Advancing Time Series Forecasting: Insights from Deep Learning and Dynamic Mode Decomposition

Christopher Salazar

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:
Ashis Banerjee, Chair
Krithika Manohar
Shuai Huang

Program Authorized to Offer Degree:
Department of Industrial & Systems Engineering

University of Washington

Abstract

Advancing Time Series Forecasting:
Insights from Deep Learning and Dynamic Mode Decomposition

Christopher Salazar

Chair of the Supervisory Committee:
Ashis Banerjee
Industrial & Systems Engineering and Mechanical Engineering

Time series forecasting presents significant challenges across engineering and scientific disciplines, particularly in handling non-stationary real-world data and providing real-time predictions from streaming sources. Deep learning approaches, including Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformer-based models, have advanced the field but often fall short in interpretability, computational efficiency, and real-time adaptability. Despite their capacity for modeling complex non-linear dynamics, these models require extensive hyperparameter tuning and lack robust mechanisms for incremental updates. They also suffer from catastrophic forgetting in streaming scenarios, limiting their deployment in dynamic and resource-constrained environments. This dissertation addresses these limitations through two complementary research directions: enhancing deep learning interpretability through distance correlation analysis and developing efficient Dynamic Mode Decomposition (DMD) methods for batch and streaming forecasting.

First, this work introduces a distance correlation-based framework to examine the internal mechanics of RNNs in time series forecasting. This versatile metric enables systematic analysis of information flow through RNN activation layers, revealing how these networks process temporal dependencies. Empirical analysis demonstrates that RNN activation layers effectively learn lag structures in early layers but progressively lose this temporal information

in deeper layers, degrading forecast quality for series with large lag dependencies. The study further reveals fundamental limitations in RNN capabilities for modeling moving average and heteroskedastic processes. Distance correlation heatmaps provide visual comparisons across architectures and hyperparameters, demonstrating that input window size influences model behavior far more than conventional hyperparameters such as hidden units or activation functions. These findings enable practitioners to assess RNN suitability for specific time series characteristics without extensive trial-and-error experimentation.

The second direction introduces novel DMD-based forecasting methods that address deep learning limitations. For batch scenarios, Incremental Kernel Dynamic Mode Decomposition (IKDMD) enhances adaptability and efficiency by integrating incremental kernel singular value decomposition and randomized linear algebra into the kernel DMD framework. Comparative analysis across real-world datasets demonstrates that IKDMD outperforms state-of-the-art deep learning methods, particularly for highly non-stationary and volatile data, while providing interpretable eigenvalue diagnostics unavailable in black-box neural networks.

For streaming applications, this dissertation presents Windowed Online Random Kernel Dynamic Mode Decomposition (WORK-DMD), which integrates Random Fourier Features with online DMD to enable real-time forecasting from continuously arriving data. By employing explicit feature mappings rather than implicit kernel methods, WORK-DMD achieves fixed computational complexity per update while capturing nonlinear dynamics. Its adaptive windowing mechanism naturally handles non-stationary dynamics without catastrophic forgetting. Experimental evaluation across benchmark datasets demonstrates remarkable sample efficiency, requiring only single-pass learning while achieving competitive or superior accuracy compared to deep learning methods that demand multiple training epochs and extensive sample exposures. This efficiency translates to reduced computational costs, faster deployment, and viability for resource-constrained edge devices.

Together, these contributions advance time series forecasting by providing both diagnostic tools for understanding deep learning limitations and computationally efficient alternatives that balance accuracy, interpretability, and real-time adaptability. The methods presented enable practical deployment in scenarios where traditional deep learning approaches struggle with sample efficiency, computational constraints, and evolving data dynamics.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Time series forecasting is a pivotal challenge across various engineering and scientific domains, necessitating the development of advanced predictive models to effectively manage real-world data that often exhibits non-stationary behavior. Such data are characterized by continuous changes in their statistical properties, demanding robust and adaptive forecasting solutions. Beyond non-stationarity, modern applications increasingly require real-time forecasting from continuously streaming data, where traditional batch methods struggle with stringent computational constraints and the need for rapid adaptation to evolving patterns. In response, researchers have increasingly turned to deep learning models, including recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformer-based models with attention mechanisms, to address the complexities of non-linear time series dynamics. Recent advancements, such as DeepAR [99], attention-based frameworks [45], and transformer architectures [134], have significantly enhanced forecasting capabilities. However, despite their experimental successes, Dynamic Mode Decomposition (DMD) and its variants, a method often overlooked in the time series forecasting landscape, hold potential advantages for both batch and streaming forecasting scenarios. This dissertation aims to explore the capabilities of deep learning models, particularly RNNs, in time series forecasting and investigate the benefits that novel DMD methods offer over state-of-the-art machine learning models.

1.1 Interpretability of RNNs with Distance Correlation

While there are several deep learning models to choose from, RNNs are popular candidates for time series forecasting. This is due to their ability to handle sequential data, which is a

fundamental characteristic of time series. Yet, through a series of several experiments, RNNs and their adaptations [73, 61, 134, 81] have shown a wide range of forecasting results, making it difficult to evaluate their effectiveness. Experimental reviews and surveys have highlighted both the scenarios, one in which RNNs have achieved highly accurate results [74, 105, 50, 123, 117], and another in which they have been surpassed by traditional statistical regression models such as ARIMA [24] and gradient boosting trees [42]. Despite the breadth of these studies, their conclusions offer limited insights on the variations in RNN performance, often deferring to the explainability of the chosen hyperparameters or domain-specific hypotheses. These outlooks can be attributed to the restricted perspective offered by generalization error, which is the primary way of evaluating these models. While the generalization error is important to track, it does not delve into the internal mechanism of how time series inputs evolve through the RNN layers, limiting our understanding of this subject. As it stands, an evaluation tool that allows us to comprehend the inner workings of RNNs on a layer-by-layer basis is needed.

While there are several evaluation metrics to measure mathematical and statistical relationships, many lack the flexibility that is needed for studying the components of RNNs. A recently proposed statistical dependency metric, called distance correlation [110], has a few key advantages in this regard. First, it has the ability to capture the non-linear relationships present in typical RNN operations. Additionally, it can compare random variables of different dimensions, which are likely to occur in RNN architectures. It can also be used as a proximal metric to measure the information stored in the RNN activation layers through its training cycle [133, 116]. These factors make distance correlation well suited for a detailed characterization of RNN models based on their forecast performance.

1.2 Dynamic Mode Decomposition as a Forecaster

Though deep learning forecasting remains a popular framework, these models frequently struggle with interpretability, flexibility, and the complexity of hyperparameter tuning. Moreover, they lack efficient mechanisms for incremental updates, which are essential for

real-time data streaming applications. In response to these challenges, this dissertation explores the potential of Dynamic Mode Decomposition (DMD), a method traditionally used in fluid mechanics and system identification, now extended to broader time series forecasting applications [87]. DMD uses linear algebra and applied mathematics for system identification and trajectory forecasting of dynamical systems [71].

Although effective for physical systems, DMD's inherent linearity assumption is often inadequate for non-linear or non-stationary time series characteristics. Kernel DMD (KDMD) [67] has been adapted to handle such nonlinearities but lacks extensive benchmarking for generalized time series analysis in batch settings. Beyond these batch forecasting scenarios, streaming and real-time applications present additional challenges that neither standard DMD nor KDMD adequately address. Traditional DMD assumes stationary dynamics and fixed linear operators, limiting its effectiveness when system dynamics evolve continuously. While online DMD variants have been developed to enable incremental updates, these methods either remain fundamentally limited by their linear nature or face computational scalability issues when extended to kernel methods. Kernel-based approaches, despite their theoretical appeal for capturing nonlinear dynamics, encounter quadratic scaling in memory and computation as new data continuously arrive, making them impractical for streaming applications.

This research aims to fill these gaps by first enhancing KDMD's adaptability and efficiency for batch forecasting, particularly through innovations supporting incremental data integration. We then extend this foundation to streaming scenarios by developing a computationally efficient approach that captures nonlinear dynamics while maintaining fixed computational complexity per update and naturally adapting to non-stationary data without catastrophic forgetting.

1.3 Research Contributions and Organization

Given these general concepts between RNN interpretability-based methods and DMD as a forecaster for both batch and streaming scenarios, we present the following contributions

of this dissertation. First, we outline a distance correlation-based approach for evaluating and understanding time series modeling at the RNN activation layer level. Consequently, we demonstrate that the activation layers identify time series lag structures well. However, they quickly lose this important information over a sequence of a few layers, posing difficulties in modeling time series with large lag structures. Further, our experiments show that the activation layers cannot adequately model moving average and heteroskedastic processes. Last, we employ distance correlation to visualize heatmaps that compare RNNs with varying hyperparameters. These heatmaps show that certain hyperparameters, such as the number of hidden units and activation function, do not influence the activation layer outputs as much as the RNN input size. A visual overview of our work is shown in Figure 1.1.

With regards to DMD advancements in forecasting, this dissertation addresses existing gaps in time series forecasting methodologies through three major contributions. First, for batch forecasting scenarios, we develop and enhance the adaptability and efficiency of Kernel DMD for univariate time series under non-stationary behavior. To achieve this, we adapt univariate time series data into a dynamical system framework suitable for a batch Kernel DMD approach, consistent with standard DMD procedures [71]. Our specific contributions for batch forecasting are organized as follows:

1. **Mathematical Implementation:** We mathematically develop and integrate the Incremental Kernel Dynamic Mode Decomposition (IKDMD) using an incremental kernel singular value decomposition method [33]. Additionally, we introduce a Randomized Kernel DMD (RKDMD) that efficiently approximates kernel functions using randomized Fourier features [51] and randomized linear algebra techniques [94].

2. **Benchmarking:** We benchmark IKDMD and RKDMD against state-of-the-art deep learning models, such as Informer [134], Non-Stationary Transformer [85], and DLinear [128]. Our analysis shows that our methods outperform these models on five out of six real-world datasets exhibiting non-stationarities, demonstrating their effectiveness. We also show superior inference times compared to the benchmark models.

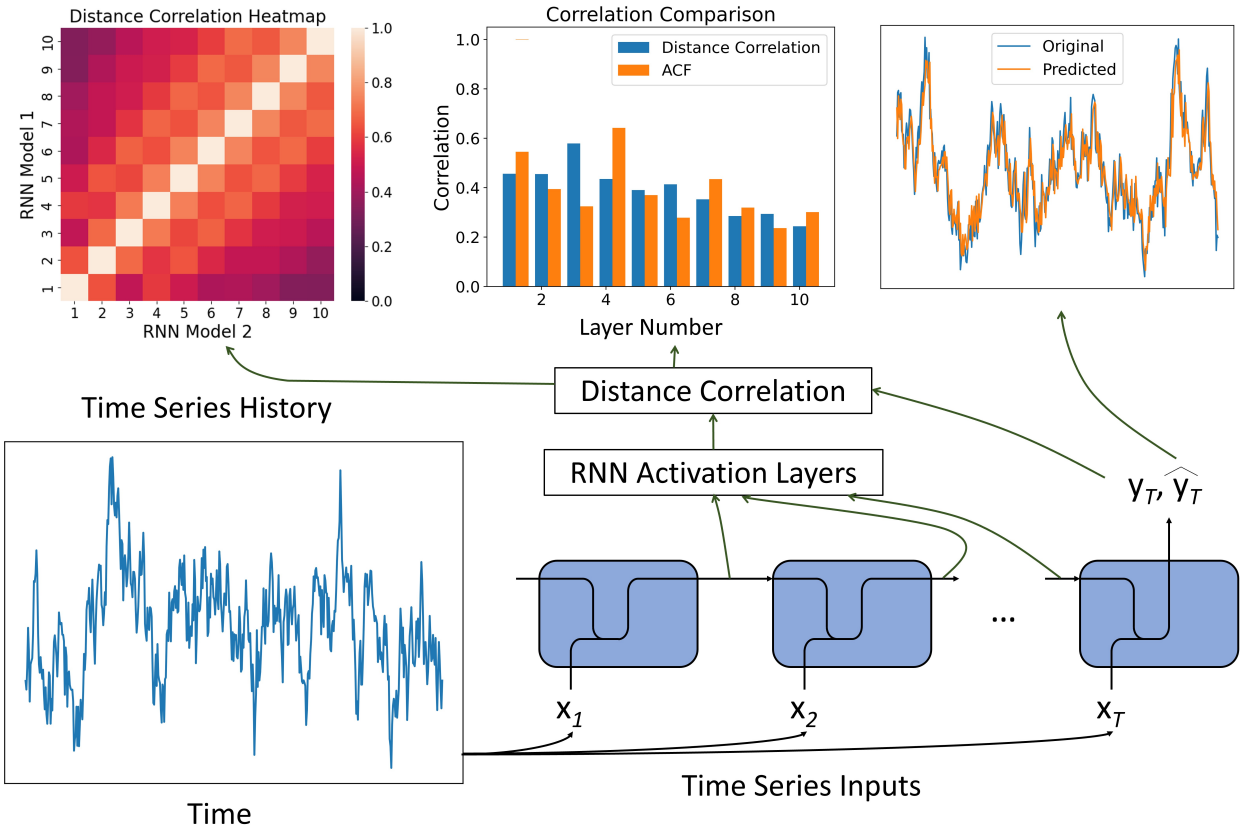


Figure 1.1: Overview of the use of distance correlation to examine time series forecasting using a recurrent neural network (RNN). We begin with a time series history that comprises the inputs x_t for the RNN and the predicted outputs $\hat{y}_T$. The outputs for each activation layer and ground truth values, y_T, are extracted and processed with distance correlation. This is then used to generate correlation plots for analyzing activation layers behaviors and visualizing heatmaps for comparisons of different RNN models.

3. **Visual Demonstrations:** We provide visual demonstrations that illustrate how our methods more effectively approximate complex, volatile time series data, such as financial returns and generalized autoregressive conditionally heteroscedastic processes. These visualizations also highlight the shortcomings of popular transformer models in

handling such data types.

Last, extending beyond batch forecasting, we develop WORK-DMD (Windowed Online Random Kernel Dynamic Mode Decomposition), which addresses the unique challenges of streaming time series forecasting. This contribution integrates Random Fourier Features with online DMD updates to enable efficient nonlinear modeling for streaming data while maintaining the interpretable eigenvalue structure of DMD. Our specific contributions for online forecasting include:

1. **Efficient Kernel Approximation:** We present the first integration of Random Fourier Features [94] with online DMD, enabling kernel-based nonlinear modeling that maintains fixed computational complexity per update, avoiding the quadratic scaling issues of implicit kernel methods.

2. **Streaming Adaptation:** We develop a rolling window mechanism coupled with online updates that naturally handles non-stationarity by continuously incorporating new patterns while discarding outdated information, without suffering from catastrophic forgetting.

3. **Single-Pass Learning:** Unlike deep learning methods requiring extensive training datasets and multiple epochs, WORK-DMD requires only the current windowed snapshot to generate predictions, making it particularly suitable for applications with limited historical data or scenarios requiring rapid deployment.

4. **Comprehensive Evaluation:** We provide extensive benchmarking against state-of-the-art online forecasting methods, demonstrating competitive or superior performance with substantially lower data requirements and computational overhead.

Through these contributions, we not only advance the theoretical framework of Kernel DMD for both batch and streaming scenarios but also provide practical insights and tools for

more accurate, robust, and computationally efficient time series forecasting across diverse application domains.

To effectively guide this research, the dissertation is organized into subsequent chapters as follows. Chapter 2 reviews the related work on RNNs, distance correlation, and interpretability-based methods for deep learning models. It also discusses the role of Dynamic Mode Decomposition (DMD) in time series forecasting for both batch and streaming approaches. Chapter 3 details the development of our interpretability-based method using RNNs and distance correlation. It then presents experimental results from this analysis. Chapter 4 introduces the novel Incremental Kernel Dynamic Mode Decomposition (IKDMD) and Randomized Kernel DMD (RKDMD) methods for batch forecasting. This chapter includes experimental outcomes demonstrating their effectiveness. Chapter 5 presents WORK-DMD, our streaming forecasting approach that extends DMD to online learning scenarios through Random Fourier Features and efficient online updates. Finally, Chapter 6 synthesizes the findings across all contributions and discusses their implications for practical forecasting applications. It concludes with future research directions for this work.

Chapter 2

BACKGROUND OF TIME SERIES FORECASTING METHODS

Time series analysis constitutes a multifaceted research domain that has evolved significantly throughout its history. It encompasses a wide array of topics such as forecasting [43], anomaly detection [106], change point detection [10], and classification [6], among others. This dissertation primarily concentrates on forecasting methods, which we conceptualize as a regression task. Notwithstanding, it is pertinent to acknowledge that the aforementioned methodologies can augment and refine forecasting techniques. Given the expansive scope of forecasting, this section will specifically narrow its focus to pertinent studies discussed in Chapters 3, 4, and 5. Consequently, this chapter is structured as follows: 1) Statistical Methods for Forecasting; 2) Machine Learning for Forecasting; 3) Interpretability of Machine Learning Forecasting Models; 4) Dynamic Mode Decomposition; and 5) Online and Streaming Forecasting Methods.

2.1 Statistical Methods for Forecasting

Statistical methods have long served as the cornerstone for time series forecasting, offering several modeling and prediction frameworks. Autoregressive (AR) models, one of the most prevalent approaches, leverage past values to predict future observations, assuming that the future value is a linear combination of past observations [25]. Complementing AR models, Moving Average (MA) models incorporate past forecast errors into the prediction equation to enhance accuracy [25]. For more complex non-stationary data, Autoregressive Integrated Moving Average (ARIMA) models extend AR and MA by differencing the data, thereby stabilizing the mean [25]. Addressing the volatility clustering common in financial time se-

ries, Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models predict future variance based on past observed variances and forecast errors, providing critical insights into risk management [47]. However, the primary limitation of these linear models is their assumption of linearity and stationary the in the data, which can lead to inaccuracies when dealing with non-linear or non-stationary behaviors prevalent in real-world data. This limitation necessitates the exploration of non-linear and decomposition methods, which offer advanced techniques for capturing complex patterns not addressable by linear models alone.

Non-linear and decomposition methods provide sophisticated alternatives for capturing the complex dynamics inherent in time series data. Decomposition techniques like Seasonal and Trend Decomposition using Loess (STL) effectively disaggregate a time series into its trend, seasonal, and residual components, enhancing the clarity of pattern analysis [106]. Exponential smoothing methods, particularly the Holt-Winters approach, adjust to shifts in trend and seasonality by applying weighted averages to past observations, proving invaluable for predicting short-term fluctuations [63]. Frequency-based methods such as the Fourier Transform offer insights into periodic patterns by analyzing time series in the frequency domain, revealing dynamics not readily apparent in the time domain. Extending this concept, spectral analysis estimates the spectral density of a series, an essential technique for uncovering hidden periodicities and cycles [62]. These methods, by challenging the basic assumptions of linearity, enable more accurate and detailed analyses. However, they also introduce challenges such as parameter sensitivity, the complexity of capturing multi-scale relationships, and a susceptibility to overfitting. These limitations emphasize the need for meticulous method selection and parameter adjustment to ensure reliable forecasting outcomes, especially in complex data scenarios.

The exploration of statistical methods for time series forecasting, both linear and non-linear, establishes a solid foundation for understanding data dynamics. However, the limitations of these methods have opened opportunities for advanced approaches. Transitioning to machine learning in the next section, we will explore how these models leverage large datasets and complex relationships, offering enhanced forecasting capabilities beyond tradi-

tional statistical techniques.

2.2 Machine Learning for Forecasting

The ascent of machine learning models has captivated the forecasting community, with supervised machine learning models predominating across diverse forecasting applications. Notably, Zhang et al. [130] utilized an XGBoost model for predictions within the retail sector. Other efforts have incorporated Gaussian Processes Regression for industrial time series forecasting and methods such as decision trees, random forests, and generalized linear regression models [31]. Despite mixed success, there is a pronounced shift towards deep learning models, which have been the focal point of recent research endeavors.

Deep learning presents a spectrum of architectures suited for time series forecasting tasks. The simplest of these, the Multi-Layer Perceptron (MLP), has been applied in various scenarios, including forecasting COVID-19 mortality rates [21]. Recurrent Neural Networks (RNNs), designed explicitly for sequential data, along with their derivatives like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), offer robust processing capabilities across numerous applications [59]. Furthermore, Convolutional Neural Networks (CNNs) have been tailored for specific forecasting tasks, such as gold price predictions [86]. The surge in proposed architectures, particularly hybrid models, aims to amalgamate the best features of individual models to create superior forecasting tools [53]. Moreover, Transformer-based architectures have recently begun to lead the field in time series forecasting innovations.

The advent of attention mechanisms in Transformers has revolutionized forecasting methodologies. Models such as the Informer [134], Non-stationary Transformers [85], and Fedformer [135] leverage the enhanced memory capabilities of Transformers to forge cutting-edge techniques for precise and efficient forecasting. This paradigm shift has inspired viewing time series as a linguistic sequence, leading to the development of the Language Based Model for forecasting, known as Chronos [13]. More recently, foundation models have emerged as a powerful paradigm in time series forecasting. These models, trained on massive datasets comprising billions of observations, demonstrate impressive zero-shot capabilities across di-

verse domains. TimesFM [37], trained on 307 billion time points, and Moirai [121], trained on 27 billion observations, represent the current state-of-the-art in foundation model approaches. These models leverage pre-training on heterogeneous time series data to achieve strong performance without task-specific fine-tuning. Nonetheless, the efficacy of Transformers in time series forecasting remains under scrutiny. Zeng et al. [128] highlighted that a well-crafted MLP could outperform several Transformer models across diverse benchmarks, indicating that while deep learning methods hold promise, their practical effectiveness varies significantly.

While the proliferation of deep learning models has significantly enhanced the capabilities of time series forecasting, their intricate architectures often lead to a 'black box' scenario where the decision-making process is not transparent. This lack of clarity presents substantial challenges, especially in industries where understanding and trust in predictive models are critical. The next section will delve into interpretability-based methods designed to clarify the inner workings of deep learning models. These methods aim to improve transparency, thereby increasing trust, aiding in the detection and correction of biases, and enhancing the reliability and accountability of forecasting systems.

2.3 Interpretability of Deep Learning Forecast Models

Interpreting deep learning models remains a formidable challenge in time series forecasting research. The opacity of these models, often referred to as the black box problem, has motivated significant research into understanding their decision-making processes [52]. Despite the diverse architectures discussed previously, this section focuses particularly on the Recurrent Neural Network (RNN) architecture. We aim to enhance understanding and interpretability within the context of time series forecasting, exploring how these complex models can be made more transparent and their decisions more comprehensible.

Neural networks, despite their tremendous success, are not fully understood from a theoretical perspective. Some strides have been made, such as the work by Telgarsky [114], where the greater approximation power of deeper networks is proved for a class of nodes that

includes ReLU and piecewise polynominal functions. Rolnick's work [95] extended this rigorous understanding of neural networks by establishing the essential properties of expressivity, learnability, and robustness. Bahri *et al.* [15] recently reviewed a breadth of research on the connections between deep neural networks and several principles of statistical mechanics to obtain a better conceptual understanding of deep learning capabilities. To estimate the overall complexity and learning ability of deep neural networks without actual training and testing, Badias and Banerjee proposed a new layer algebra-based framework to measure the intrinsic capacity and compression of networks, respectively [14]. These measurements enabled us to analyze the performance (accuracy) of state-of-the art networks on benchmark computer vision datasets.

Specifically for RNNs, one of the more well-known drawbacks is their inability to retain information from long term sequences, as found by Bengio *et al.* while using gradient-based algorithms for training RNNs [19]. This finding led to the use of alternate RNN architectures, such as the Long Short-Term Memory (LSTM) network [60], despite their increased computational complexity. More recently, Chen *et al.* [32] established generalization bounds for RNNs and their variants based on the spectral norms of the weight matrices and the network size. A different approach focused on studying the implicit regularization effects of RNN by injecting noise into the hidden states, which resulted in promoting flatter minima and overall global stability of RNNs [82]. While all these approaches are theoretically rigorous, they are typically formulated under rigid constraints, many of which are not necessarily satisfied in real-world applications.

Therefore, several researchers have turned to interpretation based methods to develop a principled yet practically useful understanding of RNNs. In this context, since RNNs are common for natural language processing (NLP) tasks, many studies have focused on tracking the hidden memories of RNNs and their expected responses to the input text [89, 109]. For time series, Shen *et al.* [107] developed a visual analytics system to interpret RNNs for high dimensional (multivariate) forecasting tasks. While this system is very useful, we aim to approach RNN interpretability from a different perspective of understanding performance

(forecast) generalizability by tracking how the hidden memories respond to specific time series characteristics. For this purpose, we look at alternate information-theoretic approaches.

One of the most promising works on information-theoretic analysis was conducted by Schwartz and Tishby [108]. They used mutual information (MI) to represent a deep neural network as a series of encoder-decoder networks, enabling analysis at the activation layer level. Such analysis goes beyond the typical generalization errors used for model assessment, and allows us to closely observe the information flow through deep learning networks such as RNNs. Although this approach is novel, MI estimation is challenging for real-world data. To address this challenge, researchers have investigated different approaches, such as a k-nearest neighbor method [70] and a semi-parametric local Gaussian method [49]. While these attempts seem promising, there are questions regarding their practical usefulness. For example, McAllester and Stratos [88] argue that without any known probability characteristics of the data, any distribution-free, high-confidence lower bound on mutual information would require an exponential amount of samples. It is quite rare to find real world datasets with known (or well-fitting) distributions, and this is generally true for time series applications [25]. Nevertheless, the underlying framework of analyzing RNNs from a layer-by-layer perspective is a useful concept that we can build upon using an alternative metric, as discussed next.

As we search for an approach that affords the flexibility of analyzing the components of RNNs, Szekely *et al.'s* [112] distance correlation comes to the forefront as a dependency measure. It is closely related to Pearson's correlation with the advantage of measuring non-linear relationships among the random variables. Naturally, some researchers directly apply it toward time series forecasting. Zhou's [136] work re-purposes distance correlation by proposing the auto-distance correlation function (ADCF), which is an extension of the auto-correlation function (ACF) [25]. Yousuf and Feng [125] use distance correlation as a screening method for deciding which lagged variables should be retained in a model.

Beyond time series applications, distance correlation has other practical uses in the deep learning space. Zhen *et al.* [133] outline a process that uses distance correlation and partial

distance correlation to compare deep learning models from a network layer level. They use it as a measure of information lost or gained in the network layers, where high distance correlation values indicate more information is stored in the activation layers. In a separate study, NoPeek [116] uses a distance correlation method that decorrelates the raw input information within the activation layers during training, which is essential for preventing the leakage of sensitive data. These studies show the viability of distance correlation as an analysis tool in time series applications and deep learning, separately. Our work aims to connect the two topics to further our understanding of time series forecasting tasks.

2.4 Dynamic Mode Decomposition

DMD has emerged as a significant analytical tool for dynamical systems. Originally adopted by the fluid dynamics community [97, 103], its application has broadened to fields such as video surveillance, epidemiology, neurobiology, and financial engineering [104]. Variants like optimized-DMD [102], DMD with control [93], and Extended DMD [120] have been introduced, providing targeted enhancements for specific applications. Standard DMD methods assume stationary dynamics with fixed linear operators governing temporal evolution. However, this stationarity assumption becomes a critical limitation when applied to non-stationary time series, where regime changes occur and system dynamics evolve continuously [93].

The application of standard DMD to univariate time series forecasting [115] offers a foundational framework, although it primarily facilitates time series reconstruction and limited short-term forecasting. The standard approach may not adequately address the non-linear or non-stationary dynamics frequently observed in real-world time series. Kernel DMD [67] attempts to overcome these limitations by applying DMD within non-linear functional spaces through implicit kernel mappings. Williams et al. [120] demonstrated that kernel methods can capture complex nonlinear dynamics by implicitly mapping data to high-dimensional feature spaces. However, kernel-based approaches face fundamental computational scalability challenges. The requirement to invert covariance matrices leads to computational complex-

ity that grows quadratically with the number of training points [20]. This quadratic scaling in both memory and computation creates particular difficulties for streaming applications, where maintaining and updating kernel matrices becomes increasingly expensive as new data continuously arrive.

To address computational efficiency while maintaining the ability to capture nonlinear dynamics, randomized methods have emerged as a promising alternative. Random Fourier Features [94] provide a mechanism to explicitly approximate kernel functions in finite-dimensional spaces, avoiding the computational pitfalls of implicit kernel methods. This approach has been successfully integrated with DMD through randomized linear algebra techniques [51], demonstrating that explicit feature mappings can achieve comparable performance to implicit kernel methods while maintaining fixed computational complexity.

2.5 Online and Streaming Forecasting Methods

The deployment of forecasting models in real-time streaming environments presents unique challenges that distinguish these scenarios from traditional batch learning settings. Streaming data applications require models that can adapt quickly to changing dynamics while maintaining computational efficiency and avoiding catastrophic forgetting, where models lose knowledge of past patterns when adapting to new data [69].

2.5.1 Deep Learning Approaches for Streaming Forecasting

Modern deep learning approaches have attempted to address streaming forecasting challenges with varying degrees of success. DeepAR [99] pioneered probabilistic forecasting with autoregressive recurrent networks, though its deployment in streaming scenarios requires careful consideration of retraining strategies. More recent transformer-based models have demonstrated impressive capabilities but face significant computational overhead in online settings. Foundation models such as TimesFM [37] and Moirai [121] achieve remarkable zero-shot performance through massive pre-training, yet their computational requirements and difficulty in efficiently adapting learned representations to evolving data distributions

make real-time deployment challenging.

Several specialized online learning methods have emerged to address these limitations. FSNet [92] leverages frequency domain representations for multi-scale temporal modeling, enabling more efficient processing of temporal patterns. OneNet [118] proposes a unified framework designed to handle diverse forecasting scenarios with improved adaptability. Continual learning techniques have also been explored to mitigate catastrophic forgetting. Methods such as DER++ [28] employ experience replay mechanisms combined with regularization strategies to preserve knowledge of past patterns while adapting to new data. However, these approaches typically still require substantial training data and multiple passes through historical observations, limiting their applicability in resource-constrained streaming environments where rapid adaptation is essential.

Beyond computational constraints, deep learning models present fundamental interpretability challenges that complicate their deployment in streaming applications. Recent work has sought to address this opacity through distance correlation-based approaches that characterize information flow through neural network layers [98]. This analysis reveals how RNNs gradually lose temporal information across activation layers and struggle with certain time series characteristics such as moving average and heteroskedastic processes. Such interpretability challenges, combined with their computational demands and catastrophic forgetting issues, underscore the need for more transparent and efficient alternatives for streaming forecasting scenarios.

2.5.2 Online Dynamic Mode Decomposition

To address the limitations of standard DMD in streaming contexts, several online variants have been developed. Streaming DMD [58] introduced methods for handling large datasets through incremental updates, enabling the model to process new data as it arrives without requiring complete retraining. Liew et al. [80] demonstrated practical applications of streaming DMD for short-term wind farm forecasting, showing domain-specific benefits. However, these linear approaches remain fundamentally limited in their ability to model complex non-

linear dynamics present in broader real-world time series.

Zhang et al. [129] proposed a more sophisticated online DMD framework that addresses computational efficiency through Sherman-Morrison matrix updates. This method provides an elegant solution for incremental updates of matrix decompositions, demonstrating effectiveness for time-varying dynamical systems. The Sherman-Morrison formula enables efficient rank-one updates to matrix inverses, allowing the DMD model to incorporate new observations with fixed computational complexity per update. While this approach establishes a solid foundation for online DMD, it has not been extended to handle nonlinear dynamics through kernel methods, and its evaluation has been limited to specific dynamical systems rather than general time series forecasting benchmarks.

The challenge of extending online DMD to capture nonlinear dynamics while maintaining computational efficiency remains largely unaddressed. Kernel methods, despite their theoretical appeal, face the quadratic scaling issues discussed previously. This gap motivates the development of approaches that combine the adaptability of online DMD with efficient nonlinear modeling capabilities, forming the foundation for the WORK-DMD method presented in Chapter 5 of this dissertation.

Chapter 3

ASSESSING RNN EFFECTIVENESS IN TIME SERIES FORECASTING: A DISTANCE CORRELATION APPROACH

Time series forecasting remains a pivotal yet complex field, necessitating robust and interpretable methods to accurately predict dynamics in diverse areas such as weather, stock markets, and supply chain operations. Despite challenges such as data non-stationarity, seasonality, and inherent uncertainties, the evolution from classical models like ARIMA to sophisticated deep learning networks marks significant progress. Recent advances, exemplified by the work of Zhang et al. [132] and the innovative information-aware attention dynamic synergetic network [57], demonstrate the efficacy of Recurrent Neural Networks (RNNs) in handling sequential data [73, 61, 134, 81]. However, the opaque nature of these models often obscures the decision-making processes, creating a barrier to their practical application. This chapter introduces a novel approach employing distance correlation [110] to dissect the complexities of RNNs, aiming to enhance understanding of their internal mechanics and forecasting effectiveness. We will discuss our methods, present experimental data, and engage in discussions that reveal how RNN activation layers manage time series data, providing critical insights into the impact of model hyperparameters and structural data characteristics. These findings aim to improve both the interpretability and reliability of forecasting models.

3.1 Methodology

In this section, we first characterize the time series forecasting problem before describing the components of an RNN and providing an overview of distance correlation. We then propose a method that links these core topics to further our understanding of time series forecasting

using RNNs.

3.1.1 Time Series Forecasting

As we define the forecasting problem, we adopt the notation used by Lara-Benitez et al. [74] in their experimental review of deep learning models with forecasting tasks. We let $\mathbf{Z} = z_1, z_2, \ldots, z_L$ be the historical time series data of length L and H be the prediction horizon. The goal of a time series forecasting problem is to use $\mathbf{Z}$ to accurately predict the next horizon values $z_{L+1}, z_{L+2}, \ldots, z_{L+H}$. The time series data can either be univariate or multivariate; we focus this paper on univariate analysis where single and sequential observations are recorded over time.

One of the challenges of time series research is the lack of benchmark datasets for forecasting. Many time series experimental reviews [74, 123, 117] choose datasets from several domains with various characteristics. This makes it difficult to generalize the behavior of the forecasting models since it is not clear whether each dataset represents a class of time series processes or is uniquely domain specific. It also complicates the interpretability of the experiments and their corresponding conclusions.

To circumvent these issues, we experiment primarily with synthetic data from known time series processes (see Section 3.2.1). We, however, still use real world time series data to provide a practical context for our proposed forecasting model evaluation tool.

Given our data, it is important for us to establish a baseline of the time series structure. One of the primary methods for analyzing time series forecasting problems is by using the auto-correlation function (ACF) [25]. This function looks at the linear correlations between z_l and its previous values z_{l-h} where $l = 1, 2 \ldots, L$ at lag h. Formally, the auto-correlation function, $\rho_z(h)$, is defined as

$$\rho_z(h) = Cor(z_l, z_{l-h}) \tag{3.1}$$

where Cor represents the Pearson's correlation function. Plotting the ACF is a common method for understanding a time series lag structure. An example of this auto-correlation

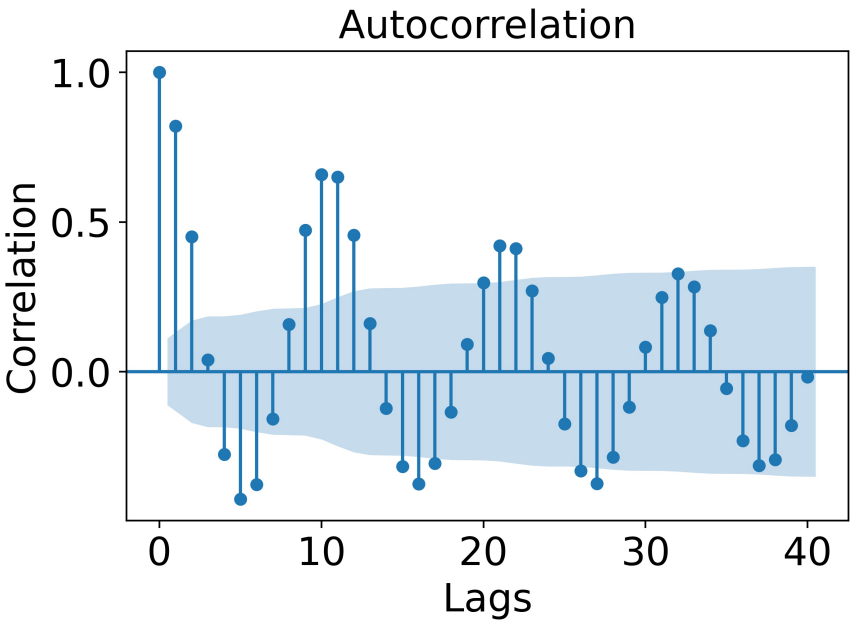


Figure 3.1: An example of an auto-correlation plot for the sun spot time series data [30] with blue shaded significance level band. Lags that fall outside of significance level band are considered important to include in a forecasting model. It also provides a way to describe the time series data, as the cyclical nature of the lags from this plot indicates a seasonal characteristic to sun spot observations.

plot is shown in Figure 3.1 for the sun spot time series data [30].

A final key aspect to consider is the size of the prediction horizon H. In this study, we stick with single-step ahead forecasting ($H = 1$) to create a simple experimental environment. This choice allows us to minimize the model complexity required for multi-step forecasting and reduces the error accumulation with having a larger H.

3.1.2 RNN Architecture

Considering that we are investigating recurrent neural networks (RNNs), we must define all its components. RNN was originally proposed by Elman [41], where they altered the

classical feedforward network layer structure into a recurrent layer. This change in architecture allowed the outputs of a previous layer to become the inputs of the current layer, thereby creating a compact model representation for sequential input data. Figure 3.3 shows an unfolded RNN displaying the inputs, an activation function operation (i.e., tanh), and outputs (hidden state). While the Elman RNN is relatively simple compared to other model architectures, focusing on this network provides a baseline of how recurrent based networks learn time series structures.

A necessary step for modeling time series with an RNN is preparing the full univariate time series history into samples that purposefully generate the input-output sequences. We adopt the moving window procedure described by [74] to create the input and output samples. Given a full time history $\mathbf{Z}$, a window of fixed size T slides over the data to create samples of input $\mathbf{x}_i$ in $\mathbb{R}^T$ and a corresponding output $\mathbf{y}_i$ in $\mathbb{R}^H$, where $i = 1, 2, \ldots, n$. A visual example of generating these input-output samples is shown in Figure 3.2 for $T = 5$ and $H = 1$.

The input-output sample vectors, $(\mathbf{x}_i, \mathbf{y}_i)$, are currently globally represented in terms of their z_l values. However, it is beneficial to convert our vectors into a sample representation that serve as local inputs for our RNN model. To do this, we denote our samples as $\mathbf{x}_i = [x_{1,i}, x_{2,i}, ..., x_{T,i}]$, which represents the ith sample of a sequence of T time series values. For each component of $\mathbf{x}_i$, we recover its relation to the time series history using $x_{t,i} = z_{t+i-1}$, where $t = 0, 1, \ldots, T$. Similarly, the output of the RNN is a vector $\hat{\mathbf{y}}_i = [\hat{x}_{T+1,i}]$, which represents the ith forecasting output for a single-step ahead value beyond T, or $H = 1$. We measure the output error with reference to the ground truth single-step ahead vector $\mathbf{y}_i = [x_{T+1,i}]$.

Lastly, we define the activation layer outputs (or hidden states) in the context of our time series data and the RNN architecture. Considering an input-output sample $(\mathbf{x}_i, \mathbf{y}_i)$, the hidden state $\mathbf{a}_{t,i}^{(p)}$ represents the ith sample output of an activation layer at time step t and epoch p, where $p = 1, 2, \ldots, P$. The activation layer output $\mathbf{a}_{t,i}^{(p)}$ is a vector in $\mathbb{R}^b$, where b specifies the width of the hidden state or the number of hidden units. Each activation layer output is produced from tensor operations with the sample inputs that pass through some

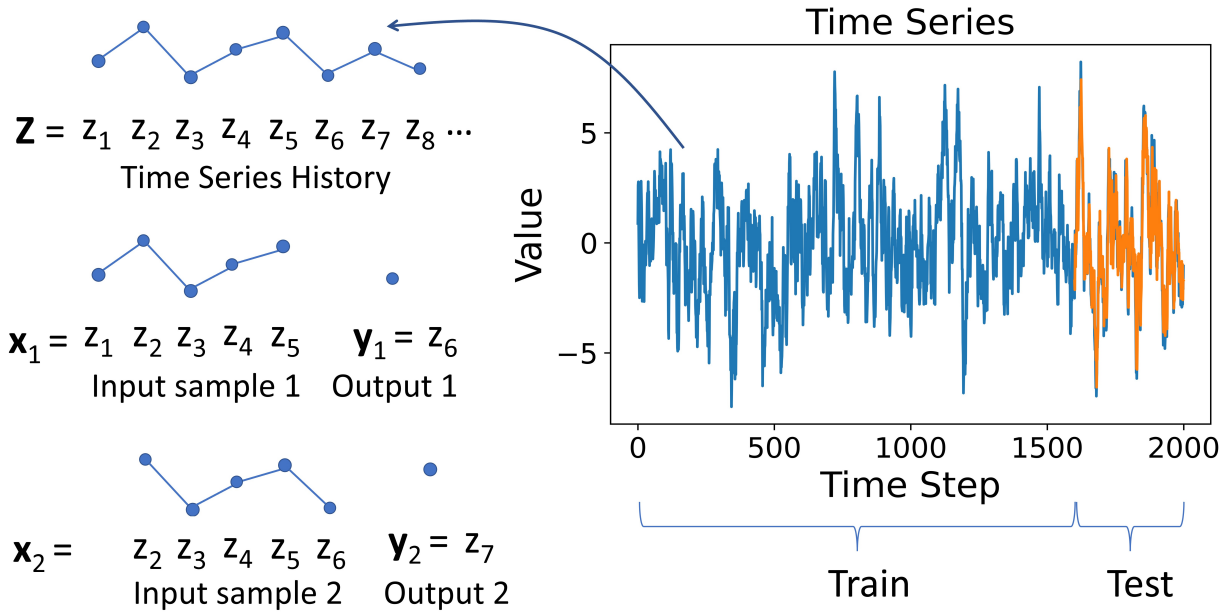


Figure 3.2: Time series sampling strategy where a full univariate time series plot is displayed. This full time series is partitioned into an 80:20 training:test split. Each of these training and testing splits are further divided into input-output samples, where each sample is generated via a sliding window. In this example, our sliding window is of size $T = 5$ and prediction horizon $H = 1$.

activation function f. The time step t for the activation layer output $\mathbf{a}_{t,i}^{(p)}$ doubles as an indicator for the layer number. For example, $t = 2$ corresponds to the input $x_{2,i}$ and $\mathbf{a}_{2,i}^{(p)}$, the second activation layer for training epoch p. In order to output a single prediction value, we must collapse the dimension of the final activation layer output, $\mathbf{a}_{T,i}^{(p)}$, with a dense layer. A visualization of this unrolled RNN setup with the time series forecasting problem is shown in Figure 3.3.

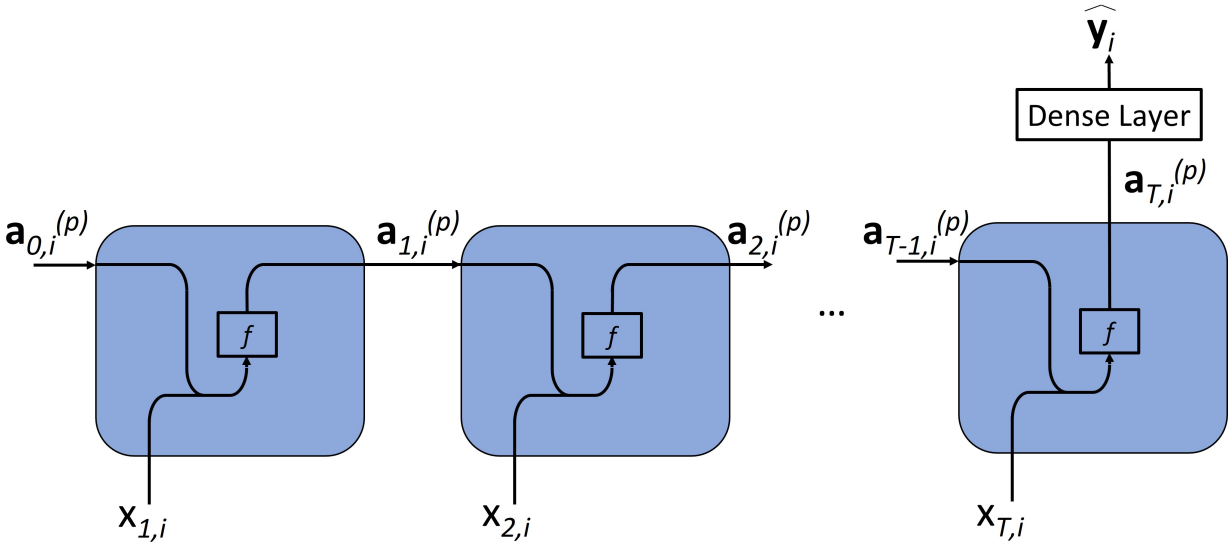


Figure 3.3: Time series input-output structure with an unfolded RNN for the ith sample and training epoch p. The ith input of the RNN $\mathbf{x}_i = [x_{1,i}, x_{2,i}, ..., x_{T,i}]$ produces activation layer outputs $\mathbf{a}_{t,i}^{(p)}$ for $t = 0, \ldots, T$ using a predetermined activation function f. The final activation layer output $\mathbf{a}_{T,i}^{(p)}$ is fed into a dense layer to produce a single-step ahead forecast value $\hat{\mathbf{y}}_i = \{\hat{x}_{T+H,i}\}$ for $H = 1$.

3.1.3 Distance Correlation

An approach for examining the relationships among random variables is through the use of energy statistics [111]. Energy statistics comprise a set of functions derived from the distances of statistical observations, inspired by the gravitational potential energy between two bodies. We primarily use the distance correlation metric from energy statistics [112] to measure the statistical dependence between random variables. Distance correlation is tailored to measure the non-linear relationships between two random variables, not necessarily of equal sizes. We build the empirical definition of distance correlation by relating it some of our previously defined time series components. We start with our input-output samples as $\mathbf{X} \in \mathbb{R}^{T \times n}$ and $\mathbf{Y} \in \mathbb{R}^{H \times n}$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \\ | & | & & | \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_n \\ | & | & & | \end{bmatrix}$$

where the columns $\mathbf{x}_i \in \mathbb{R}^T, \mathbf{y}_i \in \mathbb{R}^H$ are the input-output sample vectors defined in section 3.1.2. From the observed samples $\mathbf{X}, \mathbf{Y}$, define

$$b_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| \quad \overline{b}_{i\cdot} = \left(\sum_{j=1}^{n} b_{ij}\right)/n \quad \overline{b}_{\cdot j} = \left(\sum_{i=1}^{n} b_{ij}\right)/n$$

$$\overline{b}_{\cdot\cdot} = \left(\sum_{i,j=1}^{n} b_{ij}\right)/n^2 \quad B_{ij} = b_{ij} - \overline{b}_{i\cdot} - \overline{b}_{\cdot j} + \overline{b}_{\cdot\cdot}$$

These are similarly defined for $\overline{c}_{i\cdot}, \overline{c}_{\cdot j}, \overline{c}_{\cdot\cdot}$ where $c_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|$ and $C_{ij} = c_{ij} - \overline{c}_{i\cdot} - \overline{c}_{\cdot j} + \overline{c}_{\cdot\cdot}$. Given this set of double centered matrices, the sample distance covariance function [112] is defined as

$$\hat{V}^2(\mathbf{X}, \mathbf{Y}) = \frac{1}{n^2} \sum_{i,j=1}^{n} B_{ij} C_{ij} \quad . \tag{3.2}$$

Finally, the empirical distance correlation is defined as

$$\hat{R}^2(\mathbf{X},\mathbf{Y}) = \begin{cases} \frac{\hat{V}^2(\mathbf{X},\mathbf{Y})}{\sqrt{\hat{V}^2(\mathbf{X})\hat{V}^2(\mathbf{Y})}}, & \hat{V}^2(\mathbf{X})\hat{V}^2(\mathbf{Y}) > 0 \\ 0, & \hat{V}^2(\mathbf{X})\hat{V}^2(\mathbf{Y}) = 0 \end{cases} \tag{3.3}$$

The empirical distance correlation function has the following properties [39]:

- $\hat{R}(\mathbf{X},\mathbf{Y})$ converges almost assuredly to the theoretical counterpart [112] provided $n \to \infty$ and $E[\|\mathbf{X}\|] < \infty$ and $E[\|\mathbf{Y}\|] < \infty$.
- $\hat{R}(\mathbf{X},\mathbf{Y}) = 0$ denotes the independence of $\mathbf{X}$ and $\mathbf{Y}$
- $0 \leq \hat{R} \leq 1$

Additionally, another key property shown by [110], for a fixed n

$$\lim_{T,H\to\infty} \hat{R}(\mathbf{X},\mathbf{Y}) = 1 \tag{3.4}$$

provided that $\mathbf{X}$ and $\mathbf{Y}$ are independently and identically distributed (i.i.d.) and their corresponding second moments exist.

3.1.4 Analysis of RNN Activation Layers

We now provide the framework for using distance correlation to study how the RNN activation layer outputs learn time series structures that eventually translate to forecasting outputs. We start with a set of samples $(\mathbf{x}_1,\mathbf{y}_1)$, $(\mathbf{x}_2,\mathbf{y}_2)$, ... , $(\mathbf{x}_n,\mathbf{y}_n)$ where each $\mathbf{x}_i$ is an RNN input to produce a forecasting value $\hat{\mathbf{y}}_i$. This is compared with the ground truth output $\mathbf{y}_i$ using a loss function. For each sample fed into the RNN during training, a unique activation layer output $\mathbf{a}_{t,i}^{(p)}$ is produced depending on the sample i, time step t, and epoch p. The accumulation of all the samples for a particular activation layer output at time step t and epoch p is represented as a matrix

$$\mathbf{A}_t^{(p)} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_{t,1}^{(p)} & \mathbf{a}_{t,2}^{(p)} & \dots & \mathbf{a}_{t,n}^{(p)} \\ | & | & & | \end{bmatrix}$$

With our representation of sample ground truth outputs $\mathbf{Y}$, we estimate the distance correlation between the activation layer output at time t and epoch p, and the ground truth outputs. This is denoted by $\hat{R}(\mathbf{A}_t^{(p)}, \mathbf{Y})$, and is calculated using (3.3). We also estimate the dependency between two activation layers, $\mathbf{A}_v^{(p)}, \mathbf{A}_m^{(p)}$, at different time steps using $\hat{R}(\mathbf{A}_v^{(p)}, \mathbf{A}_m^{(p)})$, which is the distance correlation between the activation layer outputs at time steps v and m at epoch p. We note that for both the scenarios, the dimensions between the two random variables need not be the same. This framework guides the experiments in section 3.2.

3.2 Experiments

We now conduct a series of experiments to showcase the usefulness of our method for assessing time series forecasting using RNN models. Using such method, we make note of certain limitations of RNN forecasting and compare the models (network architectures) using heatmap visualizations.

3.2.1 Time Series Data

Before we begin our experiments, we must select the type of time series data we would like to analyze. As discussed in section 3.1.1, we generate synthetic time series data for each time step z_l, according to a few well established processes. Perhaps the simplest case is the auto-regressive process, signified by AR(p), where the value of the current time step l value is a linear combination of the previous p time steps. Formally, the auto-regressive process is defined as [25]

$$z_l = \sum_{i=1}^{p} c_i z_{l-i} + \epsilon_l \quad . \tag{3.5}$$

Here, c_i are the coefficients of the lagged variables and ϵ_l is white noise which follows a normal distribution $N(\mu, \sigma^2)$ at time step l. We also consider the moving average model, MA(q), which is similarly defined as [25]

$$z_l = \delta + \sum_{i=1}^{q} \theta_i \epsilon_{l-i} \tag{3.6}$$

where θ_i are the coefficients of the lagged white noised ϵ_{l-i} and δ is a predetermined average value. This moving average process generates the value of the current time step l based on an average value δ and the previous q lagged white noise. When we combine the processes from (3.5) and (3.6), we arrive at the auto-regressive moving average, or the ARMA(p, q) model.

Beyond these simple models, we consider another time series model called the generalized autoregressive conditional heteroskedasticity (GARCH) model. Full definition of the GARCH(p, q) process is presented in [25], where this process allows the time series to exhibit variance spikes during any given time step. It is generally defined as:

$$z_l = \sqrt{h_l}\epsilon_l \tag{3.7}$$

where h_l is a positive function that represents the variance. The variance h_l is defined by:

$$h_l = \alpha_0 + \sum_{i=1}^{p} \alpha_i z_{l-i}^2 + \sum_{j=1}^{q} \beta_j h_{l-j}^2 \quad . \tag{3.8}$$

In this GARCH(p, q) definition, α_i are the coefficients of the p lagged variables, and β_j are the coefficients of the q lagged variances. This process induces volatility in the time series data that is often present in real world datasets. One simple but effective way to characterize these times series processes is by observing the magnitudes of their lag structures. For large values of p, q, we consider the lag structure to be high (and correspondingly low for small values of p, q).

In addition to the synthetic time series processes, we also experiment with the following three real world datasets. (1) ETTh1, OT [134] includes the hourly oil temperature (OT) collected by electricity transformers from July 2016 to July 2018 from two different counties in China. (2) Solar-energy [73] includes the solar power production records in the year of 2006, which is sampled every 10 minutes from 137 photovoltaic plants in Alabama, USA. (3) Daily NASDAQ composite index values between 2014 to 2023 are gathered from Yahoo! Finance [5]. Note that ETTh1, OT, and Solar-energy have served as benchmarks for time series forecasting tasks [73, 85, 128].

3.2.2 RNN behavior under various time series processes

The focus of this section is to use distance correlation to determine the characteristics of RNNs for various time series processes. To investigate this, we carry out a series of experiments that track the outputs of each activation layer and compare it to the ground truth output via distance correlation upon the completion of model training. In effect, we are calculating $\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})$ for $t = 1, 2, ..., T$ at the final epoch, P. $\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})$ values near one show that the activation layer at t has a strong dependency to $\mathbf{Y}$, indicating it is a highly important layer in the training process. Conversely, values close to zero imply that the activation layer at t does not contribute to learning the output $\mathbf{Y}$.

One particular component of interest is layer number T, $\mathbf{A}_T^{(P)}$, which is the final activation layer before the RNN generates an output. Since all the previous inputs flow to this activation layer, tracking $\hat{R}(\mathbf{A}_T^{(P)}, \mathbf{Y})$ serves as a measure of what information has been retained or lost during training.

Since we are tracking the dependency structure of each layer $\mathbf{A}_t^{(P)}$ and ground truth $\mathbf{Y}$, we can make a parallel comparison to the auto-correlation function (ACF) defined by (3.1). If we are interested in understanding the linear structure between a single-step ahead time series value and a corresponding lag h, we would calculate $Cor(z_{l+1}, z_{l+1-h})$. The analogous comparison to this would be calculating the distance correlation, $\hat{R}(\mathbf{A}_{T+1-h}^{(P)}, \mathbf{Y})$. For example, given a window size of $T = 20$ and a lag of $h = 5$, activation layer number 16, $\mathbf{A}_{16}^{(P)}$, can be directly compared to the ACF value at the 5th lag. Note that this inverse relationship between the layer number and lag (e.g., layer 20 and lag 1, or, layer 1 and lag 20) is used for comparisons during our experiments. We make this comparison to establish a baseline analysis of the underlying time series structure.

Implementation Details

Before presenting the results, we choose some parameters specific to our experiments. We begin by generating a time series history $\mathbf{Z}$ of length $L = 4,000$ from the processes and

datasets in Section 3.2.1. We change L to 2,000 and 1,500 for the Solar-energy and NASDAQ datasets, respectively, to accommodate their smaller size. As is customary in training deep learning models, we standardize each time series using the z-score [74]. This is done to aid the stability of training an RNN and analyze all the time series processes or datasets using the same scale. We then split our time series history such that the first 80% points form the training set and the last 20% points are used for evaluation (test set), as shown in Figure 3.2. We only consider single step-ahead forecasts, i.e., $H = 1$.

For the Elman RNN setup, we fix the input window size $T = 20$, the number of hidden units as $b = 64$, use the ReLU activation function, batch size of 64, learning rate of 0.0001 and train the model for 35 epochs. We also initialize the networks weights via a Kaiming He initialization procedure [56], as it has been shown to be a stable and efficient initialization scheme [95]. We choose these parameters with the aim of creating a stable environment where the RNN has a high likelihood of converging. Overall, we want to ensure an experiment that captures how RNNs learn time series structures, not why they have known instabilities. We then evaluate the performance (forecast accuracy) of the RNNs using mean squared error (MSE) and mean average percentage error (MAPE) on the standardized data to allow consistent comparisons of the outcomes. In terms of the computational resources, we use TensorFlow 2.1 on a single GTX 970 GPU for model training and activation layer extraction.

Time Series Process Results

We now report the results of a series of simulations where a total of 50 runs are conducted for a given time series process. For each run, we train an RNN from scratch using a specified time series, calculate both the distance correlation values described in Section 3.2.2 , and the corresponding MSE and MAPE values. Note that for all the time series plots, the values are de-standardized to their original scales in order to view the actual results.

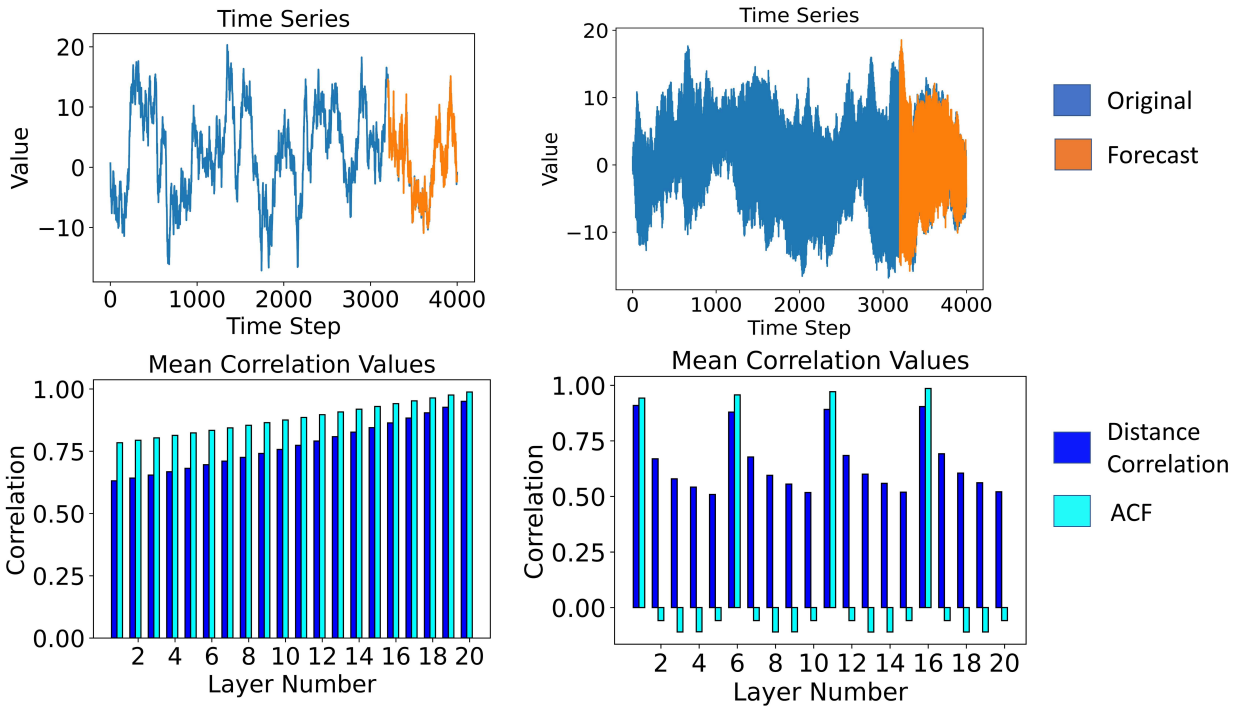
Beginning with auto-regressive time series, Figure 3.4 shows two plots each for AR(1) and AR(5) time series processes: 1) A time series plot with the original and de-standardized forecast values by the trained RNN; and 2) the mean correlation values between the ground

truth and each activation layer using both distance correlation and ACF metrics. Figure 3.5 shows the same plots for AR(5) and AR(10) time series processes. In addition, the figures also display the corresponding MSE and MAPE values, and the coefficients for the time series processes.

Figure 3.4(a) shows that for an AR(1) process, the distance correlation values closely resemble the pattern produced by the ACF values for all the layers and lags. The similarity between ACF and distance correlation is likely due to the recursive nature of the AR(1) process, where all the time steps are recursively dependent on each other. There is a gradual increase in correlation as the layer numbers increase, which remain relatively high for all the layers (above 0.7). Noting that layer number 20 is the final activation layer prior to generating a forecast, it follows that a high distance correlation in this layer yields accurate forecasts. This is supported by the low MSE and MAPE values, and well fitting de-standardized time series forecasting plots.

However, the corresponding plots for AR(5), AR(10) and AR(20) indicate that the forecasting accuracy of RNN decreases for increasing time series lag structures. Figure 3.4(b) shows the correctly identified AR(5) lag structure by displaying high correlation values at layers 1, 6, 11 and 16. Recalling the inverse relationship that lags have with the layer numbers, this aligns with the ACF values whose corresponding correlations are high at lags 20, 15, 10 and 5. This alignment, however, diminishes quickly as the distance correlation values decrease to approximately 0.5 at layer 20. This is further evident in Figure 3.5, where although the correct lags are identified by both distance correlation and ACF, distance correlation reduces to less than 0.4 for AR(10) and approximately 0.35 for AR(20) when it reaches layer 20; see Table 3.1 for the exact correlation values at activation layer $T = 20$. This reduction in distance correlation with increasing layer numbers is synonymous to the RNN losing memory of the important previous inputs. As a result, the MSE and MAPE scores increase for the AR(10) and AR(20) time series, which is supported by the poor fitting de-standardized forecasting plots.

We extend these experiments to the moving average process by looking at MA(1) and

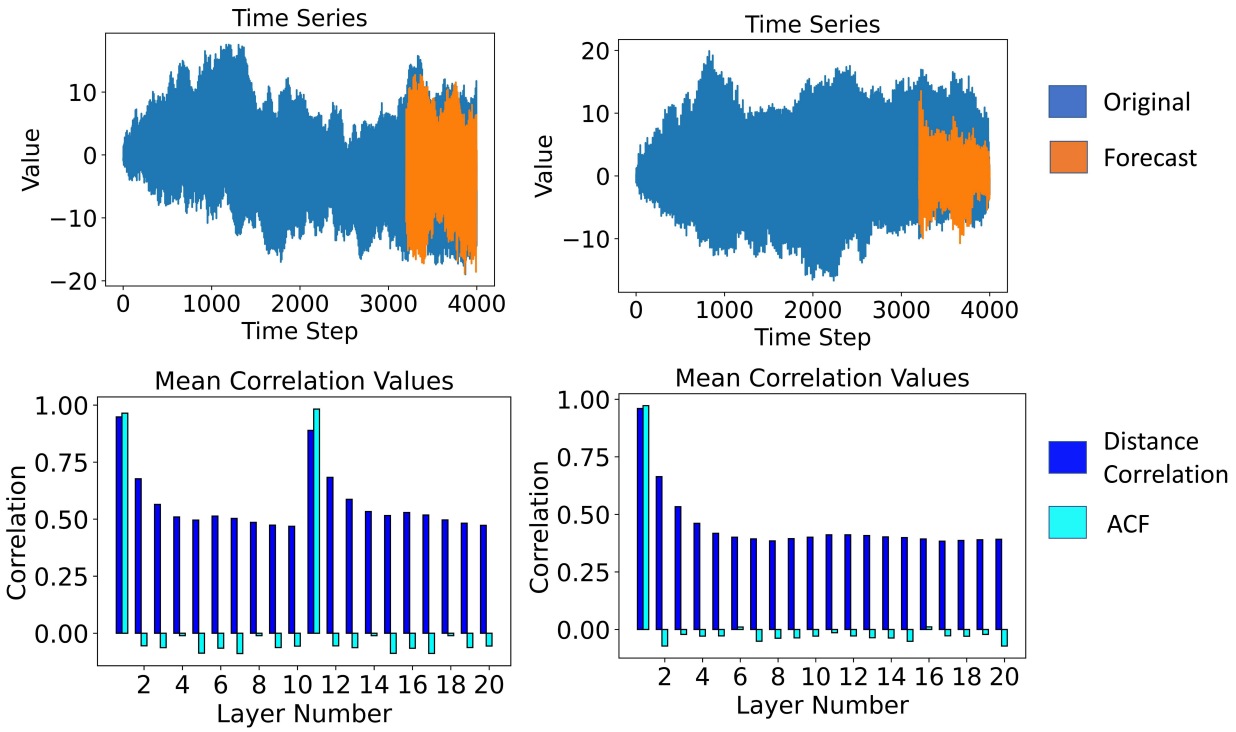


(a) AR(1)
MSE = 0.024 ± 0.006
MAPE = 0.050 ± 0.014
$z_l = 0.99 z_{l-1} + \epsilon_l$

(b) AR(5)
MSE = 0.100 ± 0.0395
MAPE = 0.111 ± 0.064
$z_l = 0.99 z_{l-5} + \epsilon_l$

Figure 3.4: AR time series plots with RNN forecasts of the test set (top row). The mean values of ACF and the distance correlations between the outputs of activation layers and the ground truth horizon values are shown in the bar plots for 50 runs (bottom row). For the AR(1) process, we observe high correlation values for both metrics and a gradual increase as the layer number increases. For the AR(5) process, we see layers 1, 6, 11, and 16 with high correlations, but they cyclically diminish for every 5 layers. This aligns with high ACF values whose corresponding lags occur at 20, 15, 10, and 5. Both the de-standardized time series plots have well fitting forecasts from the RNN, which corresponds to the relatively low MSE and MAPE scores.

MA(20) structures. Figure 3.6 shows the same mean correlation and de-standardized time series forecasting plots. In Figure 3.6(a), the mean correlation plot displays the correct layer



(a) AR(10)
MSE = 0.672 ± 0.344
MAPE = 0.319 ± 0.240
$z_l = 0.99 z_{l\text{-}10} + \epsilon_l$

(b) AR(20)
MSE = 1.28 ± 0.441
MAPE = 0.884 ± 1.153
$z_l = 0.99 z_{l\text{-}20} + \epsilon_l$

Figure 3.5: AR time series and mean correlation plots for 50 simulation runs. For AR(10), we see high distance correlation values at layers 1 and 11 with diminishing correlation values from layers 12 to 20. The high correlation values align with the ACF lags at layers 10 and 20. For AR(20), we only see high correlation at layer 1 (or lag 20), before a similar dissipation in values occur from layers 2 to 20. This decrease likely occurs due to the RNN losing some memory of the previous important inputs. We also see an increase in the MSE and MAPE values for both the processes, as compared to the AR(1) and AR(5) processes, with worse fits of the de-standardized forecasting plots.

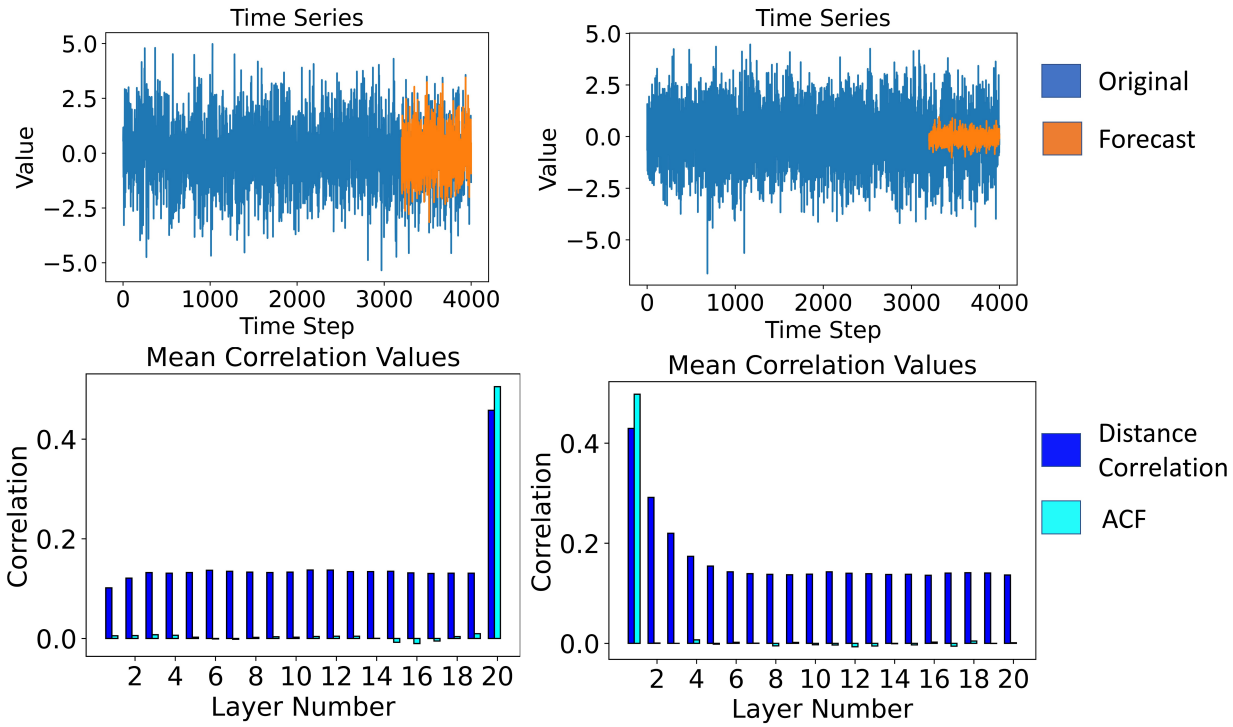
number (or lag structure) for MA(1); however, the value is relatively low (approximately 0.4). Although the distance correlation value is highest at layer number 20, the MSE and MAPE

scores are larger than their AR(1) counterparts. The relatively low distance correlation value at the final activation layer combined with worse MSE and MAPE scores indicate that the RNN struggles to model the error lag structures well, even under ideal low lag structures.

Figure 3.6(b) shows the results for the MA(20) structure. Although the proper lag is identified at layer number 1, it is again relatively low with a value of approximately 0.4. However, since the lag structure is high (larger values of p, q) and far removed from the forecast horizon, we see an asymptotic decrease in correlation values near layer number 20, exhibiting values less than 0.1. The RNN appears to compensate for this lack of information being transmitted to the final layer by modeling values near the average (of zero), as opposed to the fluctuations that characterize the MA processes. This mean trend prediction is noticeable in the de-standardized time series forecasting plot where the amplitude is truncated and centered around zero.

We also look at the combined effects of AR(p) and MA(q) processes by modeling ARMA(p, q) time series structures. Figure 3.7 shows the same set of plots for the ARMA(1,10) and ARMA(10,1) processes. Figure 3.7(a) shows the results for ARMA(1,10), which exhibit a similar correlation behavior to that of AR(1). The distance correlation values are high for all the layers with the highest occurring at layer number 20. This results in highly accurate forecasts with low MSE and MAPE values, and well fitting de-standardized time series plots. Figure 3.7(b) shows the results for ARMA(10,1), which exhibits similar correlation patterns to that of AR(10). However, the correlations do not decrease as much when approaching layer number 20 (0.5 instead of 0.4). The inclusion of the MA terms, therefore, seems to provide additional pertinent information on the underlying time series characteristics to the RNN. This slightly higher correlation values at the final layer leads to lower MSE and MAPE scores for both the ARMA processes as compared to their AR counterparts.

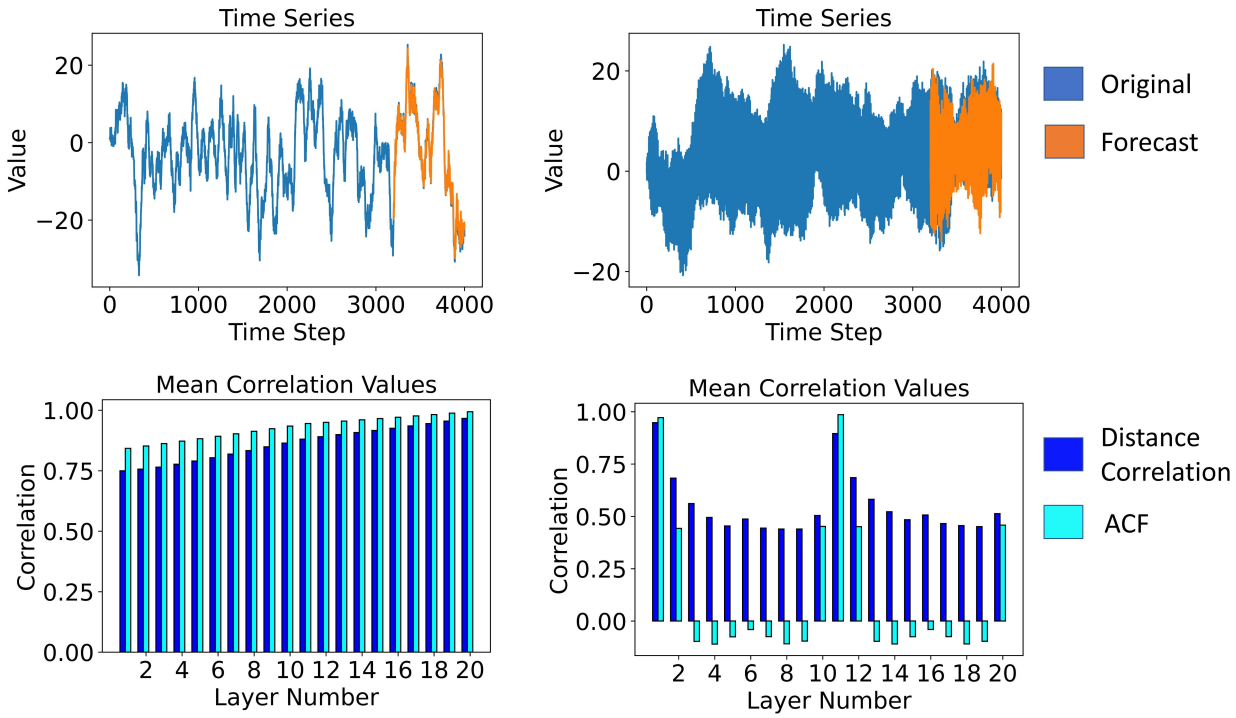
We also explore how RNNs internally learn time series processes exhibiting heteroscedasticity with GARCH processes. We report the same time series forecasts and mean value correlation plots in Figure 3.8, where we note that the distance correlation values are quite low (less than 0.25) for all the layers. The low correlation values indicate that none of the



(a) MA(1)
MSE = 0.602 ± 0.037
MAPE = 0.388 ± 0.438
$z_l = 0.99\epsilon_{l\text{-}1} + \epsilon_l$

(b) MA(20)
MSE = 1.056 ± 0.063
MAPE = 0.579 ± 0.374
$z_l = 0.99\epsilon_{l\text{-}20} + \epsilon_l$

Figure 3.6: MA time series plots with RNN forecasts of the test set (top row). The mean values of ACF and distance correlation between the outputs of the activation layers and the ground truth horizon values are shown in the bar plots for 50 runs (bottom row). For both the MA(1) and MA(20) processes, we see that the lags are identified at layers 20 and 1 (lags 1 and 20), respectively, with the correlation values around 0.4 (0.5 for ACF). We also see for MA(20), distance correlation diminishes asymptotically and converges to a value of less than 0.1 at the final activation layer. This is analogous to the RNN losing memory of the important inputs, resulting in higher MSE and MAPE scores, and a worse fitting de-standardized forecasting plot.



(a) ARMA(1,10)
MSE = 0.013 ± 0.003
MAPE = 0.037 ± 0.011
$z_l = 0.99z_{l\text{-}1} + 0.99\epsilon_{l\text{-}10} + \epsilon_l$

(b) ARMA(10,1)
MSE = 0.341 ± 0.20
MAPE = 0.216 ± 0.199
$z_l = 0.99z_{l\text{-}10} + 0.99\epsilon_{l\text{-}1} + \epsilon_l$

Figure 3.7: ARMA time series plots with RNN forecasts of the test set (top row). The mean values of ACF and the distance correlations between the outputs of the activation layers and the ground truth horizon values are shown in the bar plots for 50 runs (bottom row). For ARMA(1,10), high distance correlation values are exhibited for all the layers, which resemble the correlation plots of AR(1) and result in highly accurate forecasts. For ARMA(10,1), there is a notable lag structure similar to the AR(10) process. However, the decrease in RNN memory is less severe with a distance correlation value of 0.5 at layer 20, as compared to 0.4 for AR(10). This results in relatively well fitting de-standardized forecasting plots for both the ARMA processes.

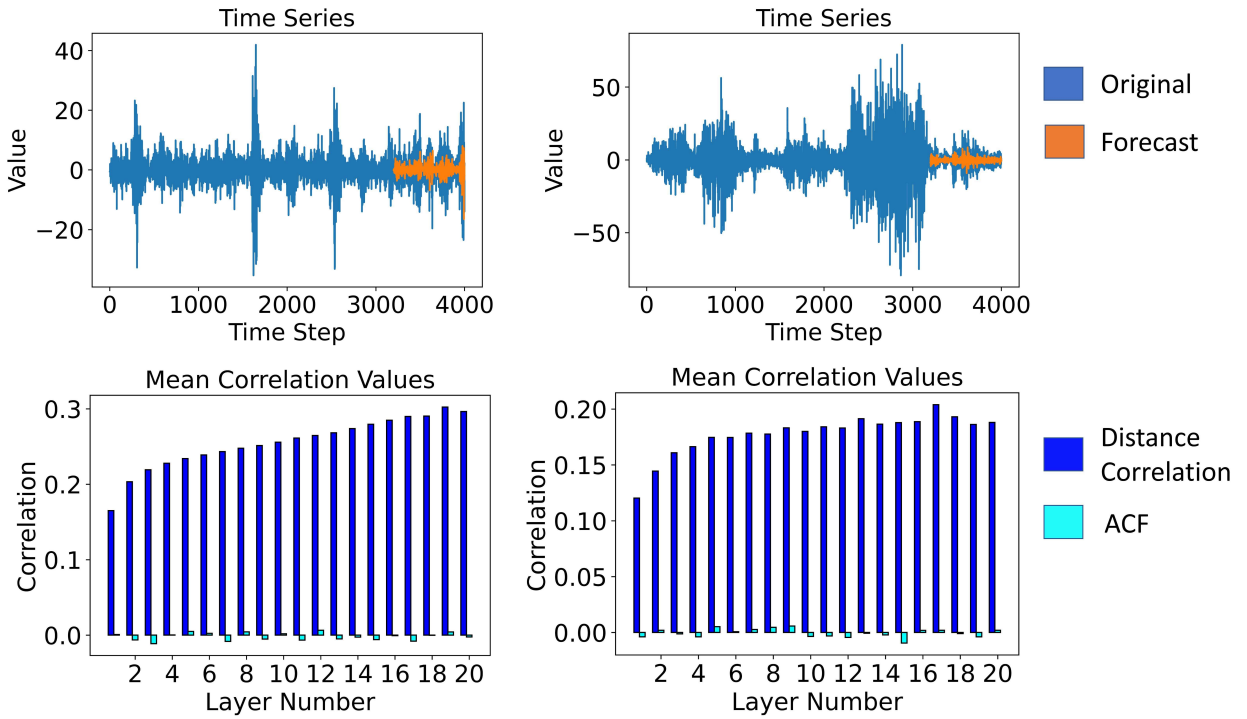
RNN activation layers make strong contributions to learning the ground truth. This is corroborated by the poor fitting de-standardized forecasting plots for both the GARCH(2,2) and GARCH(4,4) processes. In essence, the RNN activation layers are not able to effectively model the heteroscedasticity and volatility present in the GARCH process, at least without additional pre-processing.

Last, we employ our evaluation method on the real world datasets described in Section 3.2.1. We generate the same correlation plots and de-standardized forecast plots for the ETTh1, OT and the Solar-Energy datasets in Figure 3.9. Figure 3.9(a) shows that layers 1-10 have approximately constant correlation values of around 0.5 before gradually increasing to 0.8 at layer 20. This corresponds to having well fitting forecasts and lower MSE and MAPE scores. Similar observations are found in Figure 3.9(b), where the correlation values increase at a constant rate until a value of 0.85 at activation layer 20. As seen for the AR and ARMA processes, high correlation values at the final layer lead to fairly accurate forecasts.

However, the results for the NASDAQ Composite data in Figure 3.10 are quite different. The correlation plots show that none of the activation layers recognize the underlying process exhibited by the financial data. All the correlation values are near 0.1 and there is no evidence that the activation layers learn the outputs well, leading to poor forecasting results. In fact, the results are similar to those for the GARCH process. In summary, these real world examples provide additional validity to the fact that distance correlation can effectively examine the behavior, especially, learning capacity, of RNN forceasting models for any given times series.

Temporal information loss in RNNs

We recall the studies that used distance correlation as a metric of the information stored in neural network activation layers [133, 116], and apply the same principle in our study. From our experiments, we observe that the distance correlation values change as the time series inputs move from one activation layer to the next, analogous to a measurable change in information content across the RNN. Table 3.1 summarizes the results for all the time

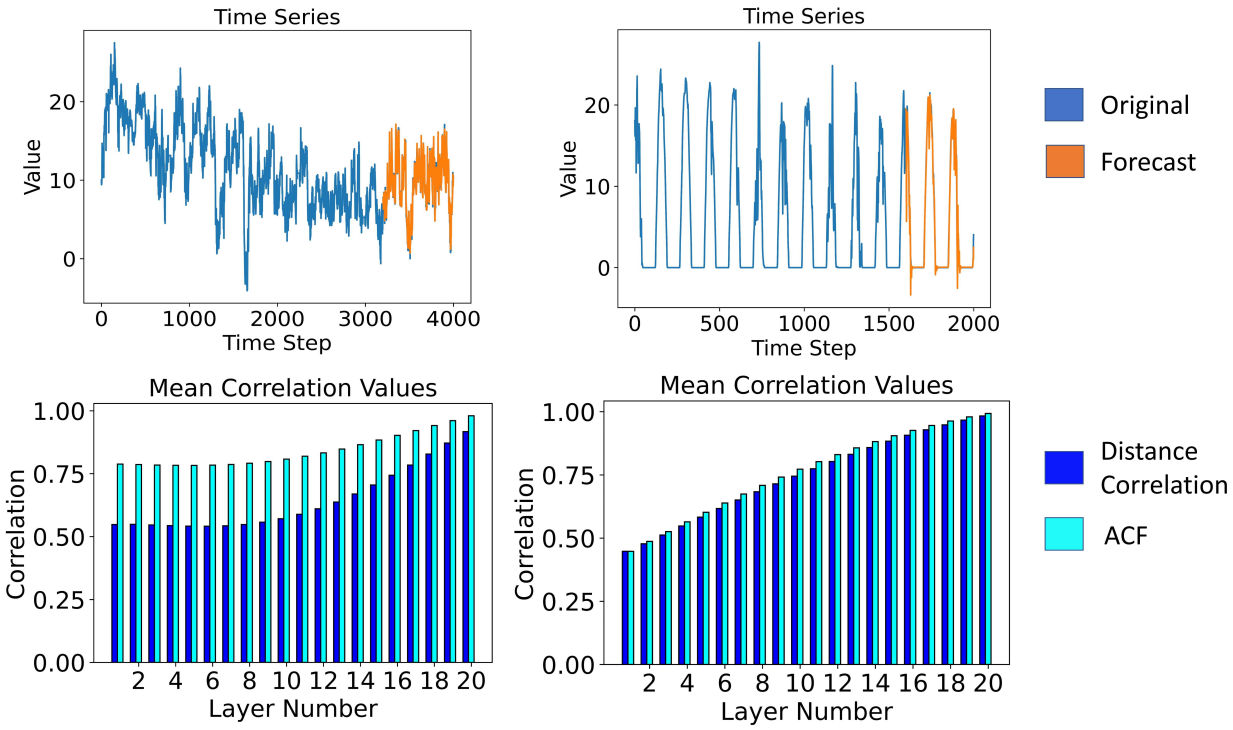


(a) GARCH(2,2)
MSE = 1.153 ± 0.990
MAPE = 0.614 ± 0.674
$h_l = 0.5 + 0.1z^2_{l-1} + 0.2z^2_{l-2} + 0.3h^2_{l-1} + 0.4h^2_{l-2}$

(b) GARCH(4,4)
MSE = 1.185 ± 0.940
MAPE = 0.720 ± 0.728
$h_l = 0.5 + 0.1z^2_{l-1} + 0.2z^2_{l-4} + 0.3h^2_{l-2} + 0.4h^2_{l-4}$

Figure 3.8: Time series plots with RNN forecasts of the test set (top row). The mean values of distance correlation and ACF between all the activation layer outputs and the ground truth horizon values are shown in a bar plot for 50 runs (bottom row). For both the GARCH(2,2) and GARCH(4,4) processes, we see that the distance correlation and ACF values are low; in fact, they are effectively zero with ACF for all the layer numbers. We also see that both the time series plots have poor RNN fitting forecasts, which corresponds to the relatively high MSE and MAPE scores. In particular, the RNN forecasts do not capture the spikes in variance well.

series forecasting experiments, and report the metrics that encode this RNN information flow. The key metric is the percentage change between the max distance correlation at any



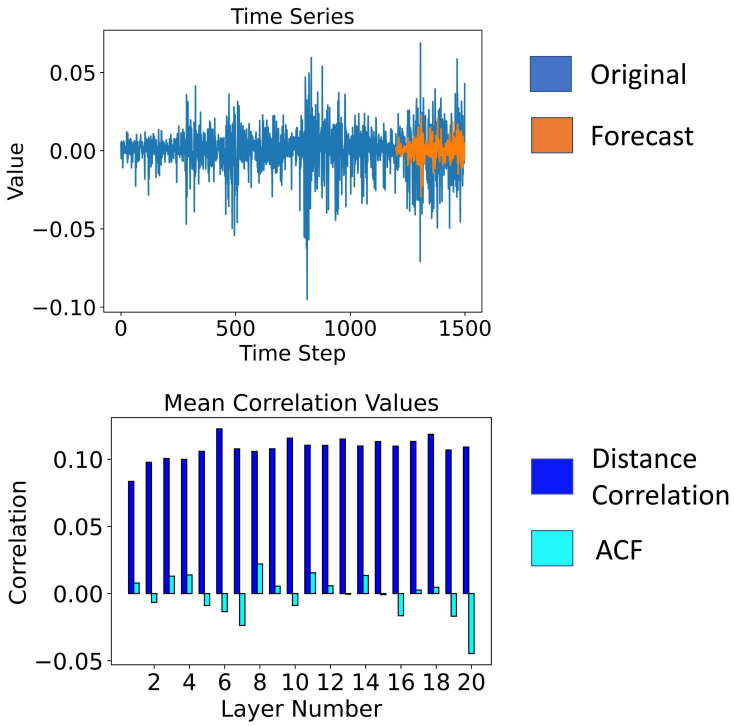
(a) ETTh1, OT
MSE = 0.040 ± 0.020
MAPE = 0.453 ± 0.324

(b) Solar-Energy
MSE = 0.017 ± 0.013
MAPE = 0.510 ± 0.691

Figure 3.9: Time series plots with RNN forecasts of the test set (top row). The mean values of distance correlation and ACF between all the activation layer outputs and the ground truth horizon values are shown in a bar plot for 50 runs (bottom row). For both the ETTh1, OT and Solar-Energy data, we see that the distance correlation and ACF values gradually increase until layer 20 is reached. This results in well fitting de-standardized forecasts, which correspond to the relatively low MSE and MAPE scores.

layer and the distance correlation at the final layer. This is calculated by $(\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})_{max} - \hat{R}(\mathbf{A}_T^{(P)}, \mathbf{Y}))/\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})_{max}$. It summarizes how much information is lost by the time the data reaches the final activation layer before the RNN produces a forecasting value.

From Table 3.1, we see that for the higher AR lag structures, the RNN is able to identify



NASDAQ-Composite Returns
MSE = 1.499 ± 0.521
MAPE = 1.077 ± 0.818

Figure 3.10: Time series plots with de-standardized RNN forecasts of the test set (top). The mean values of distance correlation and ACF between all the activation layer outputs and the ground truth horizon values are shown in a bar plot for 50 runs (bottom). For the NASDAQ Composite returns, we see that the distance correlation and ACF values are low; in fact, the ACF values are close to zero for all the layers. This leads to poor fitting forecasts, which correspond to the relatively high MSE and MAPE scores. This example is similar to the outcomes of the GARCH process in Figure 3.8.

the proper lags. However, higher lag structure AR processes lose a larger percentage of the information as they reach the final activation layer. For the MA lag structures, the proper lags are also identified, although they exhibit smaller distance correlation values.

Nonetheless, the higher lag structures still lose more information as they reach the final activation layer. This general loss of memory in the final activation layer leads to higher MSE and MAPE scores. This is also exemplified by comparing AR(10) to ARMA(10,1), where they have similar distance correlation patterns, as seen in Figures 3.5(a) and 3.7(b). However, ARMA(10,1) loses 48% of the input memory as compared to 59% for AR(1), resulting in lower MSE and MAPE values. The exception to this behavior occurs for the GARCH and NASDAQ time series, where no RNN activation layer learns the ground truth values well, leading to poor overall results.

These experiments and outcomes were replicated for different RNN hyperparameters to show that this behavior is consistent despite changes in the modeling parameters. See Appendix A for a summary of the additional results that include changes to the number of hidden units, learning rate, and dropout rate of the final RNN layer.

3.2.3 Visualizing differences in time series RNN models

As we try to gain a better understanding of RNN capabilities with time series forecasting, we leverage distance correlation to generate heatmaps as a visual way of investigating the activation layers; similar to the comparison of computer vision models [133]. We motivate this visualization method by considering which RNN hyperparameters lead to accurate forecasts. These include input size, activation functions, the number of hidden units, and even the RNN output configurations (i.e., many-to-one or many-to-many). For example, Figure 3.11 shows a distance correlation heatmap between a trained RNN with 10 inputs (activation layers) and itself, where we see a complete symmetry in the heatmap. This baseline heatmap can help us visualize the similarities between the activation layers at different time steps and reveal characteristics such as the lag structure. We expand such comparisons with some of the hyperparameters mentioned above.

Distance correlation heatmaps can show whether two networks with different hyperparameters are similar at each activation layer. Two examples of this are displayed in Figure 3.12, where we test networks with different activation functions and number of hidden units.

Table 3.1: MSE, MAPE, and distance correlation results for all the experiments

Time Series	MSE		MAPE	$\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})$ (max)	$\hat{R}(\mathbf{A}_T^{(P)}, \mathbf{Y})$	Change (%)
AR(1)	0.025	±	0.050 ± 0.014	0.927	0.927	0
	0.006					
AR(5)	0.100	±	0.111 ± 0.064	0.916	0.436	52
	0.039					
AR(10)	0.672	±	0.319 ± 0.240	0.949	0.391	59
	0.344					
AR(20)	1.281	±	0.884 ± 1.154	0.964	0.356	63
	0.441					
MA(1)	0.602	±	0.388 ± 0.438	0.418	0.418	0
	0.037					
MA(5)	0.783	±	0.525 ± 0.603	0.427	0.131	69
	0.057					
MA(10)	1.008	±	0.512 ± 0.235	0.422	0.098	77
	0.081					
MA(20)	1.056	±	0.579 ± 0.374	0.435	0.093	79
	0.063					
ARMA(1,1)	0.009	±	0.035 ± 0.012	0.935	0.935	0
	0.003					
ARMA(1,10)	0.013	±	0.037 ± 0.011	0.951	0.951	0
	0.003					
ARMA(10,1)	0.341	±	0.216 ± 0.199	0.947	0.496	48
	0.195					
GARCH(2,2)	1.037	±	0.614 ± 0.674	0.264	0.259	2
	0.830					
GARCH(4,4)	1.080	±	0.720 ± 0.728	0.215	0.202	6
	0.751					
ETTh1, OT	0.040	±	0.453 ± 0.324	0.923	0.923	0
	0.020					
Solar-Energy	0.017	±	0.510 ± 0.691	0.983	0.983	2
	0.013					
NASDAQ	1.499	±	1.077 ± 0.818	0.123	0.114	7
	0.521					

Notes: MSE and MAPE scores are calculated for 50 simulation runs. $\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})$ (max) represents the max distance correlation for any layer number t. $\hat{R}(\mathbf{A}_T^{(P)}, \mathbf{Y})$ is the distance correlation value at layer number $T = 20$. The percent change from the peak distance correlation values in any activation layer to the final layer measures how much information is lost during RNN training. In general, we see that larger lag structures tend to lose more information during training, leading to higher MSE and MAPE scores.

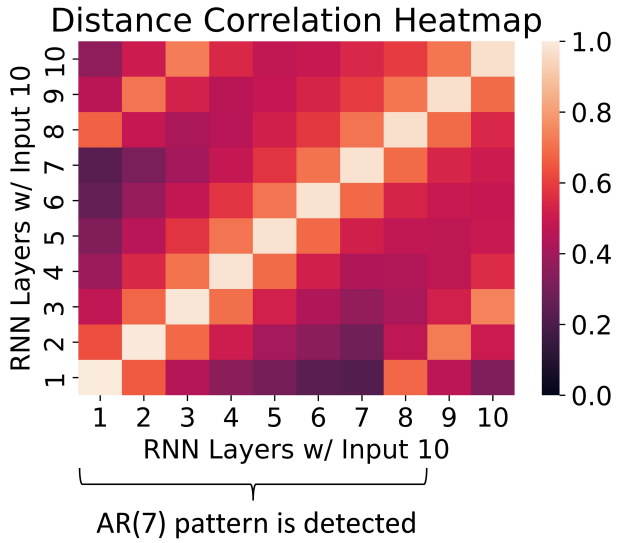
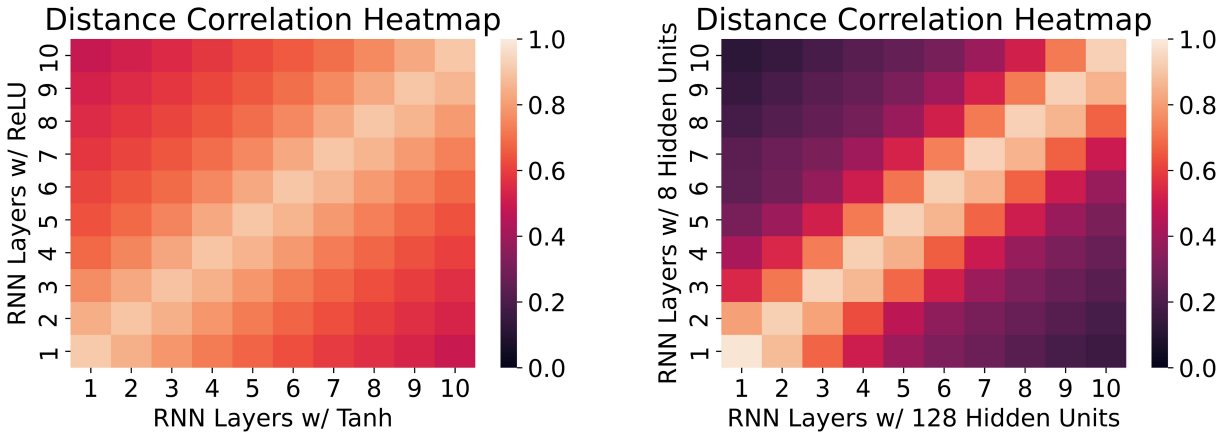


Figure 3.11: Distance correlation heatmap between RNN with 10 inputs and itself. The RNN was trained on an AR(7) process with the following governing equation: $z_l = 0.8z_{l-7} + \epsilon_l$. We observe there is an exact symmetry along the diagonal, where the distance correlation between each RNN layer and itself is 1. We also see that the AR(7) pattern is detected if we follow the progression of values from layer 1 on the y-axis to layer 8 of the x-axis.

In the case of Figure 3.12(a), the symmetric nature of the heatmap suggests that using either ReLU or Tanh activation function makes minimal difference in its evaluation of an AR(1) process. Extending this to Figure 3.12(b), we see a similar symmetry in the heatmap for networks having 8 and 128 hidden units in the activation layer for an ARMA(1,2) process. Note that all the networks were allowed to train until convergence, which exceeded 100 epochs for the RNN with 8 hidden units. Both these results show that the different hyperparameter choices do not have any noticeable effect for a given time series process, provided the computational cost is not an issue.



(a) Similarities in RNNs ReLU and Tanh activation functions
AR(1), $z_l = 0.8z_{l\text{-}1} + \epsilon_l$

(b) Similarities in RNNs 8 and 128 units in hidden state
ARMA(1,2), $z_l = 0.6z_{l\text{-}2} + 0.3\epsilon_{l\text{-}1} + 0.6\epsilon_{l\text{-}2}$

Figure 3.12: Visualization of the similarities of different RNNs in time series modeling using distance correlation heatmaps. (a) The symmetry in the heatmap suggests that using either activation function yields similar outputs at each activation layer. (b) The heatmap also suggests that both the networks arrive at similar activation layer outputs, despite the differences in the hidden unit size of the activation layers.

Another crucial choice with RNN time series forecasting is the proper window input size. While some heuristics can be applied in choosing this parameter, we can also use distance

correlation heatmaps to understand the impact. We exemplify this with a heatmap that compares an RNN with 6 inputs and 10 inputs under an AR(8) process which is shown in Figure 3.13. In this scenario, the smaller RNN shows that it can learn similar activation outputs to the larger 10 input RNN as shown by the similarities in the diagonal elements between activation layers 1-6 and 5-10, respectively. However, it seems that without having access to at least 8 inputs for the AR(8) process, suggests that the smaller network cannot adjust its learning weights via a backpropagation through time algorithm [119] to produce accurate forecasts. This is supported by the lack of good fit in the de-standardized time series forecast plot for the 6 input RNN and the better fit in the 10 input RNN.

We can then adopt a conservative strategy of choosing a window input size that is larger than necessary. Putting aside the known issues with long term dependency and exploding gradient problems in RNNs, we use distance correlation heatmaps to identify how the RNNs learn time series when the networks inputs are oversized. Figure 3.14 simulates this scenario where both the RNN input sizes (10 and 20, respectively) are greater than the AR(6) process lag structure. We notice that the diagonal streaks of similarities are 6 activation layers apart, which confirms that both the RNNs can detect the 6 lag structure of the time series. Further, the 20 input RNN encounters the lag structure multiple times. This redundancy likely aids in the backpropagation through time algorithm, where the adjusted layer weights are updated to have more accurate forecasts.

3.3 Discussion

We use distance correlation to show that the RNN activation layers can identify lag structures but have issues in transmitting that information to the final activation layer. This loss in information seems to occur over a span of 5-6 activation layers before the distance correlation values converge. The critical implication is that the RNN forecasts of univariate time series processes with large lag structures are likely to be poor since they are subject to more information loss. This can affect design decisions such as the sampling rate. For example, a sampling rate of hours, as compared to days, may subject the RNN to larger lag structures,

even though a higher resolution input sequence is obtained. Conversely, lower lag structures make RNNs a preferred model with fast forecast results as they are relatively computationally inexpensive. These results are supported by our evaluation on the ETTh1, OT and Solar-Energy datasets, where we observe high correlation values at the final activation layers. This yields favorable forecasts and indicates that the RNNs have adequate learning capacity for single-step forecasting of such data.

Additionally, low distance correlation values of the activation layers in MA processes show that the error lags are difficult to model for RNNs. This lower accuracy is exacerbated by the information loss of RNNs. We also observe that the low distance correlation values in all the activation layers for GARCH processes lead to poor performance. Thus, an RNN model of any heteroskedastic time series is not likely to perform well. Relating this to real world data, we observe that the NASDAQ Composite returns look similar to the GARCH process results. None of the activation layer correlations are high, leading to poor fitting forecasts. In these instances, we find that the low correlations in the activation layers relate to the lack of the RNN's learning capacity given this form of time series data.

These results can also help the practitioners with *a priori* assessments of the suitability of RNN forecasts given the characteristics of their time series data. For example, ACF plots, indicating whether the lag structures are high, can determine whether the corresponding time series processes are suitable for RNN forecasting. The presence of AR, MA, ARMA, and GARCH processes are also indicated in the ACF plots, providing cues on whether RNNs would be suitable for the corresponding forecast problems. This knowledge is valuable in reducing unnecessary and time-consuming model building and exploration steps. Coupling these observations with the hyperparameter-based comparisons provided by the distance correlation heatmaps, analysts are now better equipped to both interpret and explain RNN modeling of time series.

We acknowledge some ways to improve and expand our findings. First, we primarily consider well-established time series processes. This is done intentionally to create a controlled experiment, where the inputs and outputs of the time series processes are well defined. An ex-

pansion of the scope of our experiments to real world datasets, may provide a more practical context for our evaluation tool. Further, we only consider the most basic form of time series forecasting, which is univariate, single-step prediction. Single-step predictions can make our plotted forecasting results look quite favorable, but they should not be confused with the much more challenging task of forecasts with larger prediction horizons. Multivariate and multiple horizon forecasting are expected to reveal more insights into the effectiveness of RNNs, and we believe that distance correlation is powerful and generic enough to be adapted to such problems.

We also recognize that our distance correlation-based analysis is done with only Elman RNNs. However, we contend that it can be expanded to other widely-used RNN architectures, such as Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, which have more complex cell operations to alleviate some of the common pitfalls of RNNs. It may be particularly interesting to investigate how the information is partitioned among the regulatory gates and additional recurrent cell states of LSTMs. This analysis tool can also be extended to many other deep learning networks that attempt to address time series forecasting. This includes observing how a time series input evolves through every major component in a transformer or another hybrid architecture. Overall, we feel that distance correlation is a flexible metric that can potentially unlock the unexplained nature of deep learning models for time series forecasting tasks.

3.4 Conclusions

In this chapter, we develop a distance correlation framework to study the effectiveness of RNNs for time series forecasting. Specifically, we leverage the versatility of distance correlation to track the outputs of the RNN activation layers and examine how well they learn specific time series processes. Through both synthetic and real world data experiments, we find that the activation layers detect time series lag structures well, but tend to lose this information over a sequence of five-six layers. This affects the forecast accuracy of series with large lag structures. Further, the activation layers have difficulty in modeling the error lag

terms for both moving average and heteroscedastic processes. Last, our distance correlation heatmap comparisons reveal that certain network hyperparameters, such as the number of hidden units and activation function, matter less as compared to the input window size of an RNN for accurate forecasts. We, therefore, believe our framework is a foundational step toward improving our understanding of the ability of deep learning models to handle time series forecasting tasks.

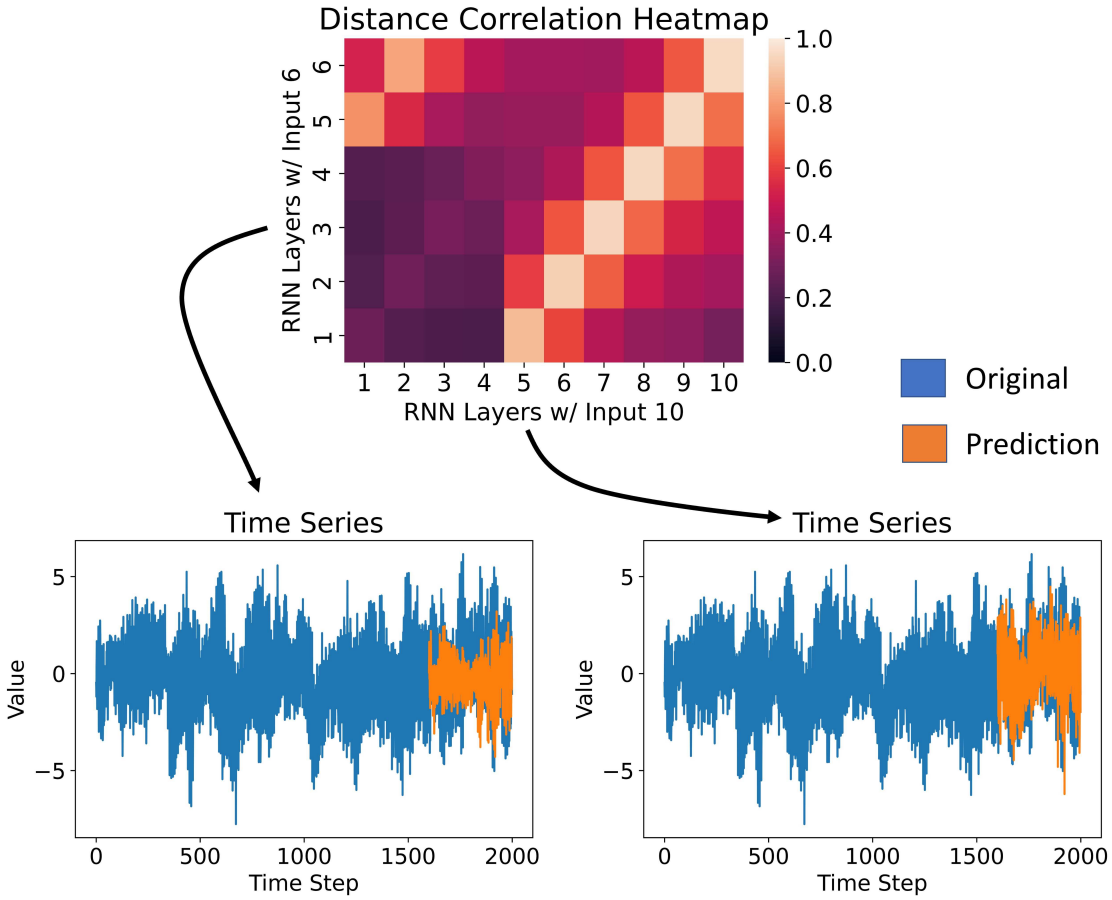


Figure 3.13: Distance correlation heatmap between RNN with 6 inputs and 10 inputs. This example invokes a scenario where the input size is smaller than the largest lag in the AR process. The RNNs were trained under an AR(8) process with the following governing equation: $z_l = 0.8z_{l-8} + \epsilon_t$. The 6 input and 10 input RNNs received an MSE $= 1.18 \pm 0.056$ (MAPE $= 0.510 \pm 0.0816$) and 0.496 ± 0.024 (MAPE $= 0.272 \pm 0.226$), respectively, after 50 simulation runs. RNN (6 input) activations layers 1-6 and RNN (10 input) activation layers 5-10 are noticeably highly correlated. This correlation suggests that the smaller network learns the final activation layer outputs of sufficiently sized networks. However, without access to the 8th lagged time step from the AR(8) process, the smaller network is missing crucial information that prevents accurate forecasts. This is evident from the worse fitting forecasts of the time series plot and the corresponding MSE and MAPE scores.

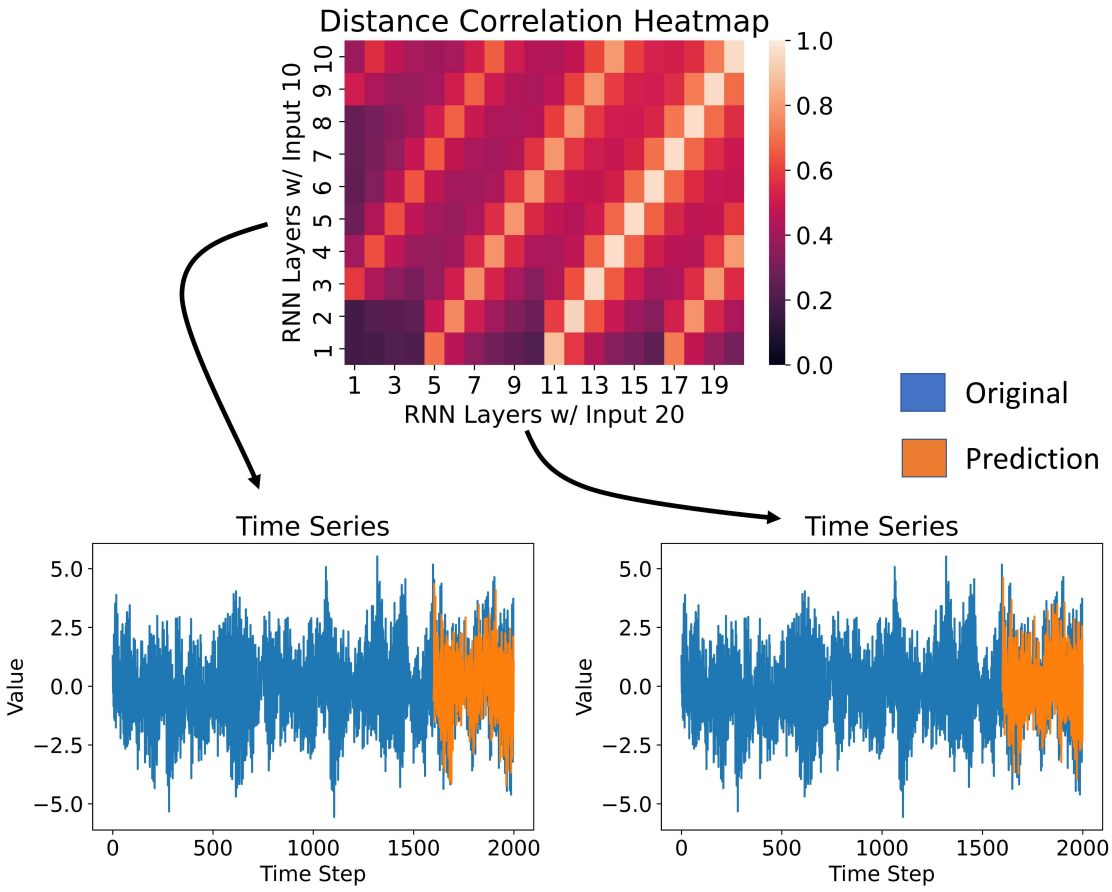


Figure 3.14: Distance correlation heatmap between RNNs with 10 inputs and 20 inputs. The RNNs were trained under an AR(6) process with the following governing equation: $z_l = 0.8z_{l-6} + \epsilon_t$. The 10 input and 20 input RNNs received an MSE of 0.501 ± 0.061 (MAPE $= 0.278 \pm 0.0790$) and 0.490 ± 0.045 (MAPE $= 0.293 \pm 0.137$), respectively, after 50 simulation runs. The heatmap shows diagonal streaks of similarities in both the networks, which are spaced apart by 6 activation layers, matching the AR(6) process. The 20 input RNN encounters the AR(6) lag structure at least thrice, as compared to twice in the 10 input RNN. This may be the source of why the former RNN yields a slightly lower MSE (and comparable MAPE) than the latter RNN. However, the corresponding time series forecasting plots indicate that both the RNNs learn the time series structure adequately as their input sizes are sufficiently large.

Chapter 4

ENHANCING NON-STATIONARY TIME SERIES FORECASTING WITH INCREMENTAL KERNEL DYNAMIC MODE DECOMPOSITION

4.1 Proposed Method

To develop our proposed method, it is essential to establish the foundational concepts related to univariate time series and the general framework of DMD. In this section, the symbol † represents the Moore-Penrose pseudoinverse of a matrix. Also $*$ represents the conjugate transpose of a matrix containing complex elements. Further, whenever a Singular Value Decomposition (SVD) of a matrix is performed, it is assumed that its components can be truncated using a parameter r. This truncation facilitates more efficient downstream calculations.

4.1.1 Univariate Time Series and DMD Framework

Let $S = (x_1, x_2, \ldots, x_T)$ be a univariate time series with T observations. To format this into a DMD snapshot matrix [115], we create a Hankel matrix of the data with m columns, which is synonymous with the number of snapshots of a dynamical system. This matrix is defined as

$$\mathbf{H} = \begin{bmatrix} x_1 & x_2 & \ldots & x_m \\ x_2 & x_3 & \ldots & x_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{T-m+1} & x_{T-m+2} & \ldots & x_T \end{bmatrix}$$

To create the pairs of snapshots used for DMD, we partition $\mathbf{H}$ into the following $\mathbf{X}, \mathbf{Y}$ data

matrices

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_{m-1} \\ x_2 & x_3 & \dots & x_m \\ \vdots & \vdots & \ddots & \vdots \\ x_{T-m+1} & x_{T-m+2} & \dots & x_{T-1} \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} x_2 & x_3 & \dots & x_m \\ x_3 & x_4 & \dots & x_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{T-m+2} & x_{T-m+3} & \dots & x_T \end{bmatrix}$$

where $\mathbf{X}$ contains the first $m-1$ snapshots and $\mathbf{Y}$ contains the final $m-1$ snapshots. For compactness, we re-write $\mathbf{X}, \mathbf{Y}$ as

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}$$

where

$$\mathbf{x}_i = [x_i, x_{i+1}, \dots, x_{i+T-m}]^T$$

and i represents the snapshot or column number of $\mathbf{H}$. Given how we arrange $\mathbf{H}$, $\mathbf{x}_i \in \mathbb{R}^d$ where $d = T - m + 1$.

As outlined by [71], for the standard DMD approach, we wish to solve $\mathbf{K}_{\text{dmd}}$ such that $\mathbf{Y} = \mathbf{K}_{\text{dmd}}\mathbf{X}$ approximately holds. Solving for $\mathbf{K}_{\text{dmd}}$ requires minimizing a least-squares problem which yields $\mathbf{K}_{\text{dmd}} = \mathbf{Y}\mathbf{X}^{\dagger}$. The eigendecomposition of $\mathbf{K}_{\text{dmd}}$ gives the eigenvalues arranged as a diagonal matrix, $\mathbf{\Lambda} \in \mathbb{C}^{r \times r}$, and eigenvectors, $\mathbf{W} \in \mathbb{C}^{r \times r}$. Furthermore, the SVD of $\mathbf{X}$ is computed:

$$\mathbf{X} = \mathbf{Q}_X \mathbf{\Sigma}_X \mathbf{V}_X^* \tag{4.1}$$

where $\mathbf{Q}_X \in \mathbb{C}^{d\times r}$, $\mathbf{\Sigma}_X \in \mathbb{C}^{r\times r}$, $\mathbf{V}_X \in \mathbb{C}^{m-1\times r}$. This decomposition allows the construction of DMD modes as

$$\mathbf{\Phi}_{\text{dmd}} = \mathbf{Y}\mathbf{V}_X\mathbf{\Sigma}_X^{-1}\mathbf{W}. \tag{4.2}$$

Finally, we use the previous matrices to construct the forecast

$$\tilde{\mathbf{x}}_t = \mathbf{\Phi}_{\text{dmd}}\mathbf{\Lambda}^t b_0 \tag{4.3}$$

which represents the prediction at time step t. The amplitudes, b_0, are formed from the initial conditions, and are typically calculated as

$$b_0 = \mathbf{\Phi}_{\text{dmd}}^{\dagger}\mathbf{x}_0 \tag{4.4}$$

where $\mathbf{x}_0$ is an initial snapshot vector (typically the last column of $\mathbf{X}$ or $\mathbf{Y}$).

If a prediction horizon of l points beyond x_T is desired, we collect the predictions from Equation (4.3) as column vectors and arrange them as the following trajectory matrix

$$\tilde{\mathbf{X}} = \begin{bmatrix} \tilde{x}_{m+1} & \tilde{x}_{m+2} & \dots & \tilde{x}_{m+l} \\ \tilde{x}_{m+2} & \tilde{x}_{m+3} & \dots & \tilde{x}_{m+l+1} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{x}_{T+1} & \tilde{x}_{T+2} & \dots & \tilde{x}_{T+l} \end{bmatrix} \tag{4.5}$$

Note that $\tilde{\mathbf{X}}$ is still arranged as a Hankel matrix, which contains multiple predictions of the same $\tilde{x}_{T+k}$ where $k = 1, 2, \dots, l$. One approach to resolve this is by collecting and averaging all the instances of $\tilde{x}_{T+k}$ as demonstrated by [115]. In effect, this method averages the anti-diagonals of the diagonals of $\tilde{\mathbf{X}}$. However, we find that this step is inefficient during inference, especially as l increases. Instead, we simply extract the last row of $\tilde{\mathbf{X}}$ to serve as our forecast values:

$$\hat{\mathbf{x}}_f = \tilde{\mathbf{X}}(d, :) \tag{4.6}$$

4.1.2 Kernel DMD

One of the limitations of the standard DMD approach is the assumption of linearity in the dynamical systems. Although this assumption suffices for many systems, it may be inadequate for predicting time series with nonlinear behavior. Therefore, we explore an advanced method known as Extended Dynamic Mode Decomposition (EDMD), which facilitates the application of DMD in infinite-dimensional spaces [120]. A practical variant of EDMD is the kernel-based (KDMD) approach developed by [67], which incorporates the kernel method or "kernel trick" within the DMD framework. We briefly present this method.

Considering our original observations $\mathbf{X}$ and $\mathbf{Y}$, we seek a transformation to a higher dimensional space, represented as $\mathbf{\Psi}_x$ and $\mathbf{\Psi}_y$:

$$\mathbf{\Psi}_x = \begin{bmatrix} | & | & & | \\ \psi(\mathbf{x}_1) & \psi(\mathbf{x}_2) & \dots & \psi(\mathbf{x}_{m-1}) \\ | & | & & | \end{bmatrix}$$

$$\mathbf{\Psi}_y = \begin{bmatrix} | & | & & | \\ \psi(\mathbf{x}_2) & \psi(\mathbf{x}_3) & \dots & \psi(\mathbf{x}_m) \\ | & | & & | \end{bmatrix}$$

Here, $\psi(\cdot)$ is the function that maps our observations to a higher dimensional space. Due to the computational costs associated with high-dimensional spaces, it is often impractical to explicitly calculate $\mathbf{\Psi}_x$ and $\mathbf{\Psi}_y$. Instead, the kernel trick facilitates an implicit calculation of these mappings through dot products $\psi(\mathbf{x}_i)^T\psi(\mathbf{x}_j)$, determined by a user-defined kernel function, such as Gaussian Radial Basis Functions (RBF), polynomial, or sigmoidal.

If we have access to $\mathbf{\Psi}_x$, we compute its SVD

$$\mathbf{\Psi}_x = \mathbf{U}\mathbf{\Sigma}\mathbf{Z}^* \tag{4.7}$$

As derived by [67], we form a new matrix $\hat{\mathbf{K}}$, analogous to $\mathbf{K}_{\text{dmd}}$:

$$\hat{\mathbf{K}} = (\mathbf{\Sigma}^\dagger\mathbf{U}^T)\hat{\mathbf{A}}(\mathbf{U}\mathbf{\Sigma}^\dagger) \tag{4.8}$$

where $\hat{\mathbf{A}} = \mathbf{\Psi}_y \mathbf{\Psi}_x^T$ is formed using the specified kernel function. Similarly, $\hat{\mathbf{G}} = \mathbf{\Psi}_x \mathbf{\Psi}_x^T$ is computed using the same kernel function, and its eigendecomposition is:

$$\hat{\mathbf{G}} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T \tag{4.9}$$

where $\mathbf{U}$ and $\mathbf{\Sigma}$ are the same matrices from Equation (4.7). Once $\hat{\mathbf{K}}$ is determined, $\hat{\mathbf{V}}$ represents the matrix whose columns are the eigenvectors of $\hat{\mathbf{K}}$, and $\hat{\mathbf{\Lambda}}$ denotes the diagonal matrix containing the corresponding eigenvalues. We then derive the dynamic modes using the following relation:

$$\mathbf{\Phi}_{\text{kdmd}} = \mathbf{X}\mathbf{U}\mathbf{\Sigma}^{\dagger}(\hat{\mathbf{V}}^{-1})^T \tag{4.10}$$

Finally, we use the eigenvalues of $\hat{\mathbf{K}}$ and $\mathbf{\Phi}_{\text{kdmd}}$ to make forecasts using Equation (4.3):

$$\tilde{\mathbf{x}}_t = \mathbf{\Phi}_{\text{kdmd}} \hat{\mathbf{\Lambda}}^t \hat{b}_0 \tag{4.11}$$

where $\hat{b}_0$ is analogous to Equation (4.4). Finally, we arrange the predictions from (4.11) similarly to (4.5) and extract the last row as in (4.6) to make our forecasts.

4.1.3 Incremental Kernel DMD

For streaming data that arrives incrementally, it is inefficient to use the kernel DMD batch method outlined in Section 4.1.1. Although incremental methods for the standard DMD approach has been proposed [58, 129], to the best of our knowledge, an incremental kernel DMD (IKDMD) method has not been explored.

The solution of converting KDMD to IKDMD boils down to the incremental kernel SVD update of $\hat{\mathbf{G}}$ from Equation (4.9). To begin, we build new data matrices by appending the newly arrived snapshots $\mathbf{x}_{\text{new}}, \mathbf{y}_{\text{new}}$ to $\mathbf{X}$ and $\mathbf{Y}$

$$\mathbf{X}_{\text{new}} = \begin{bmatrix} | & | & & | & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{m-1} & \mathbf{x}_{\text{new}} \\ | & | & & | & | \end{bmatrix}$$

$$\mathbf{Y}_{\text{new}} = \begin{bmatrix} | & | & & | & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_m & \mathbf{y}_{\text{new}} \\ | & | & & | & | \end{bmatrix}$$

Naturally, we also update

$$\mathbf{\Psi}_{x,\text{new}} = \begin{bmatrix} | & | & & | & | \\ \psi(\mathbf{x}_1) & \psi(\mathbf{x}_2) & \dots & \psi(\mathbf{x}_{m-1}) & \psi(\mathbf{x}_{\text{new}}) \\ | & | & & | & | \end{bmatrix}$$

$$\mathbf{\Psi}_{y,\text{new}} = \begin{bmatrix} | & | & & | & | \\ \psi(\mathbf{x}_2) & \psi(\mathbf{x}_3) & \dots & \psi(\mathbf{x}_m) & \psi(\mathbf{y}_{\text{new}}) \\ | & | & & | & | \end{bmatrix}$$

To incrementally update $\hat{\mathbf{G}}$, we use the work by [33], where the authors extend the incremental SVD method to the non-linear kernel case. We summarize the steps required to develop incremental kernel SVD method, specically for a rank-one update.

We begin by defining $\boldsymbol{\alpha} = \mathbf{U}\mathbf{\Sigma}^{\dagger}$ which represents the basis vector coefficients of linear expansion of the mapped input vectors, $\mathbf{\Psi}_x$. We then define L as

$$L = \boldsymbol{\alpha}^T \mathbf{\Psi}_x^T \psi(\mathbf{x}_{\text{new}}), \tag{4.12}$$

where $\mathbf{\Psi}_x^T \psi(\mathbf{x}_{\text{new}})$ can be evaluated via the kernel function since it represents an inner product. This allows the formation of $\mathbf{C}$ as

$$\mathbf{C} = \mathbf{\Psi}_{x,\text{new}} \begin{bmatrix} -\boldsymbol{\alpha} L \\ \mathbf{I}_{\text{new}} \end{bmatrix} := \mathbf{\Psi}_{x,\text{new}} \boldsymbol{\beta} \tag{4.13}$$

where $\mathbf{I}_{\text{new}}$ is the identity matrix whose size is determined the number of new snapshots being introduced; $\mathbf{I}_{\text{new}} = 1$ for rank-one updates. This allows us to form the following:

$$\mathbf{M} = \boldsymbol{\beta}^T \mathbf{\Psi}_{x,\text{new}}^T \mathbf{\Psi}_{x,\text{new}} \boldsymbol{\beta} \tag{4.14}$$

where the eigenvalue decomposition of $\mathbf{M} = \mathbf{Q}\boldsymbol{\Xi}\mathbf{Q}$. Note for a rank one update, $\mathbf{M}$ is a 1x1 matrix whose eigenvalue decomposition becomes trivial.

We use the eigenvalue decomposition of $\mathbf{M}$ to construct the following matrices,

$$\mathbf{J} = \boldsymbol{\Psi}_{x,\text{new}}\boldsymbol{\beta}\mathbf{Q}\boldsymbol{\Xi}^{-\frac{1}{2}} := \boldsymbol{\Psi}_{x,\text{new}}\boldsymbol{\Omega}, \quad \mathbf{K} = \boldsymbol{\Xi}^{\frac{1}{2}}\mathbf{Q}^T \tag{4.15}$$

Matrix $\mathbf{F}$ is then formed via the following,

$$\mathbf{F} = \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{L} \\ \mathbf{0}_{1\times r} & \mathbf{K} \end{bmatrix} \tag{4.16}$$

which is then diagonalized with $\mathbf{F} = \mathbf{U}'\boldsymbol{\Sigma}'\mathbf{V}'^*$. Given these new matrices, we are finally able to update the SVD components of equation (4.9) via the following,

$$\mathbf{U}_{\text{new}} = \boldsymbol{\Psi}_{x,\text{new}}\mathbf{B}, \quad \boldsymbol{\Sigma}_{\text{new}} = \boldsymbol{\Sigma}'(1:r, 1:r) \tag{4.17}$$

where $\mathbf{B} = \begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{0}_{1\times r} \end{bmatrix} \mathbf{U}'(1:r,:) + \boldsymbol{\Omega}\mathbf{U}'((r+1),:)$ and $\boldsymbol{\Sigma}_{\text{new}}$ formed by taking the first $r \times r$ elements of $\boldsymbol{\Sigma}'$. The matrix $\mathbf{B}$ represents the coefficients of linear expansions of our updated mapped values. We refer the interested reader to [33] for a detailed derivation of the aforementioned calculations.

The form of Equation (4.17) involves $\boldsymbol{\Psi}_{x,\text{new}}$, which is not explicitly expressed. By employing the kernel trick and leveraging dot products, we execute essential substitutions. We derive an updated $\hat{\mathbf{K}}$ from Equation (4.8) by replacing $\hat{\mathbf{A}}, \mathbf{U}, \boldsymbol{\Sigma}$ with $\hat{\mathbf{A}}_{\text{new}}, \mathbf{U}_{\text{new}}, \boldsymbol{\Sigma}_{\text{new}}$, leading to the following expression:

$$\begin{aligned} \hat{\mathbf{K}}_{\text{new}} &= (\boldsymbol{\Sigma}_{\text{new}}^{\dagger}\mathbf{U}_{\text{new}}^T)\hat{\mathbf{A}}_{\text{new}}(\mathbf{U}_{\text{new}}\boldsymbol{\Sigma}_{\text{new}}^{\dagger}) \\ &= \boldsymbol{\Sigma}_{\text{new}}^{\dagger}\mathbf{B}^T\boldsymbol{\Psi}_{x,\text{new}}^T\boldsymbol{\Psi}_{y,\text{new}}\boldsymbol{\Psi}_{x,\text{new}}^T\boldsymbol{\Psi}_{x,\text{new}}\mathbf{B}\boldsymbol{\Sigma}_{\text{new}}^{\dagger} \\ &= \boldsymbol{\Sigma}_{\text{new}}^{\dagger}\mathbf{B}^T\hat{\mathbf{A}}_{\text{new}}\hat{\mathbf{G}}_{\text{new}}\mathbf{B}\boldsymbol{\Sigma}_{\text{new}}^{\dagger}. \end{aligned} \tag{4.18}$$

Here, $\hat{\mathbf{A}}_{\text{new}} = \boldsymbol{\Psi}_{y,\text{new}}\boldsymbol{\Psi}_{x,\text{new}}^T$ and $\hat{\mathbf{G}}_{\text{new}} = \boldsymbol{\Psi}_{x,\text{new}}\boldsymbol{\Psi}_{x,\text{new}}^T$. Importantly, $\hat{\mathbf{G}}_{\text{new}}$ and $\hat{\mathbf{A}}_{\text{new}}$ do not require re-calculation from scratch; the specified kernel function is simply applied to the new vectors $\mathbf{x}_{\text{new}}, \mathbf{y}_{\text{new}}$ and the existing matrices $\mathbf{X}, \mathbf{Y}$.

With our incrementally updated $\hat{\mathbf{K}}_{\text{new}}$, we retrieve the eigenvalues and eigenvectors, $\mathbf{\Lambda}_{\text{new}}, \mathbf{V}_{\text{new}}$, allowing us to update our DMD modes

$$\mathbf{\Phi}_{\text{kdmd, new}} = \mathbf{X}_{\text{new}}\mathbf{B}\mathbf{\Sigma}^{\dagger}_{\text{new}}(\mathbf{V}^{-1}_{\text{new}})^{T}. \tag{4.19}$$

With $\mathbf{\Lambda}_{\text{new}}$, $\mathbf{\Phi}_{\text{kdmd, new}}$, we finally forecast the future trajectory:

$$\tilde{\mathbf{x}}_t = \mathbf{\Phi}_{\text{kdmd, new}}\mathbf{\Lambda}^{t}_{\text{new}}\hat{b}_0 \tag{4.20}$$

As before, we arrange the predictions from (4.20) similarly to (4.5) and extract the last row as in (4.6) to make our forecasts.

4.1.4 Randomized Kernel DMD

We now present an alternative randomized kernel DMD (RKDMD) method based on randomized linear algebra techniques [54]. This method utilizes randomized approaches to approximate the necessary DMD modes and eigenvalues for forecasting both efficiently and effectively.

Building on the methodologies outlined in [94, 51], we employ Random Fourier Features (RFF) to approximate kernel functions, notably the RBF Gaussian kernel. A key advantage of RFFs is their ability to provide an explicit approximation of the feature map $\psi(\mathbf{x})$, as described by the following formula:

$$\psi(\mathbf{x}) := \sqrt{\frac{2c}{s}}\left[\cos(\theta_i + z_i x)\right]_{i=1}^{s} \tag{4.21}$$

Here, $\theta_1, \ldots, \theta_s \in \mathbb{R}$ are uniformly distributed in $[0, 2\pi)$, and the random vectors $z_1, \ldots, z_s \in \mathbb{R}^d$ follow a normal distribution informed by the RBF kernel parameter ν. The normalization constant c is set at 1 to adhere to the statistical properties of the RBF kernel. A detailed pseudocode for implementing Equation (4.21) can be found in [51].

Equation (4.21) facilitates the explicit construction of $\mathbf{\Psi}_x$ and $\mathbf{\Psi}_y$, enabling the application of SVD. However, given that the user-defined dimension s may be substantial, traditional SVD at each incremental timestep can be computationally prohibitive. To address this issue,

we employ a randomized SVD approach [94], which significantly accelerates the kernel DMD algorithm. The following steps outline this process.

Initially, the feature map matrix $\mathbf{\Psi}_x$ is projected onto a lower-dimensional space as shown in the equation:

$$\mathbf{\Psi}_x \mathbf{P} = \mathbf{W} \tag{4.22}$$

where $\mathbf{P} \in \mathbb{R}^{s \times r}$ is a random Gaussian matrix with truncation parameter r. The resultant matrix $\mathbf{W}$ is then decomposed using QR factorization to obtain $\mathbf{W} = \mathbf{Q}_R \mathbf{R}$. Subsequently, we compute the projection $\mathbf{Q}_R^T \mathbf{\Psi}_x = \mathbf{D}$ and perform SVD on $\mathbf{D}$, resulting in $\mathbf{D} = \mathbf{U}_D \mathbf{\Sigma} \mathbf{Z}^*$, where $\mathbf{\Sigma}$ and $\mathbf{Z}^*$ correspond to the decomposed components as defined in Equation (4.7). Finally, we obtain our basis $\mathbf{U}$ by projecting back to the original space using $\mathbf{U} = \mathbf{Q}_R \mathbf{U}_D$. This procedure completes the extraction of all the components required by Equation (4.7), allowing us to proceed with the kernel DMD method described in Section 4.1.2 and eventual forecast with Equations (4.3), (4.5), and (4.6).

4.2 Experiments

We conduct a set of experiments to evaluate the performance of the incremental and randomized versions of kernel dynamic mode decomposition. We use six real-world and two synthetic time series forecasting benchmarks to quantitatively and visually validate the generality of our proposed methods.

4.2.1 Datasets

We now introduce the six datasets for our univariate time series experiments, where the chosen variables serve as target variables in previous benchmark tests [134, 85, 128]. (1) Electricity [4] records the hourly electricity consumption of 321 clients from 2012 to 2014. (2) ETTh1 [134] contains the time series of oil temperature (OT) and power load collected by electricity transformers from July 2016 to July 2018 recorded every hour; we use OT as the target variable. (3) Exchange [73] collects the panel data of daily exchange rates for 8

countries from 1990 to 2016; we use France in our experiments. We alter the exchange rates to exchange returns, as it is a common technique to measure the volatility of a exchange rate [18]. (4) ILI [2] collects the ratio of influenza-like illness patients versus the total patients in one week, which is reported weekly by Centers for Disease Control and Prevention of the United States from 2002 to 2021; we use the count for patients aged 0-4 for the target variable. (5) Traffic [1] contains the hourly road occupancy rates measured by 862 sensors in the San Francisco Bay area freeways from January 2015 to December 2016. (6) Weather [3] includes meteorological time series with 21 weather indicators collected every 10 minutes from the Weather Station of the Max Planck Biogeochemistry Institute in 2020; we model temperature measured in Celsius. To generate the main results, a standard protocol is followed that divides each dataset into training, validation, and testing subsets. The split ratio is 6:2:2 for the ETTh1 dataset and 7:1:2 for all the others [85].

These real-world datasets were selected for their non-stationary variability. To quantify this characteristic, we conduct the Augmented Dickey-Fuller Test (ADF) [40] on the univariate target variable, a widely used hypothesis test for detecting non-stationary behavior. Refer to Table 4.1 for a summary of the datasets and their respective ADF test statistics, where a smaller value suggests greater stationarity and stability in the time series.

Additionally, synthetic datasets are employed to illustrate non-stationary and volatile behaviors of the investigated methods. A synthetic non-stationary time series is generated, characterized by periodicity (sine wave), a linear trend, and stochastic noise adhering to a normal distribution. We refer to this as as the SNT-TS. Moreover, a time series based on the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) process is constructed, with a detailed description of this GARCH process provided by [25].

4.2.2 Benchmarks and Implementation details

We evaluate the IKDMD and RKDMD methods against a handful of state of the art machine learning forecasting methods. Such methods include Informer [134], Non-Stationary Transformer (NS Transformer) [85] and DLinear [128]. These benchmark methods are chosen for

Dataset	**Sampling Frequency**	**Total Observations**	**ADF Statistic**
ETTh1	1 Hour	17,420	-6.225
ILI	1 Week	966	-6.507
Electricity	1 Hour	26,304	-6.857
Weather	10 Min	52,695	-8.407
Traffic	1 Hour	17,544	-15.161
Exchange Returns	1 Day	7,588	-22.798

Table 4.1: Summary of the real-world datasets. Smaller ADF test statistic indicates a more stationary dataset. Accordingly, ETTh1 is the most non-stationary and Exchange Returns in the most stationary in this list.

their ability to predict non-stationary time series.

All the deep learning methods are implemented with PyTorch using a single NVIDIA GTX 1080 GPU. We keep many of the hyperparameters of the deep learning methods the same from previous benchmarking experiments [85]. This includes model training with the ADAM optimizer and an L2 loss function with an initial learning rate of 10^{-4} and a batch size of 32. The Transformer based methods, Informer and Non-stationary, have two encoder layers and one decoder layer. Prior to training, we normalize the time series so that so that evaluations are equalized across datasets. We evaluate the main results with test Mean Square Error (MSE) and Mean Average Error (MAE) under several prediction lengths. A lower MSE/MAE indicates better performance.

The primary hyperparameters for our proposed DMD methods include the number of snapshots m, which are used to form the Hankel matrix H, the truncation parameter r, and the RBF kernel parameter ν. Additionally, the RKDMD method requires specifying the parameter s, which defines the dimensionality of our feature map. Typically, s is fixed

relative to m according to the formula proposed by [51]:

$$s = O(\sqrt{m}\log(m)) \tag{4.23}$$

In the validation phase, we explore a range of values for m from 20% to 50% of the total time series length T, r from 50% to 100% of m, and ν from 10^{-1} to 10^{-10}. After optimizing these hyperparameters and finalizing the modes and eigenvalues used for forecasting, we proceed to evaluate IKDMD and RKDMD on the test set, incorporating an incremental update approach. In other words, for every new test data point, we update the method using the steps outlined in sections 4.1.3 and 4.1.4, which ultimately updates the DMD modes and eigenvalues that govern the forecasting Equation (4.6).

4.2.3 Main Results

Table 4.2 presents the comparative results of IKDMD and RKDMD against the other benchmark methods on the six real-world datasets for three different prediction sequence lengths. Both IKDMD and RKDMD demonstrate competitive forecasting capabilities in comparison to the deep learning methods. Notably, IKDMD and RKDMD achieve lower MSE and MAE values in all the cases, except for the Weather dataset and the 720 prediction sequence in the Traffic dataset. Furthermore, RKDMD consistently exhibits close approximation to IKDMD, as evident from the similar error metrics across all the experiments. More importantly, IKDMD and RKDMD shows superior performance compared to the other benchmarks in the most non-stationary datasets (ETTh1, ILI, Electricity) showcasing it's ability to handle unstable time series.

To illustrate the performance of each method, we provide forecasting plots for each dataset in Figure 4.1. We transform the values back to their original scale and omit RKDMD for clarity. Generally, IKDMD closely aligns with the ground truth in all the datasets. Specifically, Figures 4.1b and 4.1e demonstrate that IKDMD closely matches the ground truth shapes for the ILI and Traffic datasets, respectively. Although DLinear exhibits smaller errors in the Weather dataset, the performance of our proposed methods remains competitive,

as shown in Figure 4.1d. These results affirm the utility of our methods as effective forecasting tools.

We critically evaluate each method performance by examining their handling of non-stationary behavior. For instance, Figure 4.1a illustrates that the IKDMD forecast for the ETTh1 dataset closely mimics the inherent variability of the ground truth, unlike Informer, which tends to project a cyclical pattern uncharacteristic of the actual data. Conversely, DLinear and NS-Transformer both tend to underestimate the ground truth. A similar analysis for the Electricity dataset in Figure 4.1c reveals that while Informer predicts a constant line, failing to capture any dynamics, DLinear underestimates peak values, and NS-Transformer deviates significantly from the actual data pattern. In contrast, IKDMD generally replicates both the shape and volatility of the ground truth. For the Exchange Returns, shown in Figure 4.1f, Informer notably overestimates the ground truth. We further analyze the performances of these deep learning techniques in the next section, focusing on their handling of volatile time series such as Exchange Returns.

4.2.4 Additional Performance Visualizations

To further validate the performance of the IKDMD and RKDMD methods, additional visualizations are presented under various scenarios. First, the performance of these methods in a streaming or incremental update context is explored. Figure 4.2 illustrates the evolution of forecasting capability as new data (snapshots) are incrementally introduced for the SNT-TS and Traffic datasets. Initially, with a limited number of snapshots ($m = 100$), the forecasting accuracy is modest. However, as more data become available, the forecast curves increasingly align with the actual data trends. This underscores the robustness of the IKDMD and RKDMD methods in scenarios requiring incremental updates. Additionally, it is observed that the performance of IKDMD/RKDMD is influenced by the choice of the truncation parameter r, which dictates the extent of noise or variability of our method. Lower values of r tend to emphasize the general trends in the time series data, whereas, higher r values promote fitting to noisy data. We explore the effects of m and r further in the attached

supplementary material.

Transformer-based methods are often found to underperform in the analysis of noisy and volatile time series. This issue is highlighted in Figure 4.3, where we analyze the performance under volatile conditions present in the SNT-TS, GARCH, Exchange Returns, and Electricity datasets. For the SNT-TS dataset, both IKDMD and DLinear not only effectively capture the trends and cyclical behaviors, but also closely approximate the noisy behavior in the ground truth data. In contrast, Transformer tends to focus on predicting the overall trends and periodicities without adequately representing the noise. Similarly, for the Exchange Returns, Electricity, and GARCH time series, IKDMD and DLinear effectively capture the inherent volatility, whereas Informer and NS-Transformer forecasts generally revert to the mean of the series. Notably, IKDMD captures the volatility peaks more accurately as compared to DLinear.

To further quantify the volatility of the data and each method's predictions, we calculate the standard deviation of the predictions and compare them to the corresponding ground truth values for the datasets shown in Figure 4.3. Although other measures of volatility are employed in financial applications, standard deviation is the most commonly used metric [36]. The results, presented in Table 4.3, clearly show that IKDMD matches the ground truth standard deviation more closely than all the other methods, corroborating the visual evidence discussed earlier.

4.2.5 Inference Efficiency

One potential drawback of IKDMD and RKDMD is their tendency to increase in size as more snapshots are incorporated, as governed by mode Equations (4.10) and (4.19). Nevertheless, in the context of the benchmarks used in this study, they demonstrate superior inference times. We evaluate the average inference times across 30 predictions, each with a sequence length of 720, using the Weather dataset. Due to its extensive time series history, this dataset generates the largest matrices for the DMD modes and eigenvalues, thereby representing the worst-case scenario for inference evaluation. For IKDMD and RKDMD, inference primarily

involves matrix multiplication as outlined in Equation (4.3). We estimate the memory usage based on the matrix size, assuming each element in a PyTorch complex matrix occupies 16 bytes, resulting in memory allocations of 448 MiB for IKDMD and 436 MiB for RKDMD. In comparison, DLinear, the most efficient benchmark method, requires 687 MiB of memory and reports 0.4 ms inference time under a multivariate scenario [128]. The inference time results are presented in Table 4.4.

4.3 Discussion

The main findings from our study reveal three key outcomes. First, IKDMD and RKDMD demonstrate superior performance over state of the art deep learning methods in forecasting non-stationary time series data. Additionally, there is evidence that suggest our methods are robust to limited data samples compared to the other methods. For instance, in the ILI dataset, despite its non-stationary nature and limited sample size (see Table 4.1), the error results for our methods are substantially smaller as compared to the other deep learning techniques. Second, our visual analysis highlights that Transformer-based methods are inferior to our methods ability to predict volatile time series. This capability is crucial in financial data analysis, where precise prediction of market volatility is essential. Last, despite the increasing complexity and size of the eigen-pairs with larger data volumes, our methods continue to achieve superior inference times, outperforming even the highly efficient DLinear method.

Beyond experimental performance, our DMD methods offer significant practical benefits over traditional machine learning methods. First, IKDMD and RKDMD require only a single pass through the data, as opposed to multiple training epochs needed for deep learning methods. This substantially reduces training time and enables cost-effective fine-tuning. Additionally, unlike machine learning methods that must be trained for specific output lengths, our DMD methods use pre-computed modes and eigenvalues to forecast any desired future point, as outlined in Equation (4.20). Moreover, the eigenpairs produced by the DMD methods facilitate detailed eigen-analysis, allowing for the filtering and enhancement of specific

time series behaviors. For instance, eigenvalues with real parts indicate exponential decay or growth, while imaginary components suggest oscillatory behavior [71]. An advanced application of this is residual DMD [34], which enables the filtering of spurious DMD eigen-pairs through a spectral analysis. These advantages underscore the efficiency and flexibility our DMD methods bring to forecasting non-stationary time series data.

There are a few limitations to our proposed DMD methods. First, our experimental results have been demonstrated only for univariate cases. While extending our method to multivariate cases is theoretically straightforward and likely to yield favorable results, this extension has not yet been empirically tested. Another issue is the direct correlation between data volume and DMD eigen-pair size. Although this does not affect inference times significantly in our current benchmarks due to the truncation parameter r, the eigenvalues and DMD mode size could theoretically grow indefinitely. Practical implementations may need defined stopping or restarting criteria to facilitate sustainable, long-term deployment. Additionally, our current implementation of RKDMD is limited to RBF kernels, constrained by the use of Random Fourier Features that specifically approximate these kernels.

4.4 Conclusions

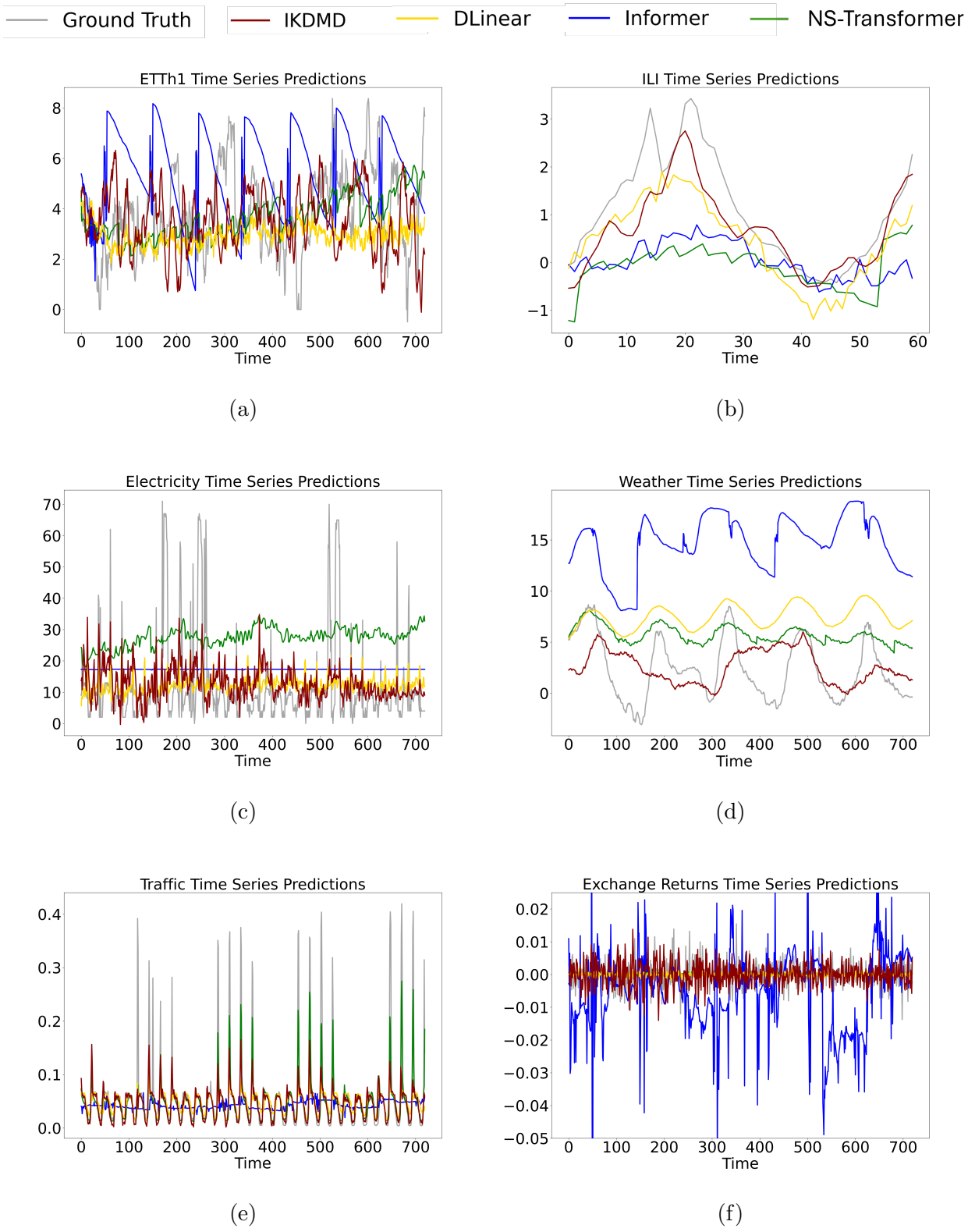
Our research introduces several significant advancements to the field of time series forecasting through the development of a novel method, the Incremental Kernel Dynamic Mode Decomposition (IKDMD). By mathematically integrating incremental kernel singular value decomposition with kernel dynamic mode decomposition, we establish a robust framework for analyzing time series data. Additionally, we introduce the Randomized Kernel Dynamic Mode Decomposition (RKDMD), which utilizes randomized linear algebra techniques to further enhance the efficiency of IKDMD. Our experimental evaluations demonstrate that IKDMD/RKDMD outperforms state-of-the-art machine learning methods, including transformers, in five out of six datasets. This is particularly notable in datasets characterized by high non-stationarity. We also provide visual evidence of this superior performance through forecasting plots, which clearly illustrate how IKDMD excels in incremental data stream

settings and outshines other methods in handling non-linear and volatile data.

Looking toward future research, there are several promising paths for enhancing our DMD methods. Although our methods have the potential to grow indefinitely, not every new data point necessitates an update. For example, streaming DMD implementations use a residual-based threshold to update the method based on reconstruction error and can resize the DMD modes once it exceeds a predefined size [58]. Adapting this solution to Kernel DMD would require a different approach due to its unique method of solving for modes and eigenvalues. Additionally, there is potential to integrate our methods into hybrid forecasting frameworks. They could serve either as a global method capturing overarching trends or as a local component focusing on specific temporal dynamics.

Methods		IKDMD		RKDMD		NS Transformer		Informer		DLinear	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.046**	**0.169**	0.073	0.193	0.106	0.250	0.294	0.447	0.091	0.243
	192	**0.052**	**0.178**	0.083	0.226	0.125	0.272	0.303	0.457	0.122	0.270
	720	**0.071**	**0.210**	0.086	0.254	0.153	0.312	0.298	0.452	0.158	0.313
ILI	24	0.723	0.622	**0.711**	**0.616**	1.871	0.954	2.813	1.123	1.872	0.993
	32	0.756	**0.631**	**0.752**	0.632	2.092	1.011	3.001	1.165	1.821	0.964
	60	1.131	0.794	**1.123**	**0.761**	2.232	1.055	3.274	3.000	1.353	0.931
Electricity	96	**0.272**	0.398	0.323	0.433	0.457	0.398	0.881	0.617	0.355	**0.326**
	192	**0.401**	0.432	0.415	0.454	0.571	0.467	0.911	0.655	0.411	**0.363**
	720	**0.784**	**0.553**	0.822	0.626	0.941	0.738	1.161	0.774	0.856	0.591
Weather	96	0.222	0.445	0.171	0.237	0.126	0.266	0.254	0.381	**0.101**	**0.225**
	192	0.225	0.423	0.179	0.338	0.226	0.371	0.402	0.505	**0.141**	**0.271**
	720	0.611	0.720	0.563	0.612	0.556	0.581	1.542	1.061	**0.327**	**0.430**
Traffic	96	0.321	0.224	**0.312**	**0.200**	0.414	0.279	1.143	0.562	0.670	0.367
	192	0.442	0.243	**0.424**	**0.230**	0.456	0.296	1.291	0.613	0.638	0.342
	720	0.735	0.280	0.721	0.278	**0.415**	**0.269**	1.632	0.647	0.725	0.374
Exchange Returns	96	0.953	0.823	0.681	**0.482**	0.674	0.508	3.992	1.470	**0.671**	0.501
	192	0.912	0.814	0.673	**0.454**	**0.663**	0.496	4.68	1.607	0.664	0.495
	720	0.634	0.605	**0.493**	**0.451**	0.523	0.457	6.037	1.897	0.530	0.458

Table 4.2: Univariate time-series forecasting results on six datasets with 3 prediction sequence lengths. The sequence lengths are 24, 32, 60 for ILI and 96, 192, 720 for all the others. We notice that IKDMD/RKDMD combined show lower error metrics for five of the six datasets with the sole exception of Weather.



(a) (b) (c) (d) (e) (f)

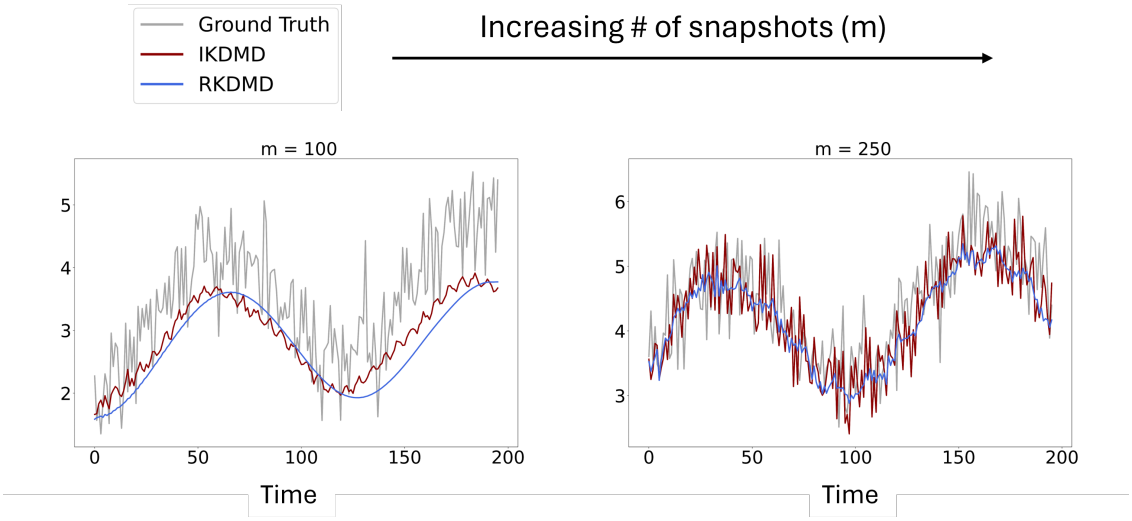
Figure 4.1: Forecast plots of each method for all the benchmark datasets: (a) ETTh1 (b) ILI (c) Electricity (d) Weather (e) Traffic (f) Exchange Returns.

Dataset	Ground Truth	IKDMD	DLinear	Informer	NS-Transformer
SNT-TS	0.91	**0.87**	0.75	0.07	0.75
GARCH	1.08	**0.64**	0.16	0.03	0.03
Exchange Returns	3.50e-3	**4.47e-3**	0.52e-3	12.81e-3	0.32e-3
Electricity	18.80	**7.50**	2.37	0.061	2.46

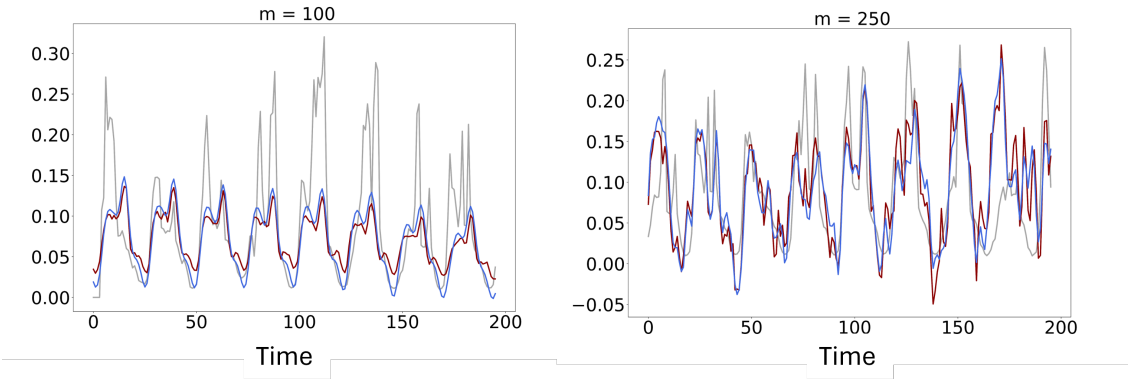
Table 4.3: Standard deviation of time series datasets and method forecasts to measure volatility. IKDMD notably closely matches the Ground Truth as compared to the other methods.

Method	Inference Time (in ms)
RKDMD	0.173 ± 0.0291
IKDMD	0.243 ± 0.0453
DLinear	0.267 ± 0.0288
NS-Transformer	5.450 ± 0.4860
Informer	22.2 ± 6.42

Table 4.4: Comparison of inference time of IKDMD/RKDMD and deep learning methods under 720 sequence prediction lengths on the Weather dataset. Both RKDMD and IKDMD report faster inferences times. The inference times are computed over 30 runs.



(a)



(b)

Figure 4.2: Incremental forecasting performance of IKDMD and RKDMD demonstrated on (a) SNT-TS and (b) Traffic Datasets. This figure illustrates how the forecasting accuracy of both the methods improves as new data (snapshots, m) are incrementally introduced into the system. The progressive enhancement of performance is evident across both the datasets.

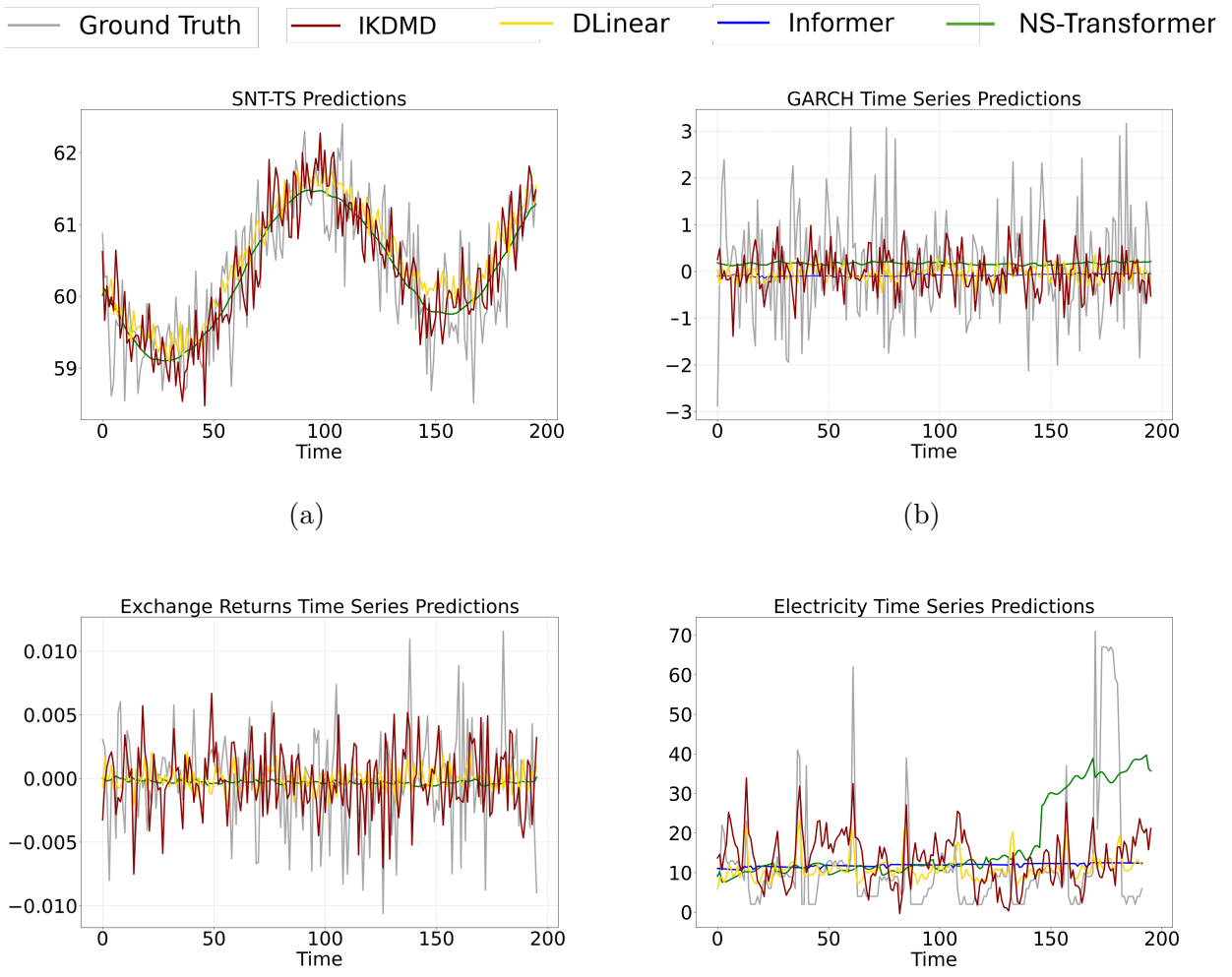


Figure 4.3: Prediction performance of each method for volatile and non-stationary time series data which include: (a) SNT-TS (b) GARCH (c) Exchange Returns (d) Electricity. IKDMD shows relatively well fitting curves for all the time series, including capturing the noisy components. In contrast, the other methods either underestimate or completely fail to capture the dynamics of volatile time series.

Chapter 5

ONLINE KERNEL DYNAMIC MODE DECOMPOSITION FOR STREAMING TIME SERIES FORECASTING WITH ADAPTIVE WINDOWING

5.1 Mathematical Background

5.1.1 Space–time transformation of the data

Consider a multivariate time series

$$\{\mathbf{x}_t\}_{t=1}^{T}, \quad \mathbf{x}_t = \left(x_t^{(1)}, \ldots, x_t^{(p)}\right)^\top \in \mathbb{R}^p,$$

where p is the number of components or features. Given a window length $w \in \mathbb{N}$, we form the windowed data matrix

$$\mathbf{X}_t = \begin{bmatrix} \mathbf{x}_{t-w+1} & \mathbf{x}_{t-w+2} & \cdots & \mathbf{x}_t \end{bmatrix} \in \mathbb{R}^{p\times w}. \tag{5.1}$$

Its jth row $\mathbf{X}_t^{(j)} \in \mathbb{R}^{1\times w}$ collects component j over the last w time steps:

$$\mathbf{X}_t^{(j)} = \left[x_{t-w+1}^{(j)}, \ldots, x_t^{(j)}\right]. \tag{5.2}$$

Next, we choose an autoregressive depth $d \leq w$. For each row $\mathbf{X}_t^{(j)},\ j = 1, \ldots, p$, we then construct the univariate Hankel matrix as

$$\boldsymbol{\mathcal{X}}_t^{(j)} = \begin{bmatrix} x_{t-w+1}^{(j)} & \cdots & x_{t-d+1}^{(j)} \\ x_{t-w+2}^{(j)} & \cdots & x_{t-d+2}^{(j)} \\ \vdots & \ddots & \vdots \\ x_{t-w+d}^{(j)} & \cdots & x_t^{(j)} \end{bmatrix} \in \mathbb{R}^{d\times(w-d+1)}. \tag{5.3}$$

Finally, we stack these p blocks to form the complete block–Hankel embedding:

$$\boldsymbol{\mathcal{X}}_t = \begin{bmatrix} \boldsymbol{\mathcal{X}}_t^{(1)} \\ \boldsymbol{\mathcal{X}}_t^{(2)} \\ \vdots \\ \boldsymbol{\mathcal{X}}_t^{(p)} \end{bmatrix} \in \mathbb{R}^{pd \times (w-d+1)}. \tag{5.4}$$

By employing a block–Hankel embedding to capture multi-step temporal correlations, we facilitate the discovery of sparse, robust eigen-analysis that is required for dynamic mode decomposition [26, 75].

5.1.2 Dynamic Mode Decomposition

Fix $m = w - d$. From the current block–Hankel matrix in (5.4), we extract

$$\begin{aligned} \mathbf{X} &= \big[\mathbf{x}_{t,1},\, \ldots,\, \mathbf{x}_{t,m}\big], \\ \mathbf{Y} &= \big[\mathbf{x}_{t,2},\, \ldots,\, \mathbf{x}_{t,m+1}\big]. \end{aligned} \tag{5.5}$$

Here, $\mathbf{x}_{t,i}$ denotes the ith column of the block–Hankel matrix $\boldsymbol{\mathcal{X}}_t$. Consequently, the columns of $\mathbf{Y}$ are each shifted one time step ahead of those in $\mathbf{X}$. Given snapshot matrices $\mathbf{X}, \mathbf{Y}$ from (5.5), we seek an operator $\mathbf{A}$ satisfying

$$\mathbf{Y} \;\approx\; \mathbf{A}\,\mathbf{X}.$$

The minimum-norm least-squares solution defines the standard DMD operator:

$$\mathbf{A}^{\mathrm{DMD}} \;=\; \mathbf{Y}\,\mathbf{X}^{\dagger}, \tag{5.6}$$

where † denotes the Moore–Penrose pseudoinverse. Further, we perform a rank-r truncated singular value decomposition (SVD)

$$\mathbf{X} = \mathbf{Q}_X\,\boldsymbol{\Sigma}_X\,\mathbf{V}_X^{*}, \tag{5.7}$$

with $\mathbf{Q}_X \in \mathbb{C}^{pd\times r}$, $\boldsymbol{\Sigma}_X \in \mathbb{C}^{r\times r}$, $\mathbf{V}_X \in \mathbb{C}^{m\times r}$, and $(\cdot)^*$ the conjugate transpose. Projecting $\mathbf{A}^{\mathrm{DMD}}$ into this reduced basis yields

$$\widetilde{\mathbf{A}} = \mathbf{Q}_X^{*}\,\mathbf{A}^{\mathrm{DMD}}\,\mathbf{Q}_X = \mathbf{Q}_X^{*}\,\mathbf{Y}\,\mathbf{V}_X\,\boldsymbol{\Sigma}_X^{-1}. \tag{5.8}$$

Its eigen-decomposition

$$\widetilde{\mathbf{A}}\,\mathbf{W} = \mathbf{W}\,\mathbf{\Lambda}, \quad \mathbf{W}, \mathbf{\Lambda} \in \mathbb{C}^{r\times r},$$

provides eigenvalues $\mathbf{\Lambda}$ and eigenvectors $\mathbf{W}$, from which the DMD modes are recovered as

$$\mathbf{\Phi} = \mathbf{Q}_X\,\mathbf{W} \;\in\; \mathbb{C}^{pd\times r}. \tag{5.9}$$

Finally, the initial amplitudes are obtained from the last column of the snapshot matrix $\mathbf{Y}$:

$$\mathbf{b}_0 = \mathbf{\Phi}^{\dagger}\,\mathbf{x}_{t,m+1}\,. \tag{5.10}$$

Forecasting k steps into the future in delay-coordinate space then proceeds by

$$\widehat{\mathcal{X}}_k = \mathbf{\Phi}\,\mathbf{\Lambda}^{\,k}\,\mathbf{b}_0\,. \tag{5.11}$$

Selecting the appropriate p rows of $\widehat{\mathcal{X}}_k$ yields the p-dimensional forecast $\widehat{\mathbf{x}}_{t+k}$.

5.2 Methodology

We now introduce our new method, which we call Windowed Online Random Kernel-Dynamic Mode Decomposition, or WORK-DMD for short. Our approach consists of three main steps. **Step 1** explicitly lifts the data to approximate a Gaussian kernel using Random Fourier Features (RFF) [94], to enable nonlinear modeling while keeping the feature dimension manageable. **Step 2** performs an online kernel-DMD update each time a new snapshot arrives and produces forecasts. **Step 3** decodes those forecasts through the current DMD modes and eigenvalues to obtain the final prediction. A general overview of the entire method is shown in Figure 5.1.

5.2.1 Random-feature lifting

Kernel methods usually exploit the kernel trick to sidestep high-dimensional computations, as shown in kernel DMD [67]. In our setting, the product of series dimension p, Hankel depth d, and window size w can be large, so direct kernel arithmetic (cf. Section 5.1) can be impractical.

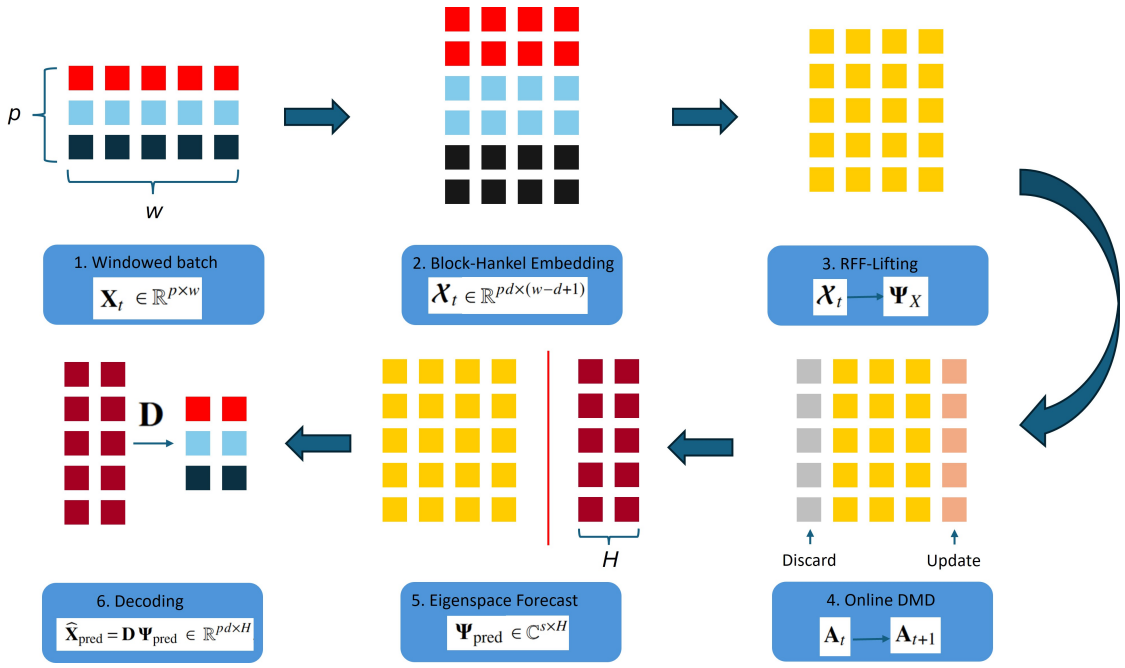


Figure 5.1: WORK-DMD methodology pipeline. The method processes multivariate time series data through the following steps: (1) Input multivariate windowed time series $\mathbf{X}_t \in \mathbb{R}^{p\times w}$ with p features, (2) Construction of block-Hankel embedding $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{pd\times m}$ to capture temporal correlations across multiple lag orders, (3) Random Fourier Feature (RFF) lifting to transform data into kernel feature space $\boldsymbol{\Psi}_X, \boldsymbol{\Psi}_Y \in \mathbb{R}^{s\times m}$ using Gaussian kernel approximation, (4) Online Dynamic Mode Decomposition update using Sherman-Morrison formulation to maintain forward update $\mathbf{A}_{t+1} \in \mathbb{R}^{s\times s}$ with streaming data by discarding old snapshots (gray) and incorporating new observations (orange), (5) Generate forecast in feature space via eigendecomposition of SVD-compressed features $\boldsymbol{\Psi}_{\text{pred}} \in \mathbb{C}^{s\times H}$ (red divider separates current features from future predictions), and (6) Decoding via matrix $\mathbf{D}$ to transform feature-space predictions back to physical coordinates $\hat{\mathbf{x}}_{t+h} \in \mathbb{R}^p$. Color coding maintains feature identity throughout the pipeline: original time series features (red, teal, blue) are preserved through Hankel embedding, transformed to kernel features (yellow), processed through online updates, and decoded back to multivariate predictions.

To control dimensionality, we *explicitly* lift each block–Hankel snapshot from (5.4) into an s-dimensional space with RFF. For a Gaussian kernel of bandwidth γ, the mapping is

$$\psi(\mathbf{x}) = \sqrt{\frac{2}{s}}\Big[\cos\big(\theta_i + \mathbf{z}_i^\top \mathbf{x}\big)\Big]_{i=1}^{s}, \tag{5.12}$$

where θ_i are drawn uniformly from $[0, 2\pi)$ and the random vectors $\mathbf{z}_i \in \mathbb{R}^{pd}$ follow a multivariate normal distribution whose covariance is determined by γ. A detailed pseudocode for computing (5.12) is given in [51]. Applying ψ column-wise to the data blocks $\mathbf{X}$ and $\mathbf{Y}$ from (5.5) yields the lifted snapshots

$$\mathbf{\Psi}_X = \psi(\mathbf{X})\,, \quad \mathbf{\Psi}_Y = \psi(\mathbf{Y}) \in \mathbb{R}^{s\times m}, \tag{5.13}$$

which are then used in the subsequent online kernel-DMD step.

5.2.2 Online Kernel DMD

We adapt the update steps described in [129] for the online DMD problem. For the initial window size, we compute the equivalent DMD operator from (5.6) with

$$\mathbf{P}_t = \big(\mathbf{\Psi}_X \mathbf{\Psi}_X^\top + \varepsilon \mathbf{I}\big)^{-1}, \qquad \mathbf{A}_t = \mathbf{\Psi}_Y \mathbf{\Psi}_X^\top \mathbf{P}_t\,. \tag{5.14}$$

In practice, $\varepsilon\mathbf{I}$ is used to avoid numerical problems during the inverse operation; ε taken to be $10^{-6}\|\mathbf{\Psi}_X \mathbf{\Psi}_X^\top\|_2$. We use the Sherman-Morrison formulation [129] to update (5.14), which ultimately contains the components that allow forecasting.

Consider a rolling window of $\mathbf{X}, \mathbf{Y}$, where the oldest snapshot columns $\mathbf{x}_{t,1}, \mathbf{x}_{t,2}$ are discarded and new snapshots $\mathbf{x}_{t,m+1}, \mathbf{x}_{t,m+2}$ are introduced. We define the following matrices:

$$\begin{aligned}
\mathbf{U} &= \begin{bmatrix}\psi(\mathbf{x}_{t,1}) & \psi(\mathbf{x}_{t,m+1})\end{bmatrix}, \\
\mathbf{V} &= \begin{bmatrix}\psi(\mathbf{x}_{t,2}) & \psi(\mathbf{x}_{t,m+2})\end{bmatrix}, \\
\mathbf{C} &= \begin{bmatrix}-1 & 0 \\ 0 & 1\end{bmatrix}.
\end{aligned} \tag{5.15}$$

We also define the auxiliary matrix $\mathbf{\Gamma}$ as

$$\mathbf{\Gamma} = \left(\mathbf{C}^{-1} + \mathbf{U}^{\top}\mathbf{P}_t\mathbf{U}\right)^{-1}, \tag{5.16}$$

which captures how much the current $\mathbf{P}_t$ matrix aligns with the directions of the new and old snapshots. The set of Sherman-Morrison equations [129] is then used to update the following:

$$\mathbf{P}_{t+1} = \mathbf{P}_t \; - \; \mathbf{P}_t\mathbf{U}\,\mathbf{\Gamma}\,\mathbf{U}^{\top}\mathbf{P}_t, \tag{5.17}$$

$$\mathbf{A}_{t+1} = \mathbf{A}_t \; + \; \left(\mathbf{V} - \mathbf{A}_t\mathbf{U}\right)\mathbf{\Gamma}\,\mathbf{U}^{\top}\mathbf{P}_t. \tag{5.18}$$

The intuitive interpretation of the above steps is that the term $\left(\mathbf{V} - \mathbf{A}_t\mathbf{U}\right)$ in equation (5.18), scaled by $\mathbf{\Gamma}$, can be considered a prediction error of the current model $\mathbf{A}_t$. The above formulae are meant to be used in a recursive manner.

5.2.3 Feature-space forecasting

After each online update, we have the current forward update operator $\mathbf{A}_{t+1} \in \mathbb{C}^{s\times s}$ and its rank-r SVD basis $\mathbf{Q}_r \in \mathbb{R}^{s\times r}$, obtained from the refreshed feature matrix $\mathbf{\Psi}_X$. Forecasting then proceeds in three algebraic steps.

(i) Projection onto the leading SVD modes.

$$\mathbf{K}_{t+1} = \mathbf{Q}_r^{\top}\,\mathbf{A}_{t+1}\,\mathbf{Q}_r \;\in\; \mathbb{R}^{r\times r}. \tag{5.19}$$

(ii) Eigendecomposition in the reduced space.

$$\begin{aligned} &\mathbf{K}_{t+1}\,\mathbf{W}_r = \mathbf{W}_r\,\mathbf{\Lambda}_r, \\ &\mathbf{W}_r,\; \mathbf{\Lambda}_r \;\in\; \mathbb{C}^{r\times r}, \\ &\qquad \mathbf{\Lambda}_r = \mathrm{diag}\,\left(\lambda_1, \ldots, \lambda_r\right). \end{aligned} \tag{5.20}$$

(iii) Forecast in the compressed basis.

$$\mathbf{b}_0 = \mathbf{W}_r^{-1}\,\mathbf{Q}_r^{\top}\,\psi\big(\mathbf{x}_{t,m+1}\big), \tag{5.21}$$

$$\mathbf{E} = \big[\lambda_i^h\big]_{i=1,\,h=0}^{r,\,H-1} \;\in\; \mathbb{C}^{r\times H}, \tag{5.22}$$

$$\mathbf{\Psi}_{\text{pred}} = \mathbf{Q}_r\,\mathbf{W}_r\,\big(\mathbf{b}_0 \odot \mathbf{E}\big) \;\in\; \mathbb{C}^{s\times H}, \tag{5.23}$$

where $\odot$ denotes element-wise multiplication. Equation (5.23) is the compressed analog of the full-space forecast. Its columns give the feature-space trajectories $\psi\big(\widehat{\mathbf{x}}_{t,m+1+h}\big)$ for (time) horizon $h = 1, \ldots, H$, with $\mathbf{x}_{t,m+1}$ being the most recent snapshot after the update. The matrix $\mathbf{E}$ is a Vandermonde-like structure encoding the temporal evolution of each eigenvalue across all forecast horizons [115], where entry $E_{ih} = \lambda_i^h$ captures how the i-th mode propagates h steps forward in time.

5.2.4 Decoding to the physical space

To convert feature–space forecasts back to the original coordinates we build a *decoder* matrix from the current lifted snapshot pair. Let $\mathbf{\Psi}_X \in \mathbb{R}^{s\times m}$ and $\mathbf{X} \in \mathbb{R}^{pd\times m}$ denote, respectively, the lifted and physical Hankel blocks at time step $t+1$. The decoder is obtained by a one–time ridge-free least–squares fit:

$$\mathbf{D} = \mathbf{X}\,\mathbf{\Psi}_X^{\dagger} \;\in\; \mathbb{R}^{pd\times s}. \tag{5.24}$$

Applying $\mathbf{D}$ to the feature–space trajectory $\mathbf{\Psi}_{\text{pred}}$ from (5.23) yields the decoded forecast matrix

$$\widehat{\mathbf{X}}_{\text{pred}} = \mathbf{D}\,\mathbf{\Psi}_{\text{pred}} \;\in\; \mathbb{R}^{pd\times H}, \tag{5.25}$$

whose columns provide the predicted snapshots $\widehat{\mathbf{x}}_{t,m+1+h}$, for horizon $h = 1, \ldots, H$, in the original data space.

5.3 Results

This section describes the experimental data sets and the online forecasting results.

5.3.1 Datasets

The datasets used in this study have typically served as benchmarks to generalize time series forecasting performance of a model, particularly with state-of-the-art methods such as deep learning models [78]. We use it to compare to other recent models who have their own methods for the online time series forecasting problem. Electricity Transformer Temperature (ETT) [126] captures load and oil temperature data of power transformers at 15-minute intervals from July 2016 to July 2018. It is composed of hourly granularity data (ETTh2) and 15-minute granularity data (ETTm1). Traffic [1] contains the hourly road occupancy rates measured by 862 sensors in the San Francisco Bay area freeways from January 2015 to December 2016. The Weather (WTH) [64] dataset contains hourly records of 11 climate features from almost 1,600 locations across the United States.

5.3.2 Implementation details

We follow the implementation details used in the FSNet method [92], where the data are split into warm-up and online testing phases by a 25:75 ratio. However, since our method is not a traditional machine/deep learning model, we use the warm-up phase to tune our hyperparameters. We perform hyperparameter tuning for $\{r, d, \gamma, s\}$ via random search with 3-fold rolling cross-validation on the warm-up data, independently for each dataset. In addition, the mean and standard deviation are calculated in the warm-up phase to normalize the online testing samples. For all the benchmark datasets, we vary the forecasting window as $H \in \{1, 24, 48\}$. We evaluate forecasting performance using Mean Squared Error (MSE) and Mean Absolute Error (MAE) as our primary error metrics. All experiments were conducted on an Nvidia GeForce RTX 3070 GPU with 8GB of RAM using PyTorch.

The optimized hyperparameter configurations for WORK-DMD show consistent values across all the datasets for gamma ($\gamma = 1 \times 10^{-4}$) and auto-regressive depth ($d = 30$), while window sizes and RFF dimensions are adapted to dataset characteristics. Specifically, ETTh2 and WTH utilize larger window sizes ($w = 120$) with RFF dimensions of $s = 1024$

and 512, respectively, while ETTm1 and Traffic employ smaller windows ($w = 60$) with RFF dimensions of $s = 256$ and 512, respectively. The rank parameters are determined by the column rank available in each experiment. Additional sensitivity studies that examine the impact of hyperparameter variations on performance are provided in Appendix C.1.

5.3.3 Baseline of Adaptation Methods

We compare our method against five recent online forecasting approaches: **OnlineTCN** [76, 16], an adaptation of Temporal Convolutional Networks for incremental updates; **DER++** [28], a continual learning method mitigating catastrophic forgetting through replay and regularization; **Informer** [134], a transformer model with Prob-Sparse attention for efficient long-sequence prediction; **FSNet** [92], a frequency-spectral network capturing multi-scale dependencies with fast and slow learning; and **OneNet** [118], a unified framework dynamically ensembling models for online forecasting under concept drift. These baselines span diverse architectures including transformer-based, convolutional, spectral, and continual learning, providing a balanced reference for evaluating online forecasting performance.

5.3.4 Online Forecasting Results

The forecasting results in Table 5.1 show that the proposed WORK-DMD method achieves competitive or superior accuracy compared to established baselines across diverse datasets and prediction horizons. Notably, WORK-DMD consistently achieves the lowest error metrics for short-term forecasts ($H = 1$), where both MSE and MAE improvements are the most pronounced. Across all datasets, WORK-DMD matches or outperforms deep learning models such as OneNet, FSNet, and Informer, demonstrating its effectiveness at capturing both short-term dynamics and longer-term dependencies compared to the baseline methods.

Further evidence of WORK-DMD's strong performance is seen in the following prediction plots, where we primarily compare with OneNet as it is the closest competitor. Figure 5.2 demonstrates that for longer $H = 48$ horizons, our method closely matches the cyclical behavior of the ETTh2 time series across two different instances. Furthermore, there is

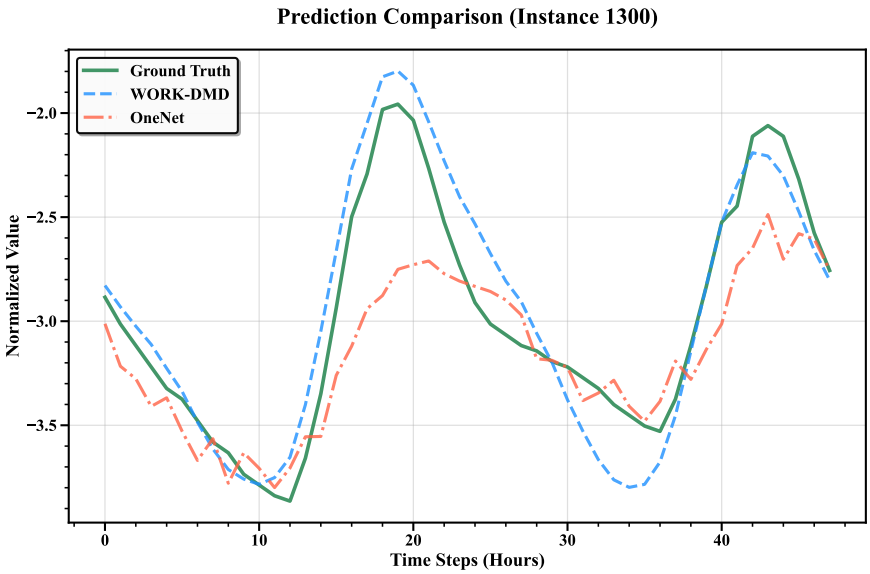
Table 5.1: Multivariate Forecasting results, averaged over all variables (horizon H).

Dataset	H	OnlineTCN		DER++		Informer		FSNet		OneNet		WORK-DMD	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	1	0.502	0.436	0.508	0.375	7.571	0.850	0.466	0.368	0.380	0.348	**0.289**	**0.334**
	24	0.830	0.547	0.828	0.540	4.629	0.668	0.687	0.467	0.532	**0.407**	**0.480**	0.453
	48	1.183	0.589	1.157	0.577	5.692	0.752	0.846	0.515	0.609	**0.436**	**0.603**	0.516
ETTm1	1	0.214	0.085	0.083	0.192	0.456	0.512	0.085	0.191	0.082	0.187	**0.004**	**0.102**
	24	0.258	0.381	0.196	0.326	0.478	0.525	0.115	0.249	**0.098**	0.225	0.102	**0.200**
	48	0.283	0.403	0.208	0.340	0.377	0.460	0.127	0.263	**0.108**	**0.238**	0.121	0.314
Traffic	1	0.315	0.283	0.289	**0.248**	0.795	0.507	0.288	0.253	-	-	**0.275**	0.265
	24	0.452	0.363	0.387	0.295	1.267	0.750	0.362	**0.288**	-	-	**0.357**	0.291
WTH	1	0.206	0.276	0.174	0.235	0.426	0.458	0.162	0.216	0.156	0.201	**0.149**	**0.197**
	24	0.308	0.367	0.287	0.351	0.370	0.417	0.188	0.276	**0.175**	0.255	0.220	**0.253**
	48	0.302	0.362	0.294	0.359	0.367	0.419	0.223	0.301	**0.200**	0.279	0.261	**0.275**

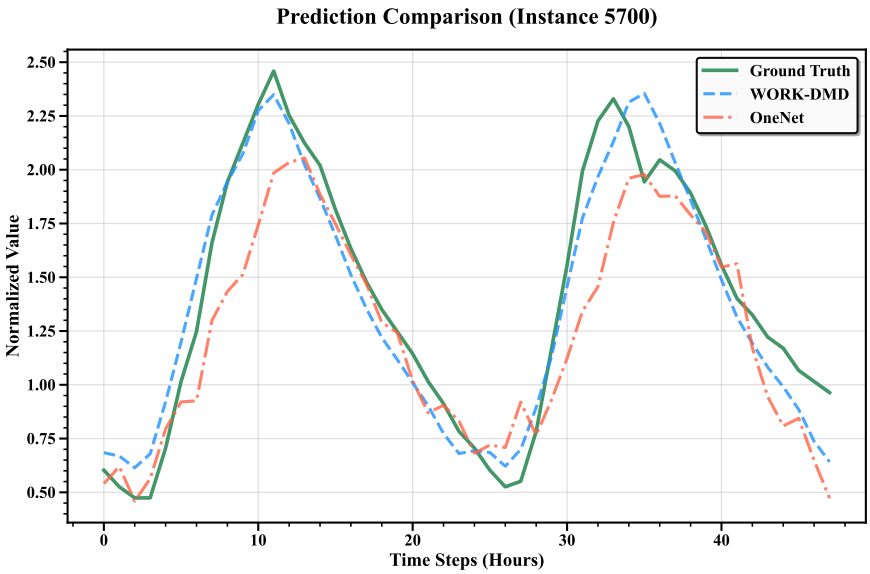
Note: OneNet results for the Traffic dataset were not provided by the authors.

evidence that our method adapts to mean shifts in the time series, as shown by the model operating in negative value space in Figure 5.2(a) and positive value space in Figure 5.2(b). Figure 5.3 shows the consecutive single-step ahead forecasts against the ground truth, showcasing how well WORK-DMD adheres to the ground truth patterns for ETTm1. Figure 5.4 illustrates two instances within the WTH time series for $H = 24$ forecasting of the Wet Bulb Temperature variable, showing that WORK-DMD can capture underlying dynamics despite erratic behavior at different stages of the time series.

Finally, to show the adaptive nature of our method, we report cumulative MSE plots for the ETTh2 and WTH datasets. Figure 5.5 shows the cumulative MSE plots for $H = 1$ and $H = 48$ for ETTh2 compared to OneNet. Despite a drastic non-stationary shift in the ETTh2 time series, WORK-DMD is able to recover, sometimes at a faster rate than OneNet,



(a) Instance 1300



(b) Instance 5700

Figure 5.2: Time series forecasting comparison on ETTh2 for the Oil Temperature (OT) variable. The figures show 48-hour ahead predictions from WORK-DMD and OneNet against ground truth for two different instances: (a) Instance 1300 and (b) Instance 5700. In both instances, WORK-DMD demonstrates better adherence to the ground truth compared to OneNet.

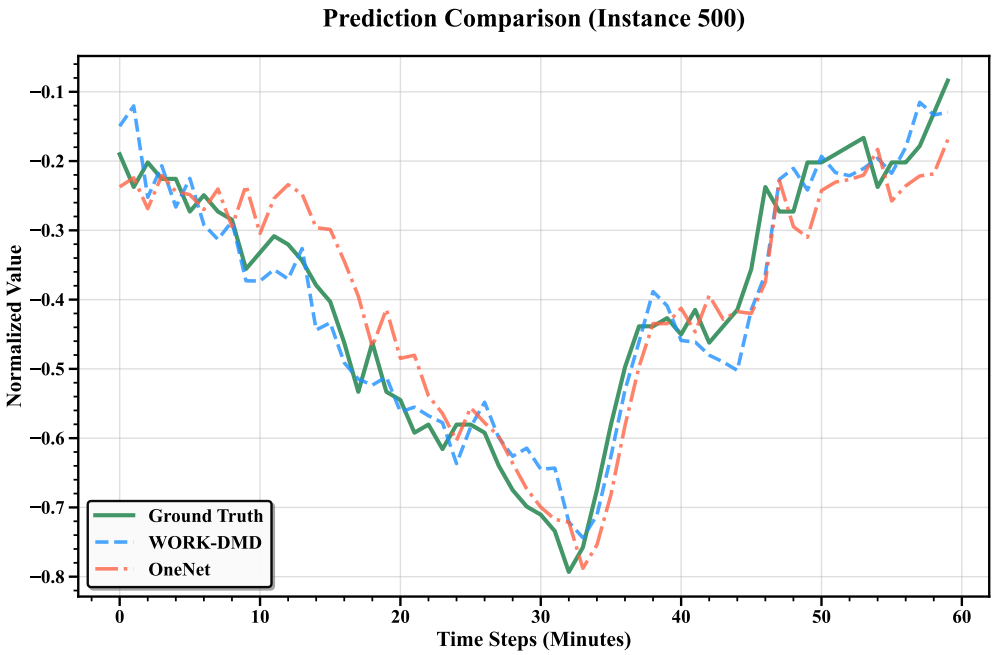
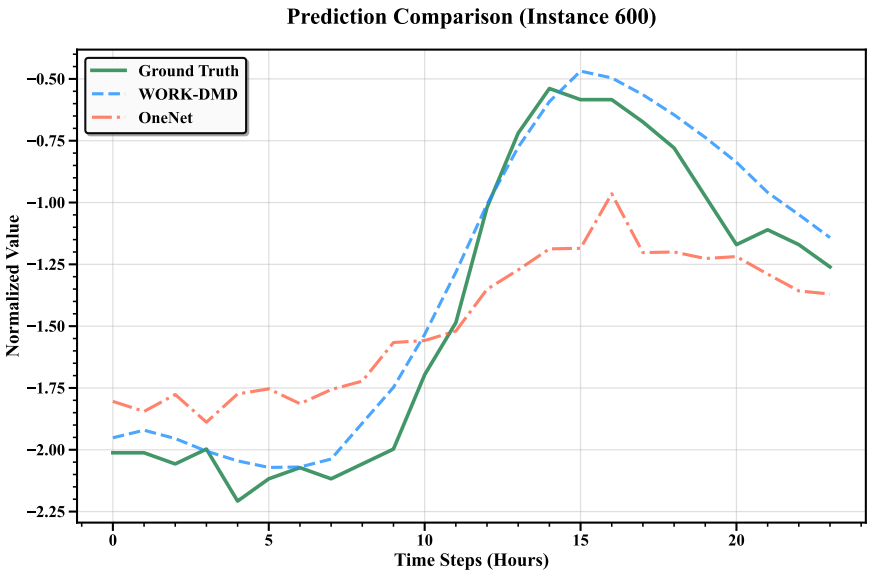
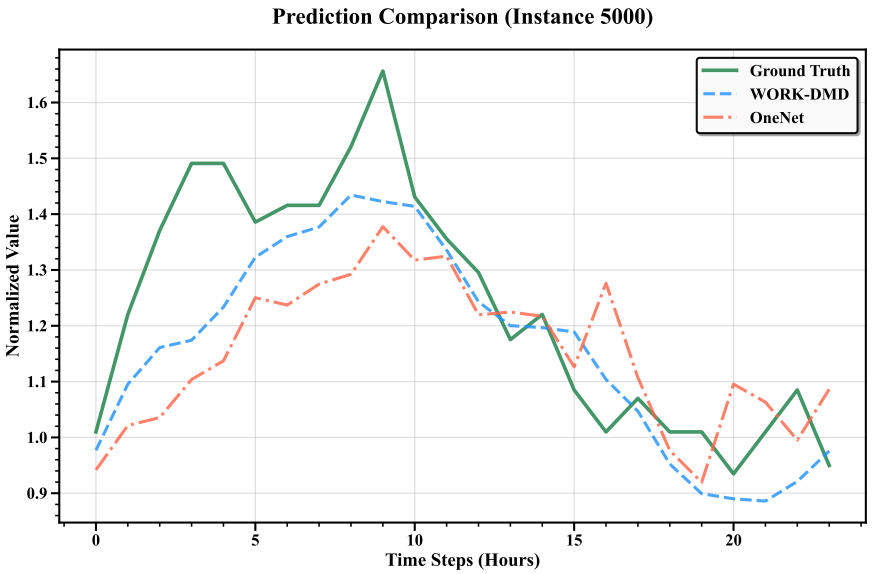


Figure 5.3: Time series forecasting comparison on ETTm1 for the Oil Temperature (OT) variable. The figure shows 1-step ahead predictions comparing WORK-DMD and OneNet against ground truth for instance 500. WORK-DMD exhibits marginally closer alignment to the ground truth, demonstrating detailed tracking of temporal patterns.



(a) Instance 600



(b) Instance 5000

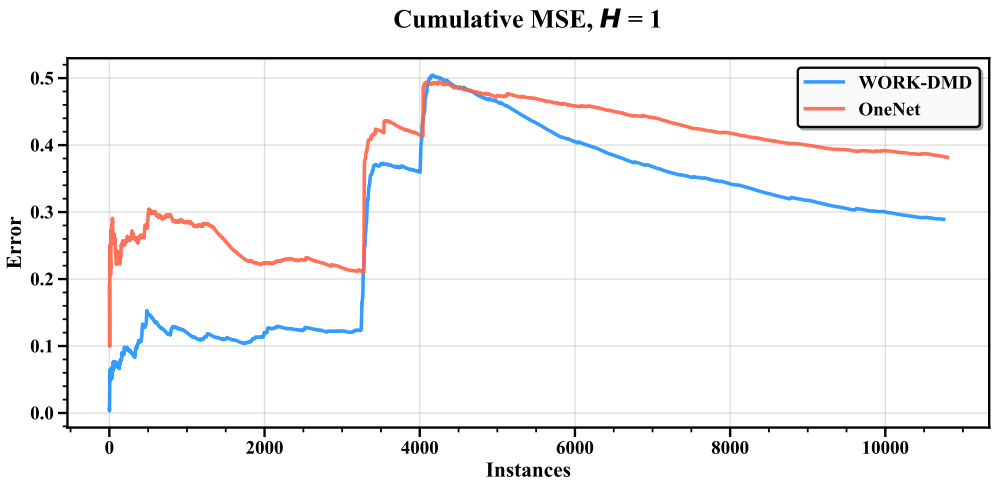
Figure 5.4: Time series forecasting comparison on WTH dataset for the Wet Bulb Temperature variable. The figures show 24-hour ahead predictions from WORK-DMD and OneNet against ground truth for two different instances: (a) Instance 600 and (b) Instance 5000. In (a), WORK-DMD demonstrates better adherence to the ground truth compared to OneNet, while in (b) both methods exhibit comparable performance. Notably, WORK-DMD showcases adaptive capabilities by accurately forecasting across different temperature regimes, operating in predominantly negative values in (a) and transitioning to positive values in (b).

eventually settling into a lower error score. Figure 5.6 presents the cumulative MSE for $H = 1$ and $H = 48$ for the WTH dataset. We see that WORK-DMD, despite structural shifts in the time series, adapts to such changes due to our updating method discarding older snapshots that no longer represent the dynamics properly. Although OneNet achieves a lower error in Figure 5.6(b), WORK-DMD remains within a competitive margin. Additional forecasting and cumulative error plots for the Traffic dataset are shown in Appendices C.2 and C.3.

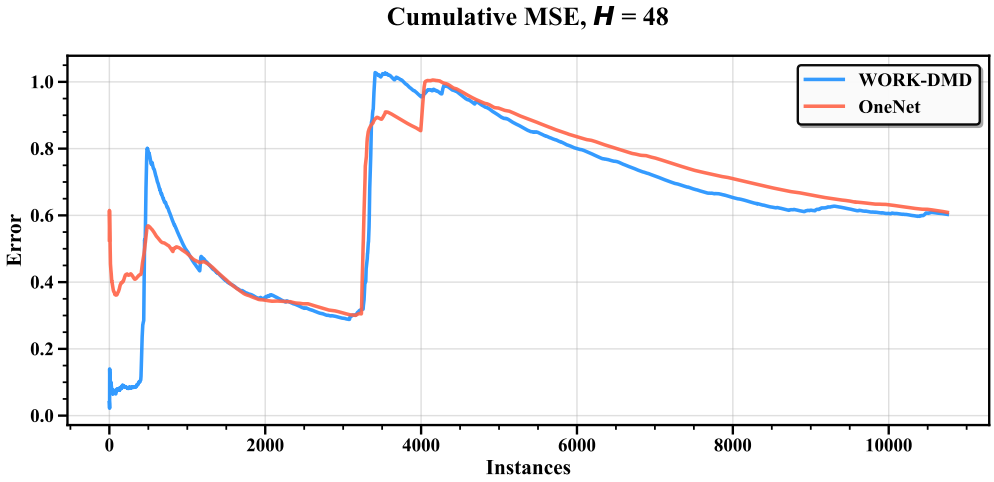
5.3.5 Sample Efficiency

A key advantage of WORK-DMD is its single-pass learning capability: each data point is processed only once, unlike deep learning methods that require multiple passes through the same data. To quantify this efficiency, we conduct a sequential online forecasting comparison between WORK-DMD, OneNet and FSNet using the Traffic dataset, tracking the total number of sample exposures each method requires during the training and online operation. Our experiment uses 100 traffic features with both methods starting from identical initial training data (60 time steps) and processing 100 streaming updates sequentially. WORK-DMD sees each sample once during training and once per online update, while OneNet/FSNet processes training data through 10 epochs and performs 1 epoch per online update using previous ground truth for supervision.

Table 5.2 shows that WORK-DMD achieves superior accuracy across all horizons while requiring 37× fewer sample exposures than OneNet/FSNet. At short horizons ($H = 1, 24$), WORK-DMD delivers approximately 2× better MSE respectively, while maintaining competitive performance. Figure 5.7 illustrates representative forecasting performance, demonstrating how WORK-DMD tracks ground truth patterns effectively with dramatically reduced sample requirements. This efficiency stems from WORK-DMD's Sherman-Morrison matrix updates that extract all necessary information in a single pass.

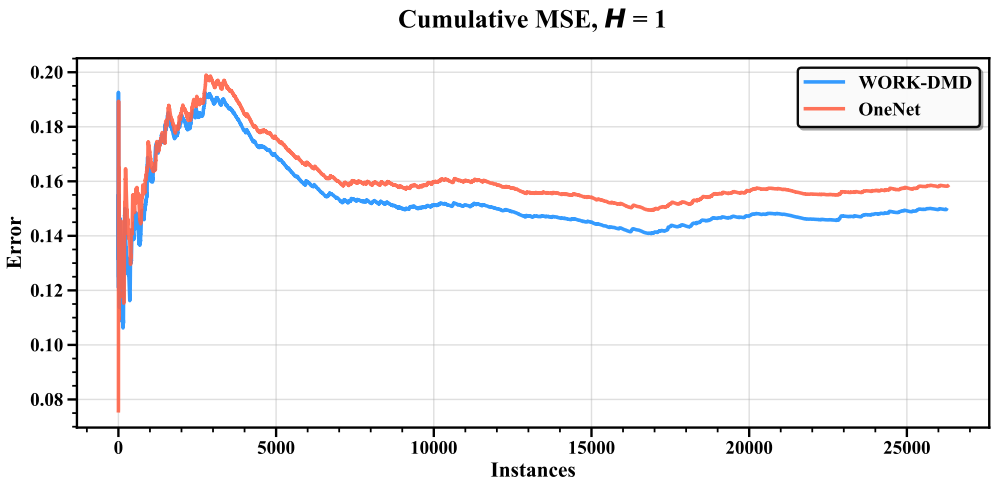


(a) ETTh2 1-step ahead

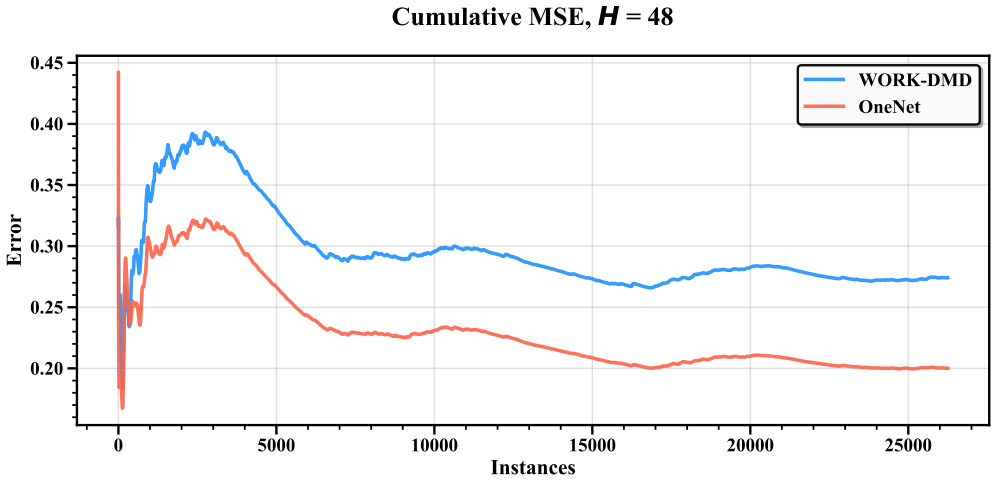


(b) ETTh2 48-step ahead

Figure 5.5: Cumulative MSE comparison on ETTh2 dataset. The figures show cumulative mean squared error progression for (a) ETTh2 1-step ahead and (b) ETTh2 48-step ahead forecasting, comparing WORK-DMD and OneNet performance. In (a), WORK-DMD demonstrates faster error correction, resulting in lower cumulative error accumulation compared to OneNet. In (b), both methods exhibit comparable performance across the forecasting horizon.



(a) 1-step ahead



(b) 48-step ahead

Figure 5.6: Cumulative MSE comparison on WTH dataset. The figures show cumulative mean squared error progression for (a) 1-step ahead and (b) 48-step ahead forecasting, comparing WORK-DMD and OneNet performance over time. In (a), WORK-DMD marginally outperforms OneNet with slightly lower cumulative error accumulation. In (b), OneNet demonstrates better performance, though WORK-DMD remains competitive and maintains close proximity throughout the forecasting horizon.

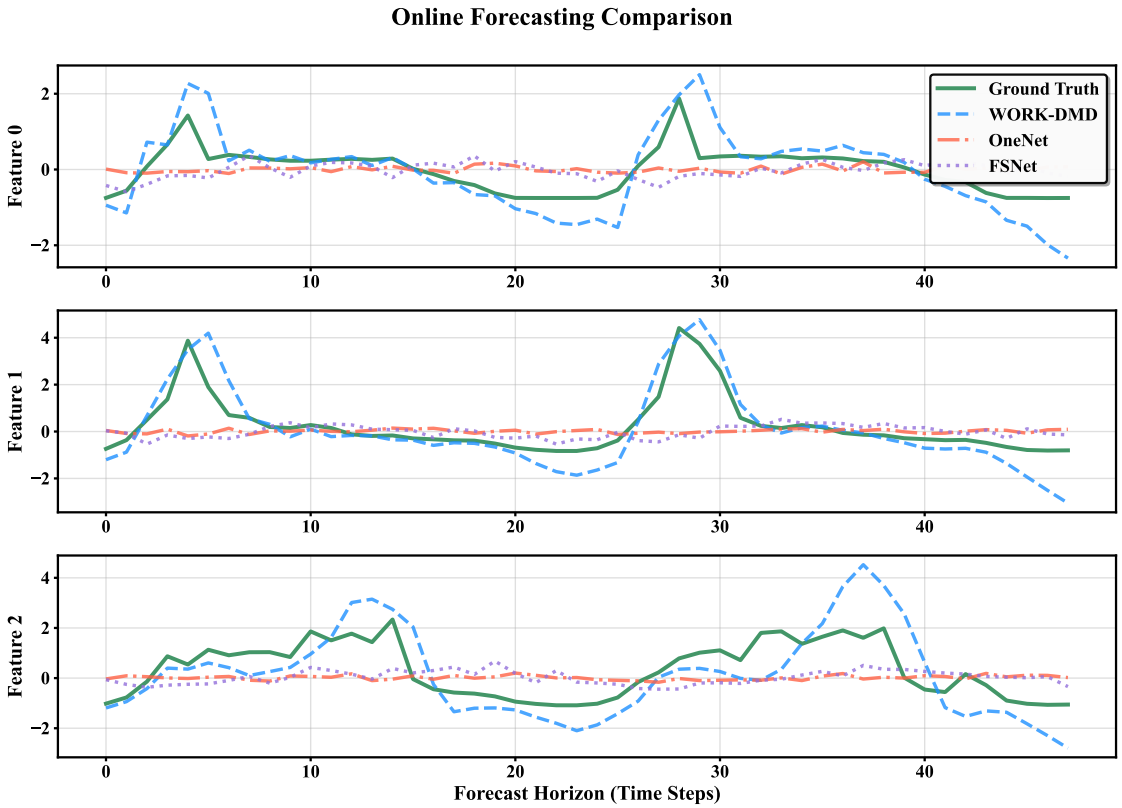


Figure 5.7: Multivariate online forecasting comparison on Traffic dataset for first three monitoring channels under limited training data. All methods start from identical initial training data and process streaming updates sequentially. WORK-DMD sees each sample once during training and once per update, while FSNet/OneNet requires multiple training epochs plus additional epochs per update. WORK-DMD achieves superior accuracy across forecasting horizons while requiring substantially fewer total sample exposures, demonstrating the efficiency of single-pass learning.

Table 5.2: Sample efficiency comparison in sequential online forecasting: FSNet vs. OneNet vs. WORK-DMD on Traffic dataset.

Horizon	FSNet	OneNet	WORK-DMD
Mean Squared Error (MSE)			
$H = 1$	0.447	0.855	0.241
$H = 24$	0.635	0.732	0.304
$H = 48$	0.597	0.675	0.448
Sample Efficiency			
FSNet/OneNet Total Exposures		5,940	
WORK-DMD Total Exposures		160	
Efficiency Ratio		37.1×	

5.4 Discussion

WORK-DMD addresses critical challenges in real-world streaming applications where traditional forecasting methods are inadequate due to computational constraints and data limitations. Our comprehensive evaluation across four benchmark datasets demonstrates that WORK-DMD consistently achieves competitive or superior performance compared to state-of-the-art online forecasting methods, with particularly strong results in short-term forecasting where it outperforms existing approaches across most datasets.

The sample efficiency analysis reveals compelling practical advantages: WORK-DMD achieves superior accuracy while requiring dramatically fewer sample exposures than competing deep learning methods. This efficiency directly translates into reduced data collection costs, faster deployment timelines, and lower computational overhead. For applications such as smart traffic management (800+ monitoring stations) or energy grid monitoring, this

efficiency may be viable for real-time processing on edge hardware.

The cumulative MSE analyses provide insights into WORK-DMD's adaptability in non-stationary environments. The results demonstrate adaptation to regime shifts in both weather and electrical transformer datasets, where the method recovers from dramatic changes that would typically require manual intervention in conventional approaches. This adaptability stems from the online Sherman-Morrison updates that automatically discard outdated information while incorporating new dynamics.

From a methodological perspective, several design choices impact performance and deployment considerations. Window size selection presents a fundamental trade-off between adaptation speed and pattern retention, with smaller windows enabling rapid response to changes but potentially missing longer-term trends. Decoder update frequency represents another practical consideration: updating after every prediction ensures maximum accuracy but increases computational load, while periodic updates based on domain knowledge or systematic approaches provide more efficient alternatives. Eigenvalue monitoring offers a principled approach to system health assessment and model interpretability. Unlike black-box deep learning methods, DMD provides access to eigenvalue analysis, which encodes information about system dynamics, dominant frequencies, and temporal patterns [72]. This spectral decomposition enables systematic approaches to detecting when model retraining or parameter adjustment is necessary. Recent work has demonstrated that DMD-based methods can be leveraged for change point detection by monitoring eigenvalue drift and reconstruction errors [68], suggesting potential extensions of WORK-DMD for automated system monitoring. However, developing robust online change point detection methods from eigenvalue patterns remains an active area of research.

Beyond efficiency metrics, the computational advantages presented here enable new deployment paradigms. WORK-DMD's Sherman-Morrison updates operate with lower computational complexity per timestep compared to full matrix operations, while maintaining constant memory requirements rather than growing with data stream length. These properties could enable deployment on embedded systems and IoT devices, where traditional neural

networks might exceed hardware constraints.

However, some limitations constrain the applicability of the method. As a system identification approach, WORK-DMD assumes underlying dynamical structure that may not exist in purely stochastic processes such as random walks, white noise processes, or series dominated by measurement noise without underlying dynamics [27]. The method excels with structured time series that exhibit recognizable patterns, such as traffic flows, energy consumption, and sensor readings, but may struggle with highly irregular or chaotic dynamics. Additionally, hyperparameter optimization remains a practical challenge, particularly for automated deployment scenarios where domain expertise is limited. This is of particular interest for the auto regressive depth, which continues to be an open research problem within the dynamic mode decomposition community.

Future research should address practical deployment challenges for widespread industrial adoption. Adaptive hyperparameter tuning mechanisms that respond to changing data characteristics could enhance robustness across diverse applications. Integration with anomaly detection systems would enable automatic model reset when the underlying dynamics shift dramatically. Extension to explicit multivariate forecasting with cross-variable modeling could expand applicability to complex systems such as chemical process control, smart manufacturing, and energy management where variable interactions drive system behavior [113, 44, 79].

The advantages of computational efficiency support sustainable AI deployment strategies by reducing resource requirements compared to traditional deep learning approaches. The method's ability to operate with limited data collection addresses privacy concerns in sensitive applications while enabling predictive capabilities in data-constrained environments. However, the interpretability advantages of eigenvalue evolution must be balanced against risks of over-reliance on automated decisions in critical infrastructure, emphasizing the importance of human oversight protocols in safety-critical deployments.

5.5 Conclusion

We present WORK-DMD, a method that integrates Random Fourier Features with online Dynamic Mode Decomposition to enable real-time forecasting from streaming data. By capturing nonlinear dynamics through explicit feature mapping while maintaining fixed computational cost per update, WORK-DMD transforms traditional kernel methods into a practical online learning framework.

Experimental evaluation shows WORK-DMD achieves competitive or superior accuracy compared to recent online forecasting methods across benchmark datasets while requiring only a single pass through the data. The method excels in short-term forecasting scenarios where rapid adaptation is critical. Unlike transformer-based models that require substantial computational resources and lengthy training, WORK-DMD generates reliable predictions from the current windowed snapshot alone. This efficiency could make it suitable for resource-constrained environments such as edge devices and IoT sensors.

By augmenting dynamical systems theory with randomized kernel methods, WORK-DMD demonstrates that sophisticated nonlinear modeling need not require deep learning's computational overhead. The combination of efficiency, minimal data requirements, and interpretable eigenvalue diagnostics positions WORK-DMD as a valuable tool for applications demanding real-time adaptation under strict computational constraints.

Chapter 6

CONCLUSIONS

This dissertation has addressed fundamental challenges in time series forecasting through two complementary perspectives: enhancing the interpretability of deep learning models and developing efficient dynamical systems-based forecasting methods. The research progresses from understanding the limitations of existing approaches to proposing novel solutions that balance accuracy, efficiency, and interpretability in both batch and streaming forecasting scenarios.

6.1 Summary of Contributions

Chapter 3 introduced a distance correlation-based framework for analyzing RNN effectiveness in time series forecasting. This approach revealed critical insights into how RNNs process temporal information through their activation layers. The analysis demonstrated that while RNNs effectively identify time series lag structures in early layers, this temporal information degrades rapidly across subsequent layers. This degradation is particularly pronounced for processes with large lag dependencies. Furthermore, the experiments exposed fundamental limitations in RNN capabilities for modeling moving average and heteroskedastic processes. The distance correlation heatmaps provided a systematic tool for comparing RNN architectures, revealing that input window size exerts substantially greater influence on model behavior than conventional hyperparameters such as the number of hidden units or activation function choice. These findings provide practitioners with actionable guidance for RNN design and highlight the need for alternative approaches when RNN limitations become prohibitive.

Chapters 4 and 5 addressed these limitations by exploring Dynamic Mode Decomposition

as a competitive alternative to deep learning methods. Chapter 4 presented an incremental kernel DMD approach that enhances forecasting performance for non-stationary time series in batch settings. By leveraging kernel methods to capture nonlinear dynamics while maintaining computational tractability through incremental updates, this work demonstrated that DMD-based approaches can achieve competitive accuracy with state-of-the-art machine learning models. These methods also offer interpretability through eigenvalue analysis. The method's ability to adapt to regime changes and evolving dynamics positions it as a valuable tool for practitioners working with non-stationary data.

Chapter 5 extended these concepts to streaming scenarios with WORK-DMD, which integrates Random Fourier Features with online DMD to enable real-time forecasting from continuously arriving data. By employing explicit feature mappings rather than implicit kernel methods, WORK-DMD achieves fixed computational complexity per update while capturing nonlinear dynamics. The experimental evaluation across benchmark datasets demonstrated remarkable sample efficiency. The method requires only single-pass learning while achieving competitive or superior accuracy compared to methods demanding multiple training epochs. This efficiency translates directly into reduced computational costs, faster deployment timelines, and the potential for edge device implementation in resource-constrained environments. The adaptive windowing mechanism enables natural handling of non-stationary dynamics without catastrophic forgetting, addressing a critical limitation of deep learning approaches in streaming contexts.

6.2 Broader Implications

The contributions of this dissertation extend beyond methodological innovations to address practical deployment considerations increasingly relevant in modern forecasting applications. The interpretability analysis of RNNs provides transparency that is essential for building trust in automated decision systems, particularly in safety-critical domains where understanding model behavior is paramount. The distance correlation framework offers practitioners a principled approach to diagnosing model failures and selecting appropriate architectures

based on data characteristics rather than trial-and-error experimentation.

The DMD-based methods presented in Chapters 4 and 5 demonstrate that sophisticated forecasting capabilities need not require the computational overhead of deep learning. For applications in smart infrastructure, industrial monitoring, and edge computing, where computational resources and training data are limited, these methods offer viable alternatives that maintain strong performance while enabling real-time deployment. The eigenvalue-based interpretability inherent to DMD provides system health monitoring capabilities unavailable in black-box neural networks, supporting proactive maintenance strategies and early warning systems.

From a sustainability perspective, the computational efficiency of WORK-DMD contributes to reducing the environmental footprint of AI systems. The single-pass learning paradigm minimizes energy consumption compared to methods requiring extensive training iterations. Meanwhile, the fixed memory requirements enable deployment on energy-efficient edge hardware. These characteristics align with growing emphasis on sustainable AI practices that balance performance with resource consumption.

6.3 Limitations and Future Directions

Despite these contributions, several limitations warrant acknowledgment. The distance correlation analysis of RNNs, while revealing fundamental behavioral patterns, was conducted primarily on univariate time series with controlled characteristics. Extension to multivariate settings and more complex real-world scenarios could reveal additional insights or nuanced behaviors not captured in the current study. The DMD-based approaches, while computationally efficient and interpretable, rely on assumptions of underlying dynamical structure that may not hold for purely stochastic or chaotic processes. Hyperparameter selection remains a practical challenge, particularly for automated deployment scenarios where domain expertise is unavailable.

Future research directions emerge naturally from these limitations. For RNN interpretability, investigating attention mechanisms and transformer architectures using distance

correlation could extend the analysis to state-of-the-art models. Developing automated diagnostic tools that leverage distance correlation patterns to recommend architecture modifications would enhance practical utility. For DMD-based methods, integrating adaptive hyperparameter tuning mechanisms that respond to changing data characteristics would improve robustness. Coupling WORK-DMD with anomaly detection systems could enable automatic model resets when underlying dynamics shift dramatically beyond the adaptation capabilities of incremental updates. Extension to explicit multivariate forecasting with cross-variable dependencies would expand applicability to complex systems such as chemical process control, smart manufacturing, and energy management.

The integration of these two research threads suggests promising hybrid approaches. Combining the pattern recognition capabilities of deep learning with the interpretability and efficiency of DMD-based methods could yield systems that leverage the strengths of both paradigms. Such hybrid architectures might employ neural networks for feature extraction and pattern recognition while using DMD for temporal modeling and forecasting. This approach could potentially achieve superior performance with enhanced interpretability.

6.4 Closing Remarks

This dissertation has demonstrated that advancing time series forecasting requires both understanding the limitations of existing approaches and developing innovative alternatives that address practical deployment challenges. The distance correlation framework provides tools for systematic RNN analysis that can guide architecture design and reveal fundamental modeling limitations. The DMD-based methods offer computationally efficient, interpretable alternatives that excel in scenarios where deep learning faces challenges of sample efficiency, computational constraints, and catastrophic forgetting.

As forecasting applications continue to expand into edge computing, real-time systems, and resource-constrained environments, the methods presented here provide practical pathways for deployment that balance accuracy, efficiency, and interpretability. The combination of rigorous theoretical foundations with comprehensive empirical validation positions these

contributions to impact both academic research and industrial practice. Future work building upon these foundations can further bridge the gap between theoretical capabilities and practical deployment, advancing the field toward forecasting systems that are not only accurate but also efficient, interpretable, and trustworthy.

BIBLIOGRAPHY

[1] Caltrans performance system, 2024.

[2] CDC for flu view, 2024.

[3] Max-Planck weather station institute, 2024.

[4] UCI machine learning repository, 2024.

[5] Yahoo finance - stock market live, quotes, business & finance news, 2024.

[6] Amaia Abanda, Usue Mori, and Jose A Lozano. A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2):378–412, 2019.

[7] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[8] Arwa Alanqary, Abdullah Alomar, and Devavrat Shah. Change point detection via multivariate singular spectrum analysis. *Advances in Neural Information Processing Systems*, 34:23218–23230, 2021.

[9] Cesare Alippi and Manuel Roveri. An adaptive cusum-based test for signal change detection. In *2006 IEEE international symposium on circuits and systems*, pages 4– pp. IEEE, 2006.

[10] Samaneh Aminikhanghahi and Diane J Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.

[11] Samaneh Aminikhanghahi and Diane J Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.

[12] T. W. Anderson. On the distribution of the two-sample cramer-von mises criterion. *The Annals of Mathematical Statistics*, 33(3):1148–1159, 1962.

[13] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[14] Alberto Badias and Ashis G Banerjee. Neural network layer algebra: A framework to measure capacity and compression in deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[15] Yasaman Bahri, Jonathan Kadmon, Jeffrey Pennington, Sam S Schoenholz, Jascha Sohl-Dickstein, and Surya Ganguli. Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*, 11:501–528, 2020.

[16] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[17] Domjan Barić, Petar Fumić, Davor Horvatić, and Tomislav Lipic. Benchmarking attention-based interpretability of deep learning in multivariate time series predictions. *Entropy*, 23(2):143, 2021.

[18] Joscha Beckmann and Rainer Schüssler. Forecasting exchange rates under parameter and model uncertainty. *Journal of International Money and Finance*, 60:267–288, 2016.

[19] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[20] Shaunak D. Bopardikar and George S. Eskander Ekladious. Towards scalable kernel machines for streaming data analytics . In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4730–4732. IEEE Computer Society, December 2017.

[21] Pedro Henrique Borghi, Oleksandr Zakordonets, and João Paulo Teixeira. A covid-19 time series forecasting model based on mlp ann. *Procedia Computer Science*, 181:940–947, 2021.

[22] Abdelhamid Bouchachia and Charlie Vanaret. Gt2fc: An online growing interval type-2 self-learning fuzzy classifier. *IEEE Transactions on Fuzzy Systems*, 22(4):999–1018, 2013.

[23] Ikram Bouchikhi, André Ferrari, Cédric Richard, Anthony Bourrier, and Marc Bernot. Kernel based online change point detection. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, 2019.

[24] Halima Bousqaoui, Ilham Slimani, and Said Achchab. Comparative analysis of short-term demand predicting models using arima and deep learning. *International Journal of Electrical and Computer Engineering*, 11(4):3319, 2021.

[25] Brockwell and Davis. *Introduction to Time Series and Forecasting*. Springer, 1996.

[26] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eric Kaiser, and J. Nathan Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8:19, 2017.

[27] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, Cambridge, UK, 2nd edition, 2022.

[28] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: A strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, pages 15920–15930, 2020.

[29] Edoardo Caldarelli, Philippe Wenk, Stefan Bauer, and Andreas Krause. Adaptive gaussian process change point detection. In *International Conference on Machine Learning*, pages 2542–2571. PMLR, 2022.

[30] National Geophysical Data Center. Solar features, Sep 2009.

[31] Vitor Cerqueira, Luis Torgo, and Carlos Soares. Machine learning vs statistical methods for time series forecasting: Size matters. *arXiv preprint arXiv:1909.13316*, 2019.

[32] Minshuo Chen, Xingguo Li, and Tuo Zhao. On generalization bounds of a family of recurrent neural networks. *arXiv preprint arXiv:1910.12947*, 2019.

[33] Tat-Jun Chin, Konrad Schindler, and David Suter. Incremental kernel SVD for face recognition with image sets. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 461–466. IEEE, 2006.

[34] Matthew J Colbrook, Qin Li, Ryan V Raut, and Alex Townsend. Beyond expectations: residual dynamic mode decomposition and variance for stochastic dynamical systems. *Nonlinear Dynamics*, 112(3):2037–2061, 2024.

[35] Bruno L Dalmazo, João P Vilela, and Marilia Curado. Online traffic prediction in the cloud: a dynamic window approach. In *2014 International Conference on Future Internet of Things and Cloud*, pages 9–14. IEEE, 2014.

[36] Kevin Daly. Financial volatility: Issues and measuring techniques. *Physica A: statistical mechanics and its applications*, 387(11):2377–2393, 2008.

[37] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 10148–10167. PMLR, 21–27 Jul 2024.

[38] Simon S Du, Yining Wang, Xiyu Zhai, Sivaraman Balakrishnan, Ruslan Salakhutdinov, and Aarti Singh. How many samples are needed to estimate a convolutional or recurrent neural network? *arXiv preprint arXiv:1805.07883*, 2018.

[39] Dominic Edelmann, Konstantinos Fokianos, and Maria Pitsillou. An updated literature review of distance correlation and its applications to time series. *International Statistical Review*, 87(2):237–262, 2019.

[40] Graham Elliott, Thomas J Rothenberg, and James H Stock. Efficient tests for an autoregressive unit root, 1992.

[41] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[42] Shereen Elsayed, Daniela Thyssens, Ahmed Rashed, Hadi Samer Jomaa, and Lars Schmidt-Thieme. Do we really need deep learning models for time series forecasting? *arXiv preprint arXiv:2101.02118*, 2021.

[43] Christos Faloutsos, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. Forecasting big time series: old and new. *Proceedings of the VLDB Endowment*, 11(12):2102–2105, 2018.

[44] Syeda Sitara Wishal Fatima and Afshin Rahimi. A review of time-series forecasting algorithms for industrial manufacturing systems. *Machines*, 12(6), 2024.

[45] Shibo Feng, Chunyan Miao, Ke Xu, Jiaxiang Wu, Pengcheng Wu, Yang Zhang, and Peilin Zhao. Multi-scale attention flow for probabilistic time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[46] Thomas Flynn and Shinjae Yoo. Change detection with the kernel cumulative sum algorithm. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 6092–6099. IEEE, 2019.

[47] Christian Francq and Jean-Michel Zakoian. *GARCH models: structure, statistical inference and financial applications*. John Wiley & Sons, 2019.

[48] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[49] Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. Efficient estimation of mutual information for strongly dependent variables. In *Artificial intelligence and statistics*, pages 277–286. PMLR, 2015.

[50] Alberto Gasparin, Slobodan Lukovic, and Cesare Alippi. Deep learning for time series forecasting: The electric load case. *CAAI Transactions on Intelligence Technology*, 7(1):1–25, 2022.

[51] Dimitrios Giannakis, Amelia Henriksen, Joel A Tropp, and Rachel Ward. Learning to forecast dynamical systems from streaming data. *SIAM Journal on Applied Dynamical Systems*, 22(2):527–558, 2023.

[52] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), August 2018.

[53] Zahra Hajirahimi and Mehdi Khashei. An optimal hybrid bi-component series-parallel structure for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11067–11078, 2023.

[54] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[55] Douglas M Hawkins, Peihua Qiu, and Chang Wook Kang. The changepoint model for statistical process control. *Journal of quality technology*, 35(4):355–366, 2003.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[57] Xiaoyu He, Suixiang Shi, Xiulin Geng, and Lingyu Xu. Information-aware attention dynamic synergetic network for multivariate time series long-term forecasting. *Neurocomputing*, 500:143–154, 2022.

[58] Maziar S Hemati, Matthew O Williams, and Clarence W Rowley. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids*, 26(11), 2014.

[59] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[61] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. Dsanet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2129–2132, 2019.

[62] W Kenneth Jenkins. Fourier series, fourier transforms and the dft. In *Mathematics for Circuits and Filters*, pages 83–111. CRC Press, 2022.

[63] Prajakta S Kalekar et al. Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi school of information Technology*, 4329008(13):1–13, 2004.

[64] Diana Kantor, Nancy W. Casey, Matthew J. Menne, and Andrew Buddenberg. Local climatological data (LCD), version 2. NOAA National Centers for Environmental Information, 2023. https://www.ncei.noaa.gov/data/local-climatological-data/.

[65] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[66] Yoshinobu Kawahara and Masashi Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 389–400. SIAM, 2009.

[67] I Kevrekidis, Clarence W Rowley, and M Williams. A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2016.

[68] Victor K. Khamesi, Niall M. Adams, Dean A. Bodenham, and Edward A. K. Cohen. Online changepoint detection via dynamic mode decomposition. *arXiv preprint arXiv:2405.15576*, 2024.

[69] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[70] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

[71] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: Data-driven modeling of complex systems*. SIAM, 2016.

[72] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.

[73] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[74] Pedro Lara-Benítez, Manuel Carranza-García, and José C Riquelme. An experimental review on deep learning architectures for time series forecasting. *International Journal of Neural Systems*, 31(03):2130001, 2021.

[75] Soledad Le Clainche and José M. Vega. Higher order dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 16(2):882–925, 2017.

[76] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data augmentation for time series classification using convolutional neural networks. In *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2016.

[77] Shuang Li, Yao Xie, Hanjun Dai, and Le Song. M-statistic for kernel change-point detection. *Advances in Neural Information Processing Systems*, 28, 2015.

[78] Wenxiang Li and K. L. Eddie Law. Deep learning models for time series forecasting: A review. *IEEE Access*, 12:92306–92327, 2024.

[79] Yue Li, Hongtao Cao, Zhongmei Li, Wenli Du, and Weifeng Shen. A flexible multi-step prediction architecture for process variable monitoring in chemical intelligent manufacturing. *Chemical Engineering Science*, 316:121943, 05 2025.

[80] Jaime Liew, Tuhfe Göçmen, Wai Hou Lio, and Gunner Chr. Larsen. Streaming dynamic mode decomposition for short-term forecasting in wind farms. *Wind Energy*, 25(4):719–734, 2022.

[81] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[82] Soon Hoe Lim, N Benjamin Erichson, Liam Hodgkinson, and Michael W Mahoney. Noisy recurrent neural networks. *Advances in Neural Information Processing Systems*, 34:5124–5137, 2021.

[83] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[84] Tianhong Liu, Haikun Wei, Sixing Liu, and Kanjian Zhang. Industrial time series forecasting based on improved gaussian process regression. *Soft Computing*, 24:15853–15869, 2020.

[85] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022.

[86] Ioannis E Livieris, Emmanuel Pintelas, and Panagiotis Pintelas. A CNN–LSTM model for gold price time-series forecasting. *Neural computing and applications*, 32:17351–17360, 2020.

[87] Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1):76–111, 2023.

[88] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pages 875–884. PMLR, 2020.

[89] Yao Ming, Shaozu Cao, Ruixiang Zhang, Zhen Li, Yuanzhe Chen, Yangqiu Song, and Huamin Qu. Understanding hidden memories of recurrent neural networks. In *2017 IEEE conference on visual analytics science and technology (VAST)*, pages 13–24. IEEE, 2017.

[90] Valentina Moskvina and Anatoly Zhigljavsky. An algorithm based on singular spectrum analysis for change-point detection. *Communications in Statistics-Simulation and Computation*, 32(2):319–352, 2003.

[91] Zuokun Ouyang, Philippe Ravier, and Meryem Jabloun. Stl decomposition of time series can benefit forecasting done by statistical methods but not by machine learning ones. *Engineering Proceedings*, 5(1):42, 2021.

[92] Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven CH Hoi. Learning fast and slow for online time series forecasting. *arXiv preprint arXiv:2202.11672*, 2022.

[93] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[94] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[95] David Rolnick. *Towards an integrated understanding of neural networks*. PhD thesis, Massachusetts Institute of Technology, 2018.

[96] Gordon J Ross, Niall M Adams, Dimitris K Tasoulis, and David J Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, 33(2):191–198, 2012.

[97] Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

[98] Christopher Salazar and Ashis G. Banerjee. A distance correlation-based approach to characterize the effectiveness of recurrent neural networks for time series forecasting. *Neurocomputing*, 629:129641, 2025.

[99] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[100] Rebecca Salles, Kele Belloze, Fabio Porto, Pedro H Gonzalez, and Eduardo Ogasawara. Nonstationary time series transformation methods: An experimental review. *Knowledge-Based Systems*, 164:274–291, 2019.

[101] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[102] Diya Sashidhar and J Nathan Kutz. Bagging, optimized dynamic mode decomposition for robust, stable forecasting with spatial and temporal uncertainty quantification. *Philosophical Transactions of the Royal Society A*, 380(2229):20210199, 2022.

[103] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[104] Peter J Schmid. Dynamic mode decomposition and its variants. *Annual Review of Fluid Mechanics*, 54(1):225–254, 2022.

[105] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[106] Kamran Shaukat, Talha Mahboob Alam, Suhuai Luo, Shakir Shabbir, Ibrahim A Hameed, Jiaming Li, Syed Konain Abbas, and Umair Javed. A review of time-series anomaly detection techniques: A step to future perspectives. In *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 1*, pages 865–877. Springer, 2021.

[107] Qiaomu Shen, Yanhong Wu, Yuzhe Jiang, Wei Zeng, KH Alexis, Anna Vianova, and Huamin Qu. Visual interpretation of recurrent neural network on multi-dimensional time-series forecast. In *2020 IEEE Pacific visualization symposium (PacificVis)*, pages 61–70. IEEE, 2020.

[108] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[109] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):667–676, 2017.

[110] Gábor J Székely and Maria L Rizzo. The distance correlation t-test of independence in high dimension. *Journal of Multivariate Analysis*, 117:193–213, 2013.

[111] Gábor J Székely and Maria L Rizzo. Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference*, 143(8):1249–1272, 2013.

[112] Gábor J Székely, Maria L Rizzo, and Nail K Bakirov. Measuring and testing dependence by correlation of distances. *The annals of statistics*, 35(6):2769–2794, 2007.

[113] C. Tang, L. Xu, B. Yang, Y. Tang, and D. Zhao. GRU-based interpretable multivariate time series anomaly detection in industrial control system. *Computers & Security*, 127:103094, 2023.

[114] Matus Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR, 2016.

[115] Santosh Tirunagari, Samaneh Kouchaki, Norman Poh, Miroslaw Bober, and David Windridge. Dynamic mode decomposition for univariate time series: analysing trends and forecasting. 2017.

[116] Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. NoPeek: Information leakage reduction to share activations in distributed deep learning. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 933–942. IEEE, 2020.

[117] Eyob Betru Wegayehu and Fiseha Behulu Muluneh. Short-term daily univariate streamflow forecasting using deep learning models. *Advances in Meteorology*, 2022, 2022.

[118] Qingsong Wen, Weiqi Chen, Liang Sun, Zhang Zhang, Liang Wang, Rong Jin, Tieniu Tan, et al. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. *Advances in Neural Information Processing Systems*, 36:69949–69980, 2023.

[119] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[120] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015.

[121] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

[122] Liyan Xie, George V Moustakides, and Yao Xie. Window-limited cusum for sequential change detection. *IEEE Transactions on Information Theory*, 69(9):5990–6005, 2023.

[123] Hongju Yan and Hongbing Ouyang. Financial time series prediction based on deep learning. *Wireless Personal Communications*, 102(2):683–700, 2018.

[124] Zhangjing Yang, Weiwu Yan, Xiaolin Huang, and Lin Mei. Adaptive temporal-frequency network for time-series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(4):1576–1587, 2020.

[125] Kashif Yousuf and Yang Feng. Targeting predictors via partial distance correlation with applications to financial forecasting. *Journal of Business & Economic Statistics*, 40(3):1007–1019, 2022.

[126] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. TS2Vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8980–8987, 2022.

[127] KD Zamba and Douglas M Hawkins. A multivariate change-point model for statistical process control. *Technometrics*, 48(4):539–549, 2006.

[128] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[129] Hao Zhang, Clarence W Rowley, Eric A Deem, and Louis N Cattafesta. Online dynamic mode decomposition for time-varying systems. *SIAM Journal on Applied Dynamical Systems*, 18(3):1586–1609, 2019.

[130] Lingyu Zhang, Wenjie Bian, Wenyi Qu, Liheng Tuo, and Yunhai Wang. Time series forecast of sales volume based on xgboost. In *Journal of Physics: Conference Series*, volume 1873, page 012067. IOP Publishing, 2021.

[131] Saizheng Zhang, Yuhuai Wu, Tong Che, Zhouhan Lin, Roland Memisevic, Russ R Salakhutdinov, and Yoshua Bengio. Architectural complexity measures of recurrent neural networks. *Advances in neural information processing systems*, 29, 2016.

[132] Xueli Zhang, Cankun Zhong, Jianjun Zhang, Ting Wang, and Wing W.Y. Ng. Robust recurrent neural networks for time series forecasting. *Neurocomputing*, 526:143–157, 2023.

[133] Xingjian Zhen, Zihang Meng, Rudrasis Chakraborty, and Vikas Singh. On the versatile uses of partial distance correlation in deep learning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*, pages 327–346. Springer, 2022.

[134] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[135] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

[136] Zhou Zhou. Measuring nonlinear dependence in time-series, a distance correlation approach. *Journal of Time Series Analysis*, 33(3):438–457, 2012.

Appendix A

ADDITIONAL EXPERIMENTAL RESULTS FOR CHAPTER 3

The following tables report the time series forecasting results using RNN models that are slightly different to the model used in Table 3.1. These tables expand the RNN architecture to a larger number of input hidden units of 128 (Table A.1), a learning rate of 0.001 (Table A.2), and a dropout rate of 0.2 in the final layer (Table A.3). The purpose of these additional tables is to show consistency in the results from Section 3.2.2, even with a different set of hyperparameters. For example, each table shows that for increasing lag structures, MSE and MAPE scores grow, and the activation memory loss increases (shown by the change in % column). It also shows that despite the changes to the hyperparameters, large MA lag structures and GARCH processes are not adequately modeled by an RNN. Given the agreement of these results, we contend that other changes to the RNN hyperparameters will lead to the same general outcomes.

Table A.1: MSE, MAPE, and distance correlation results for all the experiments with 128 input hidden units

Time Series	MSE	MAPE	$\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})$ (max)	$\hat{R}(\mathbf{A}_T^{(P)}, \mathbf{Y})$	Change (%)
AR(1)	0.024 ± 0.005	0.058 ± 0.039	0.952	0.952	0
AR(5)	0.070 ± 0.029	0.105 ± 0.078	0.913	0.449	45
AR(10)	0.404 ± 0.197	0.275 ± 0.192	0.947	0.402	58
AR(20)	1.112 ± 0.333	0.809 ± 1.275	0.966	0.378	61
MA(1)	0.612 ± 0.034	0.303 ± 0.050	0.434	0.434	0
MA(5)	0.772 ± 0.050	0.434 ± 0.178	0.431	0.134	69
MA(10)	0.920 ± 0.059	0.424 ± 0.159	0.431	0.105	76
MA(20)	1.080 ± 0.064	0.572 ± 0.309	0.428	0.096	78
ARMA(1,1)	0.008 ± 0.002	0.035 ± 0.019	0.947	0.947	0
ARMA(1,10)	0.012 ± 0.003	0.040 ± 0.025	0.966	0.966	0
ARMA(10,1)	0.235 ± 0.137	0.251 ± 0.344	0.946	0.482	49
GARCH(2,2)	0.85 ± 0.674	0.651 ± 0.551	0.272	0.267	2
GARCH(4,4)	1.186 ± 1.081	1.406 ± 4.608	0.218	0.208	5
ETTh1, OT	0.038 ± 0.019	0.603 ± 0.650	0.913	0.913	0
Solar-Energy	0.010 ± 0.007	1.083 ± 5.48	0.976	0.976	0
NASDAQ	1.558 ± 0.460	8.953 ± 8.554	0.120	0.112	7

Notes: Results for all time series experiments with 128 units for the hidden units of the RNN. Mean squared errors are calculated with standard deviations for 50 simulation runs. $\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})$ (max) represents the max distance correlation for any layer number t. $\hat{R}(\mathbf{A}_T^{(P)}, \mathbf{Y})$ is the distance correlation value at layer number 20. The percent change is calculated as $(\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})_{max} - \hat{R}(\mathbf{A}_T^{(P)}, \mathbf{Y}))/\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})_{max}$, which measures how much information is lost during RNN training.

Table A.2: MSE, MAPE, and distance correlation results for all the experiments with a learning rate of 0.001

Time Series	MSE	MAPE	$\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})$ (max)	$\hat{R}(\mathbf{A}_T^{(P)}, \mathbf{Y})$	Change (%)
AR(1)	0.028 ± 0.007	0.048 ± 0.012	0.964	0.964	0
AR(5)	0.070 ± 0.036	0.099 ± 0.083	0.917	0.416	55
AR(10)	0.438 ± 0.184	0.272 ± 0.282	0.952	0.426	55
AR(20)	1.111 ± 0.250	1.259 ± 2.677	0.965	0.346	64
MA(1)	0.826 ± 0.060	0.367 ± 0.199	0.472	0.472	0
MA(5)	1.011 ± 0.078	0.399 ± 0.144	0.450	0.170	62
MA(10)	1.213 ± 0.092	0.468 ± 0.179	0.439	0.115	74
MA(20)	1.534 ± 0.130	0.569 ± 0.222	0.431	0.106	75
ARMA(1,1)	0.008 ± 0.002	0.030 ± 0.018	0.967	0.967	0
ARMA(1,10)	0.012 ± 0.004	0.037 ± 0.021	0.972	0.972	0
ARMA(10,1)	0.193 ± 0.097	0.267 ± 0.359	0.945	0.508	46
GARCH(2,2)	1.201 ± 1.034	1.453 ± 4.901	0.279	0.270	27
GARCH(4,4)	1.235 ± 1.201	0.581 ± 0.471	0.209	0.068	20
ETTh1, OT	0.039 ± 0.021	0.523 ± 0.537	0.942	0.942	0
Solar-Energy	0.010 ± 0.006	1.083 ± 2.29	0.977	0.977	0
NASDAQ	1.592 ± 0.497	1.132 ± 0.907	0.112	0.103	8

Notes: Results for all time series experiments using an RNN learning rate of 0.001. The experimental setup as well as the overall outcomes are the same as Table 3.1.

Table A.3: MSE, MAPE, and distance correlation results for all the experiments with a dropout rate of 0.2

Time Series	MSE	MAPE	$\hat{R}(\mathbf{A}_t^{(P)}, \mathbf{Y})$ (max)	$\hat{R}(\mathbf{A}_T^{(P)}, \mathbf{Y})$	Change (%)
AR(1)	0.030 ± 0.009	0.060 ± 0.030	0.923	0.923	0
AR(5)	0.174 ± 0.160	0.184 ± 0.176	0.911	0.457	50
AR(10)	1.210 ± 0.470	0.610 ± 0.864	0.948	0.423	55
AR(20)	1.292 ± 0.291	0.774 ± 1.080	0.967	0.368	62
MA(1)	0.605 ± 0.037	0.378 ± 0.190	0.430	0.430	0
MA(5)	0.811 ± 0.047	0.437 ± 0.381	0.436	0.127	71
MA(10)	1.027 ± 0.056	0.492 ± 0.231	0.425	0.099	77
MA(20)	1.040 ± 0.056	0.550 ± 0.313	0.433	0.094	78
ARMA(1,1)	0.015 ± 0.007	0.042 ± 0.021	0.923	0.923	0
ARMA(1,10)	0.014 ± 0.005	0.044 ± 0.017	0.955	0.955	0
ARMA(10,1)	0.684 ± 0.271	0.387 ± 0.363	0.948	0.499	47
GARCH(2,2)	1.128 ± 0.835	0.701 ± 1.248	0.280	0.276	2
GARCH(4,4)	1.226 ± 1.252	1.666 ± 4.024	0.226	0.213	6
ETTh1, OT	0.041 ± 0.019	0.842 ± 2.136	0.940	0.940	0
Solar-Energy	0.029 ± 0.031	0.663 ± 1.47	0.985	0.985	0
NASDAQ	1.562 ± 0.487	2.514 ± 4.312	0.096	0.088	9

Notes: Results for all time series experiments with a dropout rate of 0.2. The experimental setup is similar to previous experiments.

Appendix B

ADDITIONAL EXPERIMENTAL RESULTS FOR CHAPTER 4

B.1 Introduction

This appendix serves as supplementary material for the paper titled "Enhancing Non-stationary Time Series Forecasting with Incremental Kernel Dynamic Mode Decomposition". Herein, we present additional results exploring hyperparameters efffects on our proposed methods: Incremental Kernel Dynamic Mode Decomposition (IKDMD) and Randomized Kernel Dynamic Mode Decomposition (RKDMD). We compare these methods against benchmark methods labeled DLinear, Informer, and Non-Stationary. The datasets featured in this document include Weather, Exchange Returns, ILI, Electricity, and ETTh1. For a detailed description of the methods, models, and datasets, please refer to the main paper.

B.2 Additional Incremental Results

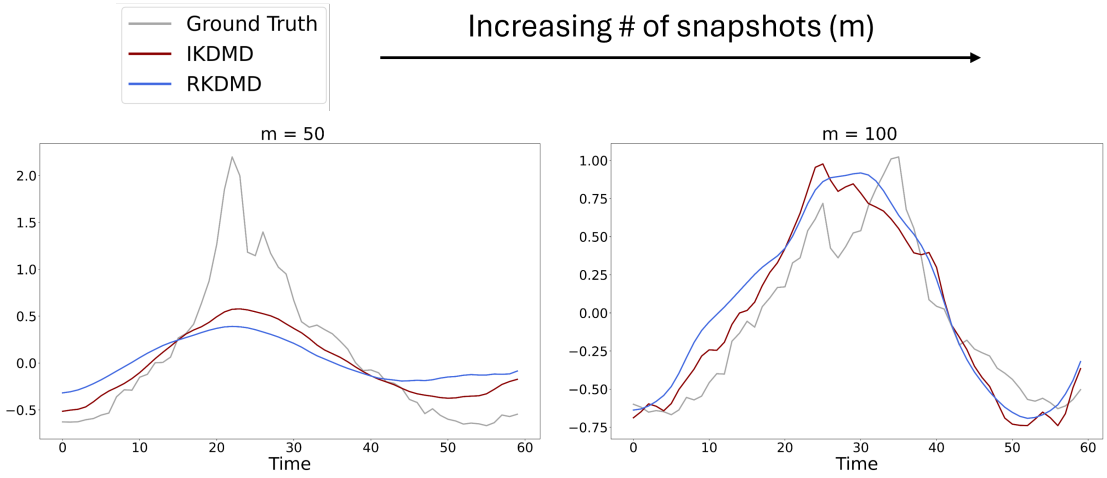
In this section, we explore the incremental improvements achieved when IKDMD and RKDMD are updated incrementally with the number of snapshots m. We also examine the effects associated with the truncation parameter r. Here, r is a value between 0 and 1, representing a percentage of m.

We begin with Figure B.1, which presents the incremental plots for the ILI dataset analyzed by the value of r. In this example, the choice of r does not lead to significant changes in the performance of the plot, except for a smoother fit to the ground truth in Figure B.1, as compared to the more jagged nature observed in Figure B.1c. Additionally, there is a noticeable improvement in prediction as the number of snapshots increases from $m = 50$ to $m = 100$.

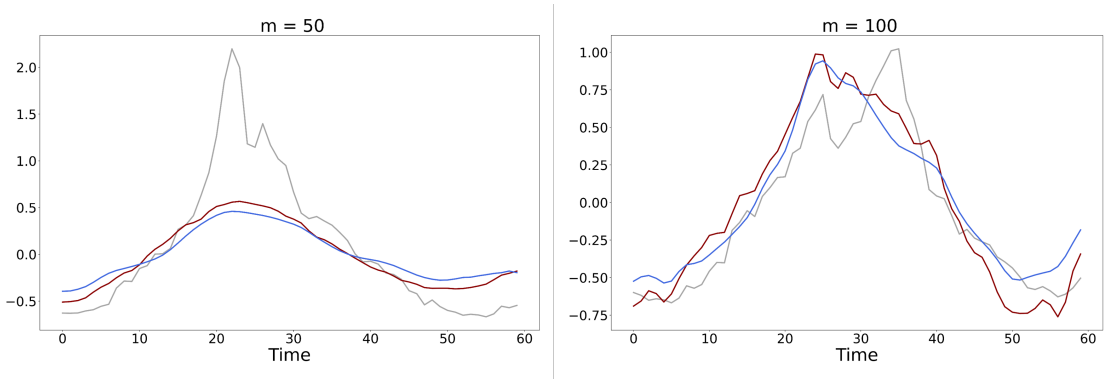
Generally, we observe that lower values of r tend to forecast the ground truth with

smoother curves that capture the mean of the data, whereas, larger values of r tend to fit the noise and fluctuations of the data more closely. As expected, greater the number of snapshots m available to the DMD models, more pronounced the overall improvement in forecasting accuracy.

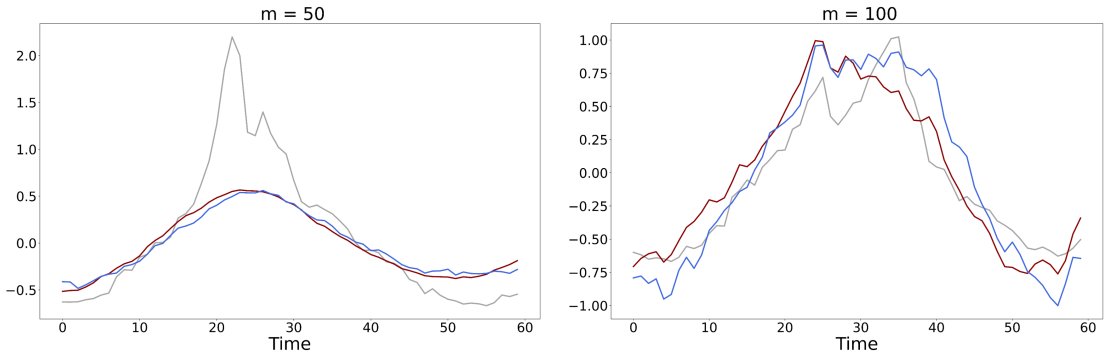
The effect of r is also evident in Figure B.2, where lower values of r produce smoother forecasts, and larger values lead to forecasts that more closely fit the ground truth. Additionally, for the ETTh1 dataset, more snapshots are needed to adequately forecast future values. These trends hold true for the Exchange Returns and Weather datasets, as demonstrated in Figures B.3 and B.4, respectively. For the Electricity dataset, it is important to note that we could not reduce the truncation beyond $r = 0.95$. In this case, the Electricity data does not appear to exhibit a low-rank dynamical structure, requiring almost all the snapshots for accurate forecasting.



(a)

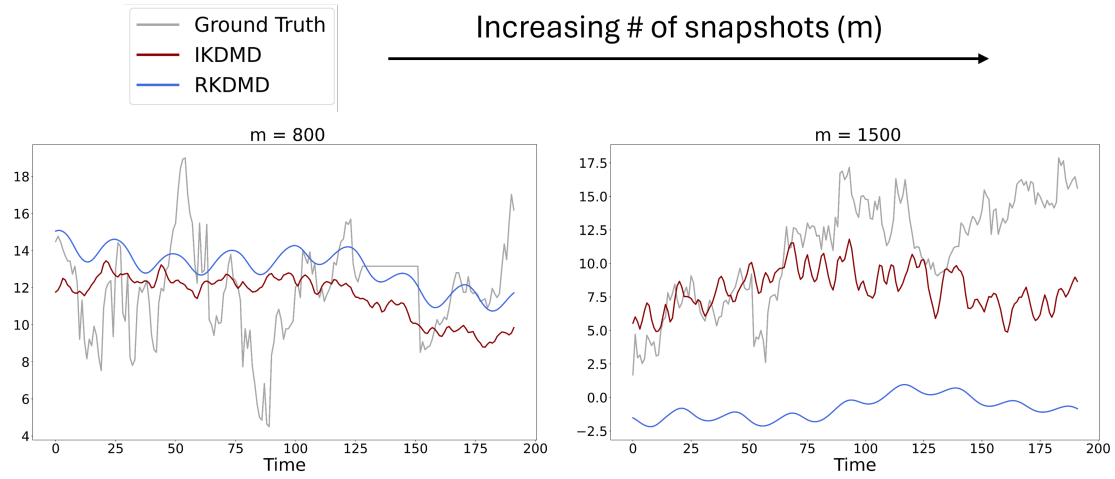


(b)

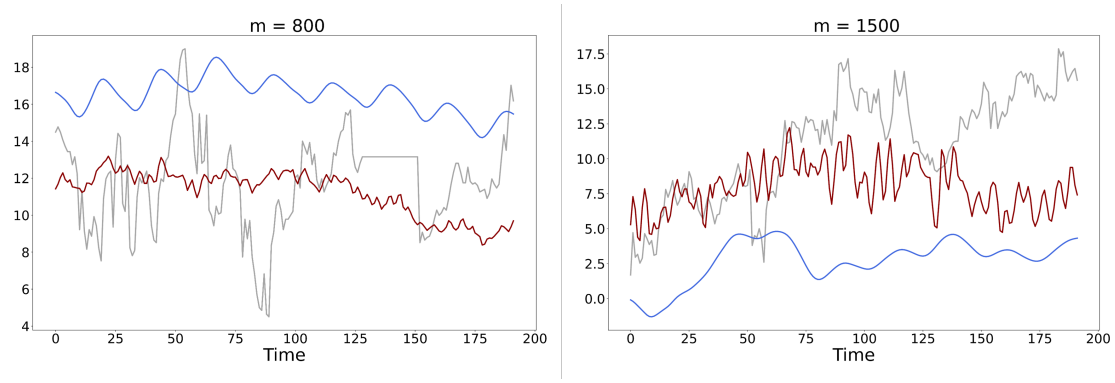


(c)

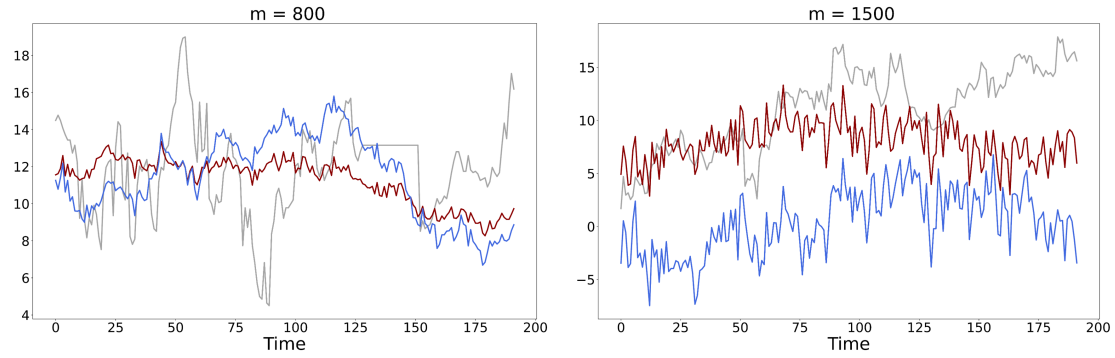
Figure B.1: Incremental forecasting performance of IKDMD and RKDMD on the ILI dataset based on the number of snapshots m and truncation parameter r. The truncation parameter is based on the percentage of snapshots m and is shown in sub-figures (a) r = 0.5 (b) r = 0.7 and (c) r = 1.



(a)

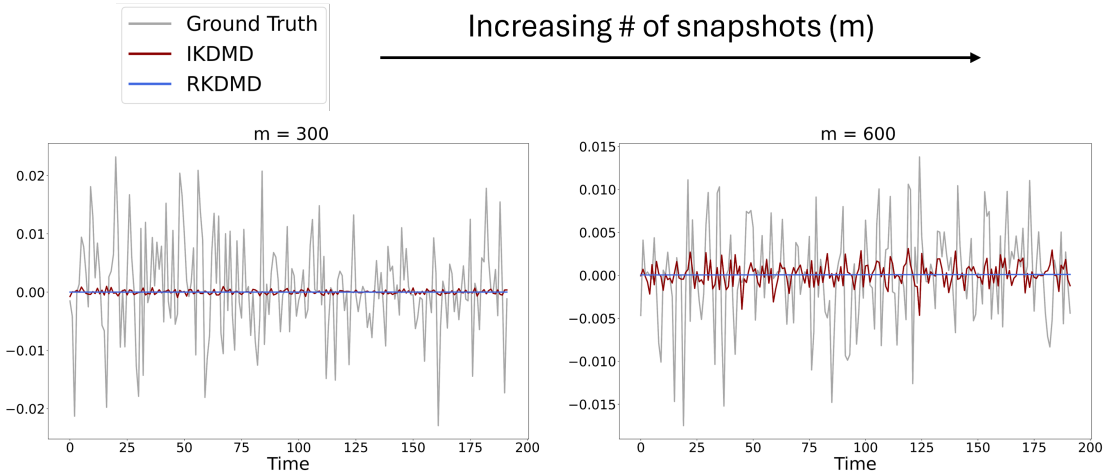


(b)

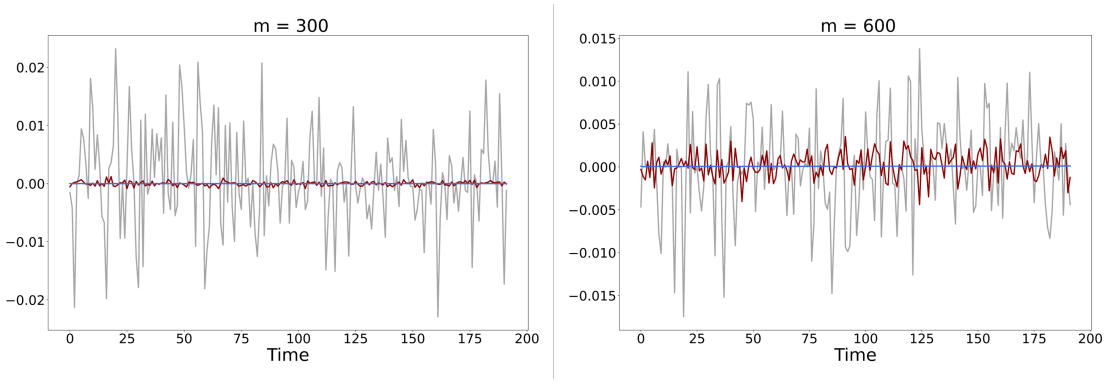


(c)

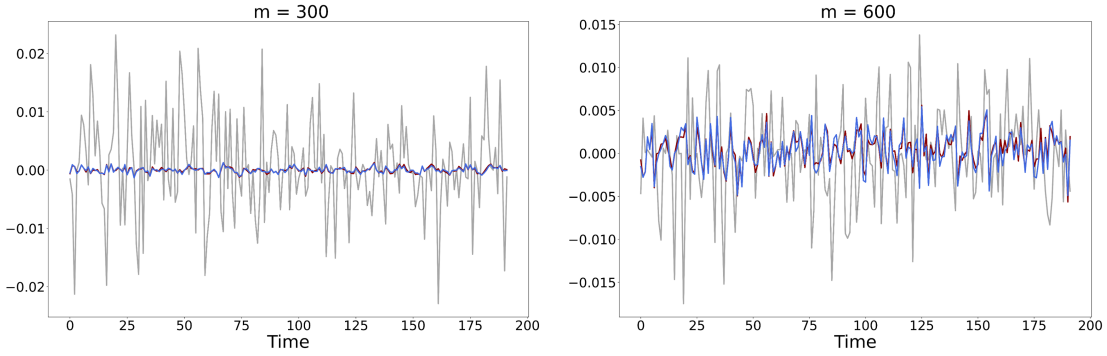
Figure B.2: Incremental forecasting performance of IKDMD and RKDMD on the ETTh1 dataset based on number of snapshots m and truncation parameter r. The truncation parameter is based on the percentage of snapshots m and is shown in sub-figures (a) r = 0.5 (b) r = 0.7 and (c) r = 1.



(a)

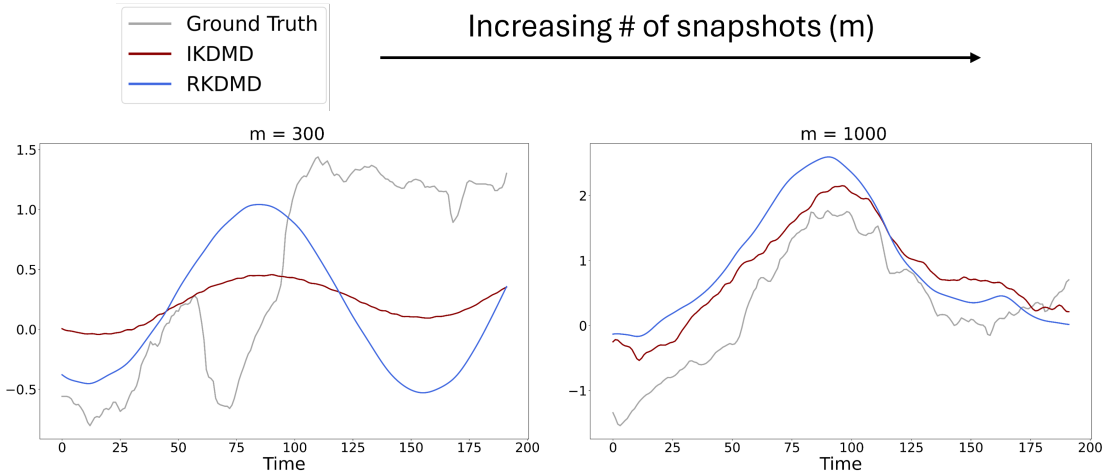


(b)

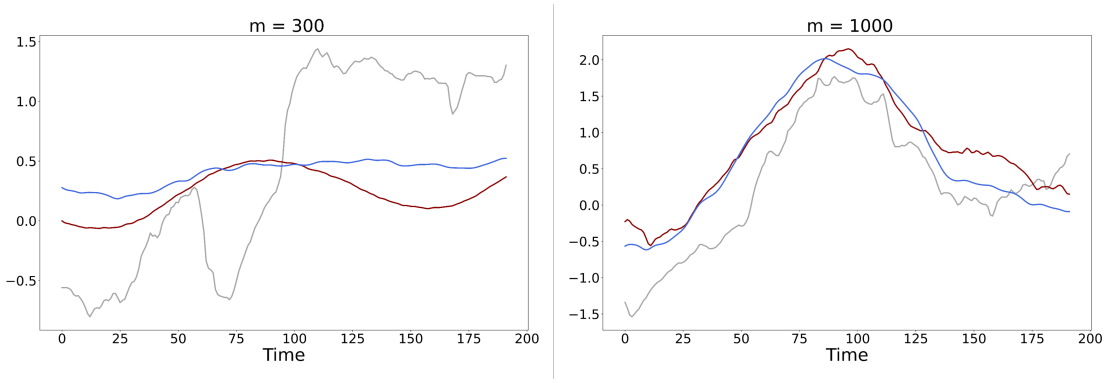


(c)

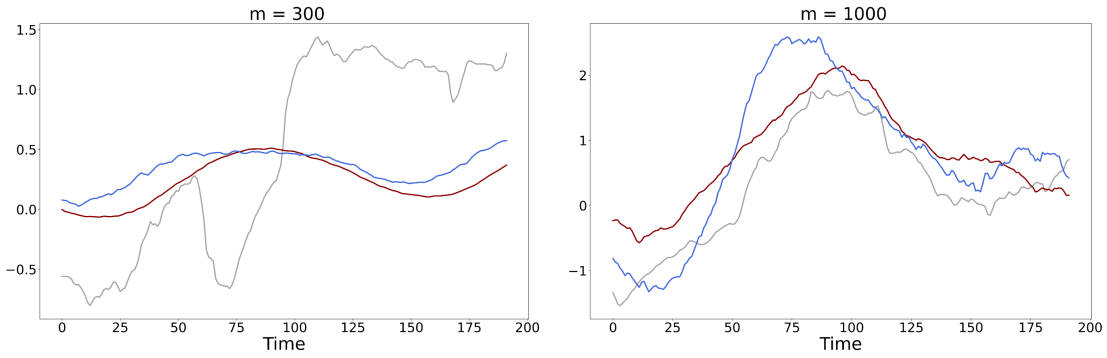
Figure B.3: Incremental forecasting performance of IKDMD and RKDMD on the exchange returns dataset based on number of snapshots m and truncation parameter r. The truncation parameter is based on the percentage of snapshots m and is shown in sub-figures (a) r = 0.5 (b) r = 0.7 and (c) r = 1.



(a)

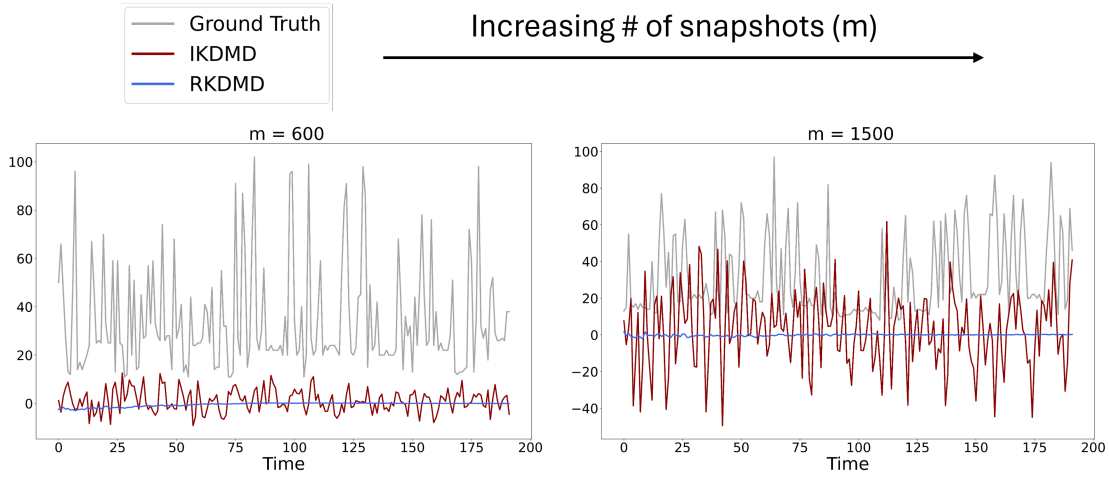


(b)

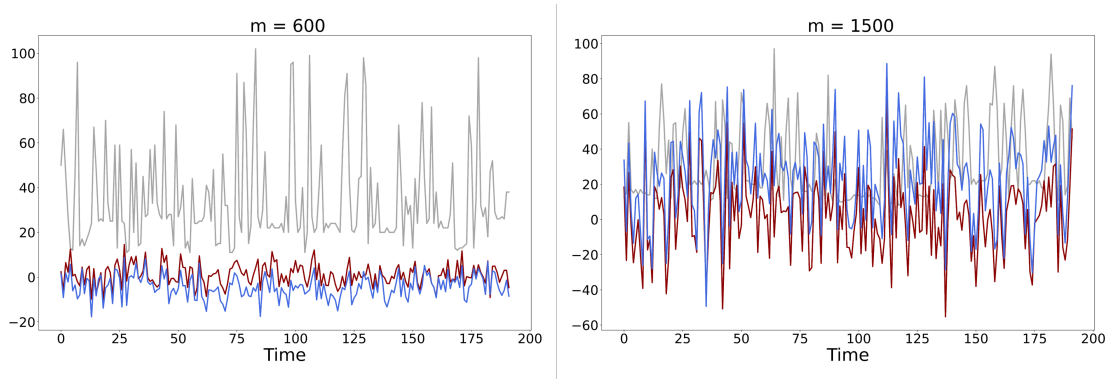


(c)

Figure B.4: Incremental forecasting performance of IKDMD and RKDMD on the weather dataset based on number of snapshots m and truncation parameter r. The truncation parameter is based on the percentage of snapshots m and is shown in sub-figures (a) r = 0.5 (b) r = 0.7 and (c) r = 1.



(a)



(b)

Figure B.5: Incremental forecasting performance of IKDMD and RKDMD on the Electricity dataset based on number of snapshots m and truncation parameter r. The truncation parameter is based on the percentage of snapshots m and is shown in sub-figures (a) r = 0.95 (b) r = 1.0. The forecasting accuracy of both the models improves as new data (snapshots, m) are incrementally introduced into the system.

Appendix C

ADDITIONAL EXPERIMENTAL RESULTS FOR CHAPTER 4

C.1 Sensitivity of Hyperparameters

To evaluate the hyperparameter robustness of WORK-DMD, we conducted a systematic sensitivity analysis of four critical hyperparameters: window size, Random Fourier Feature (RFF) dimension, kernel bandwidth parameter (γ), and matrix rank. We varied each parameter independently while maintaining baseline values (window size = 60, RFF dimension = 1024, $\gamma = 3 \times 10^{-5}$, rank = 128) for others. Each configuration was evaluated using training data of the forecasts on the Traffic dataset to ensure statistical reliability.

The analysis reveals distinct sensitivity patterns across parameters. Window size exhibits optimal performance around 60-90 time steps, where smaller windows provide insufficient temporal context while larger windows introduce computational overhead without accuracy gains (Figure C.1a). RFF dimension demonstrates diminishing returns beyond 1024 features, with performance plateauing due to sufficient kernel approximation quality (Figure C.1b). The kernel bandwidth parameter γ shows the highest sensitivity, with optimal values around 10^{-5} to 10^{-4}; values outside this range cause either over-smoothing or noise amplification (Figure C.2a). Matrix rank exhibits moderate sensitivity with stable performance between 20-40, where lower ranks lack model expressiveness and higher ranks introduce overfitting in the streaming setting (Figure C.2b).

These results demonstrate that WORK-DMD exhibits reasonable robustness within practical hyperparameter ranges, with the kernel bandwidth requiring the most careful tuning. The identified optimal ranges facilitate deployment across different domains: window size should match the temporal correlation structure of the application, RFF dimensions between 512-2048 provide good accuracy-efficiency balance, γ should be tuned within $[10^{-6}, 10^{-4}]$ us-

ing validation data, and rank values between 20-40 offer sufficient model complexity without overfitting.

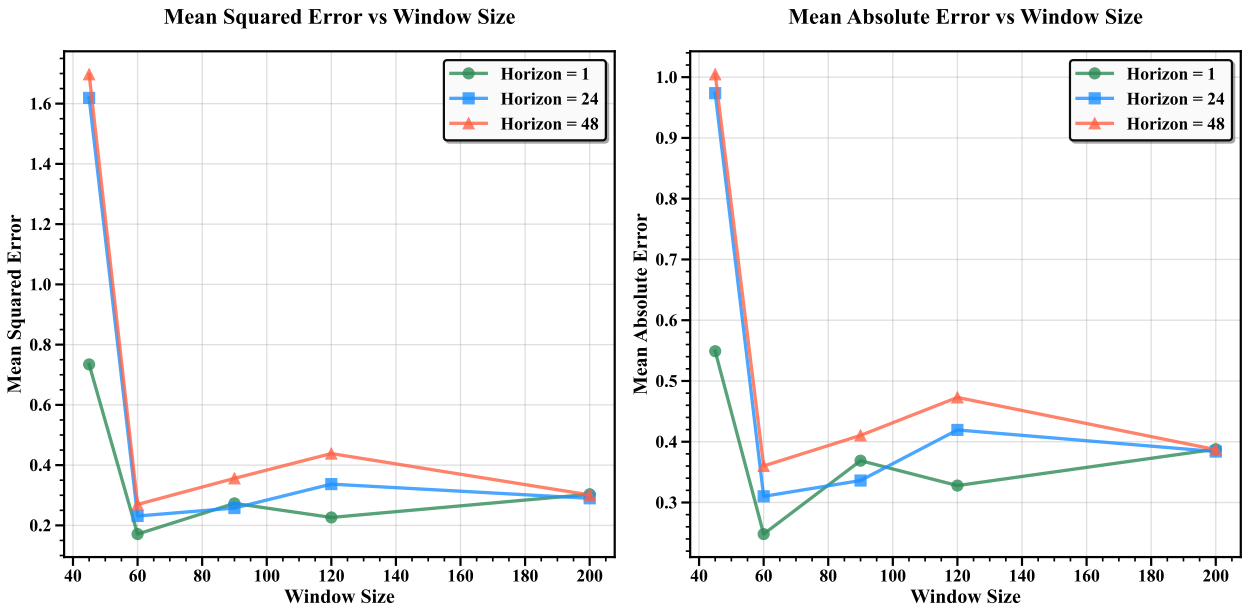
C.2 Forecasting plots for Traffic dataset

When comparing forecasting plots against the latest methods, we found that OneNet did not provide results for the traffic dataset. We therefore use FSNet as a comparison baseline. Figure C.3 shows the results for the first three channels of traffic monitoring. For prediction horizon $H = 24$, both models demonstrate comparable performance in capturing the ground truth patterns.

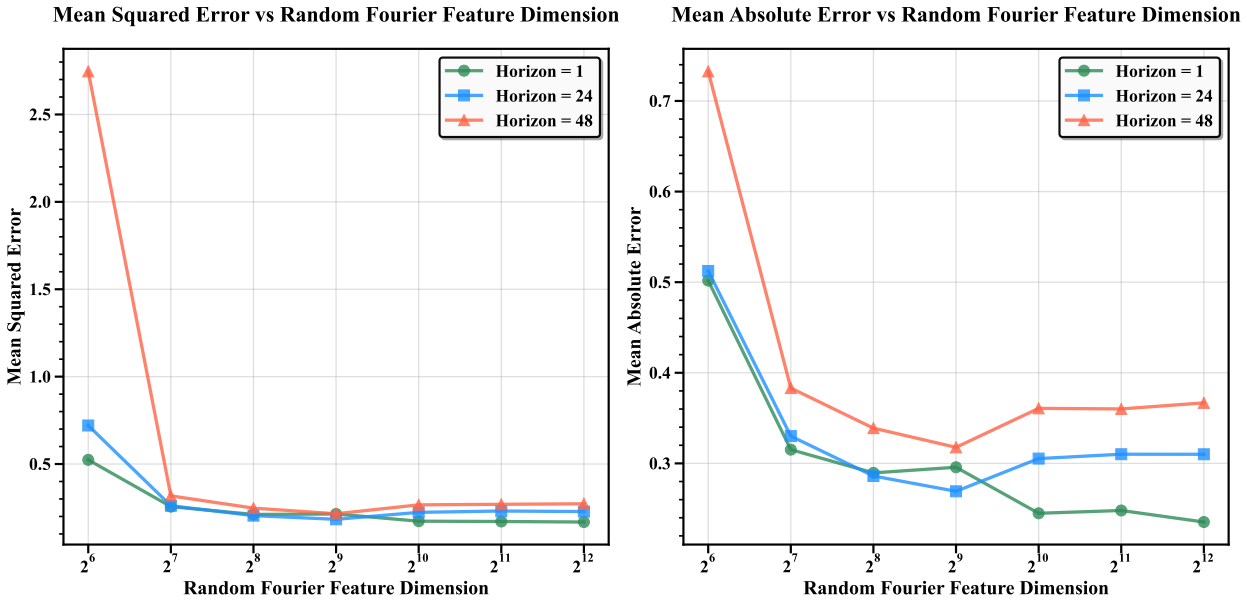
C.3 Additional Error plots

In addition to the cumulative error plots that were shown in section 5.3 for the ETTh2 and WTH forecasting results, we present the cumulative results for the ETTm1 and Traffic datasets here. It should be noted, since there were no available results for OneNet with the Traffic dataset, FSNet was the next closest comparison.

The ETTm1 results demonstrate WORK-DMD's exceptional performance for short-term forecasting, with notably superior accuracy in the 1-step ahead prediction task where the cumulative error remains consistently lower than OneNet throughout the evaluation period. For longer horizons (48-step ahead), WORK-DMD maintains competitive performance with OneNet, showing stable error accumulation patterns that indicate robust forecasting capability across different prediction timeframes. The Traffic dataset comparison with FSNet reveals consistent relative stability between the two methods, with both algorithms demonstrating similar error accumulation rates across the 1-step and 24-step ahead forecasting tasks, suggesting that WORK-DMD achieves comparable performance to established baselines even on challenging high-dimensional traffic prediction problems.

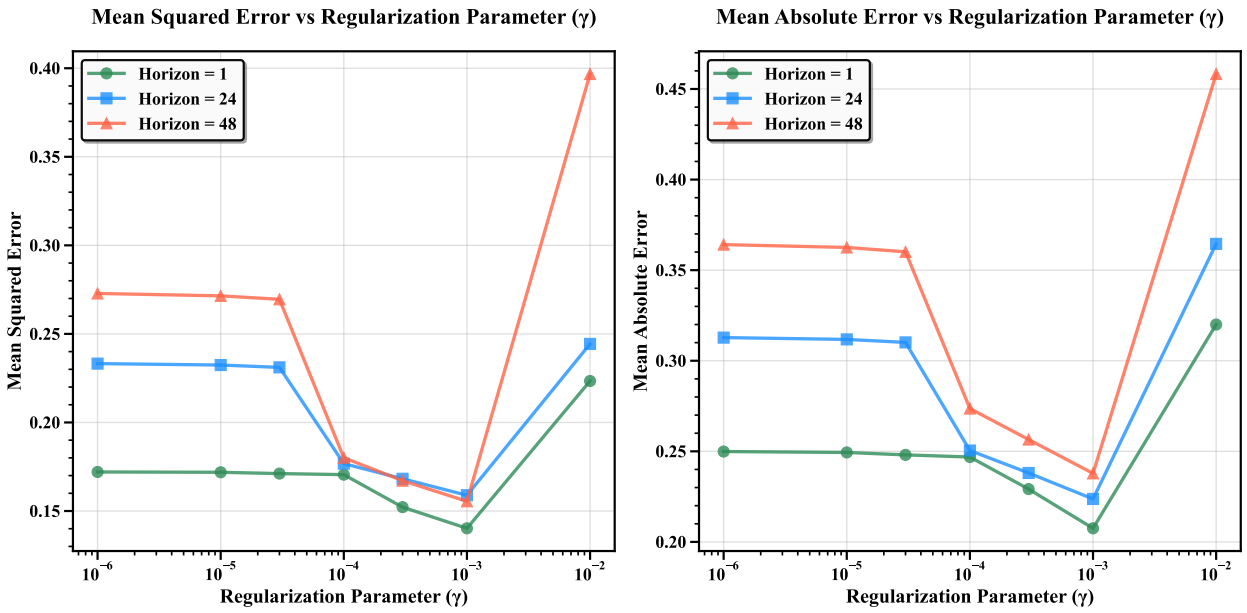


(a) Window Size Sensitivity

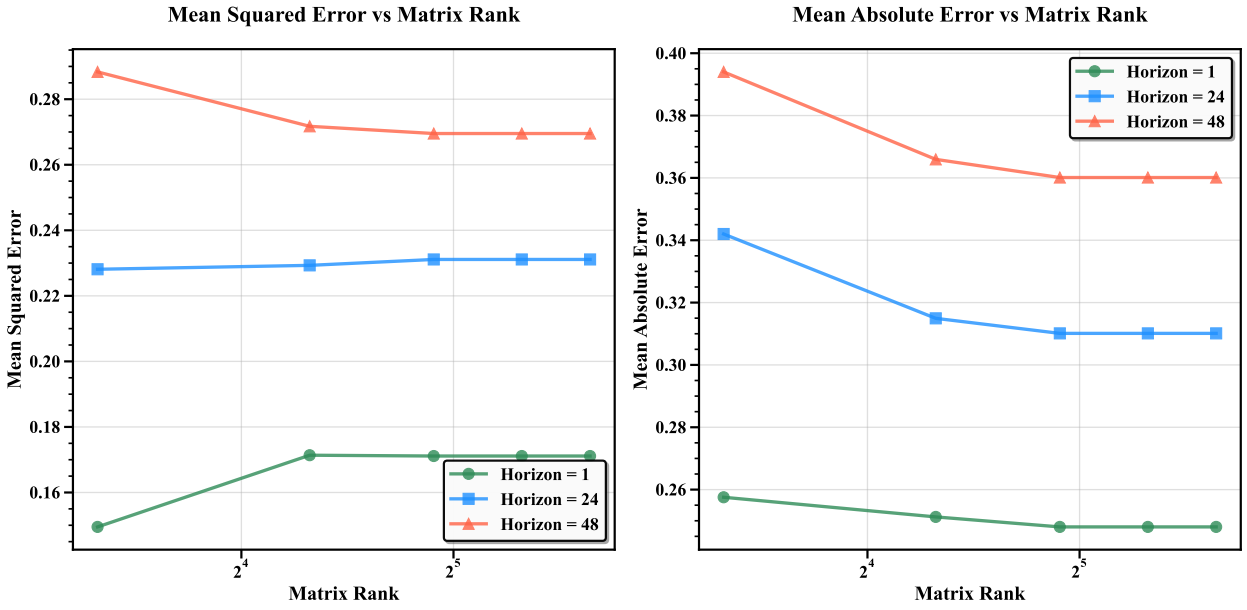


(b) RFF Dimension Sensitivity

Figure C.1: Window size and RFF dimension sensitivity analysis for WORK-DMD showing MSE and MAE performance across different forecasting horizons (H=1, 24, 48). (a) Window size analysis reveals optimal performance around 60-90 time steps, balancing temporal context with computational efficiency. Smaller windows provide insufficient historical information while larger windows introduce computational overhead without proportional gains.

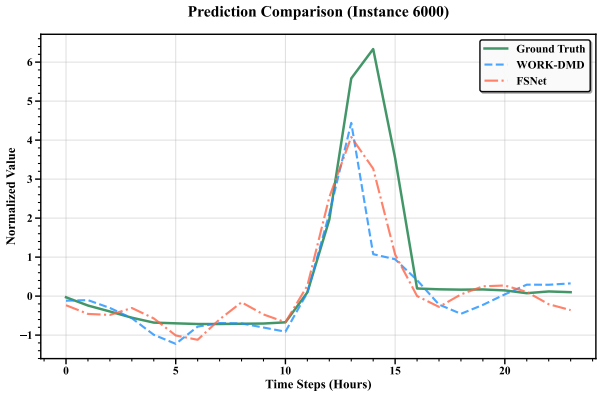


(a) Gamma Parameter Sensitivity

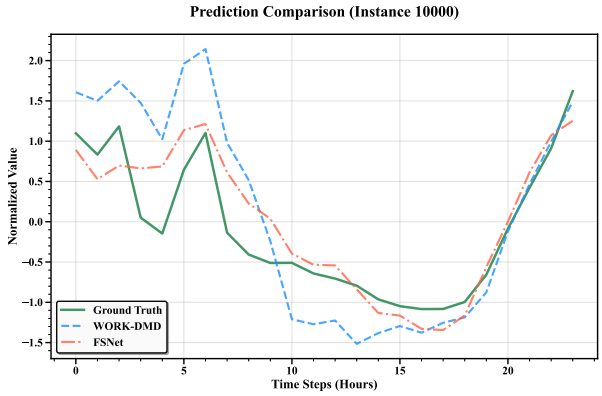


(b) Rank Sensitivity

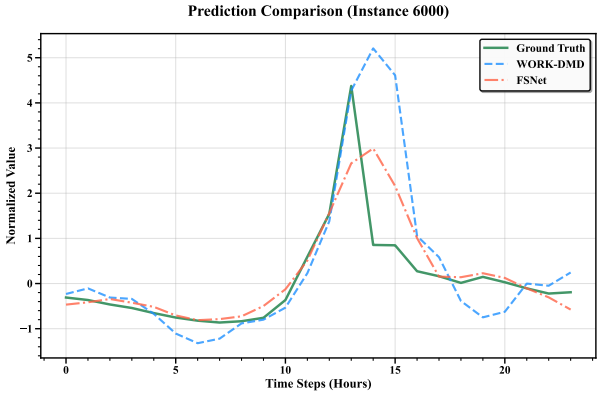
Figure C.2: Gamma parameter and rank sensitivity analysis for WORK-DMD. (c) Gamma parameter exhibits optimal range around 10^{-5} to 10^{-4}, with performance degradation at extreme values due to under/over-smoothing effects. This parameter shows the highest sensitivity among all tested hyperparameters. (d) Rank truncation demonstrates stable performance between 20-40, with degradation below 20 due to insufficient model complexity



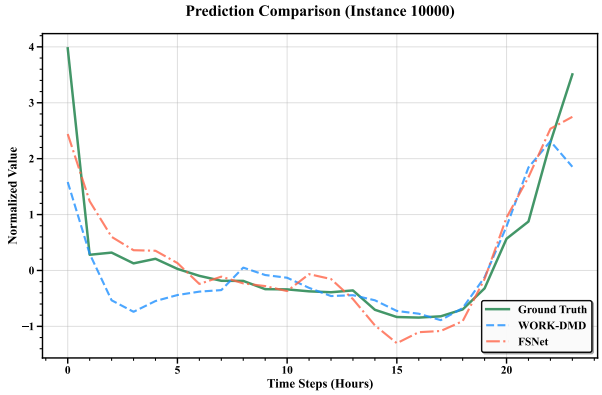
(a) Channel 0 - Instance 6000



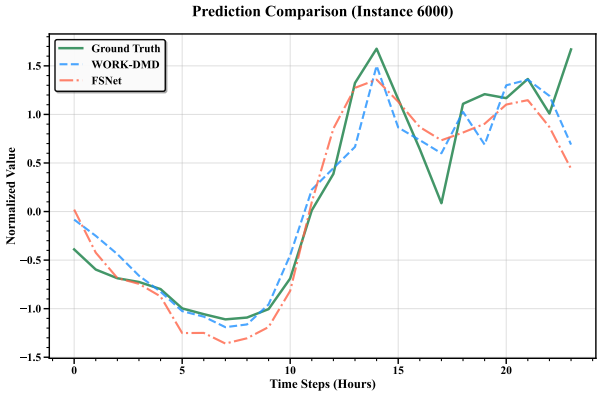
(b) Channel 0 - Instance 10000



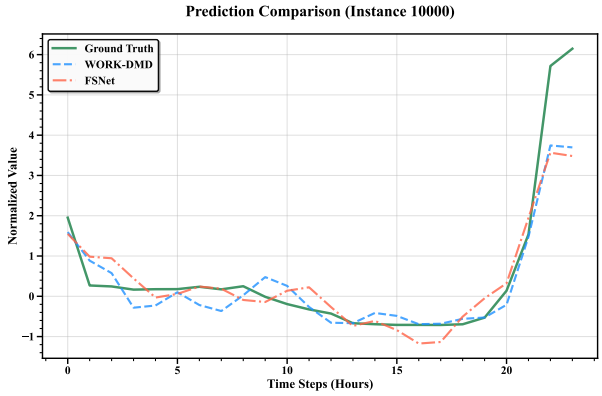
(c) Channel 1 - Instance 6000



(d) Channel 1 - Instance 10000

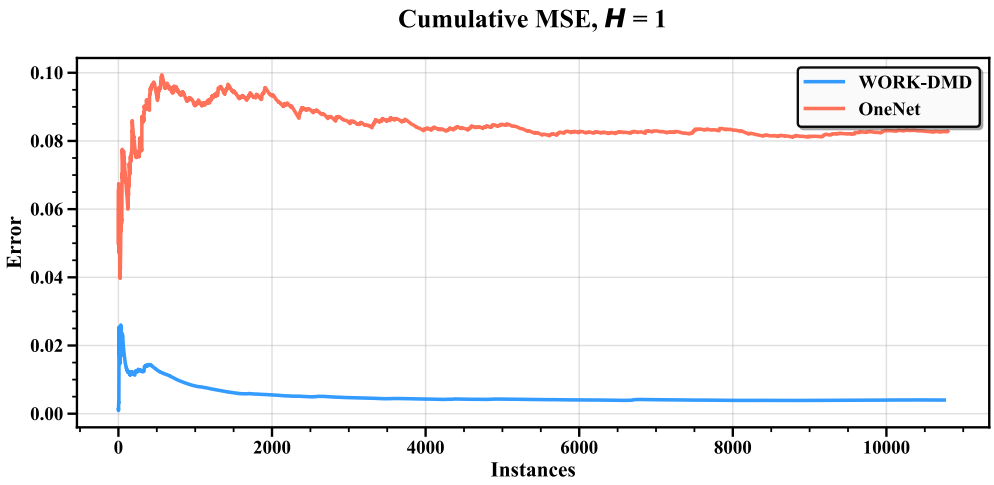


(e) Channel 2 - Instance 6000

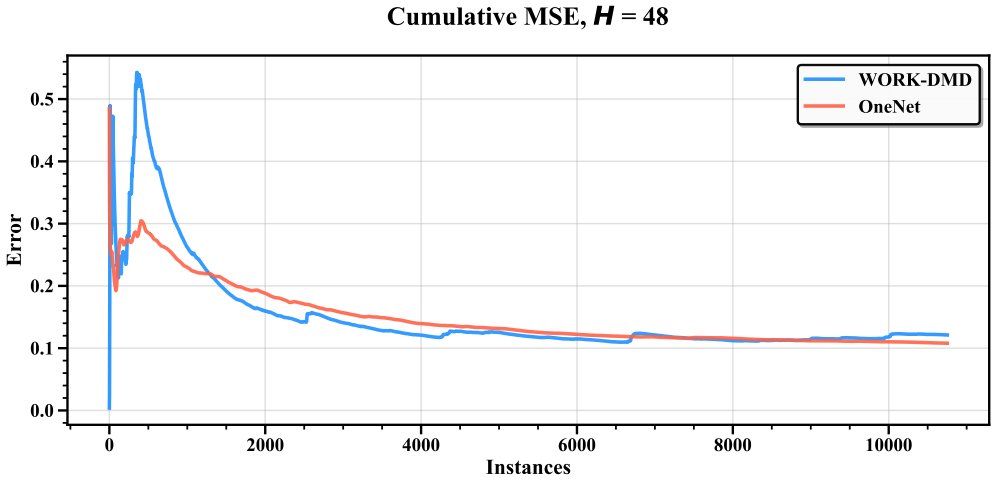


(f) Channel 2 - Instance 10000

Figure C.3: Time series forecasting comparison on Traffic dataset for three monitoring channels at two temporal instances. The figure shows 24-step ahead predictions ($H = 24$) comparing WORK-DMD and FSNet against ground truth. WORK-DMD achieves comparable performance to FSNet across all channels (only first three channels shown for clarity), demon-

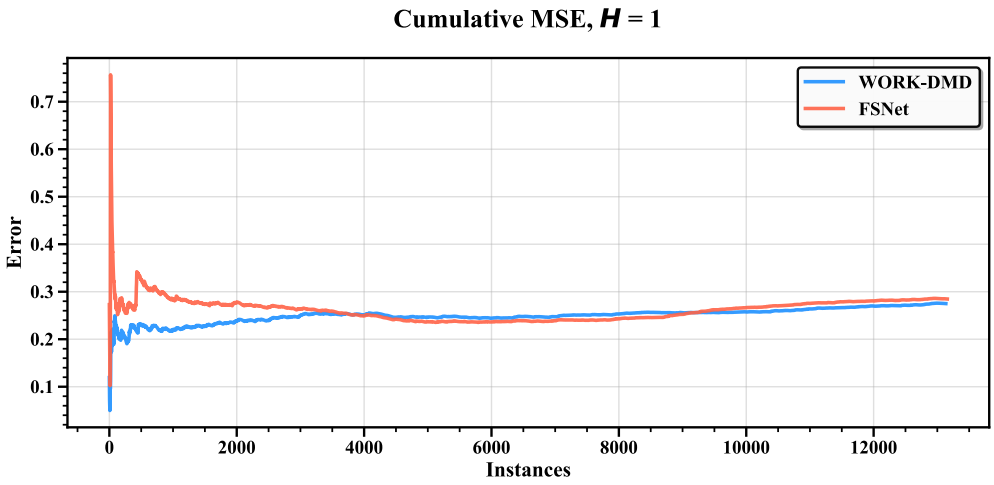


(a) 1-step ahead

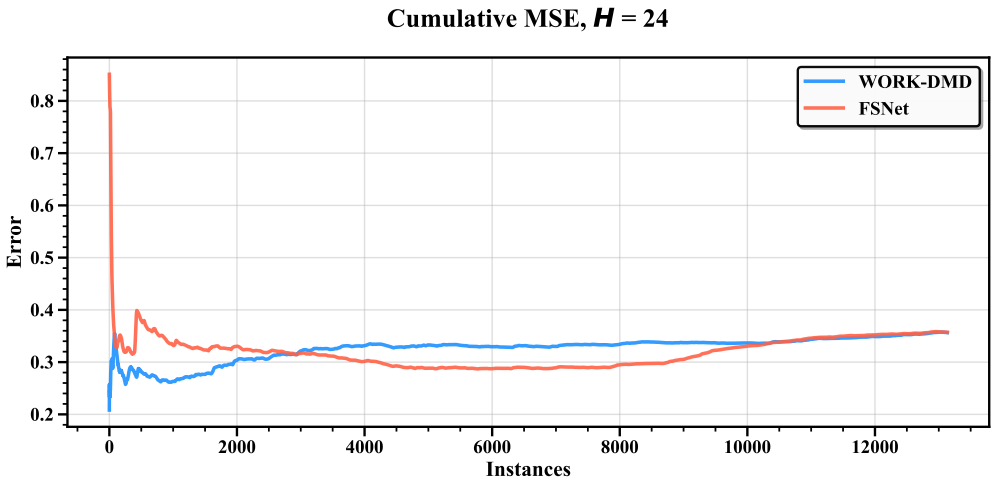


(b) 48-step ahead

Figure C.4: Cumulative MSE comparison on ETTm1 dataset. The figures show cumulative mean squared error progression for 1-step and 48-step ahead forecasting, comparing WORK-DMD and OneNet performance over time. The cumulative curves demonstrate superior performance when $H = 1$ and competitive performance when $H = 48$.



(a) 1-step ahead



(b) 24-step ahead

Figure C.5: Cumulative MSE comparison on Traffic dataset. The figures show cumulative mean squared error progression for ahead forecasting, comparing WORK-DMD and FSNet performance. The cumulative analysis reveals the relative stability and accuracy of both methods across different prediction horizons.