University of California
Santa Barbara

Modern Machine Learning in Time Series Forecasting

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

Doctor of Philosophy
in
Computer Science

by

Xiaoyong Jin

Committee in charge:

Professor Xifeng Yan, Chair
Professor Yu-Xiang Wang
Professor Ambuj K. Singh

March 2022

The Dissertation of Xiaoyong Jin is approved.

Professor Yu-Xiang Wang

Professor Ambuj K. Singh

Professor Xifeng Yan, Committee Chair

February 2022

Modern Machine Learning in Time Series Forecasting

by

Xiaoyong Jin

To my parents, who have been providing unconditional love and endless support.

Acknowledgements

My deepest gratitude goes to my advisor, Prof. Xifeng Yan, without whom I would not start or finish my pursuit of PhD. I appreciate that his introduction for me to the fantastic domain of computer science and his suggestions on time series studies. He is an experienced researcher with focus on details. During these years, he has deeply influenced my way of thinking on both research and life.

I would like to thank Prof. Yu-Xiang Wang and Prof. Ambuj K. Singh for serving on my committee. They provide invaluable comments and feedback on every stage of PhD career over the years.

I'm also very lucky to work with so many brilliant lab mates. I owe many thanks to Hanwen Zha, Zhiyu Chen, Shiyang Li, Wenhu Chen, Xiyou Zhou, Yu Su and Honglei Liu. They provides significant help in my work and I learned a lot from them.

During my internship, it was a great honor to work with my mentors and peers Yusan Lin, Hao Yang, Hao Wang, Youngsuk Park, Danielle Maddix and Bernie Wang. They were so supportive and eager to help, and I learn a lot from the experiences of working with them.

I would like to thank all my friends at UCSB and Amazon. It is always fun to chat, eat and play with you, so I can relieve stress these years. Finally, I want to thank my parents and grandparents for supporting my PhD study abroad.

Curriculum Vitæ

Xiaoyong Jin

Education

2016-2022	Ph.D. in Computer Science (Expected), University of California, Santa Barbara.
2016-2020	M.Sc. in Computer Science, Universoty of California, Santa Barbara
2012-2016	B.Sc. in Mathematics, Zhejiang University

Publications

Xiaoyng Jin, Youngsuk Park, Danielle Maddix, Hao Wang, Bernie Wang, "Adaptation for Time Series Forecasting via Attention Sharing", submitted to AAAI 2022

Xiyou Zhou, Zhiyu Chen, **Xiaoyong Jin**, William Y. Wang, "HULK: An Energy Efficiency Benchmark Platform for Responsible Natural Language Processing", EACL 2021.

Xiaoyng Jin, Yu-Xiang Wang, Xifeng Yan, "Inter-Series Attention Model for COVID-19 Forecasting", SDM 2020

Yunkai Zhang, Qiao Jiang, Shurui Li, **Xiaoyong Jin**, Xueying Ma and Xifeng Yan, "You May Not Need Order in Time Series Forecasting", NeurIPS 2019 TPP workshop

Xiaoyong Jin, Shiyang Li, Xifeng Yan, "Attention Guided Autoregression", Preprint Arxiv 2019

Shiyang Li, **Xiaoyong Jin**, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, Xifeng Yan, "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting", NeurIPS 2019

Xiaoyong Jin, Shiyang Li, Xifeng Yan, "Multi-step Deep Autoregressive Forecasting with Latent States", ICML TS workshop 2018

Abstract

Modern Machine Learning in Time Series Forecasting

by

Xiaoyong Jin

Because of its high dimensionality, complex dynamics and irregularity, forecasting of time series data has been studied by both statistics and machine learning community for decades. The massive and ever-growing volume of data created by modern applications poses even more serious challenges to practical forecasting tools. (1) Scalability. Modern forecasters should be able to process large amount of diverse time series effectively and efficiently. (2) Correlation Awareness. Modern forecasters should be able to take advantage of correlated time series in addition to the history of the current time series. (3). Generalizability. Modern forecasters should be able to generalize to a different data domain than the domain they are trained on. While much effort has been devoted to tackle these issues, forecasting in general is still an open problem.

In this dissertation, we propose complementary approaches targeting these challenges towards large-scale forecasting. We begin with novel neural architectures that are able to deal with various time series for more accurate predictions. Specifically, we present the following methods: Layerwise Recurrent Temporal Convolution Networks (LRTCN) combines the strengths of classic Recurrent Neural Nets (RNN) and Convolution Neural Nets (CNNs) to process long time series. Convolutional Transformer (ConvTrans) aims to enhance locality of attention-based forecasting model to further improve the performance, as well as to break the memory bottleneck for long time series. Attention-guided Autoregression (AGA), on the other hand, brings complicated autoregressive models and simple regreesive models together via tailored attentino mechanism to quickly respond

to change points in forecasting.

Next, we present Attention Cross Time Series (ACTS) that refers to correlated time series in order to guide current forecasting. We study its application to epidemic data and show its effectiveness in predicting cases and deaths of COVID-19 outbreak across the United States.

Finally, we present Domain Adaptation Forecasters (DATS) that enables adapting a forecaster trained on a data-rich source domain to a data-scarce target domain. Based on the idea of domain-invariant data representations in existing domain adaptation approaches, we propose to align a subset of learned features in the forecaster across domains. In the meantime, we keep remaining features domain-specific so that domain-dependent forecasts can be made for each domain.

Contents

Chapter 1

Introduction

Time-related data has been produced at an unprecedented rate in the past decades, with broad application in finance, social networks, E-commerce, video processing, etc. The collection and analysis of such data are of growing importance in various industries. For example, the stock prices are updated every second, based on which investors adjust their portfolios. The daily sales of a collection of products can be vital indicators for retailers to plan future marketing strategies. In addition, web service providers need to monitor performance indicators regularly for anomaly detection and emergency maintenance. The extraordinary growth of time-related data requires modern techniques for large-scale analysis, in which future predictions or forecasting receives increasing attention.

In principal, time-related data is generated by a temporal point process that is dominated by latent factors. At an arbitrary time point, an observation can be recorded and encoded as numerical values. By sampling the latent temporal process at certain rates, we will obtain trajectories, namely time series, as sequence of values associated with time stamps indicating when they are sampled. Formally, a point process is defined as a sequences of tuples of sampled values and time stamps $\{\{(\mathbf{x}_{it}, s_{it})\}_{t=1}^{T_i}\}_{i=1}^{N}$, where N is the number of the collected series, and T_i is the number of samples in the i-th series. For

simplicity, we consider the number of samples in each series the same, and thus omit the subscript of T_i by using T instead. In many cases, the values are sampled at a constant rate, i.e. the time interval between two consecutive observations is fixed, which make up a time series. Therefore, a time series can be simply represented by a sequence of values without explicit time stamps, $\{\{\mathbf{x}_{it}\}_{t=1}^{T_i}\}_{i=1}^{N}$. In this thesis, we will focus on time series as the representative type of time-related data. Depending on the task, time series data is usually considered in three aspects, the values at certain time points of interest, the time stamps of certain values of interest, and the dynamic patterns of changing values within a period of time. In forecasting, the patterns over time are most informative and usually the focus of the study. Essentially, forecasting requires extraction of patterns from historical data, based on which inference of reasonable extrapolation are made.

1.1 Forecasting Services

Modern forecasting tasks expect efficient predictions on massive amount of time series via a forecasting model learned using a collected repository of time series data. Figure 1.1 presents the basic framework of a forecasting service. The collected data, which is even more considerable in terms of amount, is often prepared by the service provider and used to train a set of fundamental and versatile forecasting models for various user demands. The domain where it is collected is referred to as the source domain. The data to be predicted, on the other hand, is provided by the users from a different target domain, based on which the trained forecasting models produce predictions via inference. Meanwhile, the learned forecasting models are adapted to fit the gap between the the source and target domain that usually possess distinct properties, in order to increase forecasting accuracy on the target domain. As an example service, the Amazon Forecast host by Amazon Web Service provides public interfaces to users to upload target data

and optionally related covariates. The cloud service would automatically inspect custom data, identifies key attributes and selects appropriate forecasting models that are adapted to fit the given data and to produce expected predictions in various forms [1].

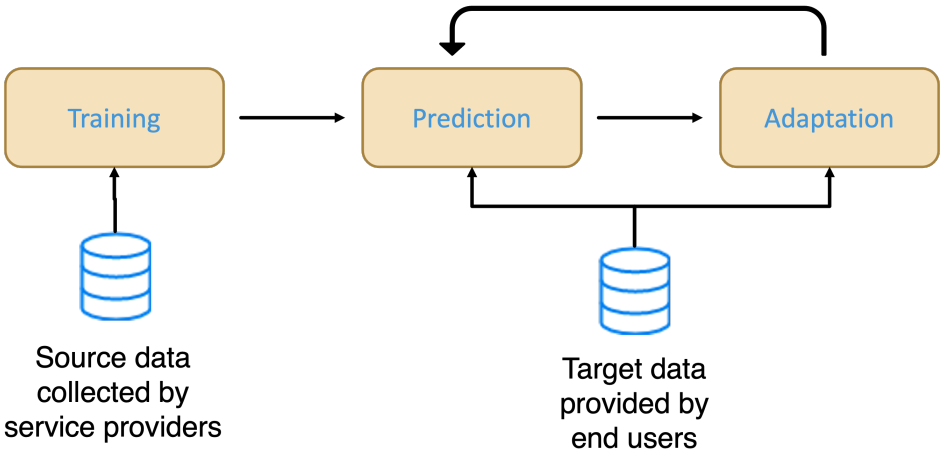


Figure 1.1: General procedure of modern forecasting services.

As per the three stages of forecasting services, there are accordingly three challenges posed on modern forecasting models

1. **Capability.** Can forecasting models capture not only patterns from individual historical series but correlations between different time series in the massive training data?

2. **Scalability.** Can forecasting models process many time series (potentially with distinct properties such as seasonality or length) efficiently?

3. **Adaptability.** Can trained forecasting models be effectively adapted to new data domains while preserving knowledge learned in existing domains?

In this dissertation, we investigate these challenges and study bottlenecks in common forecasting models hindering effective and efficient forecasting. We further propose com-

[1] https://aws.amazon.com/forecast/

plementary solutions towards effort-saving forecasting and domain adaptation. In the following sections, we will first introduce existing studies on each topic.

1.2 Forecasting Models

Generally, a forecasting model can be summarized as an encoder-decoder structure as shown in Figure 1.2. The encoder part receives historical series as input and produces an encoding of all past data. The following decoder then produce future predictions based on the encoding. In the past decades, numerous forecasting methods were proposed, and many of them focused on various designs of the encoder/decoder module. We provide a brief review in the following.

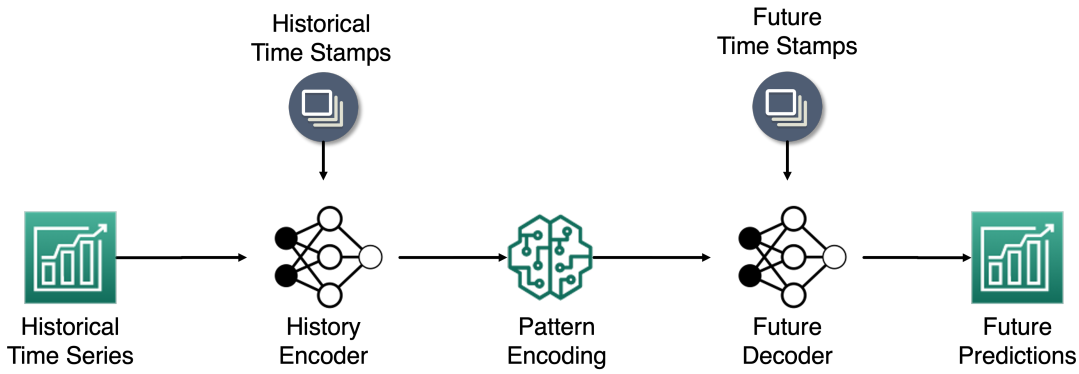


Figure 1.2: General structure of forecasting models. Besides historical time series, the time stamps of the past and the future can also be helpful.

1.2.1 Encoder Module Design

The encoder mainly accounts for pattern extraction from historical data. To accommodate quantitative analysis on the unknown future, we essentially pursue a low-dimensional representation of the long series of past observations in high-dimensional

spaces. Among the variety of representations researchers have invented, we are most interested in three typical paradigms.

Decomposition Generally, there are four types of patterns in time series, as Figure 1.3 shows. First, trends denote long-term increase or decrease, which reflects general changing direction of the entity of interest. For example, population in a country is likely to show upward or downward changes over the past decades. Second, cycles stand for rises and falls without fixed frequencies. They are usually seen in economic indicators where medium-term variations repeat over years. Third, seasonality refers to periodical fluctuations repeating in regular intervals, which are highly likely to be related to calendar events. As an example, the weather conditions such as temperatures usually show stable variations across years with respect to seasons. Finally, irregularity remains after the three patterns aforementioned are estimated and removed from original time series. It results from short-term fluctuations, which are neither predictable nor systematic, caused by random factors exogenous to the point process system.

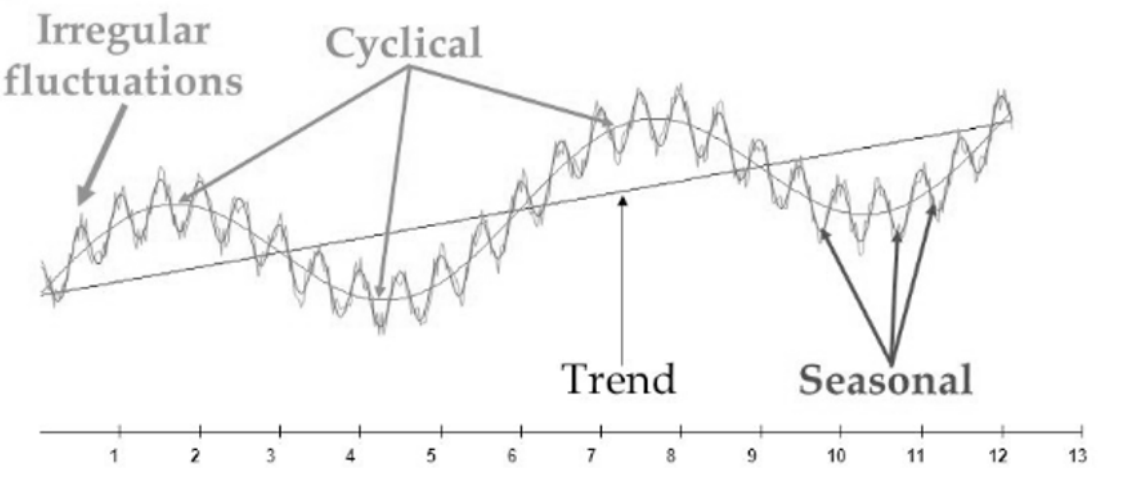


Figure 1.3: The four main components in time series.

In order to deal with highly complicated time series data, the original data is often decomposed into such components to improve systematical predictability [104]. However, there are still challenges for a forecasting tool, i.e. a forecaster, to properly process these components. Specifically, considerable memory or context capacity is usually necessary for the forecaster to capture long-term trends from other short-term variations by processing large amount of historical data. For cyclic data, while being repetitive, it is often challenging to predict the next occurrence of past patterns due to the irregular intervals. For seasonal data, the forecaster has to recognize the seasons if not specified. The remaining irregularity, on the other hand, need to be filtered and modelled by probabilistic frameworks as predicative uncertainty.

In forecasting tasks, decomposition is applied to historical series and predictions are made for each component individually before aggregating into future series. For example, [117] explicitly deconstructs historical time series into trends and seasonalities via Bayesian inference. [108] employs matrix decomposition to discover. components. [89] develops neural expansions on trend and seasonality bases. [49, 134] enables end-to-end learning of decomposition using neural nets instead of a separated pre-processing step. [124] further include wavelet decomposition to capture multiple frequencies.

Latent Variable Models Based on the assumption that time series are generated by some underlying process, low-dimensional latent variables are introduced to model the hidden dynamics, and in the meantime mappings between observed data and the latent variables are learned. Linear Dynamic Systems (LDS) [34] are the simplest latent variable model with linear hidden dynamic and observation emission

$$\mathbf{x}_t = \mathbf{A}\mathbf{z}_t + \mathbf{e}_t, \tag{1.1}$$

$$\mathbf{z}_t = \mathbf{B}\mathbf{z}_{t-1} + \mathbf{r}_t, \tag{1.2}$$

where $\mathbf{z}_t$ is the latent variable, $\mathbf{A}$ is the observation matrix, $\mathbf{B}$ is the transition matrix, and $\mathbf{e}_t, \mathbf{r}_t$ are Gaussian variations. Bayesian inference, typically expectation-maximization (EM) algorithms, are used to estimate the parameters, namely $\mathbf{A}$ and $\mathbf{B}$, and the latent $\mathbf{z}$ alternately.

Based on LDS, various extensions have been proposed to break the constraint of linear mappings and static parameters. [94, 95] assume state-dependent transition matrix. [75] enrich the model with piecewise linear mapping from latent variables $\mathbf{Z}$ to observations $\mathbf{X}$. [123, 30, 150] further introduce nonparametric Gaussian processes to allow even more flexible forms of transition dynamics and observation mappings. However, the optimization and inference in these nonlinear methods are computationally costly and thus remains an open problem.

Neural Embedding Models Instead of explicitly modeling how the dynamics evolve, embedding approaches instead directly compute low-dimensional representations of time series via highly flexible mappings. For example, autoregressive models such as ARIMA [16] compute linear combination of historical observations as pattern encodings, where the weights are determined by least square methods. Neural networks, as powerful approximating functions, are recently used for pattern encoding of massive time series data. They mainly vary in the architecture of pattern encoder, such as conditional Restricted Boltzmann Machine (cRBM) [116], Recurrent Neural Networks (RNN) [42, 39, 23] and Convolutional Neural Networks (CNN) [13, 7]. On the other hand, most of them make predictions in an autoregressive manner, which result in error cascading issue, i.e. previous errors may mislead following forecasts. Some works aim to perform multi-step prediction in parallel. [40] explores fitting the future with a polynomial function, while [129] uses temporal and static covariates to incorporate seasonal patterns and expected spikes. In addition, other works try to improve the interpretability of deep neural net-

works by incorporating traditional mechanisms. [101] aims to approximate nonlinear dynamics with locally linear segments and uses an RNN to parameterize a linear exponential smoothing system. Instead, [128] accounts for noise with a local Gaussian process for each time series instance in supplement to a global RNN modeling shared patterns.

However, even though neural networks excel at capturing patterns in complex time series data, the computational cost of training and inference remains a problem. For long-term predictions, autoregressive decoding method become a bottleneck as it needs to re-run the forward pass of the entire network at every step. Some explore to directly generate all future predictions in one step [40] at the cost of inferior accuracy. [145] introduces tensor decomposition to reduce the complexity in decoding. Nevertheless, a general solution to efficient forecasting is still under active investigation.

1.3 Domain Adaptation

Although modern forecasting models are able to capture complicated nonlinear dynamics and latent features, it is prone to overfit to the training data and suffer performance degradation when applied to a new domain for testing, especially if the training data is insufficient. Unfortunately, in practice there are always subtle distinctions (usually termed as *domain discrepancies*) between training and test data, for which transfer learning strategies need to be introduced to adapt the trained model to test data. In forecasting services, the trained model using collected data should also be updated to fit new user data.

Domain adaptation has been under active investigation for decades. Most researches focus on classification or regression settings, where a label is expected given a static set of inputs [12, 44, 121, 14, 57, 20]. In domain adaptation, knowledge learned in a source domain with sufficient data is transferred to a target domain with unlabelled data

or limited amount of labelled data [86, 130]. The mainstream methods aim to extract invariant features between the source and target domain, so that a consistent mapping from the feature space to the label space can apply to both domains. It is generally achieved by optimizing the data representation to minimize certain measure of domain shift such as maximum mean discrepancy (MMD) [120, 82], correlation distances [114, 115] or optimal transport [28]. Another way is to reconstruct target input using source representations [48]. Recently, adversarial adaptation methods have gained increasing popularity, in which an adversarial objective with respect to a domain discriminator is optimized to approximate the domain discrepancy. These methods are closely related to generative adversarial learning, which pits a generator and a discriminator against each other: the generator is trained to produce samples to confuse the discriminator while the discriminator in turn tries to distinguish generated samples from real ones [50]. Likewise, the adaptation model is trained to fit source data as well as to ensure that the domain discriminator cannot distinguish source features from target ones [2, 109].

However, time series forecasting differs from conventional settings in two aspects. First, time series contain dynamically evolving patterns so that the expected prediction may also changing accordingly. Second, instead of a fixed label space, the labels in forecasting, i.e. the future values come from the same data space as the input and thus are strongly related to the input too. Therefore, it is not straightforward to apply existing domain methods to forecasting.

1.4 Thesis Organization

In this thesis, we will discuss solutions to modern forecasting tasks as well as domain adaptation in forecasting, as the previous sections have covered.

In Chapter **??**

Chapter 2

Multi-step Deep Autoregressive Forecasting with Latent States

2.1 Introduction

Multi-step ahead forecasting is a challenging but critical task with various practical applications, from resource management to business decision making. The forecasting problem is essentially a probabilistic sequence modeling task in which the goal is to capture the stochastic evolution of a time series. Specifically, given a sequence of observations, we are supposed to estimate the trajectory within a period of time in the future, namely forecasting horizon. An example of forecasting result is illustrated in Figure 2.1.

Traditional methods can be roughly classified into two categories: State Space Models (SSMs) and AutoRegressive (AR) models. SSMs introduce latent state variables that summarize all the information coming from the past to determine the present and the future via some latent dynamic. A transition mechanism updates the latent states at each time step and then a prediction is generated with an emission system [63, 107].

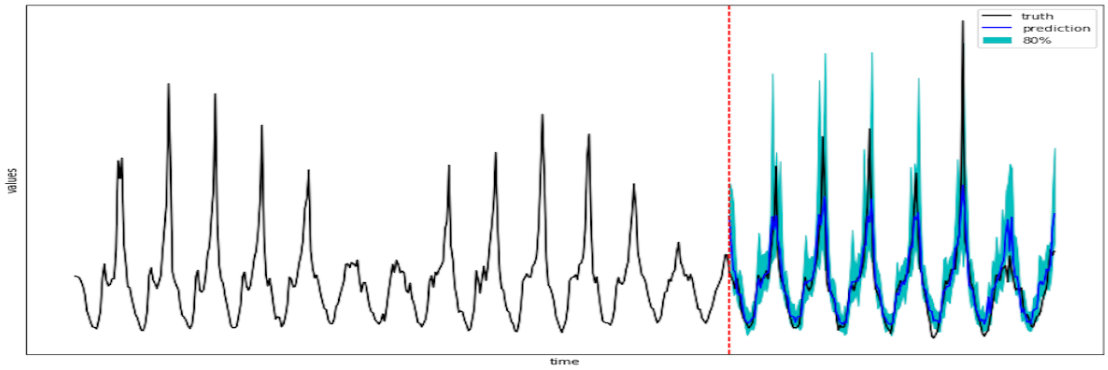


Figure 2.1: An example of probabilistic forecasting. The colored area is the uncertainty interval with 80% confidence. The red dashed line indicates the start time of forecasting, i.e. the left side is observed history and the right side shows predictions in comparison with ground truth.

AR models, on the other hand, describe a stochastic process where the current observation depends on a range of observed history. A prediction is made based on previous observations and becomes a part of history for the next step.

Deep neural networks have been proposed as an alternative approach to the forecasting problem. Most existing works employ a Recurrent Neural Network (RNN) [70, 78, 145, 39] or a Temporal Convolutional Netowrk (TCN) [13] to model a time series after an autoregressive fashion. These approaches decompose the multi-step forecasting task into a number of single-step subtasks where a sequence of history is mapped to one prediction at each time step. In essence they are generalizations of traditional linear AR models using nonlinear neural networks. Although they are able to accommodate extremely complicated dynamics, a large neural network has to be applied recursively to obtain multiple steps of predictions, which is computationally expensive and difficult to parallelize. In addition, these methods can propagate error from one prediction to the next due to dependence among steps [40].

In contrast, some works extend traditional linear Gaussian SSMs by combining latent states with deep neural networks. One approach is to replace linear transition with

Multi-layer Perceptrons (MLPs) [68] or RNNs [26]. [41] instead incorporates deterministic hidden states of an underlying RNN into the latent process. [101] keeps the linear Gaussian structure but uses an RNN to specify the transition parameters. However, at each time step the latent state can only access the previous state, either stochastic or deterministic, which is supposed to encode all information. This might cause the predefined latent dynamics to fail with heterogeneous data, especially in a large and diverse time series corpus.

In this paper, we propose a novel framework named **Deep Autoregressive Latent** (**DARL**) network that fuses the advantages of both SSMs and AR models. The framework introduces a latent state at each time step within the forecasting horizon. It consists of a prior network that models the prior distribution of latent states based on observed history, an emission network that makes predictions conditioned on latent states, and an inference network that approximates latent posteriors given the ground truth in the future. The prior network and the inference network employ a **Layer Recurrent TCN** (**LRTCN**) encoder, which maps a range of observations to a latent state. It captures temporal patterns and long-term dependencies in the history with TCNs and then implicitly restores the temporal order with recurrent connections between TCN layers. The framework is trained using stochastic variational inference [58, 66]. Note that we avoid the issues of accumulating error by predicting multiple steps in parallel, which also significantly accelerates evaluation.

We evaluate our method on three public benchmark datasets and a proprietary dataset including performance metrics from a real production environment. Our model outperforms traditional methods and obtains better or competitive results against recently proposed baselines. Moreover, our model can be evaluated much faster than RNN-based models.

2.2 Related Works

The field of time series forecasting has been supported by an abundance of classic algorithms from the statistics community. State space model (SSM) is a principled framework that models and learns time series patterns via a sequence of latent states. A prominent example is exponential smoothing, which explicitly maintains patterns such as level, trend and seasonality with linear transitions [63]. Another influential family is autoregressive (AR) models. AR models specifically depend the prediction on previous observations and a stochastic term. As a generalization, autoregressive integrated moving average (ARIMA) models combine the properties of both families. Equipped with Box-Jenkins methodology, it is the first attempt of many practitioners and there are abundant researches on its variants [16]. However, the linear assumption and high computational cost make these methods not suitable for large-scale forecasting tasks. Further, they cannot transfer learned patterns across similar time series instances as they are separately fitted.

Deep neural networks have been proposed to deal with massive time series corpus. By extracting higher-order features and modeling nonlinear dynamics, they are able to identify complex patterns and temporal-spatial correlations without much human effort [31, 146]. Recently, the successful practice of deep RNNs [25, 56] in many sequence modeling tasks such as language modeling has inspired the forecasting community. [39] augments traditional linear AR models with RNNs to accommodate complicated non-linear dynamics. [145] employs tensor-train to directly model higher-order moments and transition structures. On the other hand, to address the error accumulation issue in vanilla recurrent architectures, some works aim to perform multi-step prediction in parallel. [40] explores fitting the future with a polynomial function, while [129] uses temporal and static covariates to incorporate seasonal patterns and expected spikes. In addition,

other works try to improve the interpretability of deep neural networks by incorporating traditional mechanisms. [101] aims to approximate nonlinear dynamics with locally linear segments and uses an RNN to parameterize a linear exponential smoothing system. Instead, [128] accounts for noise with a local Gaussian process for each time series instance in supplement to a global RNN modeling shared patterns.

Convolutional neural networks (CNNs) have also been proven to be effective in sequence modeling. [70, 78] introduce a CNN layer to model temporal or spatial local patterns. Moreover, temporal Convolutional Networks (TCNs) [47, 88] achieve competitive performances without any recurrence structure and significantly accelerate training on parallel computational platforms [7]. [13] attempts to apply TCN directly to time-series data. In addition, TCNs are flexible to vary the amount of accessible history by controlling the size of their receptive fields. Consequently, practitioners are able to balance long-term dependencies and short-term concentrations [143].

Deep networks have also been proposed to augment traditional SSMs with nonlinear transition structures. [41] makes the latent process dependent on an underlying RNN, while [26] radically cuts the ties between latent states and associates them to the deterministic RNN dynamics. Both methods extend variational autoencoders to sequential data and learn the models with stochastic variational inference, where the latent priors are conditioned on the observed history.

2.3 Methodology

In multi-step ahead forecasting tasks, given a sequence of observations $\mathbf{x}_{1:t}$, the goal is to predict τ steps in the future, i.e. $\mathbf{x}_{t+1:t+\tau}$. We call $\mathbf{x}_{1:t}$ the observed history and τ the forecasting horizon. Formally, we want to model a conditional predictive distribution $p(\mathbf{x}_{t+1:t+\tau}|\mathbf{x}_{1:t})$. State Space Models introduce a series of latent state variables $\mathbf{z}_{t+1:t+\tau}$ to

encode temporal patterns [63]. They decompose the predictive model into three modules:

$$\begin{aligned} \mathbf{z}_{t+i-1} &\sim p(\mathbf{z}_{t+i-1}|\mathbf{x}_{1:t+i-1}); \\ \mathbf{z}_{t+i} &\sim p(\mathbf{z}_{t+i}|\mathbf{z}_{t+i-1}); \\ \mathbf{x}_{t+i} &\sim p(\mathbf{x}_{t+i}|\mathbf{z}_{t+i}); \end{aligned} \tag{2.1}$$

for $i = 1, 2, \cdots, \tau$. Here $\mathbf{z}_{t+i-1} \sim p(\mathbf{z}_{t+i-1}|\mathbf{x}_{1:t+i-1})$ is a filtered posterior distribution of $\mathbf{z}_{t+i-1}$ that summarizes the history. $p(\mathbf{z}_{t+i}|\mathbf{z}_{t+i-1})$ models a transition mechanism that determines the latent dynamic. $p(\mathbf{x}_{t+i}|\mathbf{z}_{t+i})$ is an emission system that produces predictions conditioned on the corresponding latent state at each step. We propose to replace the recursive latent dynamic $p(\mathbf{z}_{t+i}|\mathbf{z}_{t+i-1})$ in equation (2.1) with a straightforward translation from $\mathbf{z}_t$

$$\mathbf{z}_{t+i} \sim p(\mathbf{z}_{t+i}|\mathbf{z}_t, \mathbf{e}_{t+1:t+i}), \quad i = 1, \cdots, \tau, \tag{2.2}$$

by taking advantage of a series of known covariates $\mathbf{e}_{t+1:t+i}$ that indicate common patterns such as daily or weekly seasonality and a stochastic encoding of given history $\mathbf{z}_t$. The translation is intuitively a guess based on our knowledge about the impact of various factors inside the covariates upon the current state. For example, suppose that $\mathbf{z}_t$ carries the information about level and trend at time step t, one can estimate the level at a future time step with the elapsed time from t contained in covariates. This estimation is reasonable as long as the patterns in the past remain stationary within the forecasting horizon, which is a common assumption in forecasting.

Recent findings have shown that a single $\mathbf{z}_t$ might be ignored by the translation [15, 143]. To alleviate this, we employ a modified translation equation (2.4) with dependence

on $\mathbf{x}_{1:t}$ and thus get the following predictive process:

$$\mathbf{z}_t \sim p(\mathbf{z}_t|\mathbf{x}_{1:t}); \tag{2.3}$$

$$\mathbf{z}_{t+i} \sim p(\mathbf{z}_{t+i}|\mathbf{z}_t; \mathbf{x}_{1:t}, \mathbf{e}_{t+1:t+i}); \tag{2.4}$$

$$\mathbf{x}_{t+i} \sim p(\mathbf{x}_{t+i}|\mathbf{z}_{t+i}), \quad i = 1, \cdots, \tau. \tag{2.5}$$

From a Bayesian perspective, we call equation (2.4) a **prior network**, because it models priors over the latent states before we observe the ground truths within the horizon. We also name equation (2.5) an **emission network** as in SSMs. Furthermore, combining the prior network and the emission network makes a **generative network** because it describes the generative process of the time series.

Since exact inference for equation (2.1) is intractable, the model cannot be learned by directly maximizing the likelihood of ground truths. Stochastic variational inference [58] provides an alternative that maximizes a lower bound of likelihood by introducing an **inference network** to approximate the latent posterior with another probabilistic model $q(\mathbf{z}_{t+i}|\mathbf{x}_{1:t+i})$. Note that equation (2.3) has the same dependence structure as the inference network when $i = 0$, as it is also a filtered posterior of latent state at t. Hence we also apply the inference network to equation (2.3).

We use deep neural nets to parameterize all networks above. An overview of the framework is illustrated in Figure 2.2.

2.3.1 Emission Network

The emission network parameterizes the conditional distribution equation (2.5) through a neural network with learnable parameters $\boldsymbol{\theta}_d$. The type of distribution can be chosen with flexibility according to data being modeled. For example, we can employ negative binomial distribution for positive count data and beta distribution for percentage data

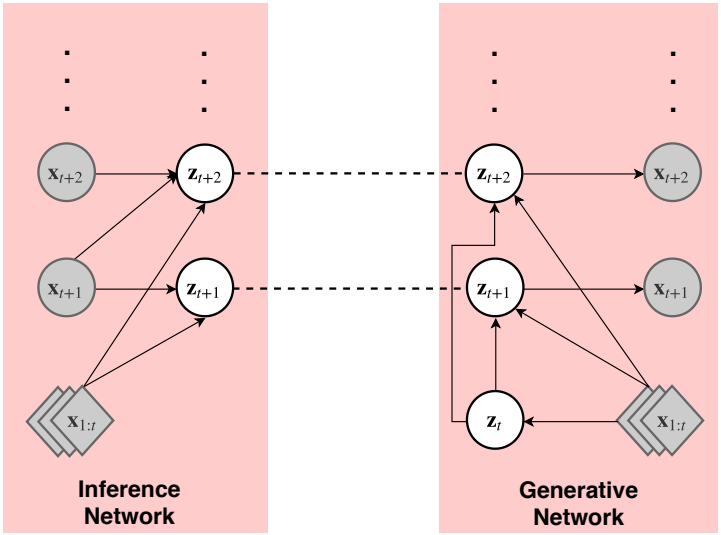


Figure 2.2: Graphical view of inference network (left) and generative network (right). Plain circles are latent states, shaded circles are observable data, and diamonds are deterministic history. The approximate posterior of $\mathbf{z}_{t+i}$ is conditioned on $\mathbf{x}_{1:t+i}$, while the prior only depends on $\mathbf{x}_{1:t}$ and $\mathbf{z}_t$. Dashed lines connect priors and their posterior counterparts.

[39]. For simplicity, we only illustrate Gaussian distribution:

$$\mathbf{x}_{t+i} = \boldsymbol{\mu}_{\boldsymbol{\theta}_d}(\mathbf{z}_{t+i}) + \boldsymbol{\sigma}_{\boldsymbol{\theta}_d}(\mathbf{z}_{t+i}) \cdot \boldsymbol{\varepsilon}_{t+i} \tag{2.6}$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}_d}, \boldsymbol{\sigma}_{\boldsymbol{\theta}_d}$ are feed-forward networks with one or more hidden layers, $\boldsymbol{\varepsilon}_{t+i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\theta}_d}$, we add a softplus activation function to the last hidden output in order to ensure positivity of the standard deviation. Note that $\boldsymbol{\theta}_d$ is shared across the forecasting horizon, as latent states encode all time-specific information.

2.3.2 Inference Network

The inference network maps a range of history to a latent variable. Instead of commonly-used RNNs, Temporal convolutional networks (TCN) has been proven to be more effective and efficient in many sequential modeling tasks [7]. Although TCNs do

not have a sense of temporal order because of their locality, the higher-level representations in TCNs have access to longer history. Hence the convolutional feature maps at different layers implicitly form a temporal hierarchy in a top-down fashion. We propose to explicitly reconstruct the temporal structure by stacking a top-down RNN layer over the hierarchy. A systematical illustration is shown in Figure 2.3. In our experiments, we choose to use GRU cells [25] to instantiate the RNN. Note that we also use covariates as an additional input associated with the observations to the TCN module. These time-based covariates can be regarded as "positional embeddings" that expose the temporal information to order-insensitive TCNs [47]. We use $\boldsymbol{\theta}_c$ to denote all trainable parameters in an LRTCN.

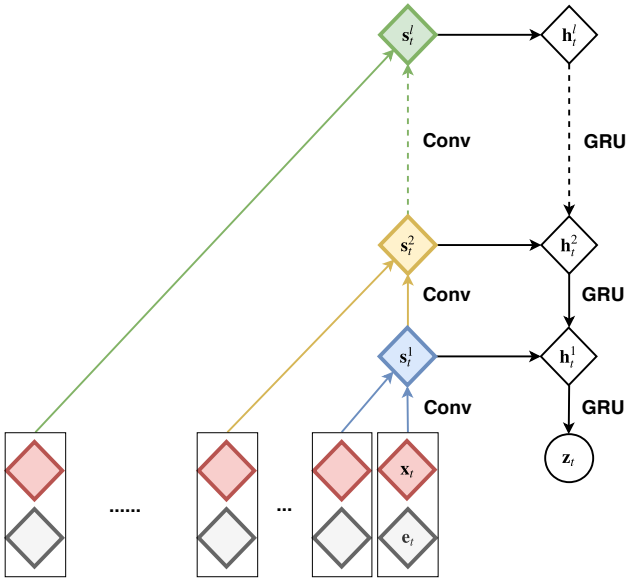


Figure 2.3: An architectural view of an LRTCN encoder. The TCN representations at different layers are consumed by a GRU network. As the upper layers have access to distant history and the lower layers focus on recent observations, the top-down hierarchy implicitly keeps the temporal order of the input sequence. We also feed covariates into the TCN associated with observations.

As shown in Figure 2.3, the approximated posterior of $\mathbf{z}_{t+i}$ is derived from the RNN output at the first layer of LRTCN, and similar to the emission network, we apply feed-

forward networks with parameter $\boldsymbol{\theta}_e$ to obtain the state

$$\begin{aligned} \mathbf{h}_{t+i}^{(1)} &= \text{LRTCN}(\mathbf{x}_{1:t+i}, \mathbf{e}_{1:t+i}; \boldsymbol{\theta}_c) \\ \mathbf{z}_{t+i} &= \boldsymbol{\mu}_{\boldsymbol{\theta}_e}\left(\mathbf{h}_{t+i}^{(1)}\right) + \boldsymbol{\sigma}_{\boldsymbol{\theta}_e}\left(\mathbf{h}_{t+i}^{(1)}\right) \cdot \boldsymbol{\varepsilon}_{t+i}. \end{aligned} \tag{2.7}$$

Note that the inference process resembles a Bayes filter of a recursive latent process. This implies the implicit temporal connection among latent states (and the intermediate state) even if they are conditionally independent in equation (2.4).

2.3.3 Prior Network

Notice that equation (2.4) also explicitly attend to past observations except that it should never have access to the current time step or unknown future. Therefore, we apply the same architecture as the inference network to the prior network but first roughly estimate $\mathbf{x}_{t+1:t+\tau}$ with corresponding covariates and $\mathbf{z}_t$ through another group of feed-forward networks $f_{\boldsymbol{\theta}_g}$

$$\tilde{\mathbf{x}}_{t+i} = f_{\boldsymbol{\theta}_g}\left(\mathbf{z}_t, \mathbf{e}_{t+i}\right) \quad i = 1, \cdots, \tau, \tag{2.8}$$

where $\boldsymbol{\theta}_g$ contains trainable parameters. $\boldsymbol{\theta}_g$ is shared across the forecasting horizon as in the emission network. We then apply LRTCN to the concatenation $[\mathbf{x}_{1:t}; \tilde{\mathbf{x}}_{t+1:t+\tau}]$ along with $\mathbf{e}_{1:t+\tau}$ as in the inference network. We illustrate an overview of this structure in Figure 2.4.

Finally, we distinguish priors from posteriors by introducing different feed-forward networks with parameter $\boldsymbol{\theta}_p$ to shape the Gaussian priors

$$\begin{aligned} \tilde{\mathbf{h}}_{t+i}^{(1)} &= \text{LRTCN}(\mathbf{x}_{1:t}, \tilde{\mathbf{x}}_{t+1:t+i}, \mathbf{e}_{1:t+i}; \boldsymbol{\theta}_c) \\ \mathbf{z}_{t+i} &= \boldsymbol{\mu}_{\boldsymbol{\theta}_p}\left(\tilde{\mathbf{h}}_{t+i}^{(1)}\right) + \boldsymbol{\sigma}_{\boldsymbol{\theta}_p}\left(\tilde{\mathbf{h}}_{t+i}^{(1)}\right) \cdot \boldsymbol{\varepsilon}_{t+i}. \end{aligned} \tag{2.9}$$

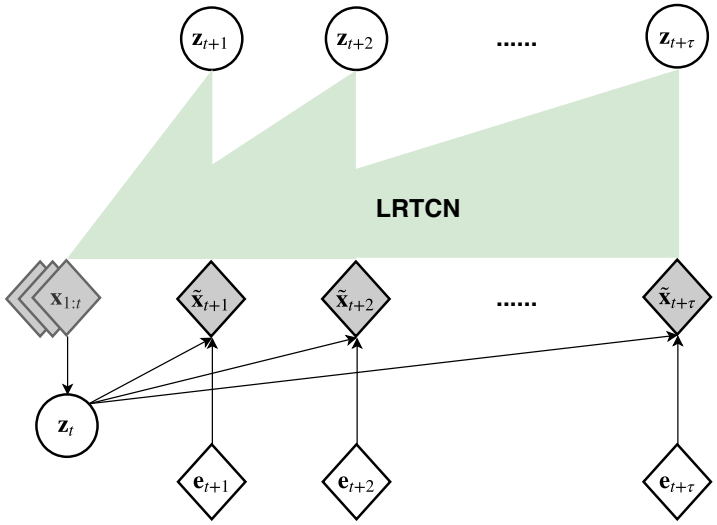


Figure 2.4: The prior network. The estimations $\tilde{\mathbf{x}}_{t+1:t+\tau}$ and the history $\mathbf{x}_{1:t}$ are concatenated and fed into the LRTCN with covariates as in the inference model.

2.3.4 Training

To learn the model, we are given a time series dataset $\{\mathbf{x}_{1:T}^{(n)}\}_{n=1}^{M}$ and associated covariates $\{\mathbf{e}_{1:T}^{(n)}\}_{n=1}^{M}$, where T is the length of all available observations and M is the number of different time series. We create training instances by selecting windows with fixed history length t and forecasting horizon τ but varying the start point of forecasting from each of the original long time series [39]. As a result, we get a training dataset with N sliding windows $\{\mathbf{x}_{1:t+\tau}^{(n)}, \mathbf{e}_{1:t+\tau}^{(n)}\}_{n=1}^{N}$.

The Evidence Lower Bound (ELBO) [58] of the log-likelihood can be derived as:

$$\log p(\mathbf{x}_{t+1:t+\tau}^{(n)}|\mathbf{x}_{1:t}^{(n)}) \geq -\mathcal{L}_{\text{NLL}}^{(n)} - \mathcal{L}_{\text{KL}}^{(n)} \tag{2.10}$$

where

$$\begin{aligned} \mathcal{L}_{\text{NLL}}^{(n)} &= -\sum_{i=1}^{\tau} \mathbb{E}_{q(\mathbf{z}_{t+i}^{(n)})} \left[\log p(\mathbf{x}_{t+i}^{(n)}|\mathbf{z}_{t+i}^{(n)})\right] \\ \mathcal{L}_{\text{KL}}^{(n)} &= \mathbb{E}_{q(\mathbf{z}_{t}^{(n)})} \sum_{i=1}^{\tau} \text{KL}\left(q(\mathbf{z}_{t+i}^{(n)}|\mathbf{x}_{1:t+i}^{(n)}) \| p(\mathbf{z}_{t+i}^{(n)}|\mathbf{z}_{t}^{(n)})\right). \end{aligned} \tag{2.11}$$

Hence we define our loss function

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\left(\mathcal{L}_{\text{NLL}}^{(n)} + \mathcal{L}_{\text{KL}}^{(n)}\right) \tag{2.12}$$

and learn the prior, emission and inference network jointly by minimizing the loss w.r.t their parameters, namely $\boldsymbol{\theta}_g, \boldsymbol{\theta}_p, \boldsymbol{\theta}_d, \boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_c$.

2.4 Experiments

We conducted experiments on three public benchmark datasets *electricity* [1], *traffic* [2] and *M4-hourly* [3]. Each dataset is split into a training set, a validation set and a test set in chronological order as in [144]. We compare our method with shallow methods ARIMA, exponential smoothing (ETS) and TRMF [144], as well as recent deep models DeepAR [39] and Deep SSM [101]. For fair comparison, we use ρ-quantile risk to evaluate the prediction accuracy. The ρ-quantile risk R_ρ with $\rho \in (0, 1)$ is defined as:

$$R_\rho(\mathbf{x}, \hat{\mathbf{x}}) = \frac{2\sum_{i,t}(\rho - \mathbf{1}_{x \le \hat{x}})(x_t^{(i)} - \hat{x}_t{}^{(i)})}{\sum_{i,t}|x_t^{(i)}|},$$

where $\hat{x}$ is the empirical ρ-quantile of the predictive distribution.

Table 2.1 summarizes the experimental results. Generally, our model achieves better or competitive results. All models have similar number of parameters.

We also compare the evaluation speed of our model with DeepAR, a representative recurrent model that makes predictions step-by-step. After both models are trained, we predict 100 time series from *electricity* dataset for a number of steps on a single Nvidia GTX 1080 Ti GPU. For each prediction, 200 samples are drawn. We repeat the evaluation 10 times and report the average elapsed time in Figure 2.5. Notice that our

[1] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[2] http://archive.ics.uci.edu/ml/datasets/PEMS-SF
[3] https://github.com/M4Competition/M4-methods/tree/master/Dataset

	Dataset	Electricity		Traffic		M4
	Horizon	1 day	1 week	1 day	1 week	2 days
ARIMA	$R_{0.5}$	0.154	0.30	0.223	0.501	0.052
	$R_{0.9}$	0.102	0.110	0.137	0.298	0.035
ETS	$R_{0.5}$	0.101	0.130	0.236	0.532	0.054
	$R_{0.9}$	0.077	0.110	0.148	0.60	0.027
TRMF	$R_{0.5}$	0.084	0.087	0.186	0.202	0.057
DeepAR	$R_{0.5}$	0.075	0.125	0.161	0.219	0.090
	$R_{0.9}$	0.040	0.080	**0.099**	0.138	0.030
DeepSSM	$R_{0.5}$	0.083	0.085	0.167	**0.168**	0.044
	$R_{0.9}$	0.056	0.057	0.113	0.114	0.027
Ours	$R_{0.5}$	**0.073**	**0.085**	**0.146**	0.169	0.037
	$R_{0.9}$	**0.038**	0.053	0.105	**0.113**	0.019

Table 2.1: Result summary of short-term (1-2 days) and long-term (1 week) forecasting.

model is much faster than DeepAR. Moreover, the evaluation time of our model almost remains constant as the forecasting horizon grows. In contrast, the time cost of DeepAR increases linearly due to sequential generation.

2.4.1 Ablation Study

To demonstrate the necessity of our design, we conducted an ablation study. We remove an individual component in our framework at a time and get the following baseline models:

- **DARLw/oGRU**: The GRU in LRTCN is removed and the encoder is thus a pure TCN;
- **DARL-straight**: The prior network directly obtains future latent states from the intermediate state and the covariates without LRTCN. Specifically, we introduce a simple MLP with one hidden layer to map $\mathbf{z}_t$ with $\mathbf{e}_{t+i}$ to $\mathbf{z}_{t+i}$. In this setting, the prior network has no direct access to the history.

For fair comparison, we modify the hidden dimensions of both baselines such that they

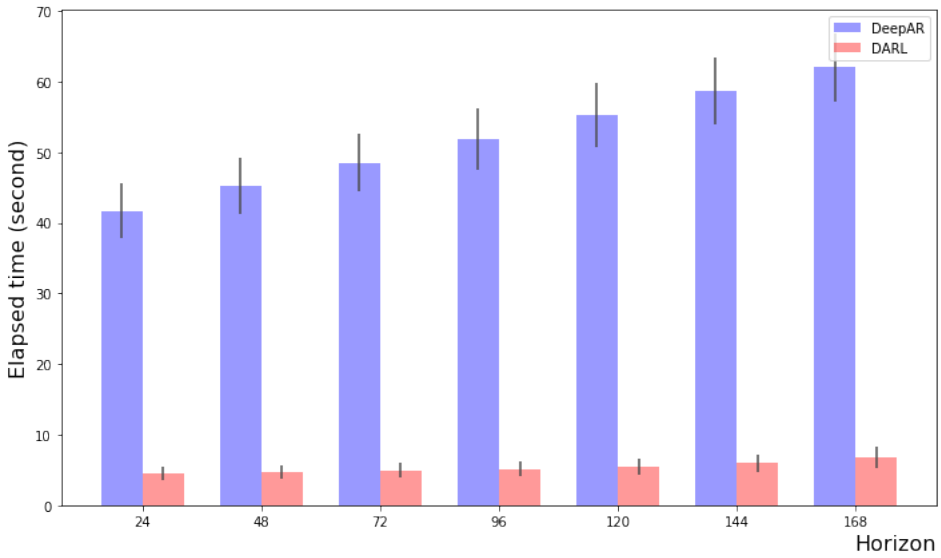


Figure 2.5: The evaluation time of DeepAR and our model on *electricity* dataset. Vertical black lines indicate standard deviations. The time cost of DeepAR increases linearly while our model is much more efficient.

have similar numbers of parameters to the complete model. The short-term predictions within 24/48 hours and long-term predictions within 168 hours are compared. The test results on *electricity* and *traffic* datasets in terms of $R_{0.5}$ and $R_{0.9}$ are shown in Figure 2.6.

The results show that our design is effective in improving forecasting performance. Note that the significant performance reduction of DARL-straight proves that naïve translation without history is not enough to capture latent dynamics, especially in the long term. On the other hand, it indicates the effectiveness of our autoregressive prior network.

2.5 Conclusions

We present a new deep learning framework for multi-step ahead time series forecasting task that combines the strengths of both autoregressive models and state space

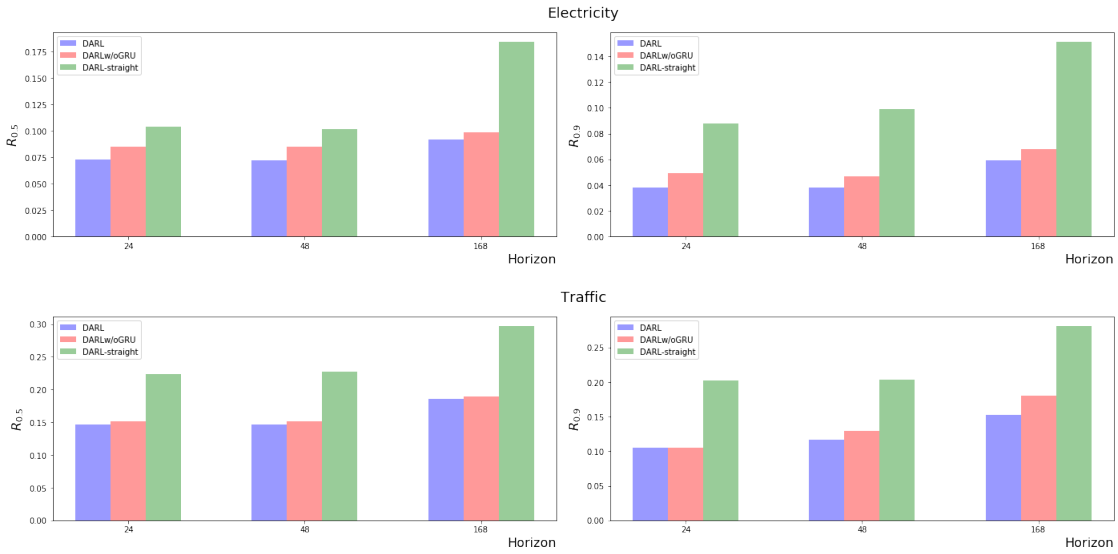


Figure 2.6: Results of DARL in the ablation tests on *Electricity* and *Traffic* dataset.

models. We avoid step-by-step generation by a fully-parallelizable latent process. This also prevents the issue of accumulating errors in recurrent models. Our model is able to achieve better or competitive performance on a variety of datasets. A main challenge we are facing is to improve long-term forecasting accuracy. Modeling latent process within the forecasting horizon efficiently is a crucial step.

Chapter 3

Enhanced Attention-based Forecasting

3.1 Introduction

Although still widely used, traditional time series forecasting models, such as State Space Models (SSMs) [36] and Autoregressive (AR) models [16], are designed to fit each time series independently. Besides, they also require practitioners' expertise in manually selecting trend, seasonality and other components. To sum up, these two major weaknesses have greatly hindered their applications in the modern large-scale time series forecasting tasks.

To tackle the aforementioned challenges, deep neural networks have been proposed as an alternative solution, where Recurrent Neural Network (RNN) [70, 145] has been employed to model time series in an autoregressive fashion. However, RNNs are notoriously difficult to train [92] because of gradient vanishing and exploding problem. Despite the emergence of various variants, the issues still remain unresolved. As an example, [65] shows that language models using LSTM have an effective context size of about 200

tokens on average but are only able to sharply distinguish 50 tokens nearby, indicating that even LSTM struggles to capture long-term dependencies. On the other hand, real-world forecasting applications often have both long- and short-term repeating patterns [70]. For example, the hourly occupancy rate of a freeway in traffic data has both daily and hourly patterns. In such cases, how to model long-term dependencies becomes the critical step in achieving promising performances.

Recently, Transformer [122, 90] has been proposed as a brand new architecture which leverages attention mechanism to process a sequence of data. Unlike the RNN-based methods, Transformer allows the model to access any part of the history regardless of distance, making it potentially more suitable for grasping the recurring patterns with long-term dependencies. However, canonical dot-product self-attention matches queries against keys insensitive to local context, which may make the model prone to anomalies and bring underlying optimization issues. More importantly, space complexity of canonical Transformer grows quadratically with the input length L, which causes memory bottleneck on directly modeling long time series with fine granularity. We specifically delve into these two issues and investigate the applications of Transformer to time series forecasting.

3.2 Related Work

Deep neural networks have been proposed to capture shared information across related time series for accurate forecasting. [39] fuses traditional AR models with RNNs by modeling a probabilistic distribution in an encoder-decoder fashion. Instead, [129] uses an RNN as an encoder and Multi-layer Perceptrons (MLPs) as a decoder to solve the so-called error accumulation issue and conduct multi-ahead forecasting in parallel. [101] uses a global RNN to directly output the parameters of a linear SSM at each step for each

time series, aiming to approximate nonlinear dynamics with locally linear segments. In contrast, [128] deals with noise using a local Gaussian process for each time series while using a global RNN to model the shared patterns.

The well-known self-attention based Transformer [122] has recently been proposed for sequence modeling and has achieved great success. Several recent works apply it to translation, speech, music and image generation [61, 97, 91]. However, scaling attention to extremely long sequences is computationally prohibitive since the space complexity of self-attention grows quadratically with sequence length. This becomes a serious issue in forecasting time series with fine granularity and strong long-term dependencies.

3.3 Methodology

Suppose we have a collection of N related univariate time series $\{\mathbf{z}_{i,1:t_0}\}_{i=1}^{N}$, where $\mathbf{z}_{i,1:t_0} \triangleq [\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \cdots, \mathbf{z}_{i,t_0}]$ and $\mathbf{z}_{i,t} \in \mathbb{R}$ denotes the value of time series i at time t[1]. We are going to predict the next τ time steps for all time series, i.e. $\{\mathbf{z}_{i,t_0+1:t_0+\tau}\}_{i=1}^{N}$. Besides, let $\{\mathbf{x}_{i,1:t_0+\tau}\}_{i=1}^{N}$ be a set of associated time-based covariate vectors with dimension d that are assumed to be known over the entire time period, e.g. day-of-the-week and hour-of-the-day. We aim to model the following conditional distribution

$$p(\mathbf{z}_{i,t_0+1:t_0+\tau}|\mathbf{z}_{i,1:t_0}, \mathbf{x}_{i,1:t_0+\tau}; \mathbf{\Phi}) = \prod_{t=t_0+1}^{t_0+\tau} p(\mathbf{z}_{i,t}|\mathbf{z}_{i,1:t-1}, \mathbf{x}_{i,1:t}; \mathbf{\Phi}). \tag{3.1}$$

We reduce the problem to learning a one-step-ahead prediction model $p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t}; \mathbf{\Phi})$ [2], where $\mathbf{\Phi}$ denotes the learnable parameters shared by all time series in the collection. To fully utilize both the observations and covariates, we concatenate them to obtain an

[1] Here time index t is relative, i.e. the same t in different time series may represent different actual time point.

[2] Since the model is applicable to all time series, we omit the subscript i for simplicity and clarity.

augmented matrix as follows:

$$\mathbf{y}_t \triangleq [\mathbf{z}_{t-1} \circ \mathbf{x}_t] \in \mathbb{R}^{d+1}, \qquad \mathbf{Y}_t = [\mathbf{y}_1, \cdots, \mathbf{y}_t]^T \in \mathbb{R}^{t \times (d+1)},$$

where $[\cdot \circ \cdot]$ represents concatenation. An appropriate model $\mathbf{z}_t \sim f(\mathbf{Y}_t)$ is then explored to predict the distribution of $\mathbf{z}_t$ given $\mathbf{Y}_t$.

Transformer We instantiate f with Transformer [3] by taking advantage of the multi-head self-attention mechanism, since self-attention enables Transformer to capture both long- and short-term dependencies, and different attention heads learn to focus on different aspects of temporal patterns. These advantages make Transformer a good candidate for time series forecasting.

In the self-attention layer, a multi-head self-attention sublayer simultaneously transforms $\mathbf{Y}$ [4] into H distinct query matrices $\mathbf{Q}_h = \mathbf{Y}\mathbf{W}_h^Q$, key matrices $\mathbf{K}_h = \mathbf{Y}\mathbf{W}_h^K$, and value matrices $\mathbf{V}_h = \mathbf{Y}\mathbf{W}_h^V$ respectively, with $h = 1, \cdots, H$. Here $\mathbf{W}_h^Q, \mathbf{W}_h^K \in \mathbb{R}^{(d+1)\times d_k}$ and $\mathbf{W}_h^V \in \mathbb{R}^{(d+1)\times d_v}$ are learnable parameters. After these linear projections, the scaled dot-product attention computes a sequence of vector outputs:

$$\mathbf{O}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = \text{softmax}\left(\frac{\mathbf{Q}_h\mathbf{K}_h^T}{\sqrt{d_k}} \cdot \mathbf{M}\right)\mathbf{V}_h. \tag{3.2}$$

Note that a mask matrix $\mathbf{M}$ is applied to filter out rightward attention by setting all upper triangular elements to $-\infty$, in order to avoid future information leakage. Afterwards, $\mathbf{O}_1, \mathbf{O}_2, \cdots, \mathbf{O}_H$ are concatenated and linearly projected again. Upon the attention output, a position-wise feedforward sublayer with two layers of fully-connected network and a ReLU activation in the middle is stacked.

[3] By referring to Transformer, we only consider the autoregressive Transformer-decoder in the following.

[4] At each time step the same model is applied, so we simplify the formulation with some abuse of notation.

3.3.1 Enhancing the locality of Transformer

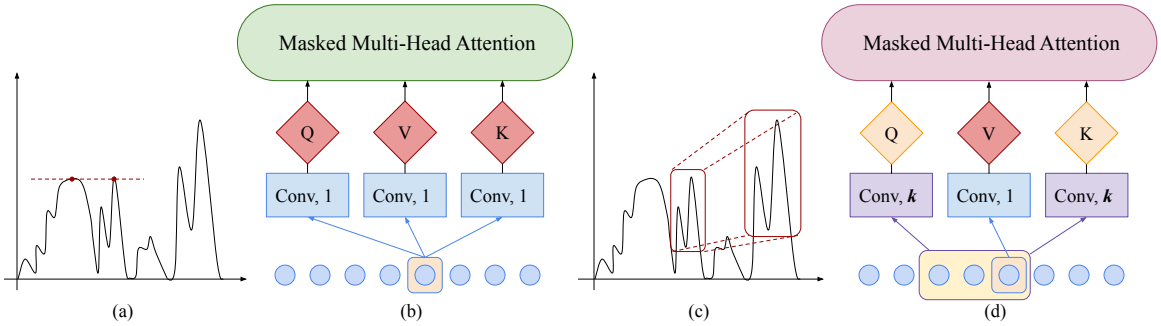


Figure 3.1: The comparison between canonical and our convolutional self-attention layers. "Conv, 1" and "Conv, k" mean convolution of kernel size $\{1, k\}$ with stride 1, respectively. Canonical self-attention as used in Transformer is shown in (b), may wrongly match point-wise inputs as shown in (a). Convolutional self-attention is shown in (d), which uses convolutional layers of kernel size k with stride 1 to transform inputs (with proper paddings) into queries/keys. Such locality awareness can correctly match the most relevant features based on shape matching in (c).

Patterns in time series may evolve with time significantly due to various events, e.g. holidays and extreme weather, so whether an observed point is an anomaly, change point or part of the patterns is highly dependent on its surrounding context. However, in the self-attention layers of canonical Transformer, the similarities between queries and keys are computed based on their point-wise values without fully leveraging local context like shape, as shown in Figure 3.1(a) and (b). Query-key matching agnostic of local context may confuse the self-attention module in terms of whether the observed value is an anomaly, change point or part of patterns, and bring underlying optimization issues.

We propose convolutional self-attention to ease the issue. The architectural view of proposed convolutional self-attention is illustrated in Figure 3.1(c) and (d). Rather than using convolution of kernel size 1 with stride 1 (matrix multiplication), we employ *causal convolution* of kernel size k with stride 1 to transform inputs (with proper paddings) into queries and keys. Note that causal convolutions ensure that the current position never has access to future information. By employing causal convolution, generated queries and

keys can be more aware of local context and hence, compute their similarities by their local context information, e.g. local shapes, instead of point-wise values, which can be helpful for accurate forecasting. Note that when $k = 1$, the convolutional self-attention will degrade to canonical self-attention, thus it can be seen as a generalization.

3.4 Breaking the memory bottleneck

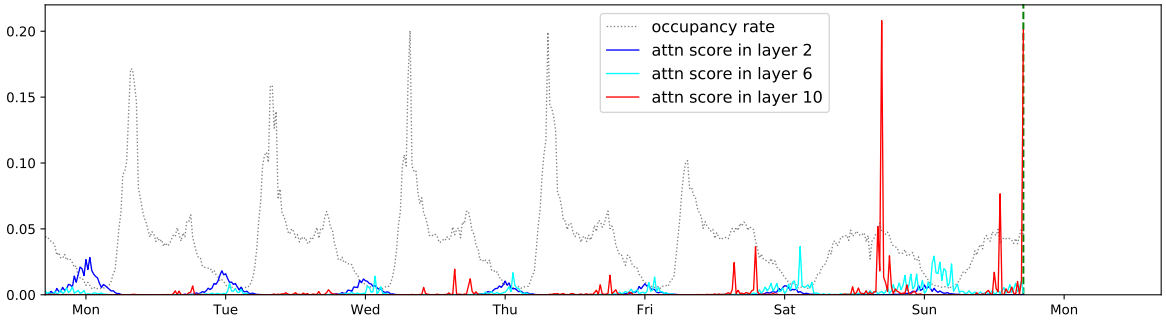


Figure 3.2: Learned attention patterns from a 10-layer canonical Transformer trained on `traffic-f` dataset with full attention. The green dashed line indicates the start time of forecasting and the gray dashed line on its left side is the conditional history. Blue, cyan and red lines correspond to attention patterns in layer 2, 6 and 10, respectively, for a head when predicting the value at the time corresponding to the green dashed line. a) Layer 2 tends to learn shared patterns in every day. b) Layer 6 focuses more on weekend patterns. c) Layer 10 further squeezes most of its attention on only several cells in weekends, causing most of the others to receive little attention.

To motivate our approach, we first perform a qualitative assessment of the learned attention patterns with a canonical Transformer on `traffic-f` dataset. The `traffic-f` dataset contains occupancy rates of 963 car lanes of San Francisco bay area recorded every 20 minutes. We trained a 10-layer canonical Transformer on `traffic-f` dataset with full attention and visualized the learned attention patterns. One example is shown in Figure 3.2. Layer 2 clearly exhibited global patterns, however, layer 6 and 10, only exhibited pattern-dependent sparsity, suggesting that some form of sparsity could be introduced without significantly affecting performance. More importantly, for a sequence with length

L, computing attention scores between every pair of cells will cause $O(L^2)$ memory usage, making modeling long time series with fine granularity and strong long-term dependencies prohibitive.

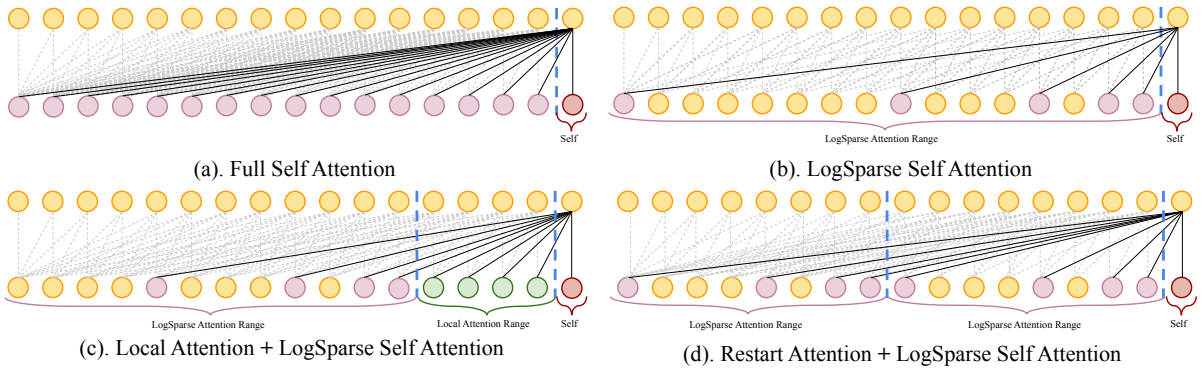
We propose *LogSparse* Transformer, which only needs to calculate $O(\log L)$ dot products for each cell in each layer. Further, we only need to stack up to $O(\log L)$ layers and the model will be able to access every cell's information. Hence, the total cost of memory usage is only $O(L(\log L)^2)$. We define I_l^k as the set of indices of the cells that cell l can attend to during the computation from k_{th} layer to $(k+1)_{th}$ layer. In the standard self-attention of Transformer, $I_l^k = \{j : j \leq l\}$, allowing every cell to attend to all its past cells and itself as shown in Figure 3.3(a). However, such an algorithm suffers from the quadratic space complexity growth along with the input length. To alleviate such an issue, we propose to select a subset of the indices $I_l^k \subset \{j : j \leq l\}$ so that $|I_l^k|$ does not grow too fast along with l. An effective way of choosing indices is $|I_l^k| \propto \log L$.

Notice that cell l is a weighted combination of cells indexed by I_l^k in kth self-attention layer and can pass the information of cells indexed by I_l^k to its followings in the next layer. Let S_l^k be the set which contains indices of all the cells whose information has passed to cell l up to k_{th} layer. To ensure that every cell receives the information from all its previous cells and itself, the number of stacked layers $\tilde{k}_l$ should satisfy that $S_l^{\tilde{k}_l} = \{j : j \leq l\}$ for $l = 1, \cdots, L$. That is, $\forall l$ and $j \leq l$, there is a directed path $P_{jl} = (j, p_1, p_2, \cdots, l)$ with $\tilde{k}_l$ edges, where $j \in I_{p_1}^1$, $p_1 \in I_{p_2}^2$, $\cdots$, $p_{\tilde{k}_l - 1} \in I_l^{\tilde{k}_l}$.

We propose *LogSparse* self-attention by allowing each cell only to attend to its previous cells with an exponential step size and itself. That is, $\forall k$ and l, $I_l^k = \{l - 2^{\lfloor \log_2 l \rfloor}, l - 2^{\lfloor \log_2 l \rfloor - 1}, l - 2^{\lfloor \log_2 l \rfloor - 2}, ..., l - 2^0, l\}$, where $\lfloor \cdot \rfloor$ denotes the floor operation, as shown in Figure 3.3(b).[5]

Theorem 1. *$\forall l$ and $j \leq l$, there is at least one path from cell j to cell l if we stack*

[5]Applying other bases is trivial so we don't discuss other bases here for simplicity and clarity.



(a). Full Self Attention (b). LogSparse Self Attention

(c). Local Attention + LogSparse Self Attention (d). Restart Attention + LogSparse Self Attention

Figure 3.3: Illustration of different attention mechanisms between adjacent layers in Transformer.

$\lfloor \log_2 l \rfloor + 1$ *layers. Moreover, for* $j < l$*, the number of feasible unique paths from cell* j *to cell* l *increases at a rate of* $O(\lfloor \log_2(l-j) \rfloor !)$.

Proof: According to the attention strategy in *LogSparse Transformer*, in each layer, cell l could attend to the cells with indices in $I_l^k = \{l - 2^{\lfloor \log_2 l \rfloor}, l - 2^{\lfloor \log_2 l \rfloor - 1}, l - 2^{\lfloor \log_2 l \rfloor - 2}, \cdots, l - 2^0, l\}$. To ensure that every cell receives the information from all its previous cells and itself, the number of stacked layers $\tilde{k}_l$ should satisfy that $S_l^{\tilde{k}_l} = \{j : j \leq l\}$ for $l = 1, \cdots, L$. That is, $\forall\ l$ and $j \leq l$, there is a directed path $P_{jl} = (j, p_1, p_2, \cdots, l)$ with $\tilde{k}_l$ edges, where $j \in I_{p_1}^1$, $p_1 \in I_{p_2}^2, \cdots, p_{\tilde{k}_l - 1} \in I_l^{\tilde{k}_l}$. We prove the theorem by constructing a path from cell j to cell l, with length (number of edges) no larger than $\lfloor \log_2 l \rfloor + 1$. Case $j = l$ is trivial, we only need to consider $j < l$ here. Consider the binary representation of $l - j$, $l - j = \sum_{m=0}^{\lfloor \log_2(l-j) \rfloor} b_m 2^m$, where $b_m \in \{0, 1\}$. Suppose $\{m_{sub}\}$ is the subsequence $\{m | 0 \leq m \leq \lfloor \log_2(l-j) \rfloor, b_m = 1\}$ and m_p is the p_{th} element of $\{m_{sub}\}$. A feasible path from j to l is $P_{jl} = \{j, j + 2^{m_0}, j + 2^{m_0} + 2^{m_1}, \cdots, l\}$. The length of this path is $|\{m_{sub}\}|$, which is no larger than $\lfloor \log_2(l-j) \rfloor + 1$. So

$$\min \{\tilde{k}_l | S_l^{\tilde{k}_l} = \{j : j \leq l\}\} \leq \max_{\{j | j < l\}} \lfloor \log_2(l-j) \rfloor + 1 \leq \lfloor \log_2 l \rfloor + 1. \tag{3.3}$$

Furthermore, by reordering $\{m_{sub}\}$, we can generate multiple different paths from cell j

to cell l. The number of feasible paths increases at a rate of $O(\lfloor \log_2(l-j) \rfloor !)$ along with l.

Theorem 1 implies that despite an exponential decrease in the memory usage (from $O(L^2)$ to $O(L \log_2 L)$) in each layer, the information could still flow from any cell to any other cell provided that we go slightly "deeper" — take the number of layers to be $\lfloor \log_2 L \rfloor + 1$. Note that this implies an overall memory usage of $O(L(\log_2 L)^2)$ and addresses the notorious scalability bottleneck of Transformer under GPU memory constraint. Moreover, as two cells become further apart, the number of paths increases at a rate of super-exponential in $\log_2(l-j)$, which indicates a rich information flow for modeling delicate long-term dependencies.

Local Attention We can allow each cell to densely attend to cells in its left window of size $O(\log_2 L)$ so that more local information, e.g. trend, can be leveraged for current step forecasting. Beyond the neighbor cells, we can resume our *LogSparse* attention strategy as shown in Figure 3.3(c).

Restart Attention Further, one can divide the whole input with length L into subsequences and set each subsequence length $L_{sub} \propto L$. For each of them, we apply the *LogSparse* attention strategy. One example is shown in Figure 3.3(d).

Employing *local attention* and *restart attention* won't change the complexity of our sparse attention strategy but will create more paths and decrease the required number of edges in the path. Note that one can combine local attention and restart attention together.

3.5 Experiments

3.5.1 Synthetic datasets

To demonstrate Transformer's capability to capture long-term dependencies, we conduct experiments on synthetic data. Specifically, we generate a piece-wise sinusoidal signals

$$f(x) = \begin{cases} A_1 \sin(\pi x/6) + 72 + N_x & x \in [0, 12), \\ A_2 \sin(\pi x/6) + 72 + N_x & x \in [12, 24), \\ A_3 \sin(\pi x/6) + 72 + N_x & x \in [24, t_0), \\ A_4 \sin(\pi x/12) + 72 + N_x & x \in [t_0, t_0 + 24), \end{cases}$$

where x is an integer, A_1, A_2, A_3 are randomly generated by uniform distribution on $[0, 60]$, $A_4 = \max(A_1, A_2)$ and $N_x \sim \mathcal{N}(0, 1)$. We aim to predict the last 24 steps given the previous t_0 data points. Intuitively, larger t_0 makes forecasting more difficult since the model is required to understand and remember the relation between A_1 and A_2 to make correct predictions after $t_0 - 24$ steps of irrelevant signals. Hence, we create 8 different datasets by varying the value of t_0 within $\{24, 48, 72, 96, 120, 144, 168, 192\}$. For each dataset, we generate 4.5K, 0.5K and 1K time series instances for training, validation and test set, respectively. An example time series with $t_0 = 96$ is shown in Figure 3.4(a).

In this experiment, we use a 3-layer canonical Transformer with standard self-attention. For comparison, we employ `DeepAR` [39], an autoregressive model based on a 3-layer LSTM, as our baseline. Besides, to examine if larger capacity could improve performance of `DeepAR`, we also gradually increase its hidden size h as $\{20, 40, 80, 140, 200\}$. Following [39, 144], we evaluate both methods using ρ-quantile loss R_ρ with $\rho \in (0, 1)$,

$$R_\rho(\mathbf{x}, \hat{\mathbf{x}}) = \frac{2 \sum_{i,t} D_\rho(x_t^{(i)}, \hat{x}_t^{(i)})}{\sum_{i,t} |x_t^{(i)}|}, \quad D_\rho(x, \hat{x}) = (\rho - \mathbf{I}_{\{x \le \hat{x}\}})(x - \hat{x}),$$

where $\hat{x}$ is the empirical ρ-quantile of the predictive distribution and $\mathbf{I}_{\{x \le \hat{x}\}}$ is an indicator

function.

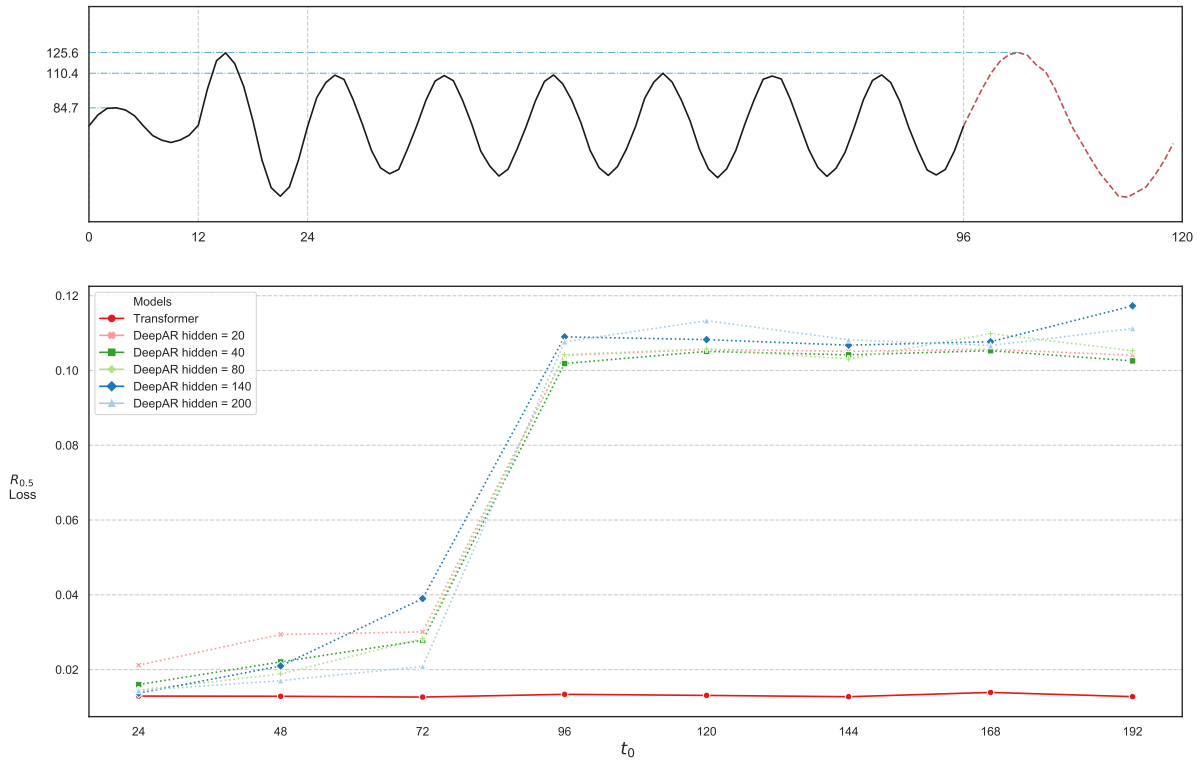


Figure 3.4: (a) An example time series with $t_0 = 96$. Black line is the conditional history while red dashed line is the target. (b) Performance comparison between `DeepAR` and canonical Transformer along with the growth of t_0. The larger t_0 is, the longer dependencies the models need to capture for accurate forecasting.

Figure 3.4(b) presents the performance of `DeepAR` and Transformer on the synthetic datasets. When $t_0 = 24$, both of them perform very well. But, as t_0 increases, especially when $t_0 \geq 96$, the performance of `DeepAR` drops significantly while Transformer keeps its accuracy, suggesting that Transformer can capture fairly long-term dependencies when LSTM fails to do so.

3.5.2 Real-world datasets

We further evaluate our model on several real-world datasets. The `electricity-f (fine)` dataset consists of electricity consumption of 370 customers recorded every 15

minutes and the `electricity-c (coarse)` dataset is the aggregated `electricity-f` by every 4 points, producing hourly electricity consumption. Similarly, the `traffic-f (fine)` dataset contains occupancy rates of 963 freeway in San Francisco recorded every 20 minutes and the `traffic-c (coarse)` contains hourly occupancy rates by averaging every 3 points in `traffic-f`. The `solar` dataset[6] contains the solar power production records from January to August in 2006, which is sampled every hour from 137 PV plants in Alabama. The `wind`[7] dataset contains daily estimates of 28 countries' energy potential from 1986 to 2015 as a percentage of a power plant's maximum output. The `M4-Hourly` contains 414 hourly time series from M4 competition.

Long-term and short-term forecasting We first show the effectiveness of canonical Transformer equipped with convolutional self-attention in long-term and short-term forecasting in `electricity-c` and `traffic-c` dataset. These two datasets exhibit both hourly and daily seasonal patterns. However, `traffic-c` demonstrates much greater difference between the patterns of weekdays and weekends compared to `electricity-c`. Hence, accurate forecasting in `traffic-c` dataset requires the model to capture both long- and short-term dependencies very well. As baselines, we use classical forecasting methods `auto.arima`, `ets` implemented in R's `forecast` package and the recent matrix factorization method `TRMF` [144], a RNN-based autoregressive model `DeepAR` and a RNN-based state space model `DeepState` [101]. For short-term forecasting, we evaluate rolling-day forecasts for seven days (i.e., prediction horizon is one day and forecasts start time is shifted by one day after evaluating the prediction for the current day). For long-term forecasting, we directly forecast 7 days ahead. As shown in Table 3.1, our models with convolutional self-attention get betters results in both long-term and short-term forecasting, especially in `traffic-c` dataset compared to strong baselines, partly due

[6] https://www.nrel.gov/grid/solar-power-data.html
[7] https://www.kaggle.com/sohier/30-years-of-european-wind-generation

	ARIMA	ETS	TRMF	DeepAR	DeepState	Ours
e-c$_{1d}$	0.154/0.102	0.101/0.077	0.084/-	0.075$^\diamond$/0.040$^\diamond$	0.083$^\diamond$/0.056$^\diamond$	**0.059/0.034**
e-c$_{7d}$	0.283$^\diamond$/0.109$^\diamond$	0.121$^\diamond$/0.101$^\diamond$	0.087/-	0.082/0.053	0.085$^\diamond$/0.052$^\diamond$	**0.070/0.044**
t-c$_{1d}$	0.223/0.137	0.236/0.148	0.186/-	0.161$^\diamond$/0.099$^\diamond$	0.167$^\diamond$/0.113$^\diamond$	**0.122/0.081**
t-c$_{7d}$	0.492$^\diamond$/0.280$^\diamond$	0.509$^\diamond$/0.529$^\diamond$	0.202/-	0.179/0.105	0.168$^\diamond$/0.114$^\diamond$	**0.139/0.094**

Table 3.1: Results summary ($R_{0.5}$/$R_{0.9}$-loss) of all methods. `e-c` and `t-c` represent `electricity-c` and `traffic-c`, respectively. In the 1st and 3rd row, we perform rolling-day prediction of 7 days while in the 2nd and 4th row, we directly forecast 7 days ahead. `TRMF` outputs points predictions, so we only report $R_{0.5}$. $\diamond$ denotes results from [101]

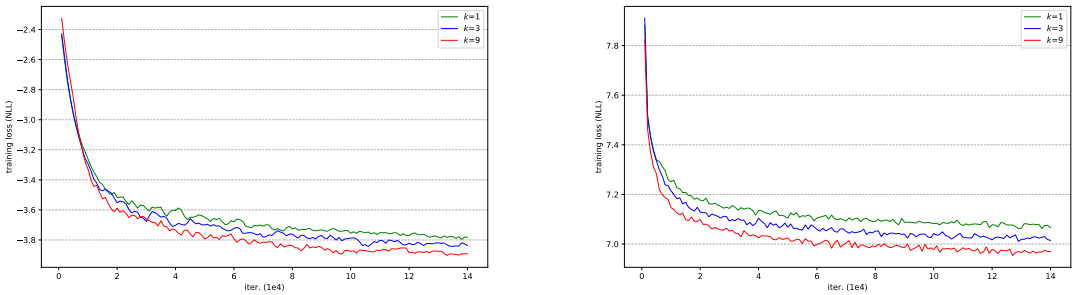


Figure 3.5: Training curve comparison (with proper smoothing) among kernel size $k \in \{1, 3, 9\}$ in `traffic-c` (**left**) and `electricity-c` (**right**) dataset. Being aware of larger local context size, the model can achieve lower training error and converge faster.

to the long-term dependency modeling ability of Transformer as shown in our synthetic data.

Convolutional self-attention In this experiment, we conduct ablation study of our proposed convolutional self-attention. We explore different kernel size $k \in \{1, 2, 3, 6, 9\}$ on the full attention model and fix all other settings. We still use rolling-day prediction for seven days on `electricity-c` and `traffic-c` datasets. The results of different kernel sizes on both datasets are shown in Table 3.2. On `electricity-c` dataset, models with kernel size $k \in \{2, 3, 6, 9\}$ obtain slightly better results in term of $R_{0.5}$ than canonical Transformer but overall these results are comparable and all of them perform very well. We argue it is because `electricity-c` dataset is less challenging and covariate vectors

	$k=1$	$k=2$	$k=3$	$k=6$	$k=9$
`electricity-c`$_{1d}$	0.060/**0.030**	0.058/**0.030**	**0.057**/0.031	**0.057**/0.031	0.059/0.034
`traffic-c`$_{1d}$	0.134/0.089	0.124/0.085	0.123/0.083	0.123/0.083	**0.122**/**0.081**

Table 3.2: Average $R_{0.5}/R_{0.9}$-loss of different kernel sizes for rolling-day prediction of 7 days.

have already provided models with rich information for accurate forecasting. Hence, being aware of larger local context may not help a lot in such cases. However, on much more challenging `traffic-c` dataset, the model with larger kernel size k can make more accurate forecasting than models with smaller ones with as large as 9% relative improvement. These consistent gains can be the results of more accurate query-key matching by being aware of more local context. Further, to verify if incorporating more local context into query-key matching can ease the training, we plot the training loss of kernel size $k \in \{1, 3, 9\}$ in `electricity-c` and `traffic-c` datasets. We found that Transformer with convolutional self-attention also converged faster and to lower training errors, as shown in Figure 3.5, proving that being aware of local context can ease the training process.

Sparse attention Further, we compare our proposed *LogSparse* Transformer to the full attention counterpart on fine-grained datasets, `electricity-f` and `traffic-f`. Note that time series in these two datasets have much longer periods and are noisier comparing to `electricity-c` and `traffic-c`. We first compare them under the same memory budget. For `electricity-f` dataset, we choose $L_{e_1} = 768$ with subsequence length $L_{e_1}/8$ and local attention length $\log_2(L_{e_1}/8)$ in each subsequence for our sparse attention model and $L_{e_2} = 293$ in the full attention counterpart. For `traffic-f` dataset, we select $L_{t_1} = 576$ with subsequence length $L_{t_1}/8$ and local attention length $\log_2(L_{t_1}/8)$ in each subsequence for our sparse attention model, and $L_{t_2} = 254$ in the full attention counterpart.

Constraint	Dataset	Full	Sparse	Full + Conv	Sparse + Conv
Memory	`electricity-f`$_{1d}$	0.083/0.051	0.084/**0.047**	**0.078**/0.048	0.079/0.049
	`traffic-f`$_{1d}$	0.161/0.109	0.150/0.098	0.149/0.102	**0.138/0.092**
Length	`electricity-f`$_{1d}$	0.082/0.047	0.084/0.047	**0.074/0.042**	0.079/0.049
	`traffic-f`$_{1d}$	0.147/0.096	0.150/0.098	0.139/**0.090**	**0.138**/0.092

Table 3.3: Average $R_{0.5}/R_{0.9}$-loss comparisons between sparse attention and full attention models with/without convolutional self-attention by rolling-day prediction of 7 days. "Full" means models are trained with full attention while "Sparse" means they are trained with our sparse attention strategy. "+ Conv" means models are equipped with convolutional self-attention with kernel size $k = 6$.

We conduct experiments on aforementioned sparse and full attention models with or without convolutional self-attention on both datasets. By following such settings, we summarize our results in Table 3.3 (Upper part). No matter equipped with convolutional self-attention or not, our sparse attention models achieve comparable results on `electricity-f` but much better results on `traffic-f` compared to its full attention counterparts. Such performance gain on `traffic-f` could be the result of the dateset's stronger long-term dependencies and our sparse model's better capability of capturing these dependencies, which, under the same memory budget, the full attention model cannot match. In addition, both sparse and full attention models benefit from convolutional self-attention on challenging `traffic-f`, proving its effectiveness.

To explore how well our sparse attention model performs compared to full attention model with the same input length, we set $L_{e_2} = L_{e_1} = 768$ and $L_{t_2} = L_{t_1} = 576$ on `electricity-f` and `traffic-f`, respectively. The results of their comparisons are summarized in Table 3.3 (Lower part). As one expects, full attention Transformers can outperform our sparse attention counterparts no matter they are equipped with convolutional self-attention or not in most cases. However, on `traffic-f` dataset with strong long-term dependencies, our sparse Transformer with convolutional self-attention can get better results than the canonical one and, more interestingly, even slightly outperform

its full attention counterpart in term of $R_{0.5}$, meaning that our sparse model with convolutional self-attention can capture long-term dependencies fairly well. In addition, full attention models under length constraint consistently obtain gains from convolutional self-attention on both `electricity-f` and `traffic-f` datasets, showing its effectiveness again.

Further Exploration In our last experiment, we evaluate how our methods perform on datasets with various granularities compared to our baselines. All datasets except `M4-Hourly` are evaluated by rolling window 7 times since the test set of `M4-Hourly` has been provided. The results are shown in Table 3.4. These results further show that our method achieves the best performance overall.

	`electricity-f`$_{1d}$	`traffic-f`$_{1d}$	`solar`$_{1d}$	`M4-Hourly`$_{2d}$	`wind`$_{30d}$
`TRMF`	0.094/-	0.213/-	0.241/-	-/-	0.311/-
`DeepAR`	0.082/0.063	0.230/0.150	0.222/0.093	0.090$^\diamond$/0.030$^\diamond$	0.286/0.116
Ours	**0.074/0.042**	**0.139/0.090**	**0.210 /0.082**	**0.067 /0.025**	**0.284/0.108**

Table 3.4: $R_{0.5}/R_{0.9}$-loss of datasets with various granularities. The subscript of each dataset presents the forecasting horizon (days). `TRMF` is not applicable for `M4-Hourly`$_{2d}$ and we leave it blank. For other datasets, `TRMF` outputs points predictions, so we only report $R_{0.5}$. $^\diamond$ denotes results from [10].

3.6 Conclusion

We propose to apply Transformer in time series forecasting. Our experiments on both synthetic data and real datasets suggest that Transformer can capture long-term dependencies while LSTM may suffer. We also showed, on real-world datasets, that the proposed convolutional self-attention further improves Transformer' performance and achieves state-of-the-art in different settings in comparison with recent RNN-based methods, a matrix factorization method, as well as classic statistical approaches. In addition,

with the same memory budget, our sparse attention models can achieve better results on data with long-term dependencies.

Chapter 4

Attention Guided Autoregression

4.1 Introduction

While abundant prior works have proposed various solutions to make accurate predictions for time series via statistical and machine learning approaches, they generally encode all available historical information into a fixed-size vector that is later fed into a decoder module. When there is an abrupt change taking place within the observed time interval, the unwanted information before the change can be misleading in prediction. For example, RNNs make predictions based on a hidden state that tries to memorize all inputs up to the current time step. The information prior to a change point are kept in the hidden state until sufficient new data points are observed, even if the model is equipped with gating mechanisms [56, 25] that are able to control the information flow for better memory management.

To detect such changes in the temporal evolution of a system, considerable attention has been paid from machine learning and data mining community [11, 106, 17, 77, 22]. A detection algorithms aims to find a point such that the respective distributions of the observations before and after it are significantly different. A new forecaster will

then be trained using the observations after the change point only, based on the natural assumption that the changed pattern will be preserved in the future [5]. However, the criterion of unsupervised change point detectors is highly task-dependent, and supervised methods require expensive human-annotated labels. On the other hand, the detector has to search the change point step by step, and the composite approach suffers from error cascading issue. In fact, we do not have to find the exact change point to make good predictions, as long as we can identify observations subject to new patterns and exclude observations before them, as shown in Figure 4.1.

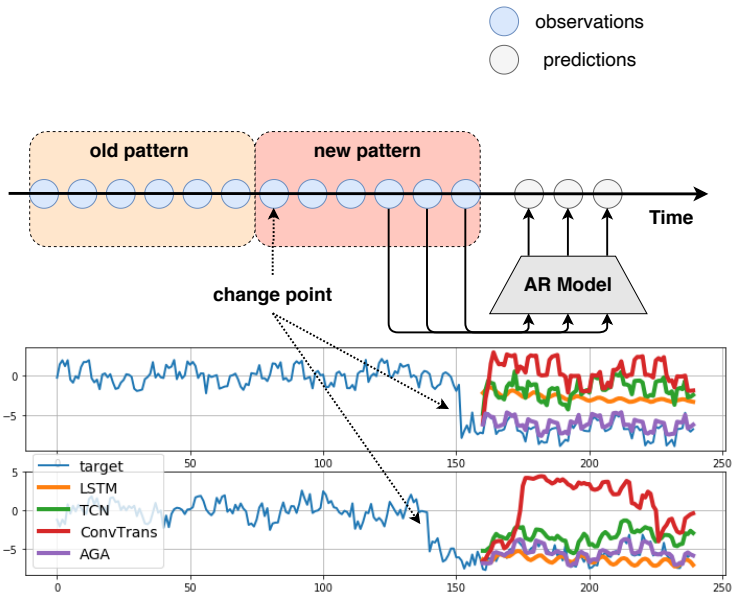


Figure 4.1: The top panel depicts the concept of Localized autoregressive forecasting in case of change points. Even though there are 6 observations after the change points, the AR model may encode the last 3 samples only and ignore all previous time steps when predict the next 3 steps. The bottom panel shows a simulation study as an example. We try to forecast noisy cyclic signals with randomly injected shifts as abrupt changes. Various deep autoregressive models are trained and evaluated. Two test samples in which the forecasts start shortly after a change point are shown. While most baselines manage to gradually adapt to the new patterns after observing the changes, AGA can address the changes more quickly thanks to the localized AR component.

Self-attention based networks, also known as Transformers [122], have been a prefer-

able choice in many sequence modeling tasks, especially in language modeling [33, 140]. Transformers locate the observations that are most informative for current predictions via pairwise comparison between an input variable and all historical encodings. On top of that, multi-head attention mechanism is able to focus on multiple subspaces of hidden activations for better comparison, which resembles the concept of feature maps in convolution neural nets [72]. Some previous works apply it to time series analysis and get promising results [113, 76, 80], but they did not consider the challenge imposed by change points. Nevertheless, the flexibility of transformers in selecting observations makes it an ideal choice to combat change points in forecasting tasks.

We propose a novel self-attention architecture to address the aforementioned issues. The idea is to first encode observations and other auxiliary covariates with local processing operations to obtain temporal contextual information at each time step. Then we use self-attention to find historical time points at which temporal contextual information is similar to that of the current time step and scale the observed values at those points accordingly as prediction. In case no such time points exist, it's likely that a potential change point has occurred and we use a relatively simple and localized autoregressive module to produce a "backup" estimate. The two estimates are dynamically fused with self-attention. Finally we employ a fully-connected network to emit predictions and use quantile regression [67] to train the model.

4.2 Related Work

Self-attention Attention mechanism was firstly introduced in [6, 83] for machine translation task. [122] proposed a architecture named Transformer based solely on multi-head self-attention which became a game changer in the domain. Later on, self-attention was applied to language modeling and achieved impressive success in many tasks [33, 141,

142, 19]. While most variants of Transformer takes a single input, it has been shown that improvements in many aspects can be obtained by manipulating the inputs. For example, [81] compress the length of sequence with a convolution layer to mitigate the memory bottleneck of Transformer. Notably, [141] introduced another self-attention operation to maintain an independent contextual representations, which is different from the standard self-attention that is aware of the target content. Our model follows this idea and transforms raw inputs into two sets of representations and maintain them separately.

Change Point Detection Change point detection (CPD) aims to find critical times when important events have occurred and caused significant changes in observed variables. Classical parametric approaches for change point detection (CPD) usually rely on strong assumption on data distributions [11, 137, 1]. Non-parametric methods are free of such assumptions but depend heavily on carefully engineered divergence metrics or kernel functions [106, 77]. Recently deep learning methods are introduced for better detection accuracy [37, 22]. However, these methods still rely on manually annotated labels and have to perform a binary classification per observation, which is usually expensive and unnecessary in forecasting non-stationary time series. Moreover, the imperfect detectors introduce extra error for the downstream forecasting task. Hence we focus on forecasting only and implicitly detect changes within self-attention.

4.3 Problem setup

Given a time series dataset with N unique instances, such as different stores for demand forecasting or households in energy consumption analysis. Each instance i consists of a sequence of observations $\boldsymbol{x}_{i,0:T} \triangleq \{\boldsymbol{x}_{i,0}, \boldsymbol{x}_{i,1}, \cdots, \boldsymbol{x}_{i,T}\}$, where the t-th observation is $\boldsymbol{x}_{i,t} \in \mathbb{R}^{d_x}$, a static covariate $\boldsymbol{s}_i \in \mathbb{R}^s$ that is time-invariant, as well as a series of

time-varying covariates $\boldsymbol{a}_{i,0:T+\tau}$ where $\boldsymbol{a}_{i,t} \in \mathbb{R}^{d_a}$, such as date/time information. Note that the auxiliary inputs $\boldsymbol{a}_{0:T+\tau}$ are known within the forecasting horizon as well. Our goal is to predict the next τ time steps in the future, i.e. $\boldsymbol{x}_{i,T+1:T+\tau}$. In the following sections, we will call $\boldsymbol{x}_{0:T}$ the condition and τ the forecasting horizon. Formally, we want to model a conditional distribution $p(\boldsymbol{x}_{T+1:T+\tau}|\boldsymbol{x}_{0:T}, \boldsymbol{a}_{0:T+\tau})$.

An autoregressive model tackles this problem by decomposing the distribution according to the chain rule, i.e.

$$p(\boldsymbol{x}_{i,T+1:T+\tau}|\boldsymbol{x}_{i,0:T}) = \prod_{t=T+1}^{T+\tau} p(\boldsymbol{x}_{i,t}|\boldsymbol{x}_{i,0:t-1}, \boldsymbol{a}_{i,0:t}, \boldsymbol{s}_i; \boldsymbol{\theta}). \tag{4.1}$$

This reduces the problem to learning a single-step-ahead predictor $f(\boldsymbol{x}_{i,0:t-1}, \boldsymbol{a}_{i,0:t}, \boldsymbol{s}_i; \boldsymbol{\theta})$ and iteratively applying it τ times by feeding the prediction $\hat{\boldsymbol{x}}_{i,t}$ back into the model. In our experiments, we simply concatenate $\boldsymbol{s}_i$ with each $\boldsymbol{a}_{i,t}$. We still use $\boldsymbol{a}_{i,t}$ to denote the concatenated vector by slightly abusing notations, and omit the subscript i for simplicity throughout the paper if not stated explicitly. Here $\boldsymbol{\theta}$ denotes the set of trainable parameters of the model that are shared across and learned jointly from all N instances.

For probabilistic forecasting, it is usually desired to quantify the uncertainty of predictions. Previous works [39, 101] assume $p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1})$ to be Gaussian and explicitly output the maximum likelihood estimate of its mean and variance. Instead, when the provision for prediction intervals is specified, we can directly predict the required intervals via quantile regression [67], e.g. outputting the 10^{th}, 50^{th} and 90^{th} percentiles of $p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1})$ [129, 38, 80], so that we do not need to make any assumption on the data distribution.

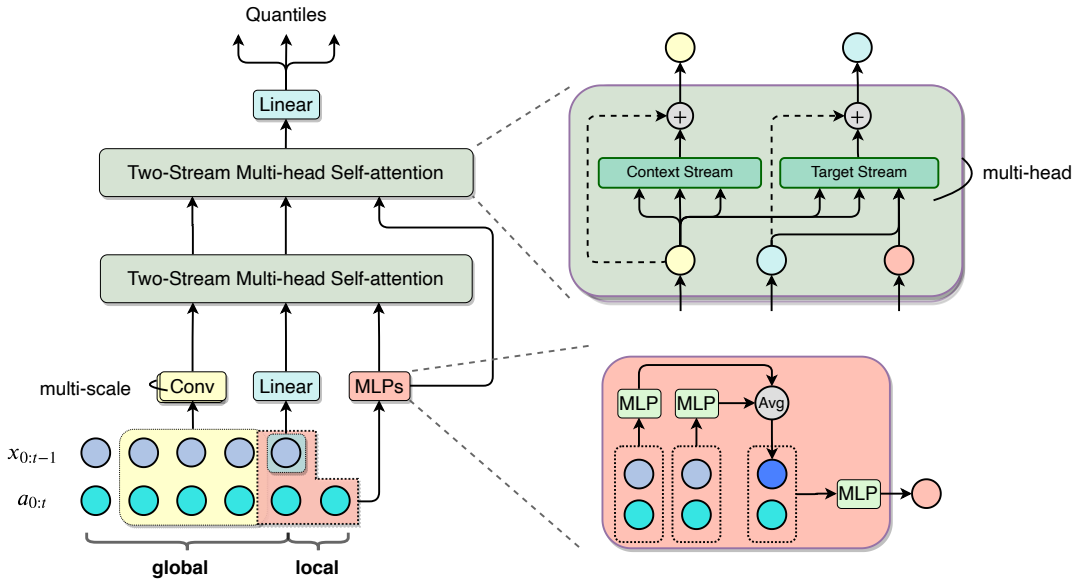


Figure 4.2: The overall architecture of AGA. Best viewed in color.

4.4 Methodology

4.4.1 Basic Multi-head Self-attention Network

We have introduced self-attention mechanism in Chapter 3, and here we briefly review its multi-head version [122]. Formally, suppose we have a sequence of values with length L which is packed into a matrix $\boldsymbol{V}$, and similarly a query sequence $\boldsymbol{Q}$ and a key sequence $\boldsymbol{K}$, then

$$\begin{aligned} \text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= A(\boldsymbol{Q}, \boldsymbol{K})\boldsymbol{V} \\ A(\boldsymbol{Q}, \boldsymbol{K}) &= \text{Softmax}(\boldsymbol{Q}\boldsymbol{K}^T/\sqrt{d_s}) \end{aligned} \tag{4.2}$$

Instead of a single attention function, the self-attention can be enriched by producing H versions of queries, keys and values via different linear projections, which is usually called "heads". Then we apply attention function to each of these H heads and obtain H different outputs. They are concatenated and linearly projected again, resulting in

the final output.

$$\begin{aligned} \text{MHAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}(\boldsymbol{A}_1, \cdots, \boldsymbol{A}_H)\boldsymbol{W}_O \\ \boldsymbol{A}_i &= \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_Q^{(i)}, \boldsymbol{K}\boldsymbol{W}_K^{(i)}, \boldsymbol{V}\boldsymbol{W}_V^{(i)}) \end{aligned} \tag{4.3}$$

Here $\boldsymbol{W}_Q, \boldsymbol{W}_K$ and $\boldsymbol{W}_V$ are parameters to be learned, and they are usually compressing projections so that different heads attend to information from different hidden subspaces at each time step. In addition, to make the function autoregressive, attention from a query at step i to a key at step j is prohibited when $i < j$.

4.4.2 Motivation

Single-step-ahead forecasting basically aims at finding a mapping from a number of observed measures or events to an estimate of unknown future. While non-stationary time series are generally not predictable [63], a plausible prediction can be made if the observed patterns repeat themselves. Intuitively, if we currently observe a temporal pattern that has been observed in the history after which an event immediately took place, we expect that a similar event is likely to occur again at this moment. In other words, to measure the relation between two time points, their surrounding observations are often considered. Self-attention mechanism exactly follows this intuition and actively compares the contextual patterns at two arbitrary time steps.

In light of this, we are supposed to maintain a temporal-context encoding (context encoding in short) $\boldsymbol{c}_t$ to capture such patterns prior to each time step t for comparison in attention. Simultaneously, we expect a prediction-target encoding (target encoding) $\boldsymbol{p}_t$ that contains the event or measure that we want to predict, i.e. the ground-truth $\boldsymbol{x}_t$. The context encodings will then be queries and keys in attention, and the target encodings will play the role of values. On the other hand, if no recurring patterns are found within observations, then another predictor $\hat{\boldsymbol{p}}_t$ should be introduced in place of target encodings.

In the next two subsections, we propose the formulation of these components, and the left panel of figure 4.2 illustrates the overall architecture of our design.

4.4.3 Context and Target Encoding

Since the context encodings $\boldsymbol{c}_t$ are expected to encode shapes and variations before t. Hence we use convolution to extract local patterns, following [76]:

$$\boldsymbol{c}_t = \tanh(\text{Conv}_k(\boldsymbol{x}_{t-k:t-1}, \boldsymbol{a}_{t-k:t-1})) \tag{4.4}$$

where k is the filter size of convolution. Note that the input is padded with zero on the left side so that the output will have the same shape as the input and contain no information about future. Meanwhile, we compute the target encodings via a simple step-wise linear layer so that it contains $\boldsymbol{x}_t$:

$$\boldsymbol{p}_t = \tanh(\text{Linear}(\boldsymbol{x}_t)) \tag{4.5}$$

4.4.4 Global-Local Autoregressive Structure

We then set $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{C} \triangleq [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_t]$ as part of inputs to self-attention layers, so that the attention will compare the temporal contextual patterns extracted from the local convolution layer. For example, suppose we are predicting time step t, then $\boldsymbol{c}_t$ attends to $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_{t-1}$, as well as itself. In this way, when there is a similar context pattern $\boldsymbol{c}_i$ found, then the corresponding target encoding $\boldsymbol{p}_i$ is likely to be from an observation $\boldsymbol{x}_i$ that is close to the ground-truth $\boldsymbol{x}_t$, e.g. periods in cyclical time series. In this case, $\boldsymbol{p}_i$ will receive a large attention weight in composing the attention output. Note that here $i < t$ since we know neither the ground-truth $\boldsymbol{x}_t$ nor its target encoding $\boldsymbol{p}_t$. We call this component "global" since it attends to all available history. If no such similar patterns are found, it is likely that a potential change occurs, and attention weight will

concentrate on the current step since the compared context encodings are exactly the same. In this case, it is safer to rely on recent observations which are more likely to belong to the new patterns.

We propose to use a simpler component to generate a localized autoregressive estimate of the unknown $\boldsymbol{p}_t$. Specifically, we use a step-wise MLP to encode both the observations and covariates at step $\{t-u, t-u+1, \cdots, t-1\}$, where $u \geq 1$ is a small integer, usually less than 3. The encodings are then averaged and fed into another MLP together with $\boldsymbol{a}_t$ to decode the esitmate $\hat{\boldsymbol{p}}_t$.

$$\begin{aligned} \boldsymbol{h}_i &= \text{MLP}(\boldsymbol{x}_i, \boldsymbol{a}_i), \\ \hat{\boldsymbol{h}} &= \frac{1}{u}\sum_{j=1}^{u} \boldsymbol{h}_{t-j}, \\ \hat{\boldsymbol{p}}_t &= \text{MLP}(\hat{\boldsymbol{h}}, \boldsymbol{a}_t) \end{aligned} \tag{4.6}$$

In a sense, we expect the decoder to fit a regression function from $\boldsymbol{a}_t$ to $\boldsymbol{p}_t$, which is dependent on recent patterns encoded by the encoder. This architecture resembles that of Conditional Neural Processes [46], in which an accurate predictor can be learned from a handful of training data points. The right bottom panel of figure 4.2 shows the local autoregressive structure.

Provided with both the global and localized estimates, we assemble both components into the framework of self-attention. We obtain the value matrix $\boldsymbol{V} = \boldsymbol{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_{t-1}, \hat{\boldsymbol{p}}_t]$ by concatenation, and $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are then fed into multi-head self-attention network to aggregate both estimates.

Distance encoding Self-attention has no clue about the temporal order of inputs and thus breaks the temporal dependencies. To recover the temporal order in non-recurrent structures, [122, 47] proposed to use sinusoidal functions or learned embeddings to encode absolute position as additional input to the network. In our setting, the absolute

positional information can be reconstructed from the given date/time information in covariates. A more flexible alternative is to encode relative distances between the compared pair of positions [110, 29]. Hence we introduce a sinusoidal distance encoding $\boldsymbol{R}$, in which the d^{th} row $\boldsymbol{r}_d \in \mathbb{R}^{d_s}$ corresponds to distance d. We adopt the sinusoidal formulation for its generalizability to arbitrarily long time series. Formally, we modify equation (4.2) as follows:

$$\begin{aligned} \hat{\boldsymbol{r}}_{i-j} &= \boldsymbol{W}_R \boldsymbol{r}_{i-j} \\ \hat{A}_{i,j}(\boldsymbol{Q}, \boldsymbol{K}) &= \boldsymbol{q}_i^T \boldsymbol{k}_j + \hat{\boldsymbol{r}}_{i-j}^T \boldsymbol{q}_i + \hat{\boldsymbol{r}}_{i-j}^T \boldsymbol{k}_j \\ A &= \text{Softmax}(\hat{A}) \end{aligned} \tag{4.7}$$

where $\hat{A}_{i,j}$ is the $(i,j)^{th}$ entry of pre-normalized attention weight matrix; $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ are rows of $\boldsymbol{Q}$ and $\boldsymbol{K}$, respectively. In essence we add a query-dependent and a key-dependent bias term to the original attention that are both dependent on the distance $i-j$. This formulation is similar to the relative positional encoding employed in [29], but they differ essentially in how to connect keys and distance encodings. While [29] used an additive combination of keys and relative position encodings, we use a dot-product function instead that does not introduce extra parameters. Empirically we find that this formulation performs as well as the additive one.

Multi-scale context encodings Preliminary experiments in [76] shows that the filter sizes of convolutions that enhances locality can be highly data-dependent and have significant impact on the forecasting performance. A practical problem is then to choose the appropriate kernel size k. Intuitively, larger filters are more reliable in pattern matching as they encode more contexts, but smaller kernels usually focus on finer details that might be more important in certain cases, e.g. outliers and changes.

Since multi-head self-attention itself combines information gathered from multiple tries of attention, we propose to divide attention heads into a number of groups, each

receives convolutional output with different filter sizes. In other words, we employ G convolution layers equation (4.4) with various filter sizes $k_1, k_2, \cdots, k_G$. The number of attention heads H is selected to be divisible by G so that the heads can be evenly divided into G groups. Each group i has H/G heads and takes the output of the convolution layer with filter size k_i as input. Moreover, inside the self-attention operation, the linear projections that produce queries and keys will also be restricted so that only intra-group interactions take place. This can be achieved by using block-diagonal weight matrices, i.e. $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$. In this way, these groups compute distinct attention scores and aggregate values based on temporal patterns at various scales, and the results are eventually assembled with the output projection. This design improves the robustness of attention in case of subtle variations that a single attention may fail to recognize. And more importantly, the model is less likely to detect a false change point due to limited receptive field of context encodings.

4.4.5 Two-Stream Self-Attention

As shown in equation (4.4)equation (4.5), at each time step t, the context encoding $\boldsymbol{c}_t$ uses the observations before $\boldsymbol{x}_t$ yet the target encoding $\boldsymbol{p}_t$ contains $\boldsymbol{x}_t$. However, a standard Transformer fuses all input into a single output, which makes the following self-attention layers inconsistent with the first one computationally and mix up two encodings conceptually. To resolve such a contradiction, we borrow the idea of two-stream self-attention from [141] that maintains the two types of representations separately. To be specific, we obtain the initial encodings $\boldsymbol{c}_t^{(0)}, \boldsymbol{p}_t^{(0)}$ according to equation (4.4) equation (4.5). Then each self-attention layer $m \in [1, 2, \cdots, M]$ performs two attention

operations equation (4.3) with shared parameters:

$$\boldsymbol{c}_t^{(m)} = \text{MHAttention}(\boldsymbol{c}_t^{(m-1)}, \boldsymbol{C}_{0:t}^{(m-1)}, \boldsymbol{C}_{0:t}^{(m-1)}), \tag{4.8}$$

$$\boldsymbol{p}_t^{(m)} = \text{MHAttention}(\boldsymbol{c}_t^{(m-1)}, \boldsymbol{C}_{0:t}^{(m-1)}, \boldsymbol{P}_{0:t}^{(m-1)}), \tag{4.9}$$

We call equation (4.8) context stream and equation (4.9) target stream. The target stream is the structure covered in Section 4.4.4. The context stream is a typical self-attention in which keys and values are tied, such that the updated context encodings summarize all $\boldsymbol{c}_{1:t}$ instead of a restricted convolutional representation. On the other hand, a skip connection keeps context encodings locality-aware by adding lower-level features to upper-level. Note that we use the local estimate from equation (4.6) in all subsequent layers, i.e. $\boldsymbol{p}_t^{(m-1)}$ is always replaced by $\hat{\boldsymbol{p}}_t$, so that it stays localized. The top right panel of figure 4.2 summarizes the concept of two-stream self-attention.

Despite being conceptually similar to [141], our design keeps both encodings distinct throughout all attention layers in terms of encoded values, so they are not likely to be mixed up. In addition, we do not need to use different masks in two streams of attention, which simplifies implementation.

4.4.6 Training and Evaluation

The target encoding from the last self-attention layer $\boldsymbol{p}_t^{(M)}$ will be fed into a fully-connected layer that emits quantile predictions of interest in parallel. For example, we are predicting the 10^{th}, 50^{th}, 90^{th} percentiles at each time step, then

$$[\hat{\boldsymbol{x}}_{t,10}; \hat{\boldsymbol{x}}_{t,50}; \hat{\boldsymbol{x}}_{t,90}] = \boldsymbol{W}_P \boldsymbol{p}_t^{(M)} + \boldsymbol{b}_P \tag{4.10}$$

where $\boldsymbol{W}_P \in \mathbb{R}^{3d_x \times d_V}$ and $\boldsymbol{b}_P \in \mathbb{R}^{3d_x}$ are linear coefficients.

As per [129], the AGA is trained by minimizing cumulative the quantile loss across

all predictions, horizons and time series instances

$$L(\mathrm{P}, \boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{t=1}^{\tau} \sum_{\rho \in \mathrm{P}} D(\boldsymbol{x}_{i,T+t}, \hat{\boldsymbol{x}}_{i,T+t,\rho}) \\ D(\boldsymbol{x}, \hat{\boldsymbol{x}}_\rho) = (\rho - \boldsymbol{I}_{[x \leq \hat{x}]})(x - \hat{x}_\rho) \tag{4.11}$$

where P represents required quantiles, and $\boldsymbol{I}_{[\cdot]}$ is an indicator function. We evaluate the performance in terms of normalized ρ-quantile risks R_ρ.

$$R_\rho(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{2 \sum_{i,t} D(\boldsymbol{x}_{i,t}, \hat{\boldsymbol{x}}_{i,t,\rho})}{\sum_{i,t} |\boldsymbol{x}_{i,t}|} \tag{4.12}$$

4.5 Experiments

4.5.1 Simulation Study

To verify the motivation of dealing with change points in forecasting task, we conduct experiments on a synthetic dataset and compare AGA with other deep autoregressive models, namely DeepAR, TCN and ConvTrans. Specifically, we generate noisy cyclic signals consisting of a sinusoidal component, a square wave and a triangle wave, corrupted by white noise with standard variance 0.2. They have a fixed length 240 with period 40 and 8. Afterwards, we randomly select a time point λ from $[0, 160)$ and a shift $s \sim \mathrm{Unif}(3, 10)$. Then the shift is added to or subtracted from x_t for all $t \in (\lambda, 240)$ according to coin toss. We generate 5000 such signals and 5000 normal sinusoids without shifts to make a dataset, on which we train the mentioned AR models. We then evaluate the forecasting performance on an independent test set with 1000 instances generated in the same way. During training and evaluation, the known periodicity is revealed to the models as covariates.

As a result, we observe that the forecasting accuracy of all models are influenced by the injected shifts when they occur. Meanwhile, the models are able to adapt to the

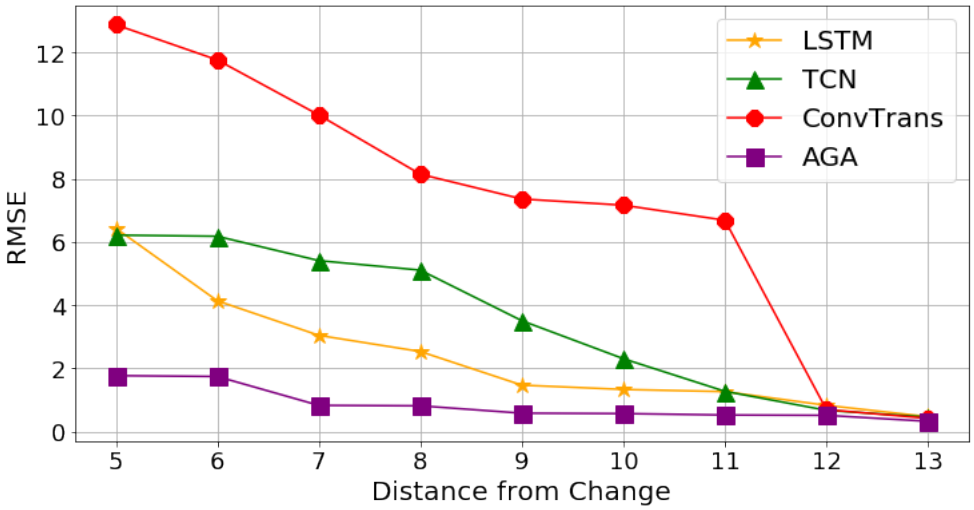


Figure 4.3: RMSE of forecasts against the distance from the start of forecasts to the time point where a random shift takes place. All models are able to make reasonable predictions after sufficient data from new patterns is observed, but AGA does the best.

changed patterns after observing a few time steps. However, it takes different amount of time for each model to accomodate the changes. Figure 4.3 illustrates the averaged root mean square error (RMSE) depending on the elapsed time from the occurrence of a shift to the start of forecast. It shows that AGA is able to respond to the changes much more quickly than baselines. On the other hand, TCN and ConvTrans are more likely to be misled by the patterns before the changes, due to their reliability on all historical observations. Some examples are shown in figure 4.1 from a qualitative perspective.

4.5.2 Quantitative Evaluation

Dataset

We then evaluate our model on a variety of publicly available real-world datasets across a wide range of domains. First we use two benchmark datasets that present clear seasonality. Then we conduct experiments on two datasets with highly non-stationary time series, one of which is large and the other is small but has labelled anomalies or

change points.

Electricity The *electricity* dataset contains hourly energy comsumption of 370 households from Jan 1, 2011 to Dec 31, 2015. We use the last 4 weeks

Traffic The *traffic* dataset contains hourly occupancy rates of 963 lanes in San Francisco bay area. The occupancy rates are in the range of $[0, 1]$ and were collected from Jan 1, 2008 to Mar 30, 2009.

Web The *web* dataset contains web traffic to $145K$ Wikipedia pages from July 1, 2015 to Sept 10, 2017. Each time series represent a number of daily views of the Wikipedia article. The data has no clear seasonalities, and the patterns varies significantly according to the popularity of the corresponding webpage.

Yahoo The *yahoo* dataset contains metrics of various Yahoo services with labeled anomalies and change points for a few weeks. It consists of 67 real production records and 100 of synthetic time series.

For each dataset, we split it into a training set, a validation and a test set in temporal order. A sample, which is a window of trajectory with specified length of condition and forecasting horizon, i.e. T and τ defined in section Section 4.3, is extracted from each subset during training and evaluation. Due to the large variance in magnitude across trajectories, we apply z-score normalization to numerical inputs. Specifically, the mean and standard variance of condition is computed and used to normalize both condition and prediction target.

	w/ labels	w/o labels
DeepAR	0.183/0.313	0.035/0.019
TCN	0.208/0.186	0.039/0.020
ConvTrans	0.228/0.276	**0.033/0.019**
AGA	**0.161/0.171**	0.033/0.023

Table 4.1: Performance on *yahoo* subsets with or without labeled change points, in terms of $R_{0.5}/R_{0.9}$. AGA not only outperforms baselines on the subset with changes, but also works well on regular data.

Baselines

To place our model in a line of researches, we compare AGA with a variety of machine learning models proposed recently.

- TRMF[144] is a method based onmatrix factorization
- DeepAR[39] is an autoregressive model based on stacked LSTMs
- TCN[13] is an autoregressive model based on temporal convolution networks
- MQ-RNN[129] is an LSTM-based quantile forecaster that makes multi-horizon forecasts directly
- DeepSSM[101] is a state space model parameterized by an LSTM.
- ConvT[76] is an autoregressive model based on Transformer architecture.

We used the open-sourced implementation for TRMF [1], DeepAR, MQ-RNN and DeepSSM [2] and our own implementation for the rest. Hyperparameters of all models are tuned using the same validation set.

Results

Table 4.2 presents the overvall results. As we can see, our model outperforms all existing methods in most of experiments, especially on *web*, suggesting the effectiveness

[1] https://github.com/rofuyu/exp-trmf-nips16/tree/master/python
[2] https://github.com/awslabs/gluon-ts

Dataset	electricity	traffic	web	yahoo
T	168	168	380	168
τ	24	24	30	24
TRMF	0.084/-	0.186/-	0.091/-	0.039/-
TCN	0.073/0.060	0.151/0.118	0.062/0.052	0.044/0.030
DeepAR	0.075/0.040*	0.161/0.099*	0.056/0.037	0.038/0.030
MQ-RNN	0.077/0.036	0.132/0.110	0.076/0.054	0.042/**0.027**
DeepSSM	0.083/0.056*	0.161/0.113*	0.065/0.050	0.041/0.042
ConvT	0.059/0.034	0.122/0.081	0.042/0.036	0.041/0.031
AGA(Ours)	**0.054/0.029**	**0.121/0.078**	**0.039/0.026**	**0.036/0.027**

Table 4.2: Performance comparison with state-of-the-art models on various datasets in terms of $R_{0.5}/R_{0.9}$. T represents the length of condition and τ is forecasting horizon. As TRMF only produces a point prediction, the normalized deviation (ND) is reported, which is equivalent to $R_{0.5}$. * denotes results from [101]. Our results are medians of 3 runs over the test sets.

of our model design. Notably, AGA is less effective on *traffic* dataset because it is more stationary and a ConvTrans model is good enough to capture the seasonal patterns. In contrast, AGA is more favorable in *web* and *yahoo*, indicating our model works better on noisy and non-stationary time series datasets with different scales.

4.5.3 Ablation Analysis

However, it is not clear which design contributes most to the empirical results. In this regard, we conduct ablation study for short-term forecasting task to examine the effects of the proposed techniques:

Global-local Autoregression (GL) We remove the local estimates in prediction, i.e. replace $[\boldsymbol{p}_1, \cdots, \boldsymbol{p}_{t-1}, \hat{\boldsymbol{p}}_t]$ with $[\boldsymbol{p}_0, \cdots, \boldsymbol{p}_{t-1}]$ in compoing the value matrix of self-attention according to 4.4.4.

Two-stream Self-attention (TS) We employ a standard self-attention block in [122] that maintains one hidden representation only instead of the two-stream self-attention strucuture in 4.4.5. For upper-level self-attention, queries and keys come from the same input as values.

Multi-scale Convolution (MC) Instead of using different sizes for convolution filters, we fix a single kernel size as in [76]. For example, if the complete model uses $5, 9$ as filter sizes, we select the larger 9 as the filter size in ablation study.

Distance Encoding (DE) We try removing the distance encoding and replacing the dot-product key-distance interaction with an parametric addition.

The results can be found in Figure 4.4. We first notice that global-local autoreression and two-stream self-attention result in better performance in general, except *traffic* in which change points are rare and simple self-attention works well enough. Then we observe that multi-scale convolution also contributes to the overall performance. In addition, the pattern-dependent distance encodings empirically improves pure self-attention as well, and the parameter-free dot-product formulation works as well as the widely-used additive function. An interesting exception is the smaller and simpler *yahoo* dataset, on which multi-scale and distance encodings worsen the forecasting accuracy. A possible reason is over-fitting since these operations introduce extra complexity to the model.

4.5.4 Analysis of Changes

To illustrate the capability of AGA to capture change points and other potential anomalies, we take advantage of the labels in *yahoo* dataset. To be specific, we separate test instances containing labeled anomalies (about 4%) from those without positive la-

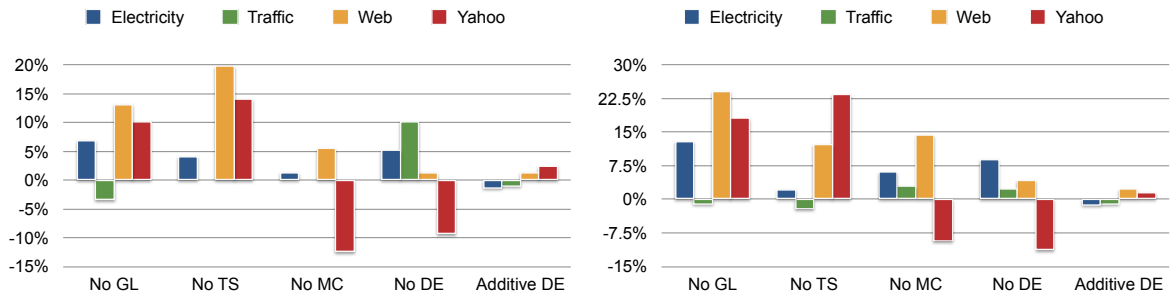


Figure 4.4: Ablation study on four datasets. The base model contains all components introduced in the paper. The tested configurations are derived from the base setting by removing each key components according to 4.5.3. The percentage of changes in $R_{0.5}$ (left panel) and $R_{0.9}$ (right panel) are reported. Smaller values are better.

bels and evaluate models on two groups of data individually in comparison with other AR models. The results are presented in Table 4.1. AGA can significantly improve accuracy on anomalous data as well as preserve favorable performance on normal data. This observation provides preliminary quantitative proof that our model is promising in forecasting certain non-stationary time series.

In order to provide a qualitative illustration of AGA's ability, we select a test sample from *web* dataset in which change points exist and visualize the predictions along with the attention distribution in figure 4.5. Note that we only employ a single self-attention layer in this experiment for better interpretability. We can see that significantly more attention was paid to the changed trends after time step 140 across the whole forecast horizons, meaning that model is able to distinguish new patterns from deprecated ones. Moreover, we can see a peak of attention on the last step of context in every single-step-ahead prediction, which reflects the activation of the local component of AGA.

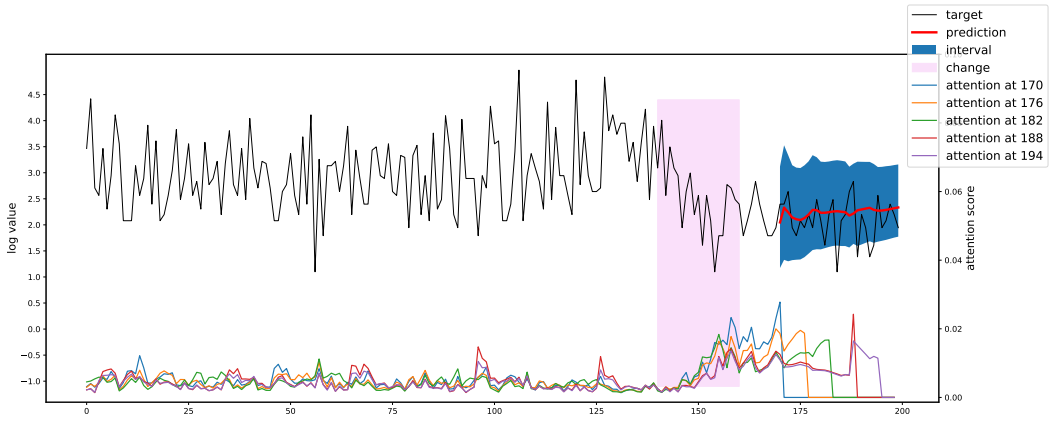


Figure 4.5: Attention distribution for predictions on *web* dataset. The red line indicates the predictive median and the rendered interval means the gap between the 10^{th} and 90^{th} percentiles. For all forecast at various horizons, the attention score over the context starts to increase significantly when a change in trend occurs. And the local estimate is activated by paying most attention to the last time steps.

4.6 Conclusion

We present a novel neural network model called Attention Guided Autoregression (AGA) based on Transformer architecture for time series forecasting task. Our model is able to achieve strong empirical performance on benchmark datasets and, also, to deal with breaking changes in non-stationary time series in a proper way. We find it beneficial to maintain two types of representations of time series data separately for more reliable attention, by which a combination of a global and a local autoregressive forecaster can intelligently cooperate to forecast both stationary and certain kinds of non-stationary time series.

Chapter 5

Inter-Series Attention Model for COVID-19 Forecasting

5.1 Introduction

The Coronavirus disease 2019 (COVID-19) has been impacting the human society since early 2020. At the time of this writing, it is an ongoing public health crisis in over 187 countries and territories around the world, with more than 30 million confirmed cases, and a growing death toll exceeding $1,000,000$. During this crisis, reliable forecasting of COVID-19 cases becomes important as it will help (1) healthcare institutes to allocate sufficient supply and resources, (2) policy-makers to consider new and further administrative interventions, (3) general public to be aware of the situation and to follow rules against the epidemic. Therefore, the Center for Disease Control and Prevention (CDC) has been actively collecting and publishing data about confirmed cases, hospitalization and deaths related to COVID-19, and hosting forecasting results in the coming weeks.

The US has been suffering the most severe loss from the pandemic, in which more than $200,000$ lives were lost. To encourage and to bring together efforts of COVID-

19 modeling, CDC has launched a forecasting challenge[1]. It calls for models that give predictions of the next 4 weeks on a daily or weekly basis. Besides COVID-19 data, other kinds of data such as demographic data, mobility data and intervention policies are also encouraged to be used in predictions.

Epidemic forecasting is regarded as a challenging task for a long time, for which many methods have been developed. They can be roughly categorized into two classes:

1. **Compartmental models** These models explicitly compartmentalize the population in groups based on their status of infection and recovery, and simulate the transmission process using differential equations. As of today, most of the CDC-featured forecasting methods fall into this category. Examples includes [4, 96, 139] that are built upon classic SIR or SEIR models [55]. Compartmental models describe disease spreading dynamics; however, it is quite hard to determine parameters in these models as they are influenced by many uncontrollable and dynamically changing factors.

2. **Statistical models** This type of methods fits the data to regression models directly, such as [3, 87, 132]. While they are more flexible in processing real data compared to compartmental models, they often assume a simplified model class such as generalized linear models [3], or require sophisticated hand-crafted features from additional, and possibly proprietary, data sources [132].

The forecasting of COVID-19 is even harder as various constantly changing factors, such as virus characteristics, social and cultural distinctions, public attitudes and behaviors, intervention policies and healthcare preparation, influence the contagious rate and death rate significantly. Will there be a better alternative that is solely data-driven without any assumptions about the underlying disease propagation mechanisms? In particular,

[1] https://www.cdc.gov/coronavirus/2019-ncov/cases-updates/forecasting.html

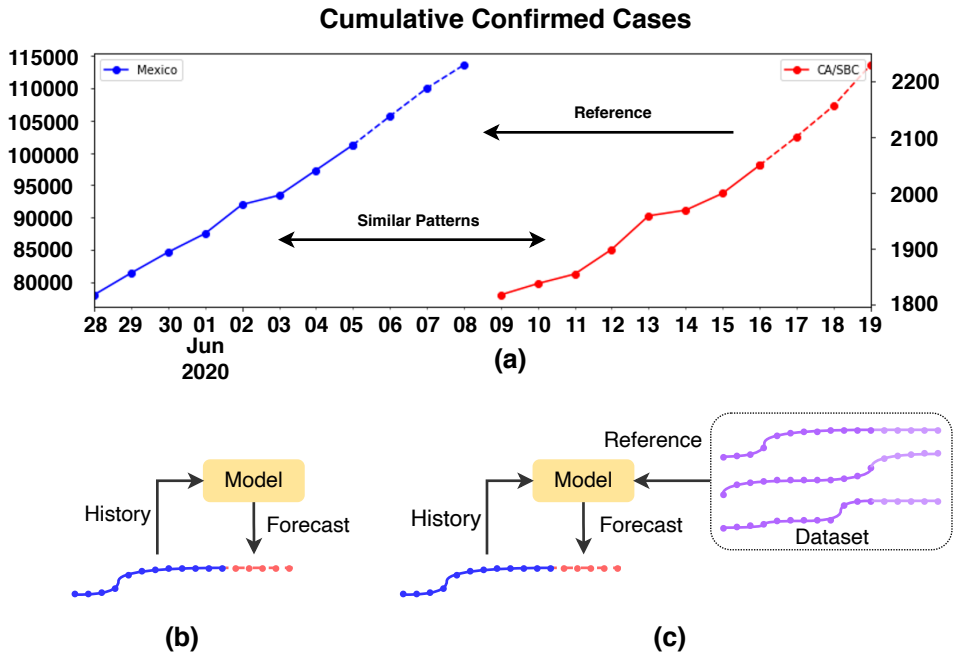


Figure 5.1: **(a)** A similar growth pattern of confirmed cases in Santa Barbara County, California, in mid June is observed in Mexico in late May and early June. **(b)** Conventional auto-regressive forecasting model. **(c)** The proposed inter-series forecasting model (**ACTS**).

we experimented a set leading neural forecasters [80, 105], but none of them gave the best result.

Since the deep models are originally designed for sufficiently long time series with hundreds of points, the scarce historical data in this task might be the reason of their failures. A natural alternative is to exploit other time series in the dataset if they reveals similar dynamics. Fortunately, even if any two regions present different disease curves over long term, it is likely to find short periods in which different regions sharing similar patterns. Figure 5.1(a) shows surprisingly that the growth pattern of confirmed cases in Santa Barbara county, California, is highly similar to that in Mexico 11 days ago even though at different scales. Moreover, the further growth in Santa Barbara is also close to that within the corresponding time period in Mexico. In light of this key observation, it is intuitively possible to do better forecasting for Santa Barbara by referring to Mexico

in this specific time window via proper transformations.

Based on this intuition, we propose to generalize the conventional auto-regressive forecasting to a novel paradigm: besides the local historical data, we also refer to the past reports in all other regions simultaneously in forecasting. Figures 5.1(b) and (c) illustrate the the fundamental difference between the two paradigms. With time series data of COVID-19 from various locations accumulating over time, we are able to deliver a model outperforming the existing methods by inter-series modeling. Note that unlike other cross-location epidemic forecasters such as [32], only certain time periods rather than the entire time series from other regions will be referred to.

In order to make the proposed paradigm work, it is critical to find small segments in reference time series that exhibit similarity with target time series. It turns out that the attention mechanism [122] is a good choice for pattern matching. Moreover, it is found that solely applying attention does not work the best as the embedded small segments do not contain long-term trends that are not directly comparable. We filter out these trends and introduce a normalization step so that the small segments can be matched at a consistent scale. In the end, we put all of these components together and achieve global optimum by joint training. Our new model called **ACTS** (Attention Crossing multiple Time Series), is able to outperform leading forecasters hosted at CDC.

5.2 Related Work

There has been a large body of work focusing on epidemic forecasting. To incorporate domain knowledge, mechanistic models [73, 147] has been favored since they often consider various factors such as epidemiological and social properties, and they make forecasts based on simulation. Moreover, geographic information can also be incorporated into the mechanistic models to better illustrate the spreading process of an infectious dis-

symbol	interpretation
$\boldsymbol{x}_t^i$	The value at time t in location i.
$\boldsymbol{x}_{s:t}^i$	The time series from s to t in location i
$\boldsymbol{W}_\cdot$	Parameter matrices to be learned.
$[\boldsymbol{a}; \boldsymbol{b}]$	The concatenation of $\boldsymbol{a}$ and $\boldsymbol{b}$.
$\langle \boldsymbol{a}, \boldsymbol{b} \rangle$	Inner product of $\boldsymbol{a}$ and $\boldsymbol{b}$.
$[\![s, t]\!]$	Consecutive index set $s, s+1, \cdots, t$

Table 5.1: Used notations

ease [8, 9]. These models have excellent interpretability but often fail to fit real observed data due to their rigid and over-simplified assumptions without careful calibration.

On the other hand, statistical methods explicitly fit historical data to a statistical model and use it to obtain predictions by extrapolation [18, 21]. For example, [102] relies on kernel density estimation, [84] uses seasonal ARIMA, [138] chooses particle filtering and [150] employs Gaussian process regression. These methods are either too simple or require laborious feature engineering. Hence, various deep learning techniques are also introduced to forecast disease spreading, such as [136, 125, 62, 118, 119, 99]. They use deep neural networks to extract complex temporal patterns from historical data and a selected set of additional features. [32, 45] are conceptually closer to our model, both of which employ attention mechanism to compare encoded temporal patterns across multiple locations. However, they require a fixed graph structure with geographic information and produce a similarity score between locations that is independent of time. Instead, in our model we generate embeddings of dynamical patterns for attention over both spatial and temporal dimensions so that the generated attention map are temporally dynamical and free from any predefined geographic structures.

5.3 Problem Statement

In COVID-19 forecasting, there are three types of incidences, namely confirmed cases, hospitalizations and deaths, to be predicted. The historical data is reported on a daily basis, and we will predict them for the coming weeks. Table 5.1 summarizes the notations we use in the following sections. Note that throughout the paper, terms "location" and "region" will be used interchangeably. Problem definition is formulated as follows.

DEFINITION 1. ***Incidence Time Series*** *We denote by* $\boldsymbol{x}_t^i$ *the reported value of a certain type of incidence data at date* t *and location* i*, for* $t = 1, 2, \cdots, T$ *and* $i = 1, 2, \cdots, N$*. Hence, the incidence time series of location* i *denoted by* $\boldsymbol{x}_{1:T}^i$*.* $\boldsymbol{x}_{s:t}^i$ *is called a time segment of* $\boldsymbol{x}^i$*, where* $[\![s, t]\!], 1 \leq s < t \leq T$ *is called a window.*

DEFINITION 2. ***Target Region*** *At the last date* T*, we predict the future incidences for location* $i_0 \in [1, N]$ *beyond* T*. We call* i_0 *the target region and* $\boldsymbol{x}_{1:T}^{i_0}$ *the target time series.*

DEFINITION 3. ***Reference Regions*** *The regions other than the target region* i_0 *are called reference regions. The reference time series are* $\boldsymbol{x}_{1:T}^i$ *where* $i \neq i_0$*. In a generalized definition, reference regions could include the target region.*

DEFINITION 4. ***Additional Features*** *Besides historical incidences in each region, other features might be available including demographic information, mobility index, and interventions. For each region* i*, time-independent features are concatenated into a single vector* $\boldsymbol{u}^i$*, and time-dependent ones into another time series* $\boldsymbol{r}_t^i$*.*

Problem Statement Given N time series $X_{1:T}^i$ ($i \in [1, N]$) and additional features, we aim to predict future incidences in a target region $i_0 \in [1, N]$ over H consecutive days after T, i.e. $\boldsymbol{x}_{T+1:T+H}^{i_0}$.

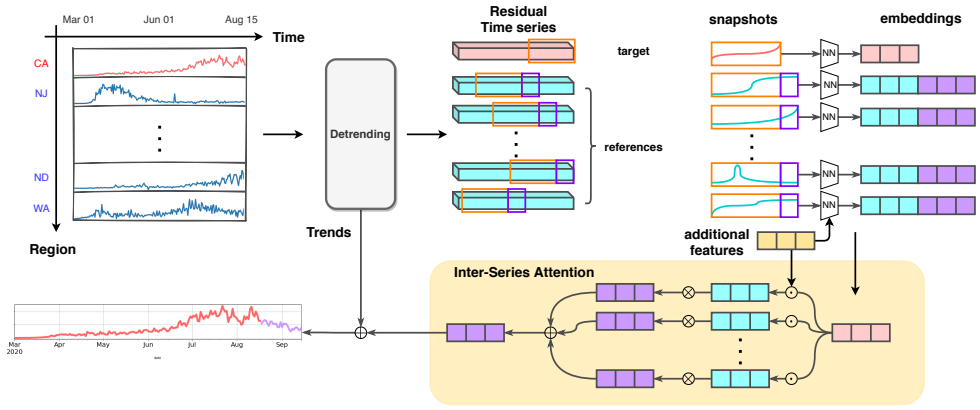


Figure 5.2: Our proposed Inter-series Attention Network. Best view in color.

5.4 Methodology

Traditionally, epidemic forecasts are made by analyzing only the growth pattern of incidences. [8, 9, 32, 45] take the incidences from neighboring regions into consideration as diseases spread through social interaction. Rather than explicitly modeling the disease spreading process, we take a bold step to directly compare the incidence curves across regions. Once the similarities between the current incidences in the target region with the past time segments in reference regions are identified, the following incidences in the reference regions can be used to forecast the future incidences in the target region.

Formally, we introduce an embedding function $\phi(\cdot)$ to encode a time series segment $\boldsymbol{x}_{t-l+1:t}$ into a vector, and then use dot-product of vectors to measure similarity. The following incidences $\boldsymbol{x}_{t+1:t+h}$ is also encoded by another embedding function $\psi(\cdot)$ for further aggregation. However, while there are comparable short-term patterns that can be extracted from time series segments, there are also non-stationary long-term trends that hinder reasonable comparison and aggregation of local patterns within segments.

We resolve the problem in two steps. First, we apply a trainable detrending module to the raw time series to remove long-term trends so that incidences across different regions

are more comparable. Second, we take rolling windows from residual time series and transform them into a common feature space using normalized convolution as embedding functions $\phi(\cdot)$ and $\psi(\cdot)$. The embedding of the recent window in the target region is then compared with windows from references to produce weights for combining the following incidences of each reference window. In such pairwise comparisons, differences in both time-dependent and time-independent features are taken into account so that the curves in corresponding windows can be better aligned. The combinations are then added to the extrapolation of filtered trends to generate the final prediction. We jointly train both modules in an end-to-end manner so that both the long- and short-term patterns can be decoupled in an adaptive way.

Figure 5.2 gives an overview of the framework. In the following subsections, we introduce each component in details.

5.4.1 Detrending

We adopt a learnable Holt smoothing model ([112]) to remove long-term trends from the raw time series. Specifically, we introduce a set of parameters $\theta_e^i = [a_0^i; b_0^i; \alpha^i; \beta^i]$ per series, where a_0^i is the initial level, b_0^i is the initial trend, α^i is the level smoothing coefficient and β^i is the trend smoothing coefficient. Then Holt's equations ([59]) are launched to iteratively derive levels and piecewise linear slopes in $\boldsymbol{x}_{1:T}^i$,

$$\begin{aligned} a_t^i &= \alpha^i \boldsymbol{x}_t^i + (1-\alpha^i)(a_{t-1}^i + b_{t-1}^i), \\ b_t^i &= \beta^i (a_t^i - a_{t-1}^i) + (1-\beta^i) b_{t-1}^i, \\ \hat{\boldsymbol{x}}_t^i &= \boldsymbol{x}_t^i - a_t^i. \end{aligned} \tag{5.1}$$

After detrending, the residual time series $\hat{\boldsymbol{x}}_{1:T}$ will contain short-term patterns for further processing. Projection from the long-term trend is generated by simple linear extrapola-

tion,

$$\bar{\boldsymbol{x}}^i_{t+h} = a^i_t + hb^i_t. \tag{5.2}$$

A more sophisticated detrending process might further boost performance; we leave it for future study. The detrending process is applied to all the time series and the residual time series are fed into the following attention module.

5.4.2 Attention Module

As COVID-19 is a new disease, we do not have its historical data in the past seasons. Hence, it is critical to leverage limited data from the same season, but across different regions, i.e. model the correlations between regions that have been undergoing the pandemic. Without detailed information about spatial dynamics such as population movement, we instead employ attention mechanism to measure the relation of one region to other regions by directly comparing the incidence curves after trend filtering. Since there are many stages in a dynamical epidemiological process, it is necessary to learn a representation for each time period in a region for alignment in attention. In light of this idea, we apply a convolution layer to encode the residual time series segment $\hat{\boldsymbol{x}}_{t-l+1:t}$ to a vector, based on which attention scores measuring similarity between regions are computed.

Segment Embedding

Even after detrending, the scales of reported numbers in residual time series are still quite different across regions. It is important to normalize residuals before embedding. We empirically find it better to apply min-max normalization to the cumulative sum of incidence time series, which can be regarded as a kind of smoothing. Specifically, for a rolling window of size l representing a period of time, i.e. $\hat{\boldsymbol{x}}^i_{t-l+1:t}, t \in [l, T]$, we

compute its cumulative sums and apply the min-max normalization to the monotonically increasing series,

$$\boldsymbol{c}_j^i = \sum_{k=t-l+1}^{j} \hat{\boldsymbol{x}}_k^i; \qquad \tilde{\boldsymbol{c}}_j^i = \frac{\boldsymbol{c}_j^i - \boldsymbol{c}_{t-l+1}^i}{\boldsymbol{c}_t^i - \boldsymbol{c}_{t-l+1}^i}, \tag{5.3}$$

for $j \in [t-l+1, t]$. As a result, the first and last values of the normalized series will consistently be 0 and 1 respectively.

We then instantiate the function $\phi(\cdot)$ using a convolution layer with d feature maps to the scaled segment and time-dependent features. The kernel size is empirically selected, and when it is smaller than l, average pooling is applied in order to reduce a sequence to a vectorized embedding,

$$\boldsymbol{p}_t^i = \text{AvgPool}\left(\text{Conv}\left(\left[\tilde{\boldsymbol{c}}_{t-l+1:t}^i; \boldsymbol{r}_{t-l+1:t}^i\right]\right)\right) \in \mathbb{R}^d. \tag{5.4}$$

These segment embeddings are used to model similarity in different temporal periods across different regions.

Likewise, we employ another convolution-pooling layer as $\psi(\cdot)$ to encode the following incidences over H days after each segment into so-called **development embedding**,

$$\boldsymbol{g}_t^i = \text{AvgPool}\left(\text{Conv}\left(\tilde{\boldsymbol{c}}_{t+1:t+H}^i\right)\right) \in \mathbb{R}^d. \tag{5.5}$$

They represent the succeeding development after encoded segments and will be the references for the prediction of the given target region. In fact, we can pair the segments and references by aligning the time indices, i.e. $\{\boldsymbol{p}_t^i, \boldsymbol{g}_t^i\}$ for $t \in [l, T-H]$.

Inter-series Attention

Given the embeddings, we use dot-product attention to compare segments and combine the values. Specifically, we linearly map the segment embeddings to query vectors $\boldsymbol{q}_t^i$ and key vectors $\boldsymbol{k}_i^t$, from which the similarity score is computed. The development

embeddings are projected to value vectors $\boldsymbol{v}_t^i$. On the other hand, the additional time-independent features $\boldsymbol{u}^i$ are also incorprated into queries and keys.

$$\begin{aligned} \boldsymbol{q}_t^i &= \boldsymbol{W}_Q \boldsymbol{p}_t^i + \boldsymbol{W}_{u,q} \boldsymbol{u}^i; \\ \boldsymbol{k}_t^i &= \boldsymbol{W}_K \boldsymbol{p}_t^i + \boldsymbol{W}_{u,k} \boldsymbol{u}^i; \\ \boldsymbol{v}_t^i &= \boldsymbol{W}_V \boldsymbol{g}_t^i; \end{aligned} \tag{5.6}$$

For a target region i_0, we take $\boldsymbol{q}_T^{i_0}$ for the last segment and compute its similarity with all the keys from other time segments across all the regions, which is then used to obtain a weighted sum of values.

$$\hat{\boldsymbol{v}}_T^{i_0} = \sum_{i,t \in \Omega} \frac{\exp\left(\langle \boldsymbol{q}_T^{i_0}, \boldsymbol{k}_t^i \rangle\right)}{\sum_{i,t \in \Omega} \exp\left(\langle \boldsymbol{q}_T^{i_0}, \boldsymbol{k}_t^i \rangle\right)} \boldsymbol{v}_t^i, \tag{5.7}$$

where $\Omega = [1, N] \times [l, T - H]$. In this way, the past observations in both the target region and reference regions are fully utilized. The weighted combination of values $\hat{\boldsymbol{v}}_T^{i_0}$ is then linearly projected to an estimate of $\tilde{\boldsymbol{c}}_{T+1:T+H}^{i_0}$. We apply the inverse transformation of equation (5.3) to get an estimate of $\hat{\boldsymbol{x}}_{T+1:T+H}^{i_0}$, denoted by $\hat{\boldsymbol{y}}_{T+1:H}^{i_0}$.

In the end, the estimate from attention module is added to the extrapolations in the detrending module to produce the final forecast $\boldsymbol{y}_{T+1:T+H}^{i_0}$, where

$$\boldsymbol{y}_t^{i_0} = \bar{\boldsymbol{x}}_t^{i_0} + \hat{\boldsymbol{y}}_t^{i_0}, \qquad t \in [T+1, T+H].$$

5.4.3 Joint Training

The model can be trained by minimizing the joint loss with respect to the parameters in all the modules. The joint loss is an aggregation of prediction error $E(\cdot, \cdot)$ computed in two steps. First, for a single target region, we compare our forecasts and ground truths for different T, i.e. lengths of history. Second, we take the aggregated loss in the first step for every region. Formally, the joint loss is defined as

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{T=l}^{L-H} E(\boldsymbol{y}_{T+1:T+H}^i, \boldsymbol{x}_{T+1:T+H}^i) \tag{5.8}$$

where L is the total number of available historical reports, and l is the minimum required history length. In our experiments, we choose Mean Absolute Error (MAE) to be the error metric $E(\cdot, \cdot)$, i.e.

$$E(\boldsymbol{y}^i_{T+1:T+H}, \boldsymbol{x}^i_{T+1:T+H}) = \frac{1}{H} \sum_{t=T+1}^{T+H} |\boldsymbol{y}^i_t - \boldsymbol{x}^i_t|$$

5.5 Experiments

In this section, we demonstrate the effectiveness of the proposed model on real COVID-19 datasets. We intend to answer the following questions:

- Can **ACTS** outperform the popular COVID-19 forecasters referred at CDC and other state-of-the-art deep learning models?
- How much does each component of **ACTS** contribute to the model performance?
- What kind of similarity can inter-series attention capture?

5.5.1 Experimental Settings

Dataset The COVID-19 incidence data is publicly available at JHU-CSSE[2] and COVID tracking project[3]. Additional features are also publicly available [4] [5] [6]. The features we used include total population, population density, ratios of age/gender/race, available hospital beds, and traffic mobility, which are proven to bring marginal accuracy gain in the hospitalization forecasting task in our experiments. The dataset covers the reports up to September 27, 2020 from 50 states and DC in the US.

[2] github.com/CSSEGISandData/COVID-19
[3] covidtracking.com/
[4] github.com/descarteslabs/DL-COVID-19
[5] github.com/djsutherland/pummeler
[6] data.world/liz-friedman/hospital-capacity-data-from-hghi

Evaluation Protocol As required by CDC, we predict the incidence data over the next 4 weeks at a given date and compare the forecasts with the reported ground truths. Suppose we are predicting the new confirmed cases in the state of California starting from 08/16. As context, we are provided a daily time series consisting of incidences in all the states till 08/15. There are three forecasting tasks: daily forecasts for new hospitalizations, weekly forecasts for new confirmed cases and deaths.

The forecasting performance is evaluated in terms of Weighted Absolute Percentage Error (WAPE), defined by the ratio of Mean Absolute Error (MAE) and mean value of ground truths.

At each prediction date, we keep the data in the last 7 days for validation, and the remaining historical data for training. We use the validation data to tune the hyperparameters and to avoid overfitting by early stopping. Other implementation details can be found in Appendix too.

Baselines We compare the performance of the epidemic models featured at CDC, including

- **YYG** [51]: An SEIR model with learnable parameters that attracts a lot of attention from media;
- **CU** [96]: A metapopulation SEIR model developed by researchers in Columbia University;
- **UCLA** [151]: An SuEIR model using machine learning developed by Statistical Machine Learning Lab at UCLA;
- **ERDC**[7]: An SEIR model that considers unreported infections and isolated popu-

[7] https://github.com/reichlab/covid19-forecast-hub/blob/master/data-processed/USACE-ERDC_SEIR/metadata-USACE-ERDC_SEIR.txt

lation developed by US Army Engineer Research and Development Center;

- **LANL** [71]: A statistical dynamical growth model accounting for population susceptibility developed by Los Alamos National Laboratory;
- **CovidSim**[8]: Machine learning model based on generalized random forests.

The first four are compartmental models and the last two rely on statistical modelling. Other than these conventional models, we also evaluate three deep learning models for time series forecasting,

- **DeepCOVID** [105] An operational deep learning framework designed for real-time COVID-19 forecasting developed by Georgia Tech;
- **ConvTrans** [76] A self-attention based Transformer model that also employs convolutions for pattern representations;
- **TFT** [80] A self-attention based deep learning model with feature selection.

We implement the ConvTrans and TFT and tune the hyperparameters using the validation data. All of our implementations run on a server with an Intel i7-6700K CPU and a single GTX 1080Ti GPU. For other baselines, since their implementations are not open-sourced, we take their forecasts submitted to the challenge hosted by CDC [9].

5.5.2 Performance Comparison

Table 5.2 shows the forecasting performance on 6 different dates. Three types of incidence data, namely confirmed cases (C), hospitalizations (H) and deaths (D) are separately predicted. We have three key observations: (1) In 13 out of 18 cases, **ACTS**

[8] https://www.covid19sim.org/documents/outbreak-methods
[9] https://github.com/reichlab/covid19-forecast-hub

		Method									
		YYG	CU	UCLA	ERDC	LANL	Covid Sim	Deep COVID	Conv Trans	TFT	ACTS
06/21	C	-	-	-	-	0.51	-	-	1.09	0.51	**0.39±0.01**
	H	-	1.91	-	-	1.08	0.95	**0.63**	1.22	0.80	0.80±0.02
	D	0.52	1.48	0.56	-	0.58	1.46	0.66	1.09	0.67	**0.45±0.01**
07/05	C	-	-	-	-	0.37	-	-	0.37	0.39	**0.33±0.01**
	H	-	0.98	1.23	0.66	0.95	-	0.65	1.08	0.84	**0.61±0.04**
	D	0.45	0.65	0.53	**0.38**	0.52	-	0.85	0.60	0.51	0.60±0.01
07/19	C	-	-	-	-	**0.27**	-	-	0.50	0.44	0.31±0.01
	H	-	0.67	1.24	0.77	0.78	1.71	0.70	0.99	0.66	**0.60±0.03**
	D	0.30	0.43	0.39	1.10	0.48	0.33	0.4506	0.54	0.67	**0.28±0.01**
08/02	C	-	-	-	-	0.30	-	-	0.24	0.24	**0.16±0.04**
	H	-	0.67	0.95	0.71	0.68	1.66	0.79	0.93	0.92	**0.66±0.09**
	D	0.24	0.37	0.27	0.57	0.44	0.26	0.29	0.45	0.38	**0.21±0.01**
08/16	C	-	0.67	0.35	0.28	0.29	0.23	-	0.33	0.55	**0.20±0.03**
	H	-	0.64	0.99	0.60	0.65	1.38	0.98	0.96	0.92	**0.57±0.02**
	D	**0.19**	0.42	0.25	0.53	0.34	0.27	0.28	0.44	0.31	0.23±0.01
08/30	C	-	0.43	0.31	0.34	0.33	0.23	-	0.36	0.29	**0.23±0.03**
	H	-	0.66	0.91	0.68	0.69	1.31	0.83	0.93	0.82	**0.58±0.03**
	D	**0.20**	0.41	0.23	0.56	0.34	0.25	0.36	0.42	0.40	0.25±0.02

Table 5.2: Forecasting performances across different time periods for different types of incidence data in terms of WAPE. A smaller value indicates better performance. We also include the variance of our model's performance by running 5 times with different random initalizations. "-" means the forecasting results of the corresponding baseline are not available.

outperforms other algorithms by a considerable margin. On average, it improves 9%, 5% and 4% over the best of these algorithms for C, H and D, respectively. (2) **ACTS** is more favorable on recent days when there are more abundant data available, showing that data-driven methods benefit from more data. (3) The two deep learning approaches ConvTrans and TFT do not exhibit strong performance. The main difference between ours and these approaches is the employment of attention across multiple time series, which dramatically boosts the performance. Note that our model can be trained in less than 5 minutes and inference takes only seconds.

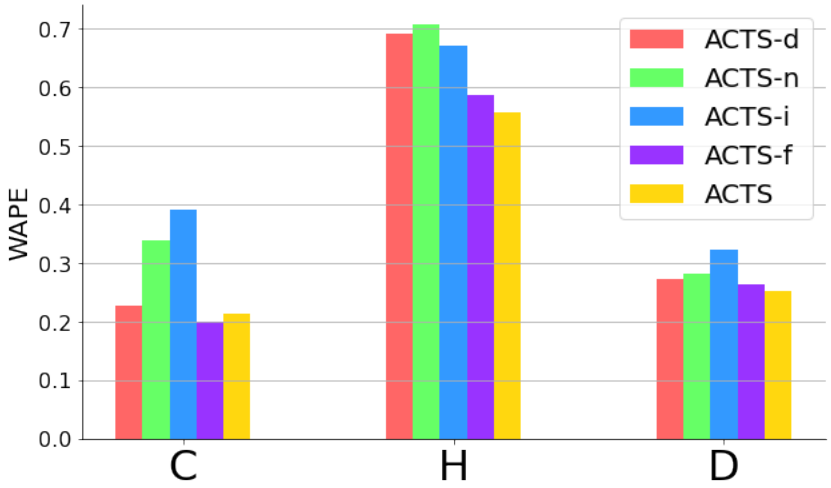


Figure 5.3: Empirical effects of each component of **ACTS** on forecasting error.

5.5.3 Ablation Study

For deeper understanding of our model, we disable each component of **ACTS** to examine its contribution:

- **ACTS-d** We remove the detrending module and obtain an attention-only forecaster;
- **ACTS-n** We remove the normalization in segment embedding;
- **ACTS-i** We restrict the attention to the target time series only. The model degenerates to an auto-regressive model similar to ConvTrans and TFT;
- **ACTS-f** We remove the additional features in the model and only rely on incidence data.

The hyperparameters of all variants are kept the same. We compare their performance against **ACTS** using training data up to August 30, 2020. Figure 5.3 depicts the results, based on which we have the following observations:

- Overall every component of **ACTS** has positive effects on forecasting accuracy, except that the introduction of additional features has mixed effect. We suspect that either better modelling could help or their effect has been absorbed by the incidence time series;

- Among all the components, inter-series attention has the most significant impact on the performance, which proves that our design of attention crossing multiple time series is valid. It can capture cross-region similarity in COVID-19 forecasting;

- The detrending module makes some contribution. We believe it has the potential for further improvement, e.g. employing advanced trend filtering or even epidemic models.

5.5.4 Cross-region Similarity

A key feature of **ACTS** is that it can capture similarity between regions via attention from data. According to equation (5.7), the reference set Ω is common for any target regions i_0, and the learned attention distribution is determined by $\boldsymbol{q}_T^{i_0}$. Hence, we directly take those d-dim queries for every region and apply K-means clustering to group them. In this experiment, we use the death forecasting model as an example, where T is August 30, 2020, and $K = 4$ is selected based on the Elbow method [85].

A colored map is shown in Figure 5.4 based on obtained clusters. We can see that California, Texas and Florida, the three states recently hit most seriously are grouped together. Furthermore, states like Arizona, Illinois, North Carolina and Georgia are recognized since they also suffer severe crisis. Interestingly, the states of Wyoming and Vermont are distinguished by our model, in which few deaths are observed for a long period. Overall, our method is able to identify similarities between regions to a certain degree.

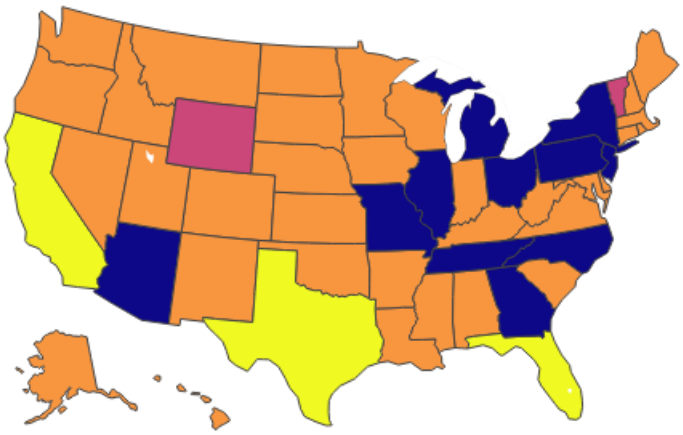


Figure 5.4: Groups of the US states learned by inter-series attention on death tolls by August 30, 2020.

5.6 Conclusion

We present **ACTS** for COVID-19 forecasting, a purely data-driven framework for an urgent forecasting problem concerning the entire world. It extends the popular deep learning technique, namely attention mechanism, to learning inter-series similarity for time series forecasting. Above that, we also introduce a detrending component to model long-term trends that are difficult for attention model to capture. Both modules are learned jointly based solely on COVID-19 incidence data and a handful of simple features. Without any domain knowledge, our model can empirically outperform many strong forecasters featured by CDC.

Chapter 6

Domain Adaptation for Time Series Forecasting via Attention Sharing

6.1 Introduction

While deep forecasting models excel at capturing complex temporal dynamics from a sufficiently large time series dataset, it is often challenging in practice to collect enough data. A common solution to the data scarcity problem is to introduce another dataset with abundant data samples from a so-called source domain related to the dataset of interest, referred to as the target domain. For example, traffic data from an area with an abundant number of sensors (source domain) can be used to train a model to forecast the traffic flow in an area with insufficient monitoring recordings (target domain). However, deep neural networks trained on one domain can be poor at generalizing to another domain due to the issue of domain shift, that is, the distributional discrepancy between domains [126].

Domain adaptation (DA) methods attempt to mitigate the harmful effect of domain shift by aligning features extracted across source and target domains [44, 14, 57, 10]. Ex-

isting approaches mainly focus on classification tasks, where a classifier learns a mapping from a learned domain-invariant latent space to a fixed label space using source data. Consequently, the classifier depends only on common features across domains, and can be applied to the target domain [130].

There are two main challenges in directly applying existing DA methods to time series forecasting. First, due to the temporal nature of time series, evolving patterns within time series are not likely to be captured by a representation of the entire history. Future predictions may depend on local patterns within different time periods, and a sequence of *local representations* can be more appropriate than using the *entire history* as done with most conventional approaches. Second, the *output space* in forecasting tasks is not fixed across domains in general since a forecaster generates a time series following the input, which is domain-dependent, e.g. kW in electrical source data vs. unit count in stock target data. Both domain-invariant and domain-specific features need to be extracted and incorporated in forecasting to model domain-dependent properties so that the data distribution of the respective domain is properly approximated. Hence, we need to carefully design the type of features to be shared or non-shared over different domains, and to choose a suitable architecture for our time-series forecasting model.

We propose to resolve the two challenges using an attention-based model [122] equipped with domain adaptation. First, for evolving patterns, attention models can make dynamic forecasts based on a combination of values weighted by time-dependent query-key alignments. Second, as the alignments in an attention module are independent of specific patterns, the queries and keys can be induced to be domain-invariant while the values can stay domain-specific for the model to make domain-dependent forecasts. Figure 6.1 presents an illustrative example of a comparison between a conventional attention-based forecaster (AttF) and its counterpart combined with our domain adaptation strategy (DAF) on synthetic datasets with sinusoidal signals. While AttF is trained using lim-

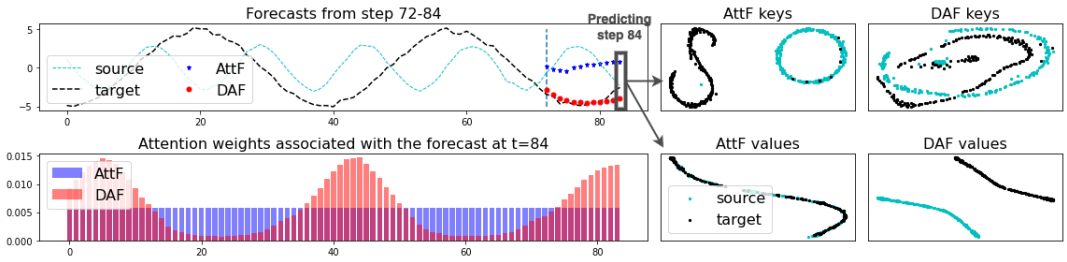


Figure 6.1: Forecasts of single-domain attention-based forecaster (AttF) and our cross-domain forecaster (DAF). Sample forecasts from steps 72-84 on traffic data where our DAF uses household electricity data as source data *(top left)*. Bar plot of the weights on the context history of the attention distributions of AttF and DAF associated with forecasting step 84 (*bottom left*). Attention keys (*top right*) and values (*bottom right*) of AttF and DAF after dimension reduction to 2D. The keys and values of AttF in the source domain are generated by simply applying AttF model trained on target data to the source data. The strategy of aligning keys rather than values between source and target domains in our DAF captures the correct attention weights, as illustrated by the accurate forecasts compared to AttF (cf. red dots vs. blue dots from steps 72-84).

ited target data, DAF is jointly trained on both domains. By aligning the keys across domains as the top rightmost panel shows, the context matching learned in the source domain helps DAF generate more reasonable attention weights that focus on the same phases in previous periods of target data than the uniform weights generated by AttF in the bottom left panel. The bottom right panels illustrate that the single-domain AttF produces the same values for both domains as the input is highly overlapped, while DAF is able to generate distinct values for each domain. As a result, the top left panel shows that DAF produces more accurate domain-specific forecasts than AttF does.

6.2 Related Work

Deep neural networks have been introduced to time series forecasting with considerable successes [39, 13, 89, 129, 128, 108, 101]. In particular, attention-based transformer-like models [122] have achieved state-of-the-art performance [76, 80, 135, 149]. A down-

side to these sophisticated models is their reliance on a large dataset with homogeneous time series to train. Once trained, the deep learning models may not generalize well to a new domain of exogenous data due to domain shift issues [127, 98, 126].

To solve the domain shift issue, domain adaptation has been proposed to transfer knowledge captured from a source domain with sufficient data to the target domain with unlabeled or insufficiently labeled data for various tasks [86, 130, 100]. In particular, sequence modeling tasks in natural language processing mainly adopt a paradigm where large transformers are successively pre-trained on a general domain and fine-tuned on the task domain [33, 54, 53, 103]. It is not immediate to directly apply these methods to forecasting scenarios due to several challenges. First, it is difficult to find a common source dataset in time series forecasting to pre-train a large forecasting model. Second, it is expensive to pre-train a different model for each target domain. Third, the predicted values are not subject to a fixed vocabulary, heavily relying on extrapolation. Lastly, there are many domain-specific confounding factors that cannot be encoded by a pre-trained model.

An alternative approach to pre-training and fine-tuning for domain adaptation is to extract domain-invariant representations from raw data [12, 27]. Then a recognition model that learns to predict labels using the source data can be applied to the target data. In their seminal works, [43, 44] propose DANN to obtain domain invariance by confusing a domain discriminator that is trained to distinguish representations from different domains. A series of works follow this adversarial training paradigm [121, 148, 2, 133], and outperform conventional metric-based approaches [82, 24, 52] in various applications of domain adaptation. However, these works do not consider the task of time series forecasting, and address the challenges in the introduction accordingly.

In light of successes in related fields, domain adaptation techniques have been introduced to time series tasks [98, 131]. [20] aim to solve domain shift issues in classification

and regression tasks by minimizing the discrepancy of the associative structure of time series variables between domains. A limitation of this metric-based approach is that it cannot handle the multi-horizon forecasting task since the label is associated with the input rather than being pre-defined. [60] propose DATSING to adopt adversarial training to fine-tune a pre-trained forecasting model by augmenting the target dataset with selected source data based on pre-defined metrics. This approach lacks the efficiency of end-to-end solutions due to its two-stage nature. In addition, it does not consider domain-specific features to make domain-dependent forecasts. Lastly, [48, 14, 111] make use of domain-invariant and domain-specific representations in adaptation. However, since these methods do not accommodate the sequential nature of time series, they cannot be directly applied to forecasting.

6.3 Domain Adaptation in Forecasting

Adversarial Domain Adaptation in Forecasting Suppose a set of N time series, and each consists of observations $z_{i,t} \in \mathbb{R}$ at time t. Given T past observations and all future input covariates, we wish to make τ multi-horizon future predictions at time T via model F:

$$z_{i,T+1}, \ldots, z_{i,T+\tau} = F(z_{i,1}, \ldots, z_{i,T}). \tag{6.1}$$

We focus on the scenario where little data is available for the problem of interest while sufficient data from other sources is provided. For example, one or both of the number of time series N and the length T is limited. We denote the dataset $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^{N}$ with past observations $\mathbf{X}_i = [z_{i,t}]_{t=1}^{T}$ and future ground truths $\mathbf{Y}_i = [z_{i,t}]_{t=T+1}^{T+\tau}$ for the i-th time series. We also omit the index i when the context is clear. To find a suitable forecasting model F in equation (6.1) on a data-scarce time series dataset, we cast the problem in terms of a domain adaptation problem, given that another "relevant" dataset

is accessible. In the domain adaption setting, we have two types of data: source data $\mathcal{D}_{\mathcal{S}}$ with abundant samples and target data $\mathcal{D}_{\mathcal{T}}$ with limited samples. Our goal is to produce an accurate forecast on the target domain $\mathcal{T}$, where little data is available, by leveraging the data in the source domain $\mathcal{S}$. Since our goal is to provide a forecast in the target domain, in the remainder of the text, we use T and τ to denote the target historical length and target prediction length, respectively, and also use the subscript $\mathcal{S}$ for the corresponding quantities in the source data $\mathcal{D}_{\mathcal{S}}$, and likewise for $\mathcal{T}$.

To compute the desired target prediction $\hat{\mathbf{Y}}_i = [\hat{z}_{i,t}]_{t=T+1}^{T+\tau}$, $i = 1, \ldots, N$, we optimize the training error on both domains jointly and in an adversarial manner in the following minimax problem:

$$\min_{G_{\mathcal{S}}, G_{\mathcal{T}}} \max_{D} \quad \mathcal{L}_{seq}(\mathcal{D}_{\mathcal{S}}; G_{\mathcal{S}}) + \mathcal{L}_{seq}(\mathcal{D}_{\mathcal{T}}; G_{\mathcal{T}}) - \lambda \mathcal{L}_{dom}\left(\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}; D, G_{\mathcal{S}}, G_{\mathcal{T}}\right), \tag{6.2}$$

where the parameter $\lambda \geq 0$ balances between the estimation error $\mathcal{L}_{seq}$ and the domain classification error $\mathcal{L}_{dom}$. Here, $G_{\mathcal{S}}, G_{\mathcal{T}}$ denote sequence generators that estimate sequences in each domain, respectively, and D denotes a discriminator that classifies the domain between source and target.

We first define the estimation error $\mathcal{L}_{seq}$ induced by a sequence generator G as follows:

$$\mathcal{L}_{seq}(\mathcal{D}; G) = \sum_{i=1}^{N} \left(\frac{1}{T} \sum_{t=1}^{T} l(z_{i,t}, \hat{z}_{i,t}) + \frac{1}{\tau} \sum_{t=T+1}^{T+\tau} l(z_{i,t}, \hat{z}_{i,t}) \right), \tag{6.3}$$

where l is a loss function and estimation $\hat{z}_{i,t}$ is the output of a generator G, and each term in equation (6.3) represents the error of input reconstruction and future prediction, respectively. Next, let $\mathcal{H} = \{h_{i,t}\}_{i=1,t=1}^{N,T+\tau}$ be a set of some latent feature $h_{i,t}$ induced by generator G. Then, the domain classification error $\mathcal{L}_{dom}$ in equation (6.2) denotes the cross-entropy loss in the latent spaces as follows:

$$\mathcal{L}_{dom}(\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}; D, G_{\mathcal{S}}, G_{\mathcal{T}}) = -\frac{1}{|\mathcal{H}_{\mathcal{S}}|} \sum_{h_{i,t} \in \mathcal{H}_{\mathcal{S}}} \log D(h_{i,t}) - \frac{1}{|\mathcal{H}_{\mathcal{T}}|} \sum_{h_{i,t} \in \mathcal{H}_{\mathcal{T}}} \log\left[1 - D(h_{i,t})\right], \tag{6.4}$$

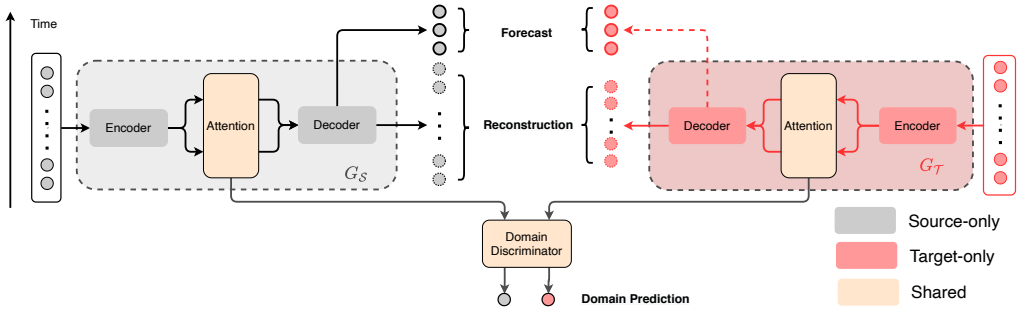


Figure 6.2: An architectural overview of DAF. The grey modules belong to the source domain, and red modules belong to target domain. The attention modules and domain discriminators shown in beige are shared by both domains. The model takes the historical portion of a time series as input, and produces a reconstruction of input and a forecast of the future time steps. The domain discriminator is a binary classifier, and predicts the origin of an intermediate representation within the attention module, either the source or the target.

where $\mathcal{H}_S$ and $\mathcal{H}_T$ are latent feature sets associated with the source $\mathcal{D}_S$ and target $\mathcal{D}_T$, and $|\mathcal{H}|$ denotes the cardinality of a set $\mathcal{H}$. The minimax objective equation (6.2) is optimized via adversarial training alternately. In the following subsections, we propose specific design choices for G_S, G_T and the latent features $\mathcal{H}_S, \mathcal{H}_T$ in our DAF model.

6.4 The Domain Adaptation Forecaster (DAF)

We propose a novel strategy based on attention mechanism to perform domain adaptation in forecasting. The proposed solution, the Domain Adaptation Forecaster (DAF), employs a sequence generator to process time series from each domain. Each sequence generator consists of an encoder, an attention module and a decoder. As each domain provides data with distinct patterns from different spaces, we keep the encoders and decoders privately owned by the respective domain. The core attention module is shared by both domains for adaptation. In addition to computing future predictions, the generator also reconstructs the input to further guarantee the effectiveness of the learned representations. Figure 6.2 illustrates an overview of the proposed architecture.

6.4.1 Sequence Generators

In this subsection, we discuss our design of the sequence generators $G_{\mathcal{S}}, G_{\mathcal{T}}$ in equation (6.2). Since the generators for both domains have the same architecture, we omit the domain index of all quantities and denote either generator by G in the following paragraphs by default. The generator G in each domain processes an input time series $\mathbf{X} = [z_t]_{t=1}^T$ and generates the reconstructed sequence $\hat{\mathbf{X}}$ and the predicted future $\hat{\mathbf{Y}}$.

Private Encoders The private encoder transforms the raw input $\mathbf{X}$ into the pattern embedding $\mathbf{P} = [\mathbf{p}_t]_{t=1}^T$ and value embedding $\mathbf{V} = [\mathbf{v}_t]_{t=1}^T$. For the value embedding, we apply a position-wise MLP with parameter $\boldsymbol{\theta}_v$ to encode input $\mathbf{X} = [z_t]_{t=1}^T$

$$\mathbf{v}_t = \mathrm{MLP}(z_t; \boldsymbol{\theta}_v).$$

For the pattern embedding $\mathbf{P}$, we apply M independent temporal convolutions with various kernel sizes in order to extract short-term patterns at different scales. Specifically, for $j = 1, \ldots, M$, each convolution with parameter $\boldsymbol{\theta}_p$ takes the input $\mathbf{X}$ to give a sequence of local representations,

$$\mathbf{p}^j = \mathrm{Conv}\left(\mathbf{X}; \boldsymbol{\theta}_p^j\right).$$

We concatenate each $\mathbf{p}_t^j$ to build a multi-scale pattern embedding $\mathbf{p}_t = [\mathbf{p}_t^j]_{j=1}^M$ and $\mathbf{P} = [\mathbf{p}_t]_{t=1}^T$ with parameters $\boldsymbol{\theta}_p = [\boldsymbol{\theta}_p^j]_{j=1}^M$ accordingly. To avoid dimension issues from the concatenation, we keep the dimension of $\mathbf{P}$ and $\mathbf{V}$ the same. The extracted pattern $\mathbf{P}$ and value $\mathbf{V}$ are fed into the shared attention module.

Shared Attention Module We design the attention module to be shared by both domains since its primary task is to generate domain-invariant queries $\mathbf{Q}$ and keys $\mathbf{K}$ from pattern embeddings $\mathbf{P}$ for both source and target domains. Formally, we project $\mathbf{P}$

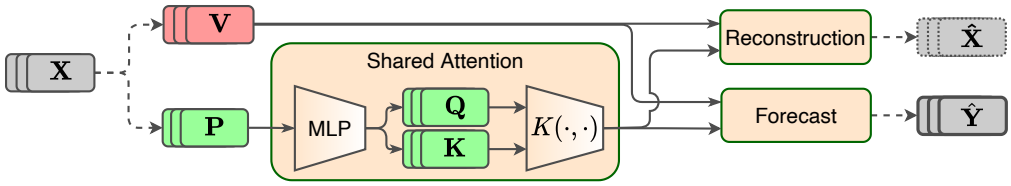


Figure 6.3: In DAF, the shared attention module processes pattern and value embeddings from either domain. A kernel function encodes pattern embeddings to a shared latent space for weight computation. We combine value embeddings by different groups of weights to obtain the interpolation $t \leq T$ for reconstruction $\hat{\mathbf{X}}$ and the extrapolation $t = T + 1$ for the forecast $\hat{\mathbf{Y}}$.

into d-dimensional queries $\mathbf{Q} = [\mathbf{q}_t]_{t=1}^T$ and keys $\mathbf{K} = [\mathbf{k}_t]_{t=1}^T$ via a position-wise MLP

$$(\mathbf{q}_t, \mathbf{k}_t) = \text{MLP}(\mathbf{p}_t; \boldsymbol{\theta}_s).$$

As a result, the patterns from both domains are projected into a common space, which is later induced to be domain-invariant via adversarial training. At time t, an attention score α is computed as the normalized alignment between the query $\mathbf{q}_t$ and keys $\mathbf{k}_{t'}$ at neighborhood positions $t' \in \mathcal{N}(t)$ using a positive semi-definite kernel $\mathcal{K}(\cdot, \cdot)$,

$$\alpha(\mathbf{q}_t, \mathbf{k}_{t'}) = \frac{\mathcal{K}(\mathbf{q}_t, \mathbf{k}_{t'})}{\sum_{t' \in \mathcal{N}(t)} \mathcal{K}(\mathbf{q}_t, \mathbf{k}_{t'})}, \tag{6.5}$$

e.g. an exponential scaled dot-product $\mathcal{K}(\mathbf{q}, \mathbf{k}) = \exp\left(\frac{\mathbf{q}^T \mathbf{k}}{\sqrt{d}}\right)$. Then, a representation $\mathbf{o}_t$ is produced as the average of values $\mathbf{v}_{\mu(t')}$ weighted by attention score $\alpha(\mathbf{q}_t, \mathbf{k}_{t'})$ on neighborhood $\mathcal{N}(t)$, followed by a MLP with parameter $\boldsymbol{\theta}_o$:

$$\mathbf{o}_t = \text{MLP}\left(\sum_{t' \in \mathcal{N}(t)} \alpha(\mathbf{q}_t, \mathbf{k}_{t'}) \mathbf{v}_{\mu(t')}; \boldsymbol{\theta}_o\right), \tag{6.6}$$

where $\mu : \mathbb{N} \to \mathbb{N}$ is a position translation. The choice of $\mathcal{N}(t)$ and $\mu(t)$ depends on whether G is in interpolation mode for reconstruction when $t \leq T$ or extrapolation mode for forecasting when $t > T$.

Private Decoders The private decoder produces prediction $\hat{z}_t$ out of $\mathbf{o}_t$ through another position-wise MLP: $\hat{z}_t = \text{MLP}(\mathbf{o}_t; \boldsymbol{\theta}_d)$. By doing so, we can generate reconstructions

$\hat{\mathbf{X}} = [\hat{z}_t]_{t=1}^{T}$ and the one-step prediction $\hat{z}_{T+1}$. This prediction $\hat{z}_{T+1}$ is fed back into the encoder and attention model to predict the next one-step ahead prediction. We recursively feed the prior predictions to generate the predictions $\hat{\mathbf{Y}} = [\hat{z}_t]_{t=T+1}^{T+\tau}$ over τ time steps.

6.4.2 Domain Discriminator

In order to induce the queries and keys of the attention module to be domain-invariant, a domain discriminator is introduced to classify the origin of a given query or key. We employ a position-wise MLP $D : \mathbb{R}^d \to [0, 1]$:

$$D(\mathbf{q}_t) = \text{MLP}(\mathbf{q}_t; \boldsymbol{\theta}_D), \;\; D(\mathbf{k}_t) = \text{MLP}(\mathbf{k}_t; \boldsymbol{\theta}_D).$$

The discriminator D performs binary classifications on whether $\mathbf{q}_t$ and $\mathbf{k}_t$ originate from the source or target domain by minimizing the cross entropy loss of $\mathcal{L}_{dom}$ in equation (6.4). We design the latent features $\mathcal{H}_{\mathcal{S}}, \mathcal{H}_{\mathcal{T}}$ in equation (6.4) to be the keys $\mathbf{K} = [\mathbf{k}_t]_{t=1}^{T+\tau}$ and queries $\mathbf{Q} = [\mathbf{q}_t]_{t=1}^{T+\tau}$ in both source and target domains, respectively.

6.4.3 Adversarial Training

Recall we have defined generators $G_{\mathcal{S}}, G_{\mathcal{T}}$ based on the private encoder/decoder and the shared attention module. The discriminator D induces the invariance of latent features keys $\mathbf{K}$ and queries $\mathbf{Q}$ across domains. While D tries to classify the domain between source and target, $G_{\mathcal{S}}, G_{\mathcal{T}}$ are trained to confuse D. By choosing the MSE loss for l, the minimax objective in equation (6.2) is now formally defined over generators $G_{\mathcal{S}}, G_{\mathcal{T}}$ with parameters $\boldsymbol{\Theta}_G = \{\boldsymbol{\theta}_p^{\mathcal{S}}, \boldsymbol{\theta}_v^{\mathcal{S}}, \boldsymbol{\theta}_d^{\mathcal{S}}, \boldsymbol{\theta}_p^{\mathcal{T}}, \boldsymbol{\theta}_v^{\mathcal{T}}, \boldsymbol{\theta}_d^{\mathcal{T}}, \boldsymbol{\theta}_s, \boldsymbol{\theta}_o\}$ and domain discriminator D with parameter $\boldsymbol{\theta}_D$. Algorithm 1 summarizes the training routine of DAF. We alternately update $\boldsymbol{\Theta}_G$ and $\boldsymbol{\theta}_D$ in opposite directions so that $G = \{G_{\mathcal{S}}, G_{\mathcal{T}}\}$ and D are trained

Algorithm 1 Adversarial Training of DAF

Input: dataset $\mathcal{D}_{\mathcal{S}}$, $\mathcal{D}_{\mathcal{T}}$; epochs E, step sizes

Initialization: parameter $\boldsymbol{\Theta}_G$ for generator $G_{\mathcal{S}}$, $G_{\mathcal{T}}$, parameter $\boldsymbol{\theta}_D$ for discriminator D

for epoch $= 1$ **to** E **do**

  repeat

    sample $\mathbf{X}_{\mathcal{S}}, \mathbf{Y}_{\mathcal{S}} \sim \mathcal{D}_{\mathcal{S}}$ and $\mathbf{X}_{\mathcal{T}}, \mathbf{Y}_{\mathcal{T}} \sim \mathcal{D}_{\mathcal{T}}$

    generate $\hat{\mathbf{X}}_{\mathcal{S}}, \hat{\mathbf{Y}}_{\mathcal{S}} = G_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}})$ and $\hat{\mathbf{X}}_{\mathcal{T}}, \hat{\mathbf{Y}}_{\mathcal{T}} = G_{\mathcal{T}}(\mathbf{X}_{\mathcal{T}})$

    compute $\mathcal{L}_{seq}$ in equation (6.3) for $\mathcal{S}$ and $\mathcal{T}$, $\mathcal{L}_{dom}$ in equation (6.4) , and total $\mathcal{L}$ in equation (6.2)

    gradient descent with $\nabla_{\boldsymbol{\Theta}_G}\mathcal{L}$ to update $G_{\mathcal{S}}$, $G_{\mathcal{T}}$

    gradient ascent with $\nabla_{\boldsymbol{\theta}_D}\mathcal{L}$ to update D

  until $\mathcal{D}_{\mathcal{T}}$ is exhausted

end for

adversarially. Here, we use a standard pre-processing for $\mathbf{X}, \mathbf{Y}$ and post-processing for $\hat{\mathbf{X}}, \hat{\mathbf{Y}}$.

6.5 Experiments

6.5.1 Baselines and Evaluation

In the experiments, we compare DAF with the following single-domain and cross-domain baselines. The conventional single-domain forecasters trained only on the target domain include:

- DAR: DeepAR [39];
- VT: Vanilla Transformer [122];

- AttF: the sequence generator $G_{\mathcal{T}}$ for the target domain trained by minimizing $\mathcal{L}_{seq}(\mathcal{D}_{\mathcal{T}}; G_{\mathcal{T}})$ in equation (6.2).

The cross-domain forecasters trained on both source and target domain include:

- DATSING: pretrained and finetuned forecaster [60];
- RDA: RNN-based DA forecaster obtained by replacing the attention module in DAF with a LSTM module and inducing the domain-invariance of LSTM encodings. Specifically, we consider three variants:
  - RDA-DANN: adversarial DA via gradient reversing [44];
  - RDA-ADDA: adversarial DA via GAN-like optimization [121];
  - RDA-MMD: metric based DA via minimizing MMD between LSTM encodings [74].

We implement the models using PyTorch [93], and train them on AWS Sagemaker [79]. For DAR, we call the publicly available version on Sagemaker. The hyperparameter of DAF and the baselines are tuned on a held-out validation set.

We evaluate the forecasting error in terms of the Normalized Deviation (ND) [144]:

$$\text{ND} = \left(\sum_{i=1}^{N} \sum_{t=T+1}^{T+\tau} |z_{i,t} - \hat{z}_{i,t}| \right) / \left(\sum_{i=1}^{N} \sum_{t=T+1}^{T+\tau} |z_{i,t}| \right),$$

where $\mathbf{Y}_i = [z_{i,t}]_{t=T+1}^{T+\tau}$ and $\hat{\mathbf{Y}}_i = [\hat{z}_{i,t}]_{t=T+1}^{T+\tau}$ denote the ground truths and predictions, respectively.

6.5.2 Real-World Datasets

We perform experiments on four real benchmark datasets that are widely used in forecasting literature: *elec* and *traf* from the UCI data repository [35], *sales* [64] and

$\mathcal{D}_\mathcal{T}$	traf		elec		wiki		sales	
$\mathcal{D}_\mathcal{S}$	elec	wiki	traf	sales	sales	traf	wiki	elec
τ	24				7			
DAR	0.205		0.141		0.055		0.305	
VT	0.187		0.144		0.061		0.293	
AttF	0.182		0.137		0.050		0.308	
DATSING	0.184	0.189	0.137	0.149	0.049	0.052	0.301	0.305
RDA-DANN	0.181	0.180	0.133	0.135	0.047	0.053	0.297	0.287
RDA-ADDA	0.174	0.181	0.134	0.142	0.045	**0.049**	0.281	0.287
RDA-MMD	0.186	0.179	0.140	0.144	0.045	0.052	0.291	0.289
DAF	**0.169**	**0.176**	**0.125**	**0.123**	**0.042**	**0.049**	**0.277**	**0.280**

Table 6.1: Performance comparison of DAF on real-world benchmark datasets with prediction length τ in the target domain in terms of the mean +/- standard deviation ND metric. The winners and the competitive followers (the gap is smaller than its standard deviation over 5 runs) are bolded for reference.

wiki [69] from Kaggle. Notably, the *elec* and *traf* datasets present clear daily and weekly patterns while *sales* and *wiki* are less regular and more challenging. We use the following time features $\xi_t \in \mathbb{R}^2$ as covariates: the day of the week and hour of the day for the hourly datasets *elec* and *traf*, and the day of the month and day of the week for the daily datasets *sales* and *wiki*.

To evaluate the performance of DAF, we consider cross-dataset adaptation, i.e., transferring between a pair of datasets. Since the original datasets are large enough to train a reasonably good forecaster, we only take a subset of each dataset as a target domain to simulate the data-scarce situation. Specifically, we take the last 30 days of each time series in the hourly dataset *elec* and *traf*, and the last 60 days from daily dataset *sales* and *wiki*. We partition the target datasets equally into training/validation/test splits, i.e. 10/10/10 days for hourly datasets and 20/20/20 days for daily datasets. The full datasets are used as source domains in adaptation. We follow the rolling window strategy from [39], and split each window into historical and prediction time series of lengths T and $T + \tau$, respectively. In our experiments, we set $T = 168, \tau = 24$ for hourly datasets,

and $T = 28, \tau = 7$ for the daily datasets. For DA methods, the splitting of the source data follows analogously.

Table 6.1 shows that DAF outperforms all the baselines. The real-world experiments also demonstrate that in general the success of DAF is agnostic of the source domain, and is even effective when transferring from a source domain of different frequency than that of the target domain. In addition, the cross-domain forecasters, DATSING, the RDA variants and our DAF outperform the three single-domain baselines in most cases. As in the synthetic cases, DATSING performs relatively worse than RDA and DAF. The accuracy differences between DAF and RDA are larger than in the synthetic case, and in favor of DAF. This finding further demonstrates that our choice of an attention-based architecture is well-suited for real domain adaptation problems.

6.5.3 Ablation Studies

In order to examine the effectiveness of our designs, we conduct ablation studies by adjusting each key component successively. We consider the following DAF variants:

- no-adv: no domain discriminator or adversarial training;
- no-q-share: no sharing of queries across domains;
- no-k-share: no sharing of keys across domains;
- v-share: shares values across domains.

Table 6.2 shows the improved performance of DAF over its variants on the target domain on four adaptation tasks. Equipped with a domain discriminator, DAF improves its effectiveness of adaptation compared to its non-adversarial variant (*no-adv*). We see that sharing both keys and queries in DAF results in performance gains over not sharing either (*no-k-share* and *no-q-share*). Furthermore, it is clear that our design choice of the

$\mathcal{D}_\mathcal{T}$	$\mathcal{D}_\mathcal{S}$	τ	*no-adv*	*no-q-share*	*no-k-share*	*v-share*	DAF
traf	*elec*	24	0.172	0.171	0.172	0.176	**0.168**
elec	*traf*	24	0.121	0.122	0.120	0.127	**0.119**
wiki	*sales*	7	0.042	0.042	0.044	0.049	**0.041**
sales	*wiki*	7	0.294	0.283	0.282	0.291	**0.280**

Table 6.2: Results of ablation studies of DAF variants on four adaptation tasks on real-world datasets.

values to be domain-specific for domain-dependent forecasts rather than shared (*v-share*) has the largest positive impact on the performance.

6.6 Conclusions

In this work, we aim to apply domain adaptation to time series forecasting to solve the data scarcity problem. We identify the differences between the forecasting task and common domain adaptation scenarios, and accordingly propose the Domain Adaptation Forecaster (DAF) based on attention sharing. Through empirical experiments, we demonstrate that DAF outperforms state-of-the-art single-domain forecasters and various domain adaptation baselines on synthetic and real-world datasets. We further show the effectiveness of our designs via extensive ablation studies. In spite of empirical evidences, the theoretical justification of having domain-invariant features within attention models remains an open problem.

Chapter 7

Conclusion

In this dissertation, we studied time series forecasting under modern setting with the explosion of data. Essentially, we identify the basic procedure of modern forecasting services and three key challenges that practical forecasting services need to cope with. Besides scalable models that are able to process large amount of time series data efficiently, we also demonstrate the urgent demand of taking advantage of inter-series correlations and adapting trained models to brand new domains. Accordingly, we propose solutions to address these challenges and provide preliminary evidences of their effectiveness and applicability.

The ultimate goal of forecasting research is to build a system with capability to make accurate short- or long-term forecasts, scalability to process correlated and high-dimension time series data and flexibility to be customized and adapted to the considerable variety of real scenarios. With such a system, users only need to provide structured time series data and set a few key parameters of forecasting tasks, and leave complex or tedious details to the system. In addition, it is supposed to transfer learned knowledge from related tasks to the current task when provided data is insufficient. Nevertheless, forecasting in general remains an open problem, and much research efforts are being

devoted in several direction.

Long-term forecasting Longer-term forecasts are required in certain scenarios. It is not only difficult as temporal dependencies become even more complex in long-range time series, but also expensive as time and space complexity grows along with the length of data. It is an active topic to search architectures with higher efficiency in encoding long sequences, especially those able to capture diverse patterns in multiple resolutions.

Interpretable forecasting Although current neural forecasting models are able to improve predictive accuracy by a significant margin, it is not easy to interpret the predictions, and hence not easy to analyze the underlying factors or failure cases in produced forecasts. A popular solution in this regard is to incorporate time series decomposition into neural nets by learning individual components before they are combined.

Few-shot forecasting While time series data is abundant, data of certain domains or entities may be insufficient for neural nets to learn. Besides domain adaptation techniques that have been introduced to alleviate the data scarcity issue, other few-shot learning and meta-learning methods can be tailored to increase versatility of models by transferring learned knowledge across related tasks.

Bibliography

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

[2] Firoj Alam, Shafiq Joty, and Muhammad Imran. Domain Adaptation with Adversarial Training and Graph Embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1077–1087, Melbourne, Australia, 2018. Association for Computational Linguistics.

[3] Nick Altieri, Rebecca L Barter, James Duncan, Raaz Dwivedi, Karl Kumbier, Xiao Li, Robert Netzorg, Briton Park, Chandan Singh, Yan Shuo Tan, and others. Curating a COVID-19 data repository and forecasting county-level death counts in the United States. *arXiv preprint arXiv:2005.07882*, 2020.

[4] Sercan O Arık, Chun-Liang Li, Jinsung Yoon, Rajarishi Sinha, Arkady Epshteyn, Yash Sonthalia, Hootan Nakhost, Elli Kanal, and Tomas Pfister. Interpretable Sequence Learning for COVID-19 Forecasting. volume 33, page 12, 2020.

[5] Dheeraj Baby and Yu-Xiang Wang. Online Forecasting of Total-Variation-bounded Sequences. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11071–11081. Curran Associates, Inc., 2019.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv:1803.01271 [cs]*, March 2018. arXiv: 1803.01271.

[8] Duygu Balcan, Vittoria Colizza, Bruno Gonçalves, Hao Hu, José J Ramasco, and Alessandro Vespignani. Multiscale mobility networks and the spatial spreading of infectious diseases. *Proceedings of the National Academy of Sciences*, 106(51):21484–21489, 2009. Publisher: National Acad Sciences.

[9] Duygu Balcan, Bruno Gonçalves, Hao Hu, José J Ramasco, Vittoria Colizza, and Alessandro Vespignani. Modeling the spatial spread of infectious diseases: The GLobal Epidemic and Mobility computational model. *Journal of computational science*, 1(3):132–145, 2010. Publisher: Elsevier.

[10] Sergey Bartunov and Dmitry P Vetrov. Few-shot Generative Modelling with Generative Matching Networks. *International Conference on Artificial Intelligence and Statistics*, pages 670–678, 2018.

[11] Michèle Basseville and Igor V. Nikiforov. *Detection of abrupt changes: theory and application*, volume 104. prentice Hall Englewood Cliffs, 1993.

[12] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, May 2010.

[13] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[14] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain Separation Networks. volume 29, pages 345–351, 2016.

[15] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *arXiv:1511.06349 [cs]*, May 2016. arXiv: 1511.06349.

[16] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[17] E. Brodsky and Boris S. Darkhovsky. *Nonparametric methods in change point problems*, volume 243. Springer Science & Business Media, 2013.

[18] Logan C Brooks, David C Farrow, Sangwon Hyun, Ryan J Tibshirani, and Roni Rosenfeld. Flexible modeling of epidemics with an empirical Bayes framework. *PLoS Comput Biol*, 11(8):e1004382, 2015. Publisher: Public Library of Science.

[19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and others. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[20] Ruichu Cai, Jiawei Chen, Zijian Li, Wei Chen, Keli Zhang, Junjian Ye, Zhuozhang Li, Xiaoyan Yang, and Zhenjie Zhang. Time Series Domain Adaptation via Sparse Associative Structure Alignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:6859–6867, 2021.

[21] Prithwish Chakraborty, Pejman Khadivi, Bryan Lewis, Aravindan Mahendiran, Jiangzhuo Chen, Patrick Butler, Elaine O Nsoesie, Sumiko R Mekaru, John S Brownstein, Madhav V Marathe, and others. Forecasting a moving target: Ensemble models for ILI case count predictions. In *Proceedings of the 2014 SIAM international conference on data mining*, pages 262–270. SIAM, 2014.

[22] Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. Kernel Change-Point Detection with Auxilliary Deep Generative Models. page 14, 2019.

[23] Zhengping Che, Sanjay Purushotham, Guangyu Li, Bo Jiang, and Yan Liu. Hierarchical deep generative models for multi-rate multivariate time series. In *International Conference on Machine Learning*, pages 784–793. PMLR, 2018.

[24] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A Closer Look at Few-shot Classification. *arXiv:1904.04232 [cs]*, January 2020. arXiv: 1904.04232.

[25] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[26] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A Recurrent Latent Variable Model for Sequential Data. *arXiv:1506.02216 [cs]*, June 2015. arXiv: 1506.02216.

[27] Corinna Cortes and Mehryar Mohri. Domain Adaptation in Regression. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, volume 6925, pages 308–323. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. Series Title: Lecture Notes in Computer Science.

[28] Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3733–3742, 2017.

[29] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv:1901.02860 [cs, stat]*, January 2019. arXiv: 1901.02860.

[30] Andreas C Damianou, Michalis K Titsias, and Neil D Lawrence. Variational Gaussian process dynamical systems. *Adavances in Neural Information Processing Systems*, pages 2510–2518, 2011.

[31] Sakyasingha Dasgupta and Takayuki Osogami. Nonlinear dynamic Boltzmann machines for time-series prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. Issue: 1.

[32] Songgaojun Deng, Shusen Wang, Huzefa Rangwala, Lijing Wang, and Yue Ning. Graph message passing with cross-location attentions for long-term ILI prediction. *arXiv preprint arXiv:1912.10202*, 2019.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019. arXiv: 1810.04805.

[34] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003. Publisher: Springer.

[35] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2017.

[36] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*, volume 38. OUP Oxford, 2012.

[37] Zahra Ebrahimzadeh and Samantha Kleinberg. Multi-scale change point detection in multivariate time series. In *NIPS Time Series Workshop*, 2017.

[38] Chenyou Fan, Heng Huang, Yuze Zhang, Yi Pan, Xiaoyue Li, Chi Zhang, Rong Yuan, Di Wu, Wensheng Wang, and Jian Pei. Multi-Horizon Time Series Forecasting with Temporal Attention Learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19*, pages 2527–2535, Anchorage, AK, USA, 2019. ACM Press.

[39] Valentin Flunkert, David Salinas, and Jan Gasthaus. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *International Journal of Forecasting*, 36:1181–1191, 2020. arXiv: 1704.04110.

[40] Ian Fox, Lynn Ang, Mamta Jaiswal, Rodica Pop-Busui, and Jenna Wiens. Deep Multi-Output Forecasting: Learning to Accurately Predict Blood Glucose Trajectories. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18*, pages 1387–1395, London, United Kingdom, 2018. ACM Press.

[41] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential Neural Models with Stochastic Layers. *arXiv:1605.07571 [cs, stat]*, May 2016. arXiv: 1605.07571.

[42] Katerina Fragkiadaki, Sergey Levine, and Jitendra Malik. Recurrent network models for kinematic tracking. *CoRR, abs/1508.00271*, 1(2):4, 2015.

[43] Yaroslav Ganin and Victor Lempitsky. Unsupervised Domain Adaptation by Backpropagation. *International conference on machine learning*, pages 1180–1189, 2015.

[44] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *The journal of machine learning research*, 17:2096–2030, 2016.

[45] Junyi Gao, Rakshith Sharma, Cheng Qian, Lucas M Glass, Jeffrey Spaeder, Justin Romberg, Jimeng Sun, and Cao Xiao. STAN: Spatio-Temporal Attention Network for Pandemic Prediction Using Real World Evidence. *arXiv preprint arXiv:2008.04215*, 2020.

[46] Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional Neural Processes. *arXiv:1807.01613 [cs, stat]*, July 2018. arXiv: 1807.01613.

[47] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR, 2017.

[48] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep Reconstruction-Classification Networks for Unsupervised Domain Adaptation. In *European conference on computer vision*, pages 597–613, 2016.

[49] Luke B. Godfrey and Michael S. Gashler. Neural Decomposition of Time-Series Data for Effective Generalization. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2017. arXiv: 1705.09137.

[50] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[51] Youyang Gu. COVID-19 projections using machine learning.

[52] Han Guo, Ramakanth Pasunuru, and Mohit Bansal. Multi-Source Domain Adaptation for Text Classification via DistanceNet-Bandits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7830–7838, April 2020.

[53] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the*

Association for Computational Linguistics, pages 8342–8360, Online, 2020. Association for Computational Linguistics.

[54] Xiaochuang Han and Jacob Eisenstein. Unsupervised Domain Adaptation of Contextualized Embeddings for Sequence Labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4237–4247, Hong Kong, China, 2019. Association for Computational Linguistics.

[55] Tiberiu Harko, Francisco SN Lobo, and MK Mak. Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model with equal death and birth rates. *Applied Mathematics and Computation*, 236:184–194, 2014. Publisher: Elsevier.

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

[57] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. *International conference on machine learning*, pages 1989–1998, 2018.

[58] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

[59] Charles C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004. Publisher: Elsevier.

[60] Hailin Hu, MingJian Tang, and Chengcheng Bai. DATSING: Data Augmented Time Series Forecasting with Adversarial Domain Adaptation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2061–2064, Virtual Event Ireland, October 2020. ACM.

[61] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. *Music Transformer*. 2018.

[62] Chiou-Jye Huang, Yamin Shen, Ping-Huan Kuo, and Yung-Hsiang Chen. Novel Spatiotemporal Feature Extraction Parallel Deep Neural Network for Forecasting Confirmed Cases of Coronavirus Disease 2019. *medRxiv*, 2020. Publisher: Cold Spring Harbor Laboratory Press.

[63] Robin John Hyndman and George Athanasopoulos. Forecasting: Principles and Practice. 2013.

[64] Pratyusha Kar. Dataset of Kaggle Competition Rossmann Store Sales, version 2, January 2019.

[65] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, 2018.

[66] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, December 2013. arXiv: 1312.6114.

[67] R. Koenker and K. F. Hallock. Quantile Regression. *Journal of Economic Perspectives*, 15(4):143–156, 2001.

[68] Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep Kalman Filters. *arXiv:1511.05121 [cs, stat]*, November 2015. arXiv: 1511.05121.

[69] Lai. Dataset of Kaggle Competition Web Traffic Time Series Forecasting, Version 3, August 2017.

[70] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. *arXiv:1703.07015 [cs]*, April 2018. arXiv: 1703.07015.

[71] LANL COVID-19. COVID-19 Confirmed and Forecasted Case Data, 2020.

[72] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[73] Justin Lessler and Derek AT Cummings. Mechanistic models of infectious disease and their impact on public health. *American journal of epidemiology*, 183(5):415–422, 2016. Publisher: Oxford University Press.

[74] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: towards deeper understanding of moment matching network. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2200–2210, 2017.

[75] Rui Li, Tai-Peng Tian, and Stan Sclaroff. Simultaneous learning of nonlinear manifold and dynamical models for high-dimensional time series. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[76] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *Advances in Neural Information Processing Systems*, pages 5243–5253, 2019.

[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] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*, 2018.

[79] Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, Can Balioglu, Saswata Chakravarty, Madhav Jha, Philip Gautier, David Arpin, Tim Januschowski, Valentin Flunkert, Yuyang Wang, Jan Gasthaus, Lorenzo Stella, Syama Rangapuram, David Salinas, Sebastian Schelter, and Alex Smola. Elastic Machine Learning Algorithms in Amazon SageMaker. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 731–737, Portland OR USA, June 2020. ACM.

[80] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *arXiv:1912.09363 [cs, stat]*, December 2019. arXiv: 1912.09363.

[81] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating Wikipedia by Summarizing Long Sequences. In *International Conference on Learning Representations*, 2018.

[82] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning Transferable Features with Deep Adaptation Networks. pages 97–105, 2015. arXiv: 1502.02791.

[83] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[84] Edson Zangiacomi Martinez, Elisângela Aparecida Soares da Silva, and Amaury Lelis Dal Fabbro. A SARIMA forecasting model to predict the number of cases of dengue in Campinas, State of São Paulo, Brazil. *Revista da Sociedade Brasileira de Medicina Tropical*, 44(4):436–440, 2011. Publisher: SciELO Brasil.

[85] Dhendra Marutho, Sunarna Hendra Handaka, Ekaprana Wijaya, and others. The determination of cluster number at k-mean using elbow method and purity evaluation on headline news. In *2018 International Seminar on Application for Technology of Information and Communication*, pages 533–538. IEEE, 2018.

[86] Saeid Motiian, Quinn Jones, Seyed Mehdi Iranmanesh, and Gianfranco Doretto. Few-Shot Adversarial Domain Adaptation. *Advances in Neural Information Processing Systems*, pages 6670–6680, 2017. arXiv: 1711.02536.

[87] CJ Murray and others. Forecasting the impact of the first wave of the COVID-19 pandemic on hospital demand and deaths for the USA and European Economic Area countries. 2020.

[88] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *arXiv:1711.10433 [cs]*, November 2017. arXiv: 1711.10433.

[89] Boris N Oreshkin, Nicolas Chapados, Dmitri Carpov, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *International Conference on Learning Representations*, page 31, 2020.

[90] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, 2016.

[91] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image Transformer. *arXiv:1802.05751 [cs]*, June 2018. arXiv: 1802.05751.

[92] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[93] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in neural information processing systems*, pages 8026–8037, 2019.

[94] Vladimir Pavlovic, James M Rehg, Tat-Jen Cham, and Kevin P Murphy. A dynamic Bayesian network approach to figure tracking using learned dynamic models. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 1, pages 94–101. IEEE, 1999.

[95] Vladimir Pavlovic, James M Rehg, and John MacCormick. Learning switching linear models of human motion. In *NIPS*, volume 2, page 4, 2000. Issue: 3.

[96] Sen Pei and Jeffrey Shaman. Initial Simulation of SARS-CoV2 Spread and Intervention Effects in the Continental US. *medRxiv*, 2020. Publisher: Cold Spring Harbor Laboratory Press.

[97] Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, and Sanjeev Khudanpur. A time-restricted self-attention layer for ASR. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5874–5878. IEEE, 2018.

[98] Sanjay Purushotham, Wilka Carvalho, Tanachat Nilanon, and Yan Liu. Variational Recurrent Adversarial Deep Domain Adaptation. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[99] Ankit Ramchandani, Chao Fan, and Ali Mostafavi. DeepCOVIDNet: An Interpretable Deep Learning Model for Predictive Surveillance of COVID-19 Using Heterogeneous Features and Their Interactions. *IEEE Access*, 2020. Publisher: IEEE.

[100] Alan Ramponi and Barbara Plank. Neural Unsupervised Domain Adaptation in NLP—A Survey. *arXiv:2006.00632 [cs]*, May 2020. arXiv: 2006.00632.

[101] Syama Sundar Rangapuram, Matthias Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep State Space Models for Time Series Forecasting. In *Advances in neural information processing systems*, pages 7785–7794, 2018.

[102] Evan L Ray, Krzysztof Sakrejda, Stephen A Lauer, Michael A Johansson, and Nicholas G Reich. Infectious disease prediction with kernel conditional density estimation. *Statistics in medicine*, 36(30):4908–4929, 2017. Publisher: Wiley Online Library.

[103] Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. Adapt or Get Left Behind: Domain Adaptation through BERT Language Model Finetuning for Aspect-Target Sentiment Classification. *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4933–4941, 2020.

[104] Cleveland Robert, C William, and Terpenning Irma. STL: A seasonal-trend decomposition procedure based on loess. *Journal of official statistics*, 6(1):3–73, 1990.

[105] Alexander Rodriguez, Anika Tabassum, Jiaming Cui, Jiajia Xie, Javen Ho, Pulak Agarwal, Bijaya Adhikari, and B Aditya Prakash. DeepCOVID: An Operational Deep Learning-driven Framework for Explainable Real-time COVID-19 Forecasting. *medRxiv*, 2020. Publisher: Cold Spring Harbor Laboratory Press.

[106] Yunus Saatçi, Ryan Turner, and Carl Edward Rasmussen. Gaussian process change point models. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 927–934, 2010.

[107] Matthias W Seeger, David Salinas, and Valentin Flunkert. Bayesian intermittent demand forecasting for large inventories. In *Advances in Neural Information Processing Systems*, pages 4646–4654, 2016.

[108] Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon. Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting. *Advances in Neural Information Processing Systems*, 32, 2019.

[109] Yuanjie Shao, Lerenhan Li, Wenqi Ren, Changxin Gao, and Nong Sang. Domain Adaptation for Image Dehazing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2805–2814, Seattle, WA, USA, June 2020. IEEE.

[110] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, 2018.

[111] Ge Shi, Chong Feng, Lifu Huang, Boliang Zhang, Heng Ji, Lejian Liao, and Heyan Huang. Genre Separation Network with Adversarial Training for Cross-genre Relation Extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1018–1023, Brussels, Belgium, 2018. Association for Computational Linguistics.

[112] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020. Publisher: Elsevier.

[113] Huan Song, Deepta Rajan, Jayaraman J Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[114] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016. Issue: 1.

[115] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[116] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2007.

[117] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018. Publisher: Taylor & Francis.

[118] Ting Tian, Yukang Jiang, Yuting Zhang, Zhongfei Li, Xueqin Wang, and Heping Zhang. COVID-Net: A deep learning based and interpretable predication model for the county-wise trajectories of COVID-19 in the United States. *medRxiv*, 2020. Publisher: Cold Spring Harbor Laboratory Press.

[119] Yuan Tian, Ishika Luthra, and Xi Zhang. Forecasting COVID-19 cases using Machine Learning models. *medRxiv*, 2020. Publisher: Cold Spring Harbor Laboratory Press.

[120] Eric Tzeng, Judy Hoffman, Trevor Darrell, Kate Saenko, and UMass Lowell. Simultaneous Deep Transfer Across Domains and Tasks. *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015.

[121] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial Discriminative Domain Adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2962–2971, Honolulu, HI, July 2017. IEEE.

[122] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[123] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models. In *NIPS*, volume 18, page 3. Citeseer, 2005.

[124] Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. Multilevel wavelet decomposition network for interpretable time series analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2437–2446, 2018.

[125] Lijing Wang, Jiangzhuo Chen, and Madhav Marathe. DEFSI: Deep learning based epidemic forecasting with synthetic information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9607–9612, 2019.

[126] Rui Wang, Danielle Maddix, Christos Faloutsos, Yuyang Wang, and Rose Yu. Bridging Physics-based and Data-driven modeling for Learning Dynamical Systems. In *Learning for Dynamics and Control*, pages 385–398, 2021.

[127] Wen Wang, Pieter HAJM Van Gelder, and JK Vrijling. Some issues about the generalization of neural networks for time series prediction. In *International Conference on Artificial Neural Networks*, pages 559–564. Springer, 2005.

[128] Yuyang Wang, Alex Smola, Danielle C Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep Factors for Forecasting. In *International conference on machine learning*, pages 6607–6617, 2019.

[129] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A Multi-Horizon Quantile Recurrent Forecaster. *arXiv:1711.11053 [stat]*, November 2017. arXiv: 1711.11053.

[130] Garrett Wilson and Diane J. Cook. A Survey of Unsupervised Deep Domain Adaptation. *arXiv:1812.02849 [cs, stat]*, February 2020. arXiv: 1812.02849.

[131] Garrett Wilson, Janardhan Rao Doppa, and Diane J. Cook. Multi-Source Deep Domain Adaptation with Weak Supervision for Time-Series Sensor Data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1768–1778, Virtual Event CA USA, August 2020. ACM.

[132] Spencer Woody, Mauricio Garcia Tec, Maytal Dahan, Kelly Gaither, Michael Lachmann, Spencer Fox, Lauren Ancel Meyers, and James G Scott. Projections for first-wave COVID-19 deaths across the US using social-distancing measures derived from mobile phones. *medRxiv*, 2020. Publisher: Cold Spring Harbor Laboratory Press.

[133] Dustin Wright and Isabelle Augenstein. Transformer Based Multi-Source Domain Adaptation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7963–7974, Online, 2020. Association for Computational Linguistics.

[134] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Advances in Neural Information Processing Systems*, 34, 2021.

[135] Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, Ying Wei, and Junzhou Huang. Adversarial Sparse Transformer for Time Series Forecasting. *Advances in Neural Information Processing Systems*, 33:11, 2020.

[136] Yuexin Wu, Yiming Yang, Hiroshi Nishiura, and Masaya Saitoh. Deep learning for epidemiological predictions. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1085–1088, 2018.

[137] Kenji Yamanishi and Jun-ichi Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 676–681, 2002.

[138] Wan Yang, Alicia Karspeck, and Jeffrey Shaman. Comparison of filtering methods for the modeling and retrospective forecasting of influenza epidemics. *PLoS Comput Biol*, 10(4):e1003583, 2014. Publisher: Public Library of Science.

[139] Wan Yang, Jaimie Shaff, and Jeffrey Shaman. COVID-19 Transmission Dynamics and Effectiveness of Public Health Interventions in New York City during the 2020 Spring Pandemic Wave. *medRxiv*, 2020. Publisher: Cold Spring Harbor Laboratory Press.

[140] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. *arXiv:1711.03953 [cs]*, November 2017. arXiv: 1711.03953.

[141] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[142] Zhilin Yang, Thang Luong, Russ R Salakhutdinov, and Quoc V Le. Mixtape: Breaking the softmax bottleneck efficiently. *Advances in Neural Information Processing Systems*, 32, 2019.

[143] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. *arXiv:1702.08139 [cs]*, February 2017. arXiv: 1702.08139.

[144] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. *NIPS*, pages 847–855, 2016.

[145] Rose Yu, Stephan Zheng, Anima Anandkumar, and Yisong Yue. Long-term Forecasting using Tensor-Train RNNs. page 10, 2017.

[146] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62, 1998. Publisher: Elsevier.

[147] Qian Zhang, Nicola Perra, Daniela Perrotta, Michele Tizzoni, Daniela Paolotti, and Alessandro Vespignani. Forecasting seasonal influenza fusing digital indicators and a mechanistic disease model. In *Proceedings of the 26th international conference on world wide web*, pages 311–319, 2017.

[148] Han Zhao, Shanghang Zhang, Guanhang Wu, José M F Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial Multiple Source Domain Adaptation. *Advances in neural information processing systems*, pages 8559–8570, 2018.

[149] 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*, 2021. arXiv: 2012.07436.

[150] Christoph Zimmer and Reza Yaesoubi. Influenza Forecasting Framework based on Gaussian Processes. In *International Conference on Machine Learning*, pages 11671–11679. PMLR, 2020.

[151] Difan Zou, Lingxiao Wang, Pan Xu, Jinghui Chen, Weitong Zhang, and Quanquan Gu. Epidemic Model Guided Machine Learning for COVID-19 Forecasts in the United States. *medRxiv*, 2020. Publisher: Cold Spring Harbor Laboratory Press.