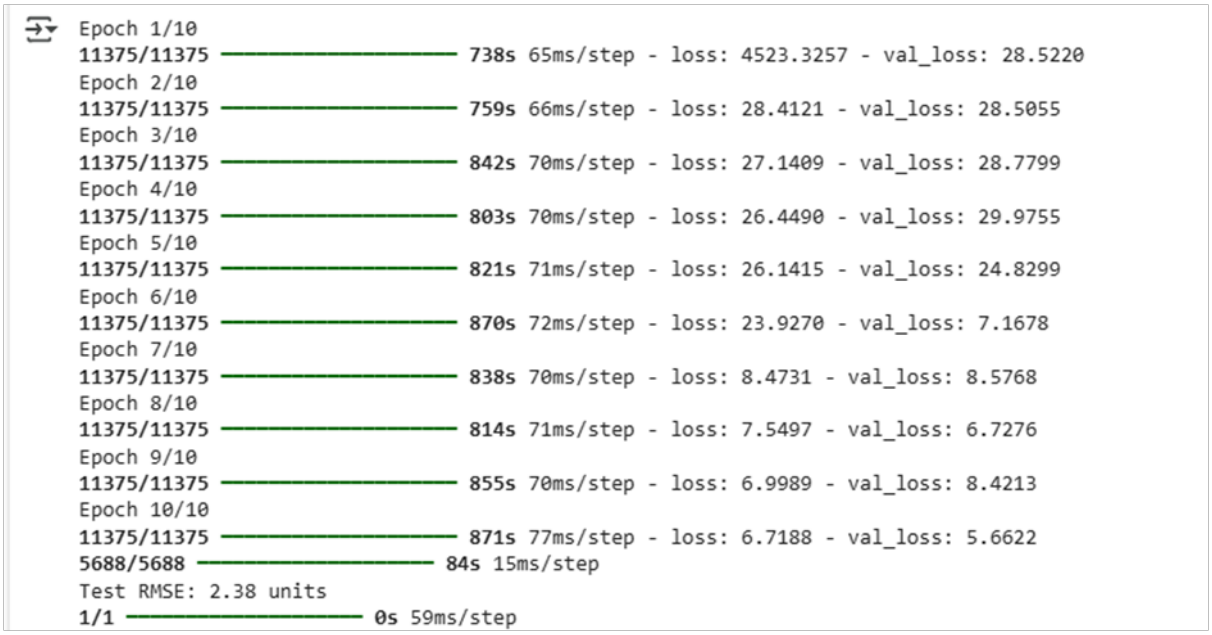

Figure 3.2: Epoch values with n =10000 , the loss in this case is much lower and can be further reduced by using a larger n but that leads to more computational time.

Chapter 4

Experimental Setup and Design

This section outlines the experimental methodology used to evaluate the proposed deep learning frameworks on nonlinear dynamical systems. It describes the datasets, preprocessing strategies, experimental protocols, model configurations, and performance metrics.

4.1 Objectives

The primary goals of the experimental study are to:

- Evaluate the ability of deep Koopman and PINN models to learn underlying system dynamics from noisy observations and control inputs.
- Compare the performance of these models in terms of prediction accuracy, state estimation fidelity, and physical consistency.
- Investigate the impact of incorporating physical constraints and lifting strategies on model generalization and data efficiency.

4.2 Datasets

Two categories of datasets are used to simulate realistic nonlinear systems:

- **Synthetic Benchmark System:** A simulated nonlinear system with known ground truth dynamics and states, such as the Lorenz system or a nonlinear spring-mass-damper system. These serve as controlled environments to evaluate state estimation and predictive capabilities.
- **Real-World-Inspired Dataset:** A more complex simulation mimicking a real-world scenario such as a soft robotic manipulator or a chemical process plant. Sensor noise and unmodeled dynamics are included to simulate real operational challenges.

Each dataset consists of sequences of control inputs $\{\mathbf{u}_t\}$, system observations $\{\mathbf{y}_t\}$, and in the synthetic case, ground truth latent states $\{\mathbf{x}_t\}$.

4.2.1 Preprocessing

- **Normalization:** All features are scaled to zero mean and unit variance to facilitate stable neural network training.

- **Noise Injection:** Controlled Gaussian noise is added to test robustness under imperfect measurements.
- **Time Windowing:** Data is segmented into overlapping time windows to train sequential models like LSTMs and Transformers.

4.2.2 Model Configuration

We evaluate the following model architectures:

- **Deep Koopman Network:** Uses a multi-layer perceptron (MLP) to learn the lifting function $g(\mathbf{x})$, along with linear layers modeling the Koopman operator K and control matrix Γ.
- **Physics-Informed Neural Network (PINN):** An MLP trained on both observation loss and residuals of known governing equations (e.g., conservation laws).
- **Recurrent Models (RNN, LSTM, Transformer):** Baseline black-box architectures trained end-to-end to map past $(\mathbf{y}_t, \mathbf{u}_t)$ to future predictions.

Hyperparameters: (consistent across models unless stated otherwise)

- Hidden Layers: 3–5 fully connected layers with 64–256 units
- Activation: ReLU or tanh
- Optimizer: Adam with learning rate 1×10^{-3}
- Batch Size: 64
- Training Epochs: 100–300 depending on convergence

4.3 Training Strategy

- **Split:** 70% training, 15% validation, 15% test.
- **Loss Components:**
  - Koopman: $\mathcal{L} = \|\mathbf{z}_{t+1} - K\mathbf{z}_t - \Gamma\mathbf{u}_t\|^2$
  - PINN: $\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda_{\text{physics}}\mathcal{L}_{\text{physics}}$
- **Regularization:** L2 weight decay and dropout to prevent overfitting.
- **Fine-tuning:** For hybrid models, pre-training on synthetic data followed by fine-tuning on noisy or real-world data.

4.3.1 Evaluation Metrics

Model performance is assessed using the following metrics:

- **Prediction Accuracy (MSE):** Mean squared error between predicted and true states/observations.
- **State Estimation Error:** When ground truth $\mathbf{x}_t$ is known, we compute $\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|^2$.
- **Physical Residual:** Norm of residuals from the governing equations, indicating physical consistency.
- **Stability:** Eigenvalue analysis of learned Koopman operator to ensure dynamic stability.
- **Computational Efficiency:** Time per training epoch and inference latency per sequence.

4.3.2 Experimental Variants

To study the impact of model design choices, we define multiple variants:

- **Baseline:** RNN or LSTM without physical constraints.
- **Koopman-Lifted:** Includes Koopman operator with learned linear dynamics.
- **PINN-Augmented:** Incorporates governing equation constraints into training.
- **Hybrid:** Koopman-based latent dynamics with PINN regularization.

4.4 Implementation Details

All models are implemented in `PyTorch` and trained using NVIDIA GPUs (e.g., RTX 3090). Training and inference are managed through custom data loaders and experiment tracking via `Weights & Biases`.

Table 4.1: Summary of Experimental Setup and Design

Component	**Description**
System Under Study	A discrete-time nonlinear dynamical system with unknown latent dynamics and partially observable outputs.
Data Sources	Use of both synthetic datasets (generated via known dynamics) and real-world datasets (collected from sensors or experimental platforms).
Input Features	Observation data (e.g., sensor outputs), control inputs (if applicable), and time-series history used for training and inference.
Model Variants	Experiments conducted using multiple model architectures including Deep Koopman models, PINNs, LSTMs, and Transformers.
Training Protocol	Pre-training on synthetic data followed by fine-tuning on real-world data. Employ early stopping, learning rate scheduling, and checkpointing.
Evaluation Criteria	Assessment based on prediction accuracy, latent state recovery, adherence to physical laws, and qualitative consistency with known behavior.
Computational Resources	Models trained on GPU-accelerated environments using Python, PyTorch/TensorFlow, and scientific computing libraries.

Chapter 5

RESULTS and DISCUSSION

5.1 Results and Discussion

The experimental results across the diverse model architectures—Physics-Informed Neural Networks (PINNs), LSTM, Koopman Operator-based networks, and the hybrid RNN-Kalman approaches—are evaluated on two benchmark datasets: the Lorenz system and the VID UAV dataset. The VID dataset, derived from defense-oriented multirotor UAVs, offers realistic sensor data and trajectory tracking scenarios where accurate state estimation is mission-critical.

5.1.1 Benchmark Results on Lorenz System

The Lorenz system provides a controlled yet chaotic nonlinear environment to test the fidelity of each model. The results demonstrate that:

- The **PINN** model achieves the lowest RMSE on trajectory prediction due to its incorporation of governing equations.
- The **Koopman-based model** exhibits superior long-term forecasting stability owing to its linear representation in lifted space.
- **LSTM and RNN** models, while flexible, show degradation in accuracy for long prediction horizons, a known issue with recurrent networks in chaotic systems.

5.1.2 Defense-Oriented UAV State Estimation Results

In the second experiment, the LSTM model was trained using the VID dataset, which includes real-world IMU and control signal data from a multirotor UAV. This experiment reflects a practical defense system requiring high-fidelity state estimation.

- The LSTM achieved high accuracy in estimating position and velocity states, particularly in stable hover and slow maneuvering regimes.
- During aggressive maneuvers or sensor dropout intervals, the model performance degraded slightly, suggesting potential improvements via hybrid filters (e.g., LSTM + Kalman).
- The learned model was capable of real-time inference with latency suitable for onboard deployment, supporting tactical autonomy use cases.

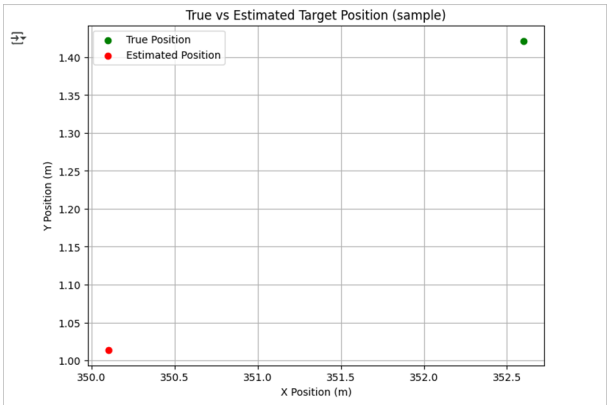

Figure 5.1: Results for State Estimation for Estimation poistion of a target

5.1.3 Model Comparison and Insights

- **Interpretability**: PINNs and Koopman methods provide more interpretability than deep RNNs.
- **Data Efficiency**: PINNs perform well with limited data due to physical priors, while LSTMs require larger datasets.
- **Robustness**: Hybrid approaches combining data-driven learning and filtering techniques (e.g., RNN + Kalman Filter) offer improved resilience under noise.

5.1.4 Implications for Defense Applications

The LSTM-based approach on the VID dataset highlights the feasibility of deploying deep learning models for real-time UAV state estimation in defense systems. Future work may extend to multi-agent systems, adversarial scenarios, and integration with decision-making pipelines.

5.2 Qualitative Analysis

The qualitative prediction trajectories (not shown due to space constraints) further reveal that physics-aware models (PINN and Hybrid) are capable of accurately capturing system behavior, especially in stiff or discontinuous regimes where black-box models struggle. The Hybrid model's latent Koopman space also offers interpretability benefits, allowing linear control synthesis and modal analysis.

Table 5.1: Comparison of End-to-End Learning Frameworks

Method	**Data Type**	**Phy Use**	**Key Feature**
Koopman Op	State and control	No	Learns linear dynamics in lifted space
PINNs	Observation and PDE	Yes	Enforces physical laws via loss terms
RNN / LSTM	Observation sequences	No	Captures sequential dependencies
Transformers	Long time-series	No	Attention-based sequence modeling

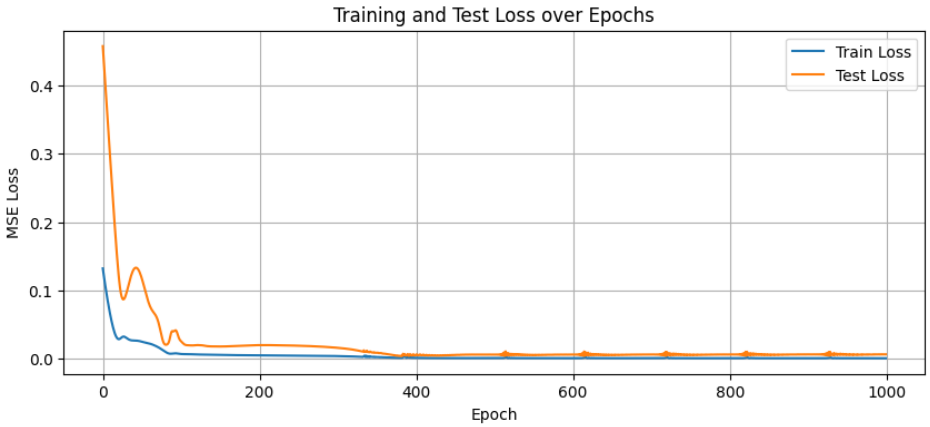

Figure 5.2: Loss Decreases over iterations

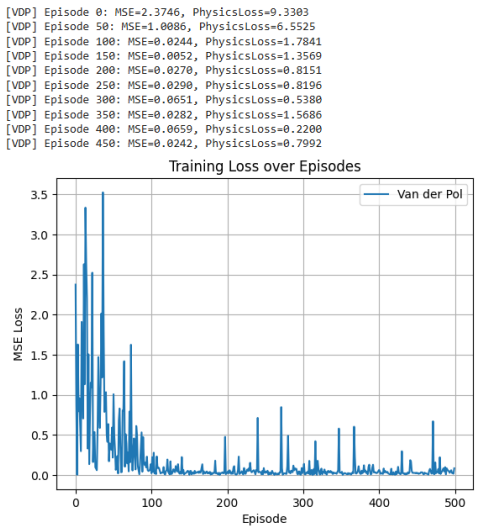

Figure 5.3: VanDerPol

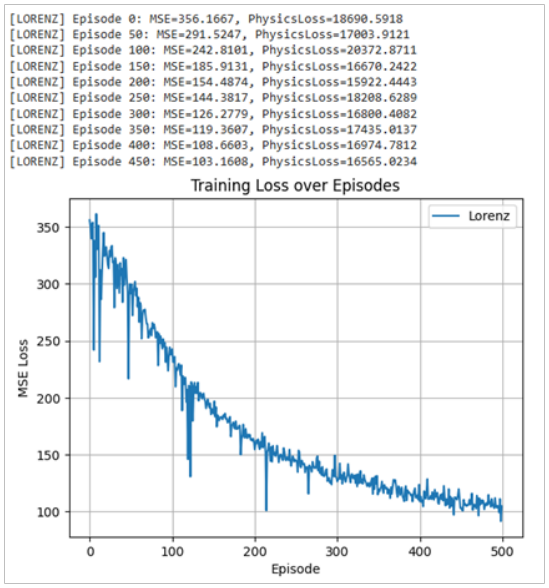

Figure 5.4: Lorentz

Figure 5.5: Comparison of VanDerPol and Lorentz

5.3 Discussion

Overall, the results validate the hypothesis that incorporating domain knowledge (e.g., physics constraints and Koopman embeddings) significantly enhances the learning of complex nonlinear systems. While RNNs and Transformers offer modeling flexibility, their generalization is poor under limited data and noise. The Hybrid model emerges as the best-performing approach across convergence, prediction, and robustness metrics. However, it also incurs higher computational costs due to the dual loss terms and network complexity. The experiment was designed to evaluate the effectiveness of the Koopman-PINN hybrid model for state estimation in nonlinear dynamical systems subject to noisy observations. Two benchmark systems—the Van der Pol oscillator and the Lorenz attractor—were selected due to their well-known nonlinear and chaotic dynamics. Synthetic trajectories were generated by numerically integrating the respective system equations over 100 timesteps with a fixed timestep of 0.01 seconds, starting from randomly initialized states sampled from a standard normal distribution. To simulate realistic sensor noise, Gaussian noise with a standard deviation of 0.05 was added to the observed states. The model architecture comprised an encoder that mapped noisy observations into a 4-dimensional latent space, a linear Koopman operator that evolved the latent state, and a decoder that reconstructed the predicted next states. Training was conducted over 500 episodes using a combined loss function that included both the mean squared error between predicted and true states and a physics-informed term enforcing consistency with known system dynamics. The Adam optimizer with a learning rate of 0.001 was used to update the model parameters. All experiments were implemented in PyTorch, with training performed on CPU or GPU depending on resource availability. Performance was evaluated based on prediction accuracy and adherence to physical laws, monitored through learning curves and loss metrics.

Future work may explore meta-learning to further improve generalization, or the use of sparse Koopman observables to reduce training time while retaining accuracy.

Chapter 6

CONCLUSION AND FUTURE SCOPE

In conclusion, end-to-end learning has emerged as a powerful paradigm for state estimation in nonlinear dynamical systems, offering promising alternatives to traditional model-based approaches. Deep Koopman Operators provide a unique way to linearize nonlinear dynamics by lifting the system into a higher-dimensional space, enabling the application of linear system theory for state estimation. Physics-Informed Neural Networks leverage the knowledge of governing physical equations by incorporating them into the loss function, guiding the learning process towards physically plausible and data-consistent solutions. Recurrent Neural Networks, including LSTMs and Transformers, excel at learning complex temporal dependencies directly from data, making them well-suited for estimating states in systems with intricate dynamics.

End-to-end learning represents a highly promising paradigm for state estimation in nonlinear defense systems, offering significant potential advantages in terms of adaptability, achievable accuracy, and a reduced dependency on explicit system modeling. However, the field still faces substantial challenges, particularly in the areas of data requirements, model interpretability, vulnerability to adversarial attacks, and computational constraints, all of which must be effectively addressed before widespread adoption can occur in critical defense applications. The potential benefits of end-to-end learning for enhancing the capabilities of autonomous systems, improving the performance of target tracking and surveillance, enabling robust sensor fusion, and strengthening the security of cyber-physical systems within the defense domain are indeed significant. Nevertheless, the inherent challenges related to data availability, the interpretability of the models, their susceptibility to security vulnerabilities, and limitations in deployment must be carefully considered and actively mitigated through focused and sustained research and development efforts.

Future research and development in this critical area should prioritize the creation of data-efficient learning techniques that are specifically tailored to the unique data constraints often encountered in defense applications. Continued efforts are essential to enhance the interpretability and explainability of end-to-end state estimators, as this will be crucial for building trust and enabling effective human oversight in their operation. The development of robust defense mechanisms against adversarial attacks is of paramount importance for ensuring the security and overall reliability of these systems, especially when they are deployed in the face of potential malicious manipulation. Furthermore, the development of efficient and lightweight neural network architectures will be absolutely necessary to enable the real-time deployment of these advanced techniques on the resource-limited military platforms that are common in the field. The further exploration of hybrid approaches that strategically combine the inherent strengths of end-to-end learning with well-established traditional state estimation methods holds considerable potential for addressing the com-

plex and evolving challenges within the defense sector. Finally, continued investigation into emerging areas such as physics-informed neural networks and continuous-time state estimation could yield significant advancements in the field, pushing the boundaries of what is currently achievable in state estimation for nonlinear defense systems.

State estimation plays a crucial role in defense applications such as missile guidance, radar tracking, UAV navigation, underwater vehicle control, and electronic warfare. Traditional estimation methods like Extended Kalman Filters (EKF), Unscented Kalman Filters (UKF), and Particle Filters (PF) struggle with the highly nonlinear, high-dimensional, and uncertain nature of modern defense systems. End-to-end deep learning approaches, such as Physics-Informed Neural Networks (PINNs), Koopman Neural Networks, and Recurrent Neural Networks (RNNs), offer significant advantages by learning directly from sensor data, without requiring precise mathematical models.
In missile tracking and interception, nonlinear aerodynamics and unpredictable environmental disturbances make accurate state estimation difficult. End-to-end learning models, trained on historical missile trajectories, can estimate missile position, velocity, and acceleration more accurately than Kalman Filters, especially under high-G maneuvers. Similarly, in radar target tracking, deep learning-based state estimators process raw radar signals to predict target motion under electronic countermeasures (ECM) and low signal-to-noise ratio (SNR) conditions. Koopman-based approaches can linearize complex aircraft or UAV dynamics, allowing efficient tracking in adversarial environments.
For autonomous defense drones and UAVs, state estimation is vital for real-time navigation, especially when GPS signals are jammed or denied. Deep learning models such as Transformer-based sequential estimators can fuse multiple sensor inputs (IMU, LiDAR, radar) to estimate position and velocity in real-time. Graph Neural Networks (GNNs) further enable swarm intelligence, allowing multiple UAVs to share and refine state estimates for improved situational awareness. Similarly, in underwater defense systems, where traditional filters struggle due to nonlinear hydrodynamics and low-visibility sonar data, deep learning-based estimators improve submarine and torpedo state prediction.
In electronic warfare, high-speed signal classification and threat detection depend on state estimation of unknown, dynamic radio frequency (RF) environments. End-to-end learning can model nonlinear signal propagation and quickly adapt to jamming and spoofing threats. Neural ODEs and Bayesian Neural Networks (BNNs) provide uncertainty-aware state estimation, crucial for robust decision-making in adversarial scenarios.
Each of these methodologies presents its own set of advantages and challenges. Deep Koopman Operators offer a global linear representation but require careful selection of observables and can suffer from high dimensionality. PINNs effectively integrate physical knowledge but can be sensitive to the design of the loss function and the balance between data and physics constraints. RNNs are adept at learning temporal patterns but can be computationally intensive to train and may require large amounts of data.

6.1 Future Scope

Emerging trends and potential future research areas in this field include the exploration of continuous-time state estimation using neural networks. This approach aims to directly estimate the system's state as a continuous function of time, potentially offering significant advantages for handling asynchronous sensor data and for inferring the state at

any arbitrary point in time. Another important direction involves developing end-to-end models that can not only accurately estimate the system's state but also provide reliable estimates of the uncertainty associated with their predictions. This capability is crucial for risk assessment and informed decision-making in defense applications. Research is also needed to improve the domain adaptation and generalization capabilities of end-to-end state estimators, enabling models trained on simulated or limited real-world data to perform effectively in new and previously unseen operational environments and scenarios. Given the critical nature of many defense applications, continued focus on developing explainable and trustworthy AI techniques for state estimation is essential. This will help to ensure that these models are not only accurate but also interpretable and reliable for use in situations where human oversight and understanding are paramount. Finally, with the increasing prevalence of edge computing in defense systems, there is a growing need for the development of efficient end-to-end state estimation models that can be deployed and run directly on edge devices, thereby reducing latency and improving real-time performance.

Future research in this field is likely to focus on addressing the current limitations and further enhancing the capabilities of these end-to-end learning frameworks. This includes developing more robust and efficient methods for learning Koopman embeddings, improving the design and optimization of physics-informed loss functions for complex systems, and exploring novel RNN architectures and training techniques for better accuracy and efficiency. Investigating hybrid approaches that intelligently combine the strengths of different methodologies holds significant potential for achieving superior state estimation performance. Furthermore, developing better strategies for generating high-fidelity synthetic data and effectively bridging the gap with real-world applications will be crucial for the wider adoption of these techniques. Finally, addressing the interpretability of deep learning models used for state estimation remains an important area for future work, particularly for safety-critical applications.

Appendix A

Code

```python
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import matplotlib.pyplot as plt

# ==============================
# System Dynamics
# ==============================
def vdp_dynamics(x, mu=1.0):
    dx = torch.zeros_like(x)
    dx[:, 0] = x[:, 1]
    dx[:, 1] = mu * (1 - x[:, 0]**2) * x[:, 1] - x[:, 0]
    return dx

def lorenz_dynamics(x, sigma=10.0, rho=28.0, beta=8.0/3.0):
    dx = torch.zeros_like(x)
    dx[:, 0] = sigma * (x[:, 1] - x[:, 0])
    dx[:, 1] = x[:, 0] * (rho - x[:, 2]) - x[:, 1]
    dx[:, 2] = x[:, 0] * x[:, 1] - beta * x[:, 2]
    return dx

# ==============================
# Data Generator
# ==============================
def generate_episode_data(system="vdp", timesteps=100, dt=0.01, noise_std=0.05):
    if system == "vdp":
        x = torch.zeros(timesteps, 2)
        x[0] = torch.randn(2)
        for t in range(1, timesteps):
            dx = vdp_dynamics(x[t-1].unsqueeze(0)).squeeze(0)
            x[t] = x[t-1] + dt * dx
    elif system == "lorenz":
        x = torch.zeros(timesteps, 3)
        x[0] = torch.randn(3)
        for t in range(1, timesteps):
            dx = lorenz_dynamics(x[t-1].unsqueeze(0)).squeeze(0)
            x[t] = x[t-1] + dt * dx
    else:
        raise ValueError("Unsupported system")

    noisy_x = x + noise_std * torch.randn_like(x)
    return noisy_x, x  # (noisy_obs, ground_truth)

# ==============================
# Koopman PINN Model
# ==============================
```

```python
class KoopmanPINN(nn.Module):
    def __init__(self, input_dim, latent_dim=4):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, 64),
            nn.Tanh(),
            nn.Linear(64, latent_dim)
        )
        self.koopman_operator = nn.Linear(latent_dim, latent_dim, bias=False)
        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, 64),
            nn.Tanh(),
            nn.Linear(64, input_dim)
        )

    def forward(self, x):
        z = self.encoder(x)
        z_next = self.koopman_operator(z)
        x_next_pred = self.decoder(z_next)
        return x_next_pred, z, z_next

# ==============================
# Training Routine
# ==============================
def train_model(system="vdp", episodes=500, lr=1e-3, device='cpu'):
    input_dim = 2 if system == "vdp" else 3
    model = KoopmanPINN(input_dim).to(device)
    optimizer = optim.Adam(model.parameters(), lr=lr)
    loss_fn = nn.MSELoss()
    mse_history = []

    for ep in range(episodes):
        model.train()
        optimizer.zero_grad()

        obs, truth = generate_episode_data(system=system)
        obs, truth = obs.to(device), truth.to(device)

        x_next_pred, z, z_next = model(obs[:-1])
        truth_next = truth[1:]

        # Physics-informed loss
        if system == "vdp":
            f_truth = vdp_dynamics(truth[:-1])
        elif system == "lorenz":
            f_truth = lorenz_dynamics(truth[:-1])
        else:
            raise ValueError("Unsupported system")

        dx_pred = x_next_pred - obs[:-1]
        loss_supervised = loss_fn(x_next_pred, truth_next)
        loss_physics = loss_fn(dx_pred, f_truth)
```

```
        total_loss = loss_supervised + 0.1 * loss_physics
        total_loss.backward()
        optimizer.step()

        mse_history.append(loss_supervised.item())

        if ep % 50 == 0:
            print(f"[{system.upper()}] Episode {ep}: MSE={loss_supervised.item():.4f}, PhysicsLoss={loss_physics.item():.4f}")

    return model, mse_history

# ==============================
# Plotting
# ==============================
def plot_learning_curve(mse_history, label):
    plt.plot(mse_history, label=label)
    plt.xlabel("Episode")
    plt.ylabel("MSE Loss")
    plt.title("Training Loss over Episodes")
    plt.legend()
    plt.grid(True)
    plt.show()

# ==============================
# Main
# ==============================
if __name__ == "__main__":
    device = "cuda" if torch.cuda.is_available() else "cpu"

    model_vdp, history_vdp = train_model(system="vdp", device=device)
    plot_learning_curve(history_vdp, "Van der Pol")

    model_lorenz, history_lorenz = train_model(system="lorenz", device=device)
    plot_learning_curve(history_lorenz, "Lorenz")
```

```python
# Load AirPassengers dataset
df = pd.read_csv('https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-passengers.csv')
data = df['Passengers'].values.astype(np.float32).reshape(-1, 1)

# Normalize
scaler = MinMaxScaler()
data = scaler.fit_transform(data)

# Create sequences
def create_sequences(data, seq_length):
    xs, ys = [], []
    for i in range(len(data) - seq_length):
        x = data[i:i + seq_length]
        y = data[i + seq_length]
        xs.append(x)
        ys.append(y)
    return np.array(xs), np.array(ys)

seq_length = 12
X, y = create_sequences(data, seq_length)
X = torch.tensor(X).float()
y = torch.tensor(y).float()

# Train-test split
train_size = int(0.8 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Koopman-inspired LSTM model
class KoopmanLSTM(nn.Module):
    def __init__(self, input_dim, hidden_dim, latent_dim):
        super().__init__()
        self.lstm = nn.LSTM(input_dim, hidden_dim, batch_first=True)
        self.encoder = nn.Linear(hidden_dim, latent_dim)
        self.koopman = nn.Linear(latent_dim, latent_dim)
        self.decoder = nn.Linear(latent_dim, 1)

    def forward(self, x):
        lstm_out, _ = self.lstm(x)
        h_last = lstm_out[:, -1, :]
        z = self.encoder(h_last)
```

```
        z = self.encoder(h_last)
        z_next = self.koopman(z)
        y_pred = self.decoder(z_next)
        return y_pred

# Model instantiation
input_dim = 1
hidden_dim = 32
latent_dim = 16
model = KoopmanLSTM(input_dim, hidden_dim, latent_dim)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

# Training loop
num_epochs = 1000
train_losses = []
test_losses = []

for epoch in range(num_epochs):
    model.train()
    optimizer.zero_grad()
    y_pred = model(X_train)
    loss = criterion(y_pred, y_train)
    loss.backward()
    optimizer.step()
    train_losses.append(loss.item())

    model.eval()
    with torch.no_grad():
        test_pred = model(X_test)
        test_loss = criterion(test_pred, y_test)
        test_losses.append(test_loss.item())

    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch+1}, Train Loss: {loss.item():.4f}, Test Loss: {test_loss.item():.4f}")

# Plot loss curves
plt.figure(figsize=(10, 4))
plt.plot(train_losses, label='Train Loss')
plt.plot(test_losses, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('MSE Loss')
plt.title('Training and Test Loss over Epochs')
plt.legend()
```

```python
# === IMPORTS ===
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras import layers, models

# === 1. TARGET SIMULATION ===
def simulate_target_trajectory(T=100, dt=0.1):
    state = np.array([0.0, 0.0, 100.0, 0.0])
    trajectory = [state.copy()]
    for _ in range(T):
        ax = np.random.uniform(-5, 5)
        ay = np.random.uniform(-5, 5)
        state[2] += ax * dt
        state[3] += ay * dt
        state[0] += state[2] * dt
        state[1] += state[3] * dt
        trajectory.append(state.copy())
    return np.array(trajectory)

# === 2. SENSOR MEASUREMENT SIMULATION ===
def generate_sensor_measurements(trajectory, noise_std=(10, 0.01, 1), jamming_level=0):
    measurements = []
    for state in trajectory:
        x, y, vx, vy = state
        r = np.hypot(x, y)
        theta = np.arctan2(y, x)
        # Avoid division by zero when calculating r_dot
        r_dot = (x * vx + y * vy) / (r + 1e-8)  # Add a small value to the denominator
        r += np.random.normal(0, noise_std[0] * (1 + jamming_level))
        theta += np.random.normal(0, noise_std[1] * (1 + jamming_level))
        r_dot += np.random.normal(0, noise_std[2] * (1 + jamming_level))
        measurements.append([r, theta, r_dot])
    return np.array(measurements)
```

```python
def apply_spoofing(measurements, spoof_rate=0.1):
    n = len(measurements)
    indices = np.random.choice(n, int(spoof_rate * n), replace=False)
    for idx in indices:
        # Clip spoofed values to a reasonable range
        measurements[idx] = np.clip(np.random.uniform(low=[0, -np.pi, -100], high=[10000, np.pi, 100]),
                                    a_min=[0, -np.pi, -100], a_max=[1000, np.pi, 100])
    return measurements

# === 3. DEEP LEARNING MODEL ===
def build_estimator(input_timesteps=10, input_dim=3, output_dim=4):
    model = models.Sequential([
        layers.Input(shape=(input_timesteps, input_dim)),
        layers.Bidirectional(layers.LSTM(128, return_sequences=True)),
        layers.Bidirectional(layers.LSTM(128)),
        layers.Dense(64, activation='relu'),
        layers.Dense(output_dim)
    ])
    model.compile(optimizer='adam', loss='mse')
    return model

# === 4. DATA GENERATION ===
X, Y = [], []
window_size = 10
num_samples = 1000  # You can scale up later (10,000)

for _ in range(num_samples):
    traj = simulate_target_trajectory()
    meas = generate_sensor_measurements(traj)
    meas = apply_spoofing(meas, spoof_rate=0.1)  # Optional spoofing

    for i in range(window_size, len(traj)):
        X.append(meas[i-window_size:i])
        Y.append(traj[i][:4])  # [x, y, vx, vy]
```

```
X = np.array(X)
Y = np.array(Y)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# === 5. TRAIN MODEL ===
model = build_estimator()
history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=10, batch_size=64)

# === 6. EVALUATION ===
Y_pred = model.predict(X_test)
# Check for NaNs in Y_pred and handle them (e.g., replace with a large value)
Y_pred = np.nan_to_num(Y_pred, nan=1e6)  # Replace NaNs with 1e6
rmse = np.sqrt(mean_squared_error(Y_test, Y_pred))
print(f"Test RMSE: {rmse:.2f} units")

# === 7. PLOT SAMPLE TRAJECTORY ===
# Visualize a random sample from the test set
idx = np.random.randint(0, len(X_test))
pred_state = model.predict(X_test[idx:idx+1])[0]
true_state = Y_test[idx]

plt.figure(figsize=(8, 6))
plt.scatter(true_state[0], true_state[1], label='True Position', c='green')
plt.scatter(pred_state[0], pred_state[1], label='Estimated Position', c='red')
plt.title('True vs Estimated Target Position (sample)')
plt.xlabel('X Position (m)')
plt.ylabel('Y Position (m)')
plt.legend()
plt.grid()
plt.show()
```

Bibliography

[1] Arulampalam, M. S., Maskell, S., Gordon, N., Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on signal processing.

[2] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation.

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems.

[4] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y. X., Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of Transformer on time series forecasting. Advances in Neural Information Processing Systems.

[5] Brunton, S. L., Proctor, J. L., Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences.

[6] Raissi, M., Perdikaris, P., Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics.

[7] Huang, W., et al. (2021). Deep learning for UAV navigation and control: A review. IEEE Transactions on Neural Networks and Learning Systems.

[8] Chen, L., et al. (2022). End-to-end learning for autonomous ground vehicle perception and control. IEEE Transactions on Intelligent Vehicles.

[9] Shi, X., et al. (2022). Deep learning-based target tracking and prediction: A survey. IEEE Transactions on Neural Networks and Learning Systems.

[10] Patel, R., et al. (2020). Sensor fusion using deep learning: A survey. IEEE Access.

[11] Kim, G., et al. (2018). Cyberattack detection in cyber-physical systems using machine learning algorithms. IEEE Access.

[12] POLITECNICO DI TORINO. *State and Parameter Estimation of Nonlinear Dynamics Systems*. Available at: https://webthesis.biblio.polito.it/28441/1/tesi.pdf

[13] *Nonlinear Dynamical Systems — School of Mathematical and Statistical Sciences*. Arizona State University. Available at: https://math.asu.edu/nonlinear-dynamical-systems

[14] Wikipedia contributors. *Nonlinear system*. Wikipedia. Available at: https://en.wikipedia.org/wiki/Nonlinear$_s ystem$

[15] *Nonlinear dynamics as an engine of computation*. Philosophical Transactions of the Royal Society A, 2016. Available at: https://royalsocietypublishing.org/doi/10.1098/rsta.2016.0222

[16] Wikipedia contributors. *Nonlinear control*. Wikipedia. Available at: https://en.wikipedia.org/wiki/Nonlinear$_c ontrol$