SPATIO-TEMPORAL FORECASTING OVER GRAPHS WITH DEEP LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Emir Ceyani
December 2020

Spatio-Temporal Forecasting Over Graphs with Deep Learning
By Emir Ceyani
December 2020

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Süleyman Serdar Kozat(Advisor)

Dr. Salih Ergüt (Co-Advisor)

Prof. Dr. Sinan Gezici

Prof. Dr. Şeref Sağıroğlu

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Ezhan Karaşan
Director of the Graduate School

ABSTRACT

SPATIO-TEMPORAL FORECASTING OVER GRAPHS WITH DEEP LEARNING

Emir Ceyani
M.S. in Electrical and Electronics Engineering
Advisor: Prof. Dr. Süleyman Serdar Kozat
Co-Advisor: Dr. Salih Ergüt
December 2020

We study spatiotemporal forecasting of high-dimensional rectangular grid graph structured data, which exhibits both complex spatial and temporal dependencies. In most high-dimensional spatiotemporal forecasting scenarios, deep learning-based methods are widely used. However, deep learning algorithms are over-confident in their predictions, and this overconfidence causes problems in the human-in-the-loop domains such as medical diagnosis and many applications of 5^{th} generation wireless networks. We propose spatiotemporal extensions to variational autoencoders for regularization, robustness against out-of data distribution, and incorporating uncertainty in predictions to resolve overconfident predictions. However, variational inference methods are prone to biased posterior approximations due to using explicit exponential family densities and mean-field assumption in their posterior factorizations. To mitigate these problems, we utilize variational inference & learning with semi-implicit distributions and apply this inference scheme into convolutional long-short term memory networks(ConvLSTM) for the first time in the literature. In chapter 3, we propose variational autoencoders with convolutional long-short term memory networks, called *VarConvLSTM*. In chapter 4, we improve our algorithm via semi-implicit & doubly semi-implicit variational inference to model multi-modalities in the data distribution . In chapter 5, we demonstrate that proposed algorithms are applicable for spatiotemporal forecasting tasks, including space-time mobile traffic forecasting over Turkcell base station networks.

Keywords: Deep Learning, Generative Models, Approximate Bayesian inference, Variational inference, Convolutional Neural Networks, Recurrent Neural Networks, Spatiotemporal Modeling, Supervised Learning.

ÖZET

DERIN ÖĞRENME İLE ÇİZGELERDE UZAY ZAMANSAL TAHMİNLEME

Emir Ceyani
Elektrik ve Elektronik Muhendisligi, Yüksek Lisans
Tez Danışmanı: Prof. Dr. Süleyman Serdar Kozat
İkinci Tez Danışmanı: Dr. Salih Ergüt
Aralık 2020

Hem karmaşık uzamsal hem de zamansal bağımlılıkları içeren yüksek boyutlu uzay-zamansal verilerin tahminlemesini inceliyoruz. Derin öğrenme bazlı yöntemler çoğu yüksek boyutlu uzay-zamansal tahminleme senaryosunda yaygın olarak kullanılmaktadır. Ancak derin öğrenme algoritmalarının yüksek güvenli tahminleri, tıbbi teşhis ve 5'inci nesil kablosuz ağların birçok uygulamasında bulunduğu döngü içinde insan alanlarında sorunlara neden olabilmektedir. Sinir ağını düzenlileştirmek, veri dağılımının kapsamı dışındaki verilere karşı sağlamlık kazanmak ve sinir ağlarının tahminlerindeki aşırı güveni düşürmek için varyasyonel özkodlayıcıları uzay-zamansal veriler için literatürde ilk defa genelliyor ve böylelikle tahminlemelerimizde belirsizliğin katkısını ekleyebiliyoruz. Lakin varyasyonel yöntemler, ardıl dağılımları üstel aile yoğunlukları ve ortalama alan varsayimi ile kestirmesi nedeniyle ardıl dağılımları yanlı ve gerçek dağılımdan oldukça uzak bir şekilde tahmin eder. Bu sorunları hafifletmek için uzay-zamansal verilerde, varyasyonel çıkarım ve öğrenmeyi yarı kapalı dağılımlarla gerçekleştiriyor ve bu yöntemi evrişimli uzun-kısa süreli bellek ağlarına (ConvLSTM) literatürde ilk kez uyguluyoruz. Bölüm 3'te, literatüre *VarConvLSTM* adlı evrişimli uzun-kısa süreli bellek ağlarına sahip varyasyonel özkodlayıcıları ilk kez öneriyoruz. Bölüm 4'te, veri dağılımındaki çoklu modaliteleri modellemek için yarı örtük ve ikili yarı örtük varyasyonel çıkarım yöntemlerini uyguluyoruz. Bölüm 5'te, önerilen algoritmaların Turkcell baz istasyonu ağlarında uzay-zaman mobil trafik tahmini dahil olmak üzere uzay-zamansal tahmin görevlerinde belirsizligi için uygulanabilir olduğunu gösteriyoruz.

Anahtar sözcükler: Derin Öğrenme, Üretici Modeller, Yakınsamalı Bayesçi Kestirim, Değişimsel Kestirim, Evrişimsel Sinir Ağları, Yinelemeli Sinir Ağları, Uzay Zamansal Modelleme, Pekiştirmeli Öğrenme.

Acknowledgement

First, I would like to thank Prof. Süleyman Serdar Kozat for his wise supervision & endless support during my undergraduate years and my M.S. study, and vouching me for the 5G & Beyond Graduate Scholarship Programme. Without this support, I could not meet my co-advisor. Thus, I'd like to continue to express sincere appreciation to my co-advisor and my mentor at Turkcell, Dr. Salih Ergüt, for his excellent guidance and unwavering support throughout my studies. Thanks to his supports during Turkcell and advising me to broaden my horizons by attending Deep|Bayes19 & PAISS, I was able to conclude this thesis. Although we began to work closely late, I believe that we together lay the groundwork for some critical projects in Turkcell. I wish for a fruitful and neverending collaboration through the years.

I express my sincere gratitude to Prof. Sinan Gezici and Prof. Şeref Sağıroğlu as my examining committee members durıng the pandemic.

I would like to thank my team members of the AI5G group at Turkcell Technology, Istanbul. I acknowledge that this work is supported by the 5G and Beyond Joint Graduate Support Programme of BTK.

From the University of Edinburgh, I would like to thank Prof. Paul Patras and his students Chaoyun Zhang & Yini Fang for their collaboration with Turkcell during my final year of M.S. studies. It was hard to be a team during the pandemic, but I am optimistic that 2021 will be beneficial for our teamwork.

I would like to thank Prof. Tolga Mete Duman, Prof. Sinan Gezici, and Prof. Erdal Arikan for their contributions to my undergraduate and graduate years throughout their courses. I also would like to thank Dr. Mehmet Alper Kutay, the EEE493-494 team, and my groups A3 & C3 from the previous year for the moments as a TA in that course.

As a tradition, I express my deepest regards to Mahmut Yurt, Oğuzhan Karaahmetoğlu, Doğan Can Çiçek, and Semih Kurt for their fruitful comments

on my thesis. I wish Mahmut, Oğuzhan, and Doğan good luck for their upcoming and future Ph.D. applications. And, Mahmut, I am looking forward to seeing the result of our promise in one year!

I have been extremely fortunate to be continously contacted with my comrades Doğan Can Çiçek, Redion Xhepa, Emre Elvan, Dolunay Sabuncuoğlu, Mümtaz Torunoğlu, and our gang. I cannot forget our amusing coffee breaks and many other adventures. I also would like to thank Arda Atalık, Kordağ Kılıç, Semih Kurt, Yiğit Ertuğrul, Atakan Altıner, Ismet Koyuncu, Omer Uyar, and Altuğ Kaya for their valuable and enlightening conversations on various topics. Besides during M.S., Atakan Serbes, Barış Ardıç, and Ege Berkay Gülcan's contributions during the CS courses we took led to memorable nights at the CS building.

From my lab, I would like to give my best regards to Doğan Can, Nuri Mert, Selin, Mine, Tolga, Ersin, Kaan & Hakan, Safa, Osman Fatih, Oğuzhan, and Mustafa. Last two years in lab would be hard without Nurı Mert. I wish him good luck for his future studies.

I also want to thank Berk Arel, Berkay Ön, Benan, Berkan, Gökhan, Ekin Bircan, Yiğit Ertuğrul, Göksenin, Ali Ercan, Doğa, Arsen Berk, Berk Tinaz, Erdem Ünal, Eray Özer, and many other friends I have met in Bilkent because I believe each connection is a strength to develop ourselves.

Last but not the least, I would like to thank and express my deepest gratitude to my family for their endless support and encouragement throughout my life. I owe them everything. And, Efe, my brother, I wish you good luck with your journey in Bilkent EEE as an undergraduate. You are the new 'Ceyani' of Bilkent EEE!

Contents

List of Figures

List of Tables

List of Abbreviations

ARIMA	Auto-Regressive Integrated Moving Average
HW-ExpS	Holt-Winters Exponential Smoothing
MLP	Multi-Layer Perceptron
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
LSTM	Long-Short Term Memory
ConvLSTM	Convolutional Long-Short Term Memory
VI	Variational Inference
SIVI	Semi-Implicit Variational Inference
DSIVI	Doubly Semi-Implicit Variational Inference
ELBO	Evidence Lower Bound
VLB	Variational Lower Bound
VAE	Variational Auto-Encoder
NF	Normalizing Flows
MCMC	Markov-Chain Monte Carlo

List of Symbols and Notation

z	Scalar representation
$\mathbf{z}$	Vector representation
$\mathbf{Z}$	Matrix representation
$\mathcal{Z}$	Tensor representation
q_ϕ	Variational distribution parameterized by parameter ϕ
ϕ	Variational parameters for semi-implicit posterior distribution
ψ	Mixing distribution for semi-implicit variational posterior distribution
Ω	Variational parameters for semi-implicit prior distribution
ζ	Mixing distribution for semi-implicit variational prior distribution

Chapter 1

Introduction

Mobile devices with an internet connection have become an essential component of the 21^{st} century, skyrocketing the mobile data traffic with their applications. The reputable forecasts indicate that annual global IP traffic consumption will reach 3.3 zettabytes by 2021, and more importantly, the smartphone traffic will exceed the PC traffic by the same year [3]. Considering the dire preference towards wireless connectivity, present mobile infrastructure faces a significantly increasing capacity demand. Earlier efforts proposed to agilely provision resources [4] and distributively tackle the mobility management [5]. However, in the long term, Internet Service Providers (ISPs) should develop architectures with intelligence and heterogenity and tools capable of spawning the 5^{th} generation of mobile systems (5G) as well as must gradually meet more requirements regarding stringent end-user applications [6, 7].

It becomes increasingly important to forecast the traffic in cellular networks for the 5G systems, smart-grid systems, and dynamic network resource allocation. This task becomes especially harder due to dense traffic networks and costly monitoring with sufficient accuracy. The monitoring systems having alarm options empower engineers to better react to instantaneous changes in the traffic volume, adversely affecting the latency perceived by interactive applications. Although

the long-term forecasting methods for network traffic have demonstrated to handle this problem for wired broadband networks [8, 9], scarce attention has been received by mobile networks [10]. Moreover, the current mobile traffic forecasting mechanisms [11, 12] yield an underperformance while performing predictive modeling of time series that represents base station networks with spatial correlations. This mostly stems from neglecting those spatial correlations associated with the movements of the users. Therefore, the available systems are undesirably limited to short-term forecasting. Yet, the capability of deep learning models [13, 14, 15] can enhance learning the representations from raw data, and as a result, mobile forecasting problem [16, 17, 18] can be elongated to spatiotemporal domains .

Given sufficient training data, deep learning models can model more robust input and output relationships and provide elevated predictive accuracy. Although their promising performance, they are prone to overfitting to training set if the amount of the data is not enough. This hugely restricts the applications in the those domains where labeled data are expensive including medical applications [19, 20, 21, 22, 23], autonomous driving [24], and human-in-the-loop systems in wireless networking [25, 26, 27]. As a matter of fact, deep network architectures trained with point estimation procedures such as MLE and MAP incline to act overconfidently and, thereby, may yield inaccurate confidence intervals [28, 29]. This phenomenon frequently happens for inputs being far from the distribution of the given training data [30], so inevitably restricts the applications for decision making, such as determining the disease of the underlying patient by considering the output of such a network. In mobile networking, mobile traffic forecasting is the heart of many applications, including smart-grid systems and network resource allocation, so accounting uncertainties in wireless networking systems would increase these systems' predictive accuracy in the long run [31, 32].

A principled strategy to handle both of the limitations mentioned above is to use a Bayesian inference procedure, which infers a posterior distribution instead of performing a point estimation for the network parameters [33]. In turn, these distributions can capture the parameter uncertainty of the network and then integrate over them to attain enhanced uncertainties about the model's predictions. In addition, being Bayesian can offer other benefits including generalization

[34, 35]. Obtaining the exact posterior distributions in high dimensions is known to be notoriously hard due to the nonlinearities in neural networks, and exact posterior estimation becomes intractable [36]. Therefore, approximate Bayesian inference methods must be used.

By exploiting deep generative models, it is even possible to model complex data distributions with deep learning. Several studies have performed the of Bayesian inference for deep neural network using one of these methods: distilling SGD with Langevin Dynamics [37], Markov Chain Monte Carlo (MCMC) with Hamiltonian Dynamics [38] or deterministic techniques such as the Laplace Approximation [39], Expectation Propagation [40], Deep ensembles [41, 42], Stochastic Weight Averaging Gaussian [43, 44], and variational inference [45, 46, 13, 47].

Despite the blessings of Bayesian methods onto deep learning methods, there is inevitably a trade-off between these solutions' scalability and the unbiasedness of posterior estimators obtained with these algorithms. MCMC methods [36] are guaranteed to find the true posterior distribution but have scalability and convergence issues, which prohibits the usage of these methods. Recently, using dropout algorithms at the output layer grants neural networks to output uncertainty estimates [48, 49] but it is reported that these estimates are overconfident compared to other methods [50]. Also, this method's accuracy is very close to what we are able to get with MAP estimates. Recently, Deep Ensembles [41] and Stochastic Weight Averaging Gaussian [43, 44] methods are proposed to deal with uncertainty. Although these methods provide strong predictive accuracies, these methods are not scalable for the spatiotemporal domain either because of the need to store and train multiple models. Finally, there is a line of work using variational inference methods which casts the inference problem into an optimization problem by constructing handcrafted posteriors to match the true distribution with probability divergences [51]. Arguably, an essential component of variational inference is the flexible nature of the approximate posterior distribution, which perfectly demonstrates how well we can capture the true posterior distribution, thereby capturing our models' true uncertainty. Even though these methods are the most scalable ones with small adjustments, the main issue is that variational methods tend to be biased and underconfident due to poor prior

choices. However, recent works on variational inference have been on the use of implicit distributions, which do not have a parametric form but we can sample from them, is a promising research field in generative modeling [52, 53, 54, 55].

With this motivation, in this thesis, we explore the deep variational learning algorithms with implicit distributions to propose a solution in spatiotemporal mobile forecasting while accounting for the uncertainty and multimodalities in the mobile traffic network for the first time in the literature.

1.1 Related Work

Time series prediction models are utilized to analyze mobile traffic networks. Several time series prediction schemes specialized for mobile traffic dynamics have been proposed to interpret and forecast the dynamics [10, 12, 56]. Widely used Exponential Smoothing [12] and ARIMA [56] are linear regression models for time-series. A Holt-Winters exponential smoothing scheme proposed by [57] is used for short-term forecasting based on historical data in GSM/GPRS networks [12]. Similarly, ARIMA is employed to predict data traffic across 9,000 base stations in Shanghai [56]. These approaches estimate user traffic at individual base stations as if stations are spatially uncorrelated, which results in a significant information loss in the spatial domain [10]. These approaches can only utilize prior temporal information and require continuous long observation periods, which is not practical to perform well. Exploratory Factor Analysis [58] is recently being used to mine non-trivial spatio-temporal mobile traffic patterns for network activity profiling and land use detection.

Deep learning has become the cornerstone technique for computer vision and natural language processing domains [14, 59, 60, 61]. For the time-series domain, recurrent neural networks(RNN) like LSTM [62, 63] are the prevalent machine learning technique that outperform the traditional approaches. An advanced version of RNNs, utilizing convolutions instead of matrix multiplications, namely

ConvLSTM, is a predominant method for precipitation nowcasting [64]. 3D-ConvNets are well known for their capability of spatiotemporal feature learning from videos [65], while their time series forecasting abilities remain mostly unexplored. Recently, [66] employs two distinct autoencoders to local and global spatial features to extract spatial features from mobile traffic and subsequently use an LSTM to perform time-series predictions. However, their approach requires to train an auto-encoder for each location, which is computationally inefficient for large-scale forecasting tasks. Moreover, this approach can only perform very short-term (i.e., one step) forecasting, limiting its applicability to real deployments.

Mobile traffic estimation is a fundamental problem for many applications. Most of the tasks mentioned above are heavily reliant on deep neural networks, which can interpolate very well but are known to be notoriously overconfident in their predictions due to their low extrapolation capabilities [67]. In short, neural networks can only deal with the instances they have seen before [30]. Even though it is almost impossible for deep learning models to be perfect, a model being sure about its prediction is valuable for practitioners. In the case of high uncertainty, extensive tests, or a person dealıng hımself dırectly on the case to avoid potentially wrong results.Various methods are proposed in the literature to distill uncertainty quantification into deep learning: Variational Bayesian Neural Networks(VBNN) [68, 36, 69, 70, 71], Monte-Carlo Dropout [49, 72, 73, 48], Deep Ensembles [42, 41], and Stochastic Weighted Average Gaussian(SWAG) [43, 34, 44]. Comparison of these methods can be found in, but to be precise, we stick to use variational methods in this thesis thanks to their fast inference as in ensembling, storing multiple models, and performing MCMC sampling would be notoriously hard in the spatio-temporal domain. VBNN utilizes uncertainty inside the weights by learning the distribution of each weight in the network [68, 74, 75, 76, 77, 78], doubling the parameters. Instead of using VBNN, we perform variational inference over the latent variables that result in the aforementioned variational autoencoder (VAE) models [13, 79, 80, 81, 82], which are also more computationally efficient for the spatio-temporal domain. It is also possible to perform variational inference over functions [83], but this is future

work.

This thesis focuses on using variational generative deep learning models for spatio-temporal data to perform mobile traffic volume predictions while utilizing uncertainty quantification. Existing mobile traffic forecasting literature does not consider uncertainty in wireless networking systems. The closest work to this thesis is by Zhang et al. [18], but the difference is threefold: First, their setting is entirely deterministic and discriminative. Second, instead of estimating the whole spatio-temporal area, they predict pixel by pixel, which may lack capturing global information over the networks. Third, they use generative adversarial networks, which may be impractical to train and serve for practical purposes. Also, these methods are very hard to train.

1.2 Contributions

This thesis provides efficient and practical variational deep learning algorithms for spatio-temporal forecasting over mobile stations. Moreover, we demonstrate that incorporating uncertainty through deep probabilistic generative modeling improves the prediction quality of spatio-temporal forecasting systems. Here, we list our contributions:

- We consider modeling the uncertainty inherent in the spatio-temporal forecasts via variational autoencoders for the first time in the mobile traffic forecasting literature.
- In chapter 3. we *Variational ConvLSTM* architecture which blends convolutional neural networks with variational autoencoder and inference paradigms, which is performed for the first time in the literature. Our model is capable of generating and predicting future spatio-temporal sequencing under uncertainty with a learned prior.
- In chapter 4, we consider the variational learning of deep spatio-temporal

models under semi-implicit distributions via semi-implicit variational inference to augment the proposed Variational ConvLSTM algorithm, called *Semi Implicit Variational ConvLSTM (SI - VarConvLSTM)* for the first time in the literature.

- Besides, we also offer improving prior choice by allowing prior as a semi-implicit distribution. For this setting, we propose *Doubly Semi-Implicit Variational ConvLSTM (DSI-VarConvLSTM)* for the first time in the literature. Our doubly semi-implicit setting in ConvLSTM networks can be used for recurrent neural network models, so we are also proposing this novelty together for the first time in the literature.

- In Chapter 5, we show that our proposed algorithms can capture the stochasticity in the spatio-temporal data. Proposed algorithms can scale to real-world spatio-temporal crowd flow datasets. Finally, we test the performance of the proposed models on a Turkcell dataset, which is a simulation of 5G mobile traffic, and show that these methods can replace Turkcell's current standards for traffic forecasting.

1.3 Thesis Outline

This thesis is organized as follows. In Chapter 2, we provide the fundamentals of spatio-temporal forecasting over graphs having a regular grid structured data along with some required mathematical tools to understand our novel algorithms. We first explain LSTM & ConvLSTM neural networks that are widely used in forecasting problems. Finally, we introduce variational inference and variational autoencoder, the most vital background topics to follow this thesis' arguments. Besides, advanced methods in variational inference to model arbitrary probability distributions are also explained as a basis to understand our novelties. In Chapter 3, we extend the aforementioned variational autoencoder paradigm for the spatio-temporal mobile traffic forecasting problem, namely VarConvLSTM. In Chapter 4, to capture uncertainties in the environment and multimodalities of data distribution, we provide improvisations to VarConvLSTM via semi-implicit

variational inference methods, so that express our formulations both for semi-implicit posterior and semi-implicit priors. In Chapter 5, we demonstrate that our algorithms are successful in spatio-temporal traffic flow and mobile traffic prediction tasks. Finally, we present our conclusions and future working directions in Chapter 6.

Chapter 2

Problem Formulation & Preliminaries

This chapter aims to provide the necessary background for the subsequent chapters elaborating on the fundamental concepts in spatio-temporal traffic forecasting over networks while incorporating uncertainty. We first formulate the mobile traffic forecasting problem as a spatio-temporal time-series prediction problem. We then review the related deep learning algorithms and variational inference to understand and to build a better intuition for later chapters in this thesis.

The chapter is organized as follows: In Section 2.1, we cast the thesis's fundamental problem as a spatio-temporal forecasting problem. In Section 2.2, we describe LSTM and ConvLSTM models. Then, in Section 2.3 & 2.4, we elaborate on the variational inference and the variational autoencoders. Finally, Section 2.5 reviews the variational inference procedures with recent trends.

2.1 Formulation of Mobile Traffic Forecasting Problem

The goal of mobile traffic forecasting is to utilize the previously observed mobile traffic sequence collected from spatially located base stations to predict a fixed or dynamic length of mobile traffic consumption in a local region.

Suppose we observe a dynamical system over a spatial region represented by a rectangular grid graph represented by $M \times N$ grid cells. Thus, our problem reduces to grid forecasting problem. Inside each cell in the grid, we obtain P time varying measurements. Therefore, the measurements at any time is modeled by a tensor $\mathcal{X} \in \mathbb{R}^{P \times M \times N}$. The problem is to predict K-step sequence given previous J observations:

$$\tilde{\mathcal{X}}_{t+1}, \ldots, \tilde{\mathcal{X}}_{t+K} = \underset{\mathcal{X}_{t+1}, \ldots, \mathcal{X}_{t+K}}{\arg\max}\ p\left(\mathcal{X}_{t+1}, \ldots, \mathcal{X}_{t+K} \mid \hat{\mathcal{X}}_{t-J+1}, \hat{\mathcal{X}}_{t-J+2}, \ldots, \hat{\mathcal{X}}_{t}\right) \quad (2.1)$$

We measure a two-dimensional downlink/uplink traffic matrix snapshot for the traffic nowcasting over spatio-temporally located base stations at every timestamp. Each entry in the matrix is a traffic measurement. Thus, the traffic forecasting problem is an instance of a spatio-temporal sequence forecasting problem.

Note that the spatio-temporal sequence forecasting problem is much more complicated than the one-step time series forecasting problem since the former problem's prediction target is a multi-dimensional sequence exhibiting both spatial and temporal correlations. Even though the number of free variables in a length-K sequence can be up to $O(M^K N^K P^K)$, in reality, we utilize the underlying structure of the space of possible predictions to make the problem feasible.

2.2 LSTM & ConvLSTM Networks

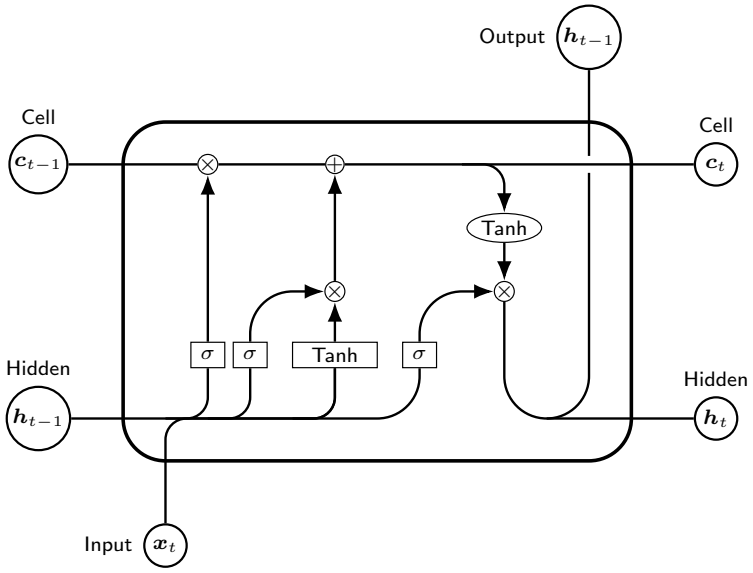


Figure 2.1: Memory cell for Long-Short term memory neural networks.

For general-purpose sequence modeling, LSTM, a special RNN structure, has proven stable and powerful for learnıng sequential representations over a very long-range in several previous studies [62, 14, 84, 85]. The striking novelty of LSTM is its memory cell c_t, which accumulates state information over time. The memory cell can be controlled via self-parameterized control gates. Whenever a new input comes, its information will be accumulated to the cell if the input gate i_t is activated. Also, the cell status from the previous time step c_{t-1} might be rewritten if the forget gate f_t is active. Whether the current cell output c_t will be propagated to the final state h_t or not is dependent to the output gate o_t . Embodying a memory cell and information control gates is that the gradient will be trapped in the cell, and the vanishing gradient problem, which is a critical problem for the vanilla RNN model [84, 62] can be averted in no time. In this thesis, we formulate LSTM, depicted in Figure 2.1, adopted as in [14], where $'\odot'$

denotes the Hadamard product:

$$i_t = \sigma\left(\boldsymbol{W}_{xi} x_t + \boldsymbol{W}_{hi} h_{t-1} + b_i\right) \tag{2.2}$$

$$f_t = \sigma\left(\boldsymbol{W}_{xf} x_t + \boldsymbol{W}_{hf} h_{t-1} + b_f\right) \tag{2.3}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh\left(\boldsymbol{W}_{xc} x_t + \boldsymbol{W}_{hc} h_{t-1} + b_c\right) \tag{2.4}$$

$$o_t = \sigma\left(\boldsymbol{W}_{xo} x_t + \boldsymbol{W}_{ho} h_{t-1} + b_o\right) \tag{2.5}$$

$$h_t = o_t \circ \tanh\left(c_t\right) \tag{2.6}$$

where $\boldsymbol{x}_t \in \mathbb{R}^m$ is the input vector, $\boldsymbol{c}_t \in \mathbb{R}^q$ is the state vector and $\boldsymbol{y}_t \in \mathbb{R}^q$ is the output vector at time t , $\boldsymbol{i}_t, \boldsymbol{f}_t$ and $\boldsymbol{o}_t$ are the input, forget and output gates, respectively. Nonlinear activation functions $g(\cdot)$, $h(\cdot)$ and $\sigma(\cdot)$ apply the point-wise operations. $\tanh(\cdot)$ is commonly used for $g(\cdot)$ and $h(\cdot)$ functions and $\sigma(\cdot)$ is the sigmoid function, i.e., $\sigma(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{x}}}$ $\boldsymbol{W}_{z\star}, \boldsymbol{W}_{i\star}, \boldsymbol{W}_{f\star}, \boldsymbol{W}_{o\star} \in \mathbb{R}^{q\times m}$ are the input weight matrices and $\boldsymbol{R}_{z\star}, \boldsymbol{R}_{i\star}, \boldsymbol{R}_{f\star}$ $\boldsymbol{R}_o \in \mathbb{R}^{q\times q}$ are the recurrent weight matrices. Multiple LSTMs can be stacked to form more complex structures[85, 86].

Despite the powerful modeling capabilities of temporal correlation, LSTM neural networks cannot handle the spatial data without further modifications as no spatial information is encoded with LSTM's input-to-state and state-to-state transitions. Unlike LSTM networks, Convolutional LSTM (ConvLSTM) neural networks employ convolutions in their input-to-state and state-to-state transitions [64].

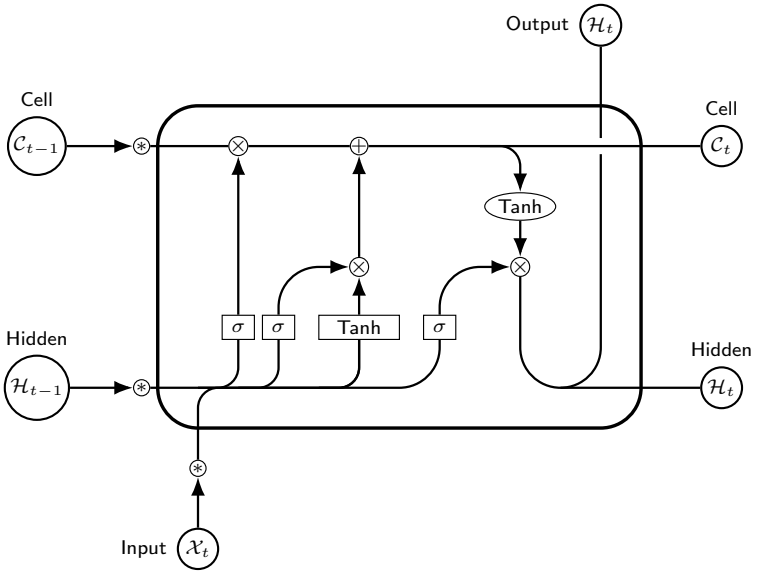


Figure 2.2: Memory cell for Convolutional long-short term memory neural network.

In ConvLSTM, inputs $\mathcal{X}_1, \ldots, \mathcal{X}_t$, cell outputs $\mathcal{C}_1, \ldots, \mathcal{C}_t$, hidden states $\mathcal{H}_1, \ldots, \mathcal{H}_t$, and gates i_t, f_t, o_t of are 3D tensors whose last two dimensions are spatial dimensions. In ConvLSTM, the future state of a certain cell in the grid is a function of the inputs and past states of its local neighbors using a convolution operation in the state-to-state and input-to-state transitions The ConvLSTM equations are shown below, where $*$ denotes the convolution operator and $'\odot'$, as before, denotes the Hadamard product:

$$i_t = \sigma\left(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \odot \mathcal{C}_{t-1} + b_i\right) \tag{2.7}$$

$$f_t = \sigma\left(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \odot \mathcal{C}_{t-1} + b_f\right) \tag{2.8}$$

$$\mathcal{C}_t = f_t \odot \mathcal{C}_{t-1} + i_t \odot \tanh\left(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c\right) \tag{2.9}$$

$$o_t = \sigma\left(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \odot \mathcal{C}_t + b_o\right) \tag{2.10}$$

$$\mathcal{H}_t = o_t \odot \tanh\left(\mathcal{C}_t\right) \tag{2.11}$$

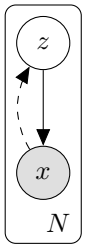


Figure 2.3: The graphical model representation of the vanilla variational inference procedure. Here x denotes the observed variable, z is the latent variable associated with x parameterized by the variational parameter ϕ, and N denotes the dataset, which means we imply the relationship inside of it for all data points[1]

.

2.3 Variational Inference and Learning

Consider a probabilistic model as in Figure 2.3:

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})} \tag{2.12}$$

where $\boldsymbol{x}$ and $\boldsymbol{z}$ denotes the observed and latent variables respectively. In Bayesian statistics, $p(\boldsymbol{z})$ is called the prior distribution of the latent variable and $p\left(\boldsymbol{x} \mid \boldsymbol{z}\right)$ is the likelihood of the observation X given the latent code Z.

The fundamental procedure in Bayesian modeling is to compute the posterior distribution, $p\left(\boldsymbol{z} \mid \boldsymbol{x}\right)$ defined in Eq. 2.12, where $p(\boldsymbol{x}) = \int_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}$ called evidence term. In practice, performing fully Bayesian Inference is notoriously hard due to the computation of the evidence term, which requires marginalization over latent variables. To be precise, for a given input, we may have no latent code for the input, or we may even have exponentially (maybe infinite) many latent codes for a given pair. Even if we have a finite number of latent vector for an input, evaluating the integral is cumbersome. In many cases, the evidence integral is intractable and cannot be computed in closed form.

[1]In Figure 2.3 and some of the figures later appearing in this thesis, we use graphical models to describe our ideas concisely. As a refresher, diamonds represent deterministic variables, while circles represent stochastic variables. Gray circles denote observed stochastic variables. Rectangle encapsulating the graphical model is called a plate, and it signifies that the procedure is repeated for all N data points. However, we sometimes neglect the place notation for brevity.

Variational inference transforms the inference problem into an optimization problem to seek the best distribution among a family of distributions parameterized by free "variational parameters" which approximates the posterior distribution $p(\boldsymbol{z} \mid \boldsymbol{x})$ [87]. It should be noted that there are other non-optimization based methods to make such approximate inference, such as MCMC providing unbiased distribution requiring too many samples to form it and is impractical [88].

Let L be the family of distributions over the latent random variables. Each $q(z) \in L$ is a candidate approximation to the true posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$. The aim is to find the best candidate who has the smallest Kullback-Leibler (KL) divergence [89] to the true posterior we want to compute. Mathematically, assuming that both approximate and true posterior distributions are continuous, our optimization problem is formulated as

$$q^*(\boldsymbol{z}) = \underset{q(\boldsymbol{z}) \in L}{\operatorname{argmin}} \operatorname{KL}(q(\boldsymbol{z}) \| p(\boldsymbol{z}|\boldsymbol{x})) \tag{2.13}$$

where $q^*(\boldsymbol{z})$ is the best approximation to the true posterior in distribution family L. Here, we one of the two problems of distribution optimization by considering a fixed distribution family L. However, we still cannot compute the divergence since it is impossible to evaluate the true posterior. If we open up KL divergence,

$$\begin{aligned}
\operatorname{KL}(q(\boldsymbol{z}) \| p(\boldsymbol{z}|\boldsymbol{x})) &= \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \left[\frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} \right] d\boldsymbol{z} \\
&= \int_{\boldsymbol{z}} [q(\boldsymbol{z}) \log q(\boldsymbol{z})] d\boldsymbol{z} - \int [q(\boldsymbol{z}) \log p(\boldsymbol{z}|\boldsymbol{x})] d\boldsymbol{z} \\
&= \mathbb{E}_q[\log q(\boldsymbol{z})] - \mathbb{E}_q[\log p(\boldsymbol{z}|\boldsymbol{x})] \\
&= \mathbb{E}_q[\log q(\boldsymbol{z})] - \mathbb{E}_q \left[\log \left[\frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{x})} \right] \right] \\
&= \mathbb{E}_q[\log q(\boldsymbol{z})] - \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z})] + \mathbb{E}_q[\log p(\boldsymbol{x})] \\
&= \mathbb{E}_q[\log q(\boldsymbol{z})] - \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z})] + \log p(\boldsymbol{x})
\end{aligned} \tag{2.14}$$

we see that we cannot optimize the KL divergence directly due to evidence term which is a constant. Since the evidence is constant, leave it alone:

$$\log p(\boldsymbol{x}) = \mathbb{E}_q[\log q(\boldsymbol{z})] - \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z})] + \operatorname{KL}(q(\boldsymbol{z}) || p(\boldsymbol{z}|\boldsymbol{x})) \tag{2.15}$$

Instead of minimizing the KL divergence, we can maximize the other terms. Since KL divergence is nonnegative,

$$\begin{aligned} \log p(\boldsymbol{x}) &= \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathbb{E}_q[\log q(\boldsymbol{z})] + \mathrm{KL}(q(\boldsymbol{z}) \mid\mid p(\boldsymbol{z} \mid \boldsymbol{x})) \\ &\geq \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathbb{E}_q[\log q(\boldsymbol{z})] = \mathrm{ELBO(q)} \end{aligned} \tag{2.16}$$

where ELBO(q) is the acronym for *evidence lower bound* which also known as *variational lower bound(VLB)*. Minimizing the KL divergence is equivalent to maximizing the ELBO:

$$\begin{aligned} q^*(\boldsymbol{z}) &= \arg\min_{q(\boldsymbol{z}) \in L} \mathrm{KL}\left(q(\boldsymbol{z}) || p(\boldsymbol{z}|\boldsymbol{x})\right) = \arg\max_{q(\boldsymbol{z}) \in L} \mathrm{ELBO(q)} \\ &= \arg\max_{q(\boldsymbol{z}) \in L} \Big\{ \mathbb{E}_q\big[\log p(\boldsymbol{x}, \boldsymbol{z})\big] - \mathbb{E}_q\big[\log q(\boldsymbol{z})\big] \Big\} \end{aligned} \tag{2.17}$$

Notice that the ELBO term still consists of a joint distribution. Using the factorization of the joint distribution, we rewrite the ELBO as

$$\begin{aligned} \mathrm{ELBO}(q) &= \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z})] - \mathbb{E}_q[\log q(\boldsymbol{z})] \\ &= \mathbb{E}_q[\log p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z})] - \mathbb{E}_q[\log q(\boldsymbol{z})] \\ &= \mathbb{E}_q[\log p(\boldsymbol{x} \mid \boldsymbol{z})] + \mathbb{E}_q[\log p(\boldsymbol{z})] - \mathbb{E}_q[\log q(\boldsymbol{z})] \\ &= \mathbb{E}_q[\log p(\boldsymbol{x} \mid \boldsymbol{z})] + \mathbb{E}_q\left[\log \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})}\right] \\ &= \mathbb{E}_q[\log p(\boldsymbol{x} \mid \boldsymbol{z})] - \mathrm{KL}(q(\boldsymbol{z}) \| p(\boldsymbol{z})) \end{aligned} \tag{2.18}$$

The first term in Equation 2.18 is the expected log-likelihood of the data and the second is the negative KL divergence between approximate posterior $q(\boldsymbol{z})$ and the prior $p(\boldsymbol{z})$. With the overall objective to maximize ELBO(q), we maximize the log-likelihood while maintaining the distance between approximate and prior distributions as much as possible.

There are two simple yet fundamental methods to compute the approximate posterior: mean-field VI and amortized parametric VI. In mean-field VI, the variational distribution is always result in the form, $q(\boldsymbol{z}) = \prod_{j=1}^{m} q_j(z_j)$, that each dimension of the latent code mutually independent and is modelled by its own density function [46]. The optimization is carried out with coordinate ascent mean-field variational inference (CAVI) algorithm. As opposed to mean-field VI,

which involves solving separate optimization problems scaling with latent space dimensionality, with amortized VI, we optimize the parameters of a parameterized function that maps from observation space to the parameters of the approximate posterior distribution, $q(\boldsymbol{z}) = p(z|\theta)$ where θ is the distributional parameter.

Variational learning generalizes variational inference in a way that prior and likelihood distributions can also be parameterizable. Consider the probabilistic model factorization in Eq. 2.12. Then, variational learning tries to find parameters maximizing the following VLB:

$$\log p\left(\mathrm{x} \mid \theta_{lh}, \theta_p\right) \geq \mathbb{E}_{p(x)} \mathbb{E}_{q_\phi(z|x)} \log \frac{p_{\theta_{lh}}(x \mid z) p_{\theta_p}(z)}{q_\phi(z \mid x)} \rightarrow \max_{\phi, \theta_{lh}, \theta_p} \tag{2.19}$$

where θ_{lh} and θ_p are the variational parameters for likelihood and prior distributions, respectively. This problem is used whenever we want to use complex priors and likelihoods, where it is going to be crucial in Chapter 4, under the variational inference framework.

2.4 Variational Autoencoders

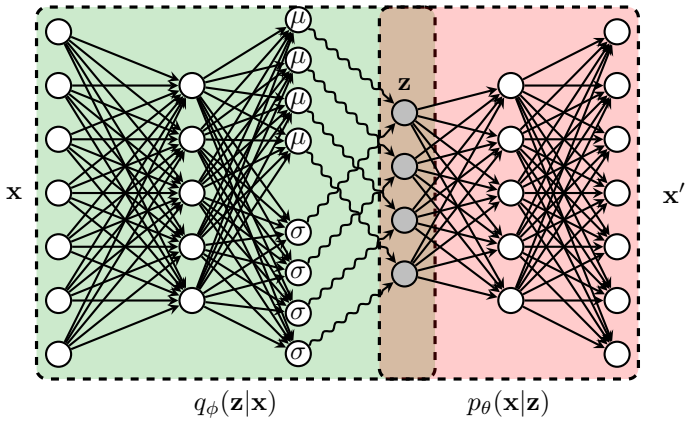


Figure 2.4: Variational Autoencoder (VAE) architecture

Introduced by Kingma and Welling [13, 80, 79], variational autoencoders use neural networks to parametrize the density distributions p and q described in the previous chapter. Here, as in standard autoencoders, we have two neural

networks: inference network (the encoder) and the generative network (the decoder). The inference network $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, parameterised via $\boldsymbol{\phi}$, models the approximate posterior q using an amortised VI scheme. Here, amortised inference is the quintiessential factor for fast inference as it saves us to store variational parameters for each data points during training and test time. After projecting the data into the latent space, we use the generative network $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ with parameters θ to reconstruct the data given the latent code $\boldsymbol{z}$. This is illustrated with the help of Figure 2.4. Replacing $q(\boldsymbol{x})$ with the approximate posterior, we write the ELBO(q) for the data $\mathcal{D} = \left\{\boldsymbol{x}^{(n)}\right\}$ in Eq. 2.18 as follows:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(n)}\right) &\geq \mathbb{E}_{\boldsymbol{z}\sim q_\phi\left(\boldsymbol{z}|\boldsymbol{x}^{(n)}\right)}\left[\log\left\{\frac{p_\theta\left(\boldsymbol{x}^{(n)},\boldsymbol{z}\right)}{q_\phi\left(\boldsymbol{z}\mid\boldsymbol{x}^{(n)}\right)}\right\}\right] \\ &= \mathbb{E}_{\boldsymbol{z}\sim q_\phi\left(\boldsymbol{z}|\boldsymbol{x}^{(n)}\right)}\left[\log p_\theta\left(\boldsymbol{x}^{(n)}\mid\boldsymbol{z}\right)\right] - \mathrm{KL}\left(q_\phi\left(\boldsymbol{z}\mid\boldsymbol{x}^{(n)}\right)\|p(\boldsymbol{z})\right)\end{aligned} \tag{2.20}$$

We are required to maximize ELBO(q), instead we minimize the −ELBO(q):

$$\begin{aligned}J^{(n)} &= -\mathbb{E}_{\boldsymbol{z}\sim q_\phi\left(\boldsymbol{z}|\boldsymbol{x}^{(n)}\right)}\left[\log p_\theta\left(\boldsymbol{x}^{(n)}\mid\boldsymbol{z}\right)\right] + \mathrm{KL}\left(q_\phi\left(\boldsymbol{z}\mid\boldsymbol{x}^{(n)}\right)\|p(\boldsymbol{z})\right) \\ &= J_{\mathrm{rec}}\left(\boldsymbol{\theta},\boldsymbol{\phi},x^{(n)}\right) + \mathrm{KL}\left(q_\phi\left(\boldsymbol{z}\mid\boldsymbol{x}^{(n)}\right)\|p(\boldsymbol{z})\right)\end{aligned} \tag{2.21}$$

The first term, the expected negative log-likelihood of data, is the reconstruction term similar to traditional deterministic autoencoders (DAE). If the likelihood is Gaussian, then we get the square loss between the input and the reconstructed input. For sequence-to-sequence models, reconstruction loss is the reconstruction errors summed across all timesteps. The second term, the KL divergence between the approximate posterior distribution $q_\phi\left(\boldsymbol{z}\mid\boldsymbol{x}^{(n)}\right)$ that the encoder network maps the original data space into, and the pre-specified prior $p(\boldsymbol{z})$. For continuous latent variables, the prior is typically assumed to be Gaussian $\mathcal{N}\left(\mathbf{0},\boldsymbol{I}\right)$ because KL divergence for the Gaussian distribution has analytic form and Gaussian is a reparameterizable. KL term is analogous to a regularizer, forcing the latent space into a pre-scribed distribution to sample and synthesize new data, unlike in DAEs, which just projects data and is prone to memorizing the mappings.

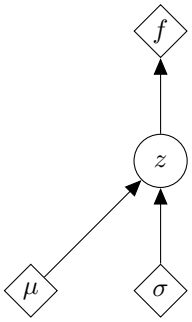
A recent work on VAEs show that the if KL term of the VAE loss becomes zero, then approximate posterior becomes exactly the same as the prior and

latent space becomes uninformative [90, 91]. To maintain a balance between the reconstruction and KL terms, they propose KL annealing shown below:

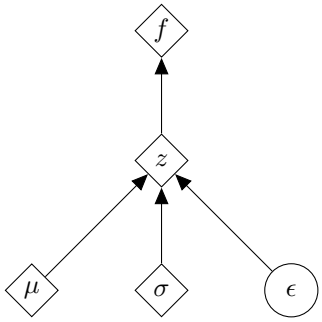
$$J^{(n)} = J_{\text{rec}}\left(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{x}^{(n)}\right) + \lambda \cdot \text{KL}\left(q_\phi\left(\boldsymbol{z} \mid \boldsymbol{x}^{(n)}\right) \| p(\boldsymbol{z})\right) \tag{2.22}$$

where λ refers to the KL weight, whose value is a function of epochs for the training only. The key idea behind this technique is that we first start with a completely deterministic autoencoder then gradually turning our autoencoder into a variational one . in this thesis, we adopt this loss function for all chapters.

2.4.1 Reparameterization Trick



(a) Graphical model for Gaussian Random Variable Generation



(b) Graphical model for Gaussian Random Variable Generation with reparameterization trick.

Figure 2.5: Reparameterization Trick for Deep Latent Variable Models.

We train VAE networks using stochastic gradient descent algorithm, but a problem arises in backpropagation. That is, as in Figure 2.5a, VAE introduces a stochastic variable in the computation graph of the network due to sampling the latent code. It is known that derivative through sampling is not defined. To circumvent this issue, Kingma and Welling proposed the reparameterization trick in which we sample from a prescribed distribution and then transform this sample in the latent space [13]. For the Gaussian distribution, reparameterization trick is to sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathcal{I})$ first then transform it to $\boldsymbol{z}$ using the learned parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ as shown below

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \otimes \boldsymbol{\epsilon} \tag{2.23}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ have already been obtained by transforming the encoder output as in Figure 2.5b. With this trick, we transfer the source of stochasticity from latent variable $\boldsymbol{z}$ to the independent random sample $\boldsymbol{\epsilon}$ so that we can backpropagate through the latent variable and the gradient passes back to the encoder network (through $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$) and the model is trained end to end. In other words, we represent the latent code with encoder outputs and a predefined simple distribution that is parameterizable like Gaussian. Thanks to sampling $\boldsymbol{\epsilon}$ first, we separate the sampling procedure from the network's backpropagation graph, which is the desired property. However, this restricts the various distributions we can use to approximate the posterior. The next section explains how to model variational approximations, which can model arbitrary distribution families.

2.5 Advanced Variational Inference Methods

In the final section of Chapter 2, we elaborate on current methods from variational inference literature, which allows us to model implicit distributions. Unlike explicit probability distributions, implicit distributions do not have a parametric form(even may be inaccessible to us), but we still can perform sampling and backpropagation over implicit distributions. The concept of implicit distributions is closely related to the generator networks in generative adversarial networks(GAN) [92, 52, 53]. The raison d'etre for implicit distributions is that explicit parametric approximations to the true posterior are too simple [53]. For the sake of efficiency, we restrict ourselves to explicit distributions for q_ϕ is chosen so that $\mathrm{L}(\phi)$ and its gradients w.r.t. ϕ are easy to compute, resulting in underestimating the variance of the posterior because we are restricted to have an analytic PDF in the variational family [46]. However, optimal distributions for the variational inference and learning problems may be implicit distributions. With implicit distributions, deep variational models are not bound to parametric posteriors and become competitive with GANs. Unlike GANs, we can perform Bayesian inference and employ easier optimization problems.

There exist mainly two ways to extend the variational family to mitigate explicit distributions' deficiencies: those that require tractable approximate posteriors, the normalizing flows [81, 93, 94, 95, 96, 97], and those that do not (implicit models) [52, 53].

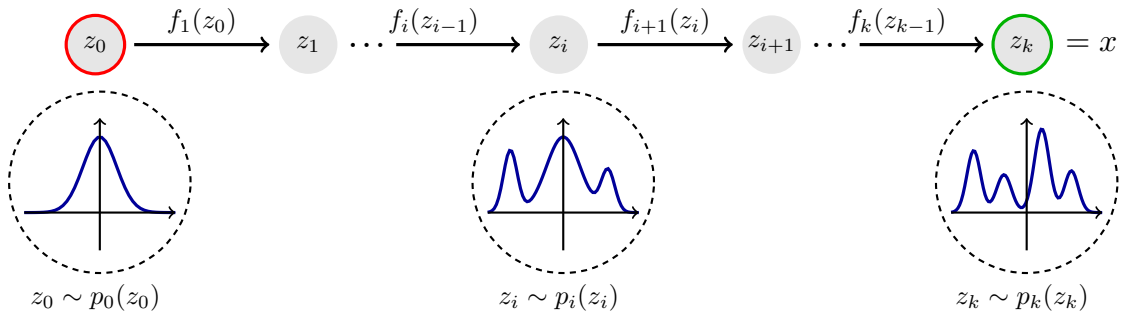


Figure 2.6: Normalizing flows transforms a simple density such as Gaussian, to a complicated mutli-modal density via a series of invertible transformations.

Normalizing flows are mappings from R^D to R^D such that densities p_X on the input space $X = R^D$ are transformed into some simple distribution p_Z (e.g. an isotropic Gaussian) on the space $Z = R^D$ [98]. This mapping $f : X \to Z$, is composed of a sequence of bijections.Using the change of variables formula, we express

$$p_{\mathcal{X}}(\mathbf{x}) = p_{\mathcal{Z}}(\mathbf{z}) \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right|, \tag{2.24}$$

where $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian of f at $\boldsymbol{x}$. Normalizing flows have the property that the $\mathbf{x} = f^{-1}(\mathbf{z})$ which is easy to evaluate since Jacobian determinant takes $O(D)$ time to compute. To model a nonlinear density map $f(\boldsymbol{x})$, a series of bijections $\mathbf{x} \to \mathbf{z}_{k-1} \to \cdots \to \mathbf{z}_1 \to \mathbf{z}_0$ are composed together while alternating the dimensions which are unchanged and transformed. Via the change of variables formula, the probability density function of the flow given a data point $\mathbf{x} = \mathbf{z}_K =$

$f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0)$ can be written as

$$\begin{aligned}
\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\
&= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log \left| \det \frac{df_{K-1}}{d\mathbf{z}_{K-2}} \right| - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\
&= \dots \\
&= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|
\end{aligned} \tag{2.25}$$

As the equation 2.25 implies that the transformation functions are bijections whose Jacobian determinants easy to compute. Normalizing flows can represent any data distribution under some reasonable conditions on $p_{\mathcal{X}}(\mathbf{x})$. The argument is based on the proof, which is similar to the proof of the existence of non-linear ICA, so a more formal treatment is provided in [99, 100]. Normalizing flows can be used in tandem with VAE to provide more expressive posteriors [81, 93, 94, 95]. In our problem, however, invertibility is a restricting condition in the high-dimensional spatiotemporal data. The prominent methods for variational inference with implicit distributions employ adversarial training but compared to other approaches, it tends to overfit in higher dimensions [53, 52, 101].

2.5.1 Semi-Implicit Variational Inference (SIVI)

A distribution $q_\phi(z)$ is semi-implicit if it has the following representation:

$$q_\phi(z) = \int q_\phi(z \mid \psi) q_\phi(\psi) \mathrm{d}\psi, \tag{2.26}$$

where $q_\phi(z|\psi)$ has analytically tractable density and both $q_\phi(z|\psi)$ and $q_\phi(\psi)$ are reparametrizable[55, 54]. In SIVI, we treat the variational parameters in Eq. 2.18 as a random variable. Thanks to this treatment, SIVI distributions form much more broader variational family while maintaining simple optimziation problem compared to the vanilla VI. If $\psi \sim q(\psi)$ degenerates to a point mass density (a delta function), then SIVI is equivalent to vanilla VI. Let $\phi \sim q_\phi(z|\psi)$, where ϕ

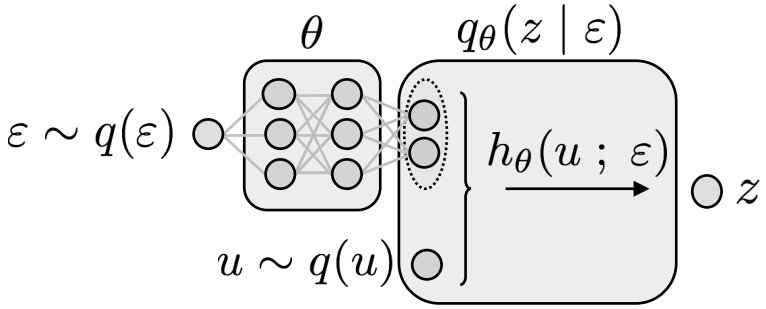


Figure 2.7: lllustration of the sampling procedure for the semi-implicit variational distribution $q_\theta(z)$. First, a sample $\epsilon \sim q(\epsilon)$ is pushed through a neural network parameterized by θ (left block). This network outputs the parameters of the reparameterizable conditional distribution $q_\theta(z|\epsilon)$. To draw a sample z, we first sample $u \sim q(u)$ and then set $z = h_\theta(u;\epsilon)$,where $h_\theta(\dot{)}$is an appropriate transformation. The transformation $h_\theta(\dot{)}$depends on ϵ and θ through the parameters of the conditional. The output $z = h_\theta(u;\epsilon)$ is a sample from the variational distribution $q_\theta(z)$ [1]

is the distribution parameters to be inferred. Then the semi-implicit variational distribution for z is defined in a hierarchical manner[102, 103, 104]

$$z \sim q_\phi(z \mid \psi), \quad \psi \sim q_\phi(\psi) \tag{2.27}$$

Marginalizing the variable ψ, we construct the random variable z as a random variable drawn from distribution $h_\phi(z)$

$$\mathcal{H} = \left\{ h_\phi(z) : h_\phi(z) = \int_\psi q(z \mid \psi) q_\phi(\psi) d\psi \right\}. \tag{2.28}$$

Since we seek to estimate the distributional parameters, $q(z \mid \psi)$ is required to be explicit, but there is no restriction over $q_\phi(\psi)$ unless mixing distribution is conjugate to $q(z \mid \psi)$. The reason behind $q(z \mid \psi)$ being reparameterizable is to maintain easy sampling from semi-implicit distribution. We transform random noise ϵ through a function $f(\epsilon, \psi)$ to generate $z \sim q(z \mid \psi)$. As $q(\phi)$ can be implicit, we can think of it as a GAN generator that transforms a random noise with neural network results in a sample from an implicit distribution due to non-invertibility.

SIVI lower bound can be derived using Jensen's inequality and semi-implicit

hierarchy:

$$\begin{aligned}\underline{\mathcal{L}}\left(q(\boldsymbol{z} \mid \boldsymbol{\psi}), q_{\boldsymbol{\phi}}(\boldsymbol{\psi})\right) =& \mathbb{E}_{\boldsymbol{\psi} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi})} \mathbb{E}_{\boldsymbol{z} \sim q(\boldsymbol{z} \mid \boldsymbol{\psi})} \log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z} \mid \boldsymbol{\psi})} \\ =& -\mathbb{E}_{\boldsymbol{\psi} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi})} \mathrm{KL}(q(\boldsymbol{z} \mid \boldsymbol{\psi}) \| p(\boldsymbol{z} \mid \boldsymbol{x})) + \log p(\boldsymbol{x}) \\ \leq& -\mathrm{KL}\left(\mathbb{E}_{\boldsymbol{\psi} \sim q_{\boldsymbol{\phi}}}(\boldsymbol{\psi}) q(\boldsymbol{z} \mid \boldsymbol{\psi}) \| p(\boldsymbol{z} \mid \boldsymbol{x})\right) + \log p(\boldsymbol{x}) \\ =& \mathcal{L} = \mathbb{E}_{\boldsymbol{z} \sim h_{\boldsymbol{\phi}}(\boldsymbol{z})} \log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{h_{\boldsymbol{\phi}}(\boldsymbol{z})}\end{aligned} \tag{2.29}$$

where we have used the fact that $\mathrm{KL}\left(\mathbb{E}_{\psi} q(z \mid \psi) \| p(z)\right) \leq \mathbb{E}_{\psi} \mathrm{KL}(q(z \mid \psi) \| p(z))$ [105]. Optimizing $\underline{\mathcal{L}}$ directly can suffer from degeneracy of SIVI to vanilla VI as $\delta(\psi) \in q_{\phi}(\psi)$. To prevent it, we first approximate the semi-implicit approximate posterior with a finite mixture:

$$q_{\phi}(z) = \int q_{\phi}(z \mid \psi) q_{\phi}(\psi)\, d\psi \approx \frac{1}{K} \sum_{k=1}^{K} q_{\phi}(z \mid \psi^{k}), \quad \psi^{k} \sim q_{\phi}(\psi). \tag{2.30}$$

and re-express the previous lower bound with the finite mixture (and upper bound, derived in the original paper via the concavity of the logarithm function, is used to check how much we have sandwiched the ELBO term):

$$\overline{\mathcal{L}}_{K}^{q} = \mathbb{E}_{q_{\phi}(z)} \log p(x \mid z) p(z) - \mathbb{E}_{\psi^{0..K} \sim q_{\phi}(\psi)} \mathbb{E}_{z \sim q_{\phi}(z \mid \psi^{0})} \log \frac{1}{K} \sum_{k=1}^{K} q_{\phi}(z \mid \psi^{k}) \tag{2.31}$$

$$\underline{\mathcal{L}}_{K}^{q} = \mathbb{E}_{q_{\phi}(z)} \log p(x \mid z) p(z) - \mathbb{E}_{\psi^{0..K} \sim q_{\phi}(\psi)} \mathbb{E}_{z \sim q_{\phi}(z \mid \psi^{0})} \log \frac{1}{K+1} \sum_{k=0}^{K} q_{\phi}(z \mid \psi^{k}). \tag{2.32}$$

To prevent degeneracy, we add the following term to the SIVI lower bound [2]:

$$B_{K} = \mathbb{E}_{\psi, \psi^{(1)}, \ldots, \psi^{(K)} \sim q_{\phi}(\psi)} \mathrm{KL}\left(q(z \mid \psi) || \tilde{h}_{K}(z)\right) \tag{2.33}$$

where $\tilde{h}_{K}(z) = \frac{q(z|\psi) + \sum_{k=1}^{K} q\left(z|\psi^{(k)}\right)}{K+1}$. Then, optimize $\underline{\mathcal{L}}_{K}^{q} + B_{K}$. Notice, $B_{K} \geq 0$. Both SIVI upper and lower bound converge to ELBO as $K \to \infty$.

[2] For the upper bound, we subtract a term. However, we consider not to discuss its derivation as the upper bound is not useful for the optimization.

In the next chapter, we discuss one of the fundamental stepstones in this thesis upon the fundamentals introduced in this chapter, which we call *Variational ConvLSTM*. This algortihm allows performing Variational Bayesian inference with deep learning over spatio-temporally structured data. The aim is to learn a latent space for space-time series. Since we model the data distribution via latent variables, it allows us to model scenarios not available in the training data. Thanks to the LSTM structure, our algorithm is analogous to a space-time Kalman Filter except we learn parameters via neural networks.

Chapter 3

Variational ConvLSTM

In this chapter, we introduce the spatio-temporal extension to variational autoencoders. We present its posterior inference, prior construction, and data generation mechanisms. Then, we present the learning algorithm for our *VarConvLSTM* architecture.

The chapter is organized as follows. In Section 3.1, we describe the Variational ConvLSTM model. In subsections 3.1.1 and 3.1.2, we describe the generative and inference models of our architecture. Then, in Section 3.2, we describe the learning procedure of VarConvLSTM and finally deriving the variational lower bound of VarConvLSTM to train the model.

3.1 Variational ConvLSTM Model

This section introduces a VAE to model spatio-temporal sequences, which we call *Variational Convolutional Long Short Term Memory*(VarConvLSTM). Variational RNN inspires variational ConvLSTM (VRNN) [106] architecture except for all our inputs and the hidden states are tensors. The description of VarConvLSTM lies in its generation, inference, and learning schemes, which are explained in the following sections in detail.

3.1.1 Generative Model

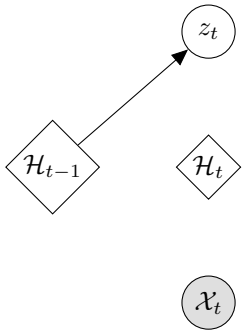


Figure 3.1: Prior model in VarConvLSTM

In classical VAE models as in Figure 2.4, prior over the latent variable follows a Gaussian distribution. However, applying this practice to sequential data does not encode the sequential nature of the data into our prior. To mitigate this issue, we define a Gaussian prior whose statistics are a function of the hidden state tensor $\mathcal{H}_{t-1}$, described in Figure 3.1, can be written mathematically as follows:

$$\mathbf{z}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_{\text{prior}}^{(t)}, \text{diag}\left((\boldsymbol{\sigma}_{prior}^{(t)})^2\right)\right), \tag{3.1}$$

where $\left[\boldsymbol{\mu}_{\text{prior}}^{(t)}, \boldsymbol{\sigma}_{prior}^{(t)}\right] = \varphi_\tau^{\text{prior}}\ (\mathcal{H}_{t-1})$, denote the prior parameters of our model. Here, $\varphi_\tau^{\text{prior}}$ is the feature extraction function for prior which can be any flexible and differentiable function. Since the hidden states are tensors, $\varphi_\tau^{\text{prior}}$ is a composition of fully convolutional layers with a flattening transformation to store latent

representations as vectors for efficiency and regularization. The use of $\varphi_\tau^{\text{prior}}$ also provides a better representation specific for learning prior parameters.

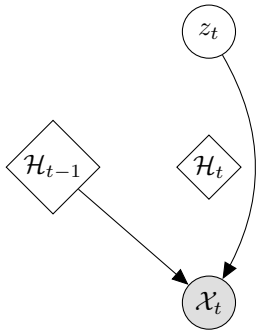


Figure 3.2: Likelihood Model in VarConvLSTM

Then, following the VAE formalism, amortised likelihood scheme has to be defined. In VAE, given a latent vector $\mathbf{z}$, we wish to generate $\mathbf{x}$. In our architecture however, at each time, we condition on the latent vector $\mathbf{z}_t$ and the hidden state tensor $\mathcal{H}_{t-1}$ depicted in Figure 3.2. This is because of the Markovian structure imposed with recurrent models. Therefore, at each time step, we generate $\mathcal{X}_t$ as follows:

$$\mathcal{X}_t \mid \mathbf{z}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_{\text{decoder}}^{(t)}, \text{diag}\left((\boldsymbol{\sigma}_{\text{decoder}}^{(t)})^2\right)\right), \tag{3.2}$$

where $\left[\boldsymbol{\mu}_{\text{decoder}}^{(t)}, \boldsymbol{\sigma}_{\text{decoder}}^{(t)}\right] = \varphi_\tau^{\text{dec}}\left(\varphi_\tau^{\mathbf{z}}\left(\mathbf{z}_t\right), \mathcal{H}_{t-1}\right)$,denote the parameters of generating distribution, and $\varphi_\tau^{\text{dec}}$ has exactly the same functionalities with $\varphi_\tau^{\text{prior}}$.

Our model can learn the sequential nature of data, but we have to carry out the model's decisions on the latent variables over time. The hidden state of a recurrent neural network is responsible for carrying out the sequential nature of the data across time conditioned given $\mathcal{X}_t$ and $\mathcal{H}_{t-1}$. Thus, recurrence, depicted in Figure 3.3, must include the latent vector using the following recurrence equation:

$$\mathcal{H}_t = f_\theta\left(\varphi_\tau^{\mathcal{X}}\left(\mathcal{X}_t\right), \varphi_\tau^{\mathbf{z}}\left(\mathbf{z}_t\right), \mathcal{H}_{t-1}\right), \tag{3.3}$$

where f_θ is the deterministic state transition function parameterized by θ while $\varphi_\tau^{\mathbf{x}}$ and $\varphi_\tau^{\mathbf{z}}$ again have the same purpose with as in prior and decoder which extract features from $\mathcal{X}_t$ and $\mathbf{z}_t$, respectively.

Finally, we model the generative model as a joint distribution of observation

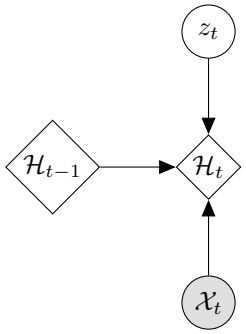


Figure 3.3: Recurrence in VarConvLSTM

tensors and latent vectors from all times. To do so, we need to harness the VarConvLSTM structure to factorize the joint distribution efficiently. Notice, from Eq. 3.3, $\mathcal{H}_t$ is a function of $\mathcal{X}_{\leq t}$ and $\mathbf{z}_{\leq t}$. Thus, likelihood and prior of VarConvLSTM described in Eq. 3.1 and Eq. 3.2 define the distributions conditioned on previous time stamps $p(\mathbf{z}_t|\mathcal{X}_{<t}, \mathbf{z}_{<t})$ and $p(\mathcal{X}_{<t}|\mathcal{X}_{<t}, \mathbf{z}_{\leq t})$, respectively. The parameterization of the generative model results in the following the factorization:

$$p\left(\mathcal{X}_{\leq T}, \mathbf{z}_{\leq T}\right) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right) p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right), \tag{3.4}$$

where $\boldsymbol{\theta}$ is the neural network parameters modeling the parameters of the decoder neural network.

3.1.2 Inference Model

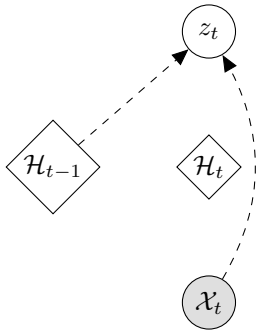


Figure 3.4: Inference mechanism in VarConvLSTM

Now, given an observation tensor at time t, $\mathcal{X}_t$, we want to infer its latent

vector, $\mathbf{z}_t$. Since prior is a function of hidden state tensor $\mathcal{H}_t$,

$$\mathbf{z}_t \mid \mathcal{X}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_{decoder}^{(t)}, (\operatorname{diag}\left((\boldsymbol{\sigma}_{decoder}^{(t)})^2\right)\right), \tag{3.5}$$

where $\left[\boldsymbol{\mu}_{decoder}^{(t)}, \boldsymbol{\sigma}_{decoder}^{(t)}\right] = \varphi_\tau^{\text{enc}}\left(\varphi_\tau^{\mathcal{X}}\left(\mathcal{X}_t\right), \mathcal{H}_{t-1}\right)$, denote the parameters of the approximate posterior. The reason why hidden state is also a input is that the encoding of the approximate posterior and the decoding for the generation are tied through the ConvLSTM hidden state. Thanks to this fact, we factorize the variational posterior as

$$q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right) = \prod_{t=1}^{T} q_{\boldsymbol{\phi}}\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right), \tag{3.6}$$

where $\boldsymbol{\phi}$ is encoder neural network parameters.

3.2 Learning in VarConvLSTM

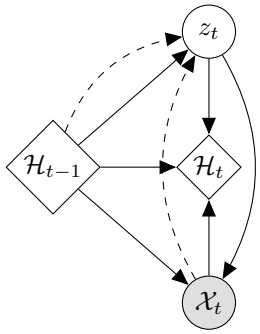


Figure 3.5: Overall Graphical Model for VarConvLSTM

As in the standard VAE, we learn the generative and inference models jointly by maximizing the variational lower bound(or minimizing the negative of variational lower bound) with respect to their parameters. The schematic for VarConvLSTM is shown in Fig. 3.5. Using Eq. 3.4 and 3.6, we found the variational lower bound as the sum of all lower bound over all time steps given as:

$$\log p\left(\mathcal{X}_{\leq T}\right) \geq \mathbb{E}_{q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right)}\left[\sum_{t=1}^{T}\left(-\operatorname{KL}\left(q\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right) \| p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right)\right) + \log p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right)\right)\right] \tag{3.7}$$

Now, We derive the variational lower bound of VarConvLSTM by expanding the KL divergence between the variational posterior and the joint likelihood:

$$
\begin{aligned}
&\mathrm{KL}(q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right) \| \, p\left(\mathcal{X}_{\leq T}, \mathbf{z}_{\leq T}\right)) = \int q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right) \log \left(\frac{p\left(\mathcal{X}_{\leq T}, \mathbf{z}_{\leq T}\right)}{q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right)}\right) d\mathbf{z}_{\leq T} \\
&= \int \sum_{t=1}^{T} \left(q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right) \log \left(\frac{p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right) p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right)}{q\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right)}\right)\right) d\mathbf{z}_{\leq T} \\
&= \sum_{t=1}^{T} \left(\int q\left(\mathbf{z}_{\leq t} \mid \mathcal{X}_{\leq t}\right) \log \left(\frac{p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right) p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right)}{q\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right)}\right) d\mathbf{z}_{\leq t}\right)
\end{aligned}
\tag{3.8}
$$

Next, we decompose the logarithm:

$$
\begin{aligned}
&= \sum_{t=1}^{T} \left(\int q\left(\mathbf{z}_{\leq t} \mid \mathcal{X}_{\leq t}\right) \log p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right) d\mathbf{z}_{\leq t}\right) + \\
&\qquad \sum_{t=1}^{T} \left(\int q\left(\mathbf{z}_{\leq t} \mid \mathcal{X}_{\leq t}\right) \log \left(\frac{p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right)}{q\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right)}\right) d\mathbf{z}_{\leq t}\right) \\
&= \sum_{t=1}^{T} \left(\int q\left(\mathbf{z}_{\leq t} \mid \mathcal{X}_{\leq t}\right) \log p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right) d\mathbf{z}_{\leq t}\right) - \\
&\qquad \sum_{t=1}^{T} \left(\int q\left(\mathbf{z}_{<t} \mid \mathcal{X}_{<t}\right) \mathrm{KL}\left(q\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right) \| p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right)\right) d\mathbf{z}_{<t}\right) \\
&= \mathbb{E}_{q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right)} \left[\sum_{t=1}^{T} \left(-\mathrm{KL}\left(q\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right) \| p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right)\right) + \log p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right)\right)\right] \\
&\simeq \sum_{t=1}^{T} \left(\log p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right) - \mathrm{KL}\left(q\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right) \| p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right)\right)\right)
\end{aligned}
\tag{3.9}
$$

where $\mathbf{z}_{\leq T} \sim q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right)$

In this chapter, we introduced the aforementioned VAE architecture for the spatio-temporal data, called *VarConvLSTM*. One caveat with algorithm and the VAE is that despite how deep are the encoder and decoder networks, distributions are modeled as Gaussian random variables. This choice of distribution may not be optimal for many Bayesian inference procedures. In the following chapter, we propose to blend newly introduced semi-implicit & doubly semi-implicit variational inference schemes into our *VarConvLSTM* architecture to model arbitrarily complex prior & posterior distributions.

Chapter 4

Semi-Implicit Extensions to Variational ConvLSTM

This chapter considers the use of non-parametric variational posterior and priors in VarConvLSTM neural networks via semi-implicit variational inference [55]. We first describe the necessity of the semi-implicit setting and present *SI-VarConvLSTM*, which models the posterior parameters as a random variable via composition of multiple stochastic layers. To remove the bias in the prior choice, we also consider a semi-implicit setting for priors by incorporating doubly semi-implicit variational inference [54]. With *DSI-VarConvLSTM*, it is possible to use non-parametric posterior and prior distributions simultaneously. We present variational lower bounds to train *SI-VarConvLSTM* & *DSI-VarConvLSTM*.

The chapter is organized as follows. In Section 4.1, we describe the first innovation to the VarConvLSTM model utilizing an amortized semi-implicit posterior scheme. In subsections 4.1.1, we describe our ulterior motive to enhance our model with semi-implicit posterior distributions. Then, in subsections 4.1.2, 4.1.3, and 4.1.4, we describe generation, inference, and learning procedures of SI-VarConvLSTM. in Section 4.2, we consider semi-implicit prior and posteriors simultaneously with DSI-VarConvLSTM and describe its details in the following subsections 4.2.1,4.2.2, 4.2.3, and 4.2.4.

4.1 Semi-Implicit VarConvLSTM

4.1.1 Generation in the Semi-Implicit Setting

The generation and prior (Eq. 3.1) structures in VarConvLSTM are carried to the SI-VarConvLSTM with no change. We model the prior distribution in our SI-VarConvLSTM as a learned prior distribution conditioned on the hidden states in previous time steps. In particular, we construct the prior distribution as

$$\mathbf{z}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_{\text{prior}}^{(t)}, \text{diag}\left(\left(\boldsymbol{\sigma}_{\text{prior}}^{(t)}\right)^2\right)\right), \tag{4.1}$$

where $\left\{\boldsymbol{\mu}_{\text{prior}}^{(t)}, \boldsymbol{\sigma}_{\text{prior}}^{(t)}\right\} = \varphi^{\text{prior}}\left(\mathcal{H}_{t-1}\right)$ denote the parameters of the prior distributution conditioned on the previous hidden state tensor. The joint distribution factorization in SI-VarConvLSTM is formalised as

$$\begin{aligned} p\left(\mathcal{X}_{\leq T}, \mathbf{z}_{\leq T}\right) &= \prod_{t=1}^{T} p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right) p\left(\mathbf{z}_t \mid \mathbf{z}_{<t}, \mathcal{X}_{<t}\right) \\ &= \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right) p\left(\mathbf{z}_t \mid \mathcal{H}_{t-1}\right) \end{aligned} . \tag{4.2}$$

4.1.2 Inference in the Semi-Implicit Setting

The prominent part is the variational posterior formulation since we propose to model the variational posterior via semi-implicit distributions [55]. In the semi-implicit setting, we first model a variational distribution that models the distribution over the posterior parameters. In a VAE model trained with vanilla variational inference, posterior parameters are deterministic [13, 80]. Compared to the traditional scheme, posterior parameters are stochastic, which are modeled via neural networks with stochastic input [107]. We may model the distribution generator function with a very deep neural network whose layers are injected with stochastic reparameterizable noise. Let $\boldsymbol{\psi}_t$ be the parameters of the distribution we want to infer, conditioned over past inputs and latent states. Then the semi-implicit hierarchy in SI-VarConvLSTM is defined as

$$\mathbf{z}_t \sim q\left(\mathbf{z}_t \mid \boldsymbol{\psi}_t\right), \quad \boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}\left(\boldsymbol{\psi}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right) \tag{4.3}$$

where $q_{\boldsymbol{\phi}}\left(\boldsymbol{\psi}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right)$ is the mixing distribution, parameterized by $\boldsymbol{\phi}$, distilling the parameters of the explicit variational posterior $q\left(\mathbf{z}_t \mid \boldsymbol{\psi}_t\right)$. Then, we sample $\mathbf{z}_t$, the latent representation of $\mathcal{X}_t$ from $q\left(\mathbf{z}_t \mid \boldsymbol{\psi}_t\right)$. Using the recurrence of VarConvLSTM model as in Eq. 3.3, the hierarchy of SI-VarConvLSTM in 4.3 simplifies to:

$$\mathbf{z}_t \sim q\left(\mathbf{z}_t \mid \boldsymbol{\psi}_t\right), \quad \boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}\left(\boldsymbol{\psi}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right) \tag{4.4}$$

Marginalizing over the distribution $\boldsymbol{\psi}_t$, we construct our variational distribution family for VarConvLSTM as in Eq. 4.25 parameterized via $\boldsymbol{\phi}$ as follows:

$$\mathcal{G} = \left\{ g_\phi\left(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right) = \int_{\boldsymbol{\psi}_t} q\left(\mathbf{z}_t \mid \boldsymbol{\psi}_t\right) q_{\boldsymbol{\phi}}\left(\boldsymbol{\psi}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right) d\boldsymbol{\psi}_t \right\} \tag{4.5}$$

In SI-VarConvLSTM, the variational posterior is constructed by modeling the paramters of the distribution as a distribution characterised by an encoder neural network φ^{encoder} as follows:

$$q\left(\mathbf{z}_t \mid \boldsymbol{\psi}_t\right) \sim \mathcal{N}\left(\boldsymbol{\mu}^{(t)}_{\text{encoder}}, \operatorname{diag}\left(\left(\boldsymbol{\sigma}^{(t)}_{\text{encoder}}\right)^2\right)\right), \tag{4.6}$$

where $\left\{\mu^{(t)}_{\text{encoder}}, \sigma^{(t)}_{\text{encoder}}\right\} = \varphi^{\text{encoder}}\left(\mathbf{z}_t, \mathcal{H}_{t-1}, \epsilon_t\right)$. Then, factorization of the variational distribution $q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right)$ can be expressed as

$$q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right) = \prod_{t=1}^{T} g_\phi\left(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right) \tag{4.7}$$

4.1.3 Learning in the Semi-Implicit Setting

The loss function for SI-VarConvLSTM, the sum of variational lower bounds from all-time steps, which is given below:

$$\begin{aligned} \underline{\mathcal{L}}_{\text{SI-VarConvLSTM}} = \sum_{t=1}^{T} \Big\{ & \mathbb{E}_{\boldsymbol{\psi}_t \sim q_\phi(\boldsymbol{\psi}_t \mid \mathcal{X}_t, \mathcal{X}_{t-1})} \mathbb{E}_{\mathbf{z}_t \sim q(\mathbf{z}_t \mid \boldsymbol{\psi}_t)} \log p\left(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{X}_{t-1}\right) \\ & - \mathbf{KL}\left(\mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \mid \mathcal{X}_t, \mathcal{X}_{t-1})} q\left(\mathbf{z}_t \mid \boldsymbol{\psi}_t\right) \| p\left(\mathbf{z}_t \mid \mathcal{H}_{t-1}\right)\right) \Big\} \end{aligned} \tag{4.8}$$

Derivation of VLB for SI-VarConvLSTM is also derived with the same steps, except our posterior is implicit. To derive the VLB, consider the KL divergence between the variational posterior and the likelihood of SI-VarConvLSTM:

$$\mathrm{KL}(q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right) \,||\, p\left(\mathcal{X}_{\leq T}, \mathbf{z}_{\leq T}\right)) = \int q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right) \log\left(\frac{p\left(\mathcal{X}_{\leq T}, \mathbf{z}_{\leq T}\right)}{q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right)}\right) d\mathbf{z}_{\leq T} \tag{4.9}$$

$$= \int \sum_{t=1}^{T} \left(q\left(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}\right) \log\left(\frac{p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right) p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right)}{q\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right)}\right)\right) d\mathbf{z}_{\leq T} \tag{4.10}$$

$$= \sum_{t=1}^{T} \left(\int q\left(\mathbf{z}_{\leq t} \mid \mathcal{X}_{\leq t}\right) \log\left(\frac{p\left(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t}\right) p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right)}{q\left(\mathbf{z}_t \mid \mathcal{X}_{\leq t}, \mathbf{z}_{<t}\right)}\right) d\mathbf{z}_{\leq t}\right) \tag{4.11}$$

After interchanging integration and summation, we write (4.11) as the sum of expectations. Then, with the help of distribution factorizations and seperation of the log-likelihood the divergence, we get:

$$= \sum_{t=1}^{T} \left(\mathbb{E}_{\mathbf{z}_t \sim g_\phi(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})} \log p\left(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right)\right) + \sum_{t=1}^{T} \left(\mathbb{E}_{\mathbf{z}_t \sim g_\phi(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})} \log\left(\frac{p\left(\mathbf{z}_t \mid \mathcal{H}_{t-1}\right)}{g_\phi\left(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right)}\right)\right) \tag{4.12}$$

Next, we factorize the expectations in 4.12 using the semi-implicit hierarchy in 4.4 :

$$= \sum_{t=1}^{T} \left(\mathbb{E}_{\psi_t \sim q_\phi(\psi_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})} \mathbb{E}_{\mathbf{z}_t \sim q(\mathbf{z}_t \mid \psi_t)} \log p\left(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right)\right) \tag{4.13}$$

$$- \sum_{t=1}^{T} \left(\mathrm{KL}(\mathbb{E}_{\psi_t \sim q_\phi(\psi_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})} q\left(\mathbf{z}_t \mid \psi_t\right) \,||\, p\left(\mathbf{z}_t \mid \mathcal{H}_{t-1}\right)\right) \tag{4.14}$$

Notice, we used the expectation with respect to the latent variable $\mathbf{z}_t$ to form the KL divergence. The reason behind the relative entropy of our variational posterior to the prior. Our aim is to bound the KL term as much as possible, but here $q_\phi\left(\psi_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right)$ is not always an explicit distribution. To circumvent this issue, we bound the KL term thanks to its convexity with respect to a functional [108],

$$\geq \sum_{t=1}^{T} \left(\mathbb{E}_{\psi_t \sim q_\phi(\psi_t | \mathcal{X}_t, \mathcal{H}_{t-1})} \mathbb{E}_{\mathbf{z}_t \sim q(\mathbf{z}_t | \psi_t)} \log p\left(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right) \right) \tag{4.15}$$

$$- \sum_{t=1}^{T} \left(\mathbb{E}_{\psi_t \sim q_\phi(\psi_t | \mathcal{X}_t, \mathcal{H}_{t-1})} \mathrm{KL}(q\left(\mathbf{z}_t \mid \psi_t\right) || \, p\left(\mathbf{z}_t \mid \mathcal{H}_{t-1}\right) \right) = \underline{\mathcal{L}}_{\text{SI-VarConvLSTM}} \tag{4.16}$$

which concludes our derivation of the variational lower bound.

While Monte Carlo estimation of $\underline{\mathcal{L}}_{\text{SI-VarConvLSTM}}$ only requires $q_\phi\left(\mathbf{z}_t \mid \psi_t\right)$ to have an analytic density function and $q_\phi\left(\boldsymbol{\psi}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right)$ to be convenient to sample from, $g_\phi\left(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right)$ is often intractable, and so the Monte Carlo estimation of the ELBO $\underline{\mathcal{L}}_{\text{SI-VarConvLSTM}}$ is prohibited. Therefore, SI-VarConvLSTM evaluates the lower bound separately from the distribution sampling. While the combination of an explicit $q_{\boldsymbol{\phi}}\left(\mathbf{z}_t \mid \boldsymbol{\psi}_t\right)$ with an implicit $q_{\boldsymbol{\phi}}\left(\boldsymbol{\psi}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right)$ is as powerful as needed, it is computationally tractable. As discussed in [55] , without early stopping optimization, $q_\phi\left(\boldsymbol{\psi}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right)$ can converge to a point mass density, making SI-VarConvLSTM degenerated to Var-ConvLSTM. To avoid degeneracy of SIVI, we add a regularizer to the variational lower bound $\underline{\mathcal{L}}_K = \underline{\mathcal{L}}_{\text{SI-VarConvLSTM}} + B_K$ as inspired by SIVI [55]:

$$B_K = \sum_{t=1}^{T} \mathbb{E}_{\psi_t, \psi_t^{(1)}, \ldots, \psi_t^{(K)} \sim q_\phi(\psi_t | \mathcal{X}_t, \mathcal{H}_{t-1})} \mathbf{KL}\left(q\left(\mathbf{z}_t \mid \boldsymbol{\psi}_t\right) \| \tilde{g}_K\left(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right)\right) \tag{4.17}$$

With the additive term, the ELBO becomes asymptotically exact, $\underline{\mathcal{L}}_0 = \mathcal{L}$ and $\lim_{K \to \infty} \underline{\mathcal{L}}_K = \underline{\mathcal{L}}_{\text{SI-VarConvLSTM}}$.

4.2 Doubly Semi - Implicit VarConvLSTM

4.2.1 Necessity of Doubly Semi-Implicit Setting

The standard Normal distribution is the defacto prior distribution for most of the VAE models, but this choice results in over-regularization. One way to alleviate

the over-regularization is to scale the KL term in VAE loss [91, 90], but this requires careful hyperparameter tuning. Recently, [109] showed that aggregated posterior distribution is the optimal prior distribution in the ELBO sense and with form

$$p^*(z) = \frac{1}{N}\sum_{n=1}^{N} q_\phi\left(z \mid x_n\right) \tag{4.18}$$

where the summation is over all training samples x_n ,$n = 1, \ldots, N$. However, this extreme case leads to overfitting and is highly computationally inefficient. A possible middle ground is to consider the variational mixture of posteriors or VampPrior [109]:

$$p^{Vamp}(z) = \frac{1}{K}\sum_{k=1}^{K} q_\phi\left(z \mid u_k\right) \tag{4.19}$$

The VampPrior is an aggregated posterior distribution with K conditioners. These conditioners can be formed from a random subset of training data or learnable. If posteriors are Gaussian, then the VampPrior becomes a mixture of Gaussians prior. However, VampPrior is a special case of semi-implicit prior, in fact, an approximation. In the final section of this chapter, we consider arbitrary and trainable semi-implicit prior distributions in the form:

$$p_\theta^{SI}(z) = \int p_\theta(z \mid \zeta) p_\theta(\zeta) d\zeta \tag{4.20}$$

where θ and ζ are the parameters required for semi-implicit variational inference construction. Thanks to the doubly semi-implicit variational inference, DSI-VarConvLSTM can operate in three different modes:

- Explicit prior & Semi-implicit Posterior (SI-VarConvLSTM case)
- Semi-Implicit prior & Explicit Posterior (Case-II)
- Semi Implicit Prior and Posterior (Case-III)

In the following subsections, while explaining the fundamental mechanisms of DSI-VarConvLSTM, we also derive upper and lower bounds over ELBO for Case-II and Case-III.

Remark. *In all cases, the posterior models are semi-implicit. Thus, the inference procedure of SI-VarConvLSTM is carried exactly for Cases I and III. For Case II, we use a Gaussian distribution as a variational posterior.*

4.2.2 Generation in the Doubly Semi-Implicit Setting

If the prior is explicit, then the generation mechanism is exactly same as in Section 4.1.1. Otherwise, we introduce a semi-implicit hierarchy for the variational prior distribution as

$$\mathbf{z}_t \sim q\left(\mathbf{z}_t \mid \boldsymbol{\zeta}_t\right), \quad \boldsymbol{\zeta}_t \sim q_{\boldsymbol{\Omega}}\left(\boldsymbol{\zeta}_t \mid \mathcal{H}_{t-1}\right) \tag{4.21}$$

where $\boldsymbol{\Omega}$ characterizes the neural network parameters for mixing distribution required for the semi-implicit prior construction. Then, the semi-implicit distribution family $\mathcal{P}$ is the set of all distributions indexed by variational parameter Ω that are marginalized over the distribution $\boldsymbol{\zeta}_t$ defined as

$$\mathcal{P} = \left\{ g_{\Omega}\left(\mathbf{z}_t \mid \mathcal{H}_{t-1}\right) = \int_{\boldsymbol{\psi}_t} q\left(\mathbf{z}_t \mid \boldsymbol{\zeta}_t\right) q_{\boldsymbol{\Omega}}\left(\boldsymbol{\zeta}_t \mid \mathcal{H}_{t-1}\right) d\boldsymbol{\zeta}_t \right\} \tag{4.22}$$

Assuming that the conditional prior is a Gaussian distribution with diagonal covariance, the latent variable becomes

$$\mathbf{z}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_{\text{prior}}^{(t)}, \text{diag}\left(\left(\boldsymbol{\sigma}_{\text{prior}}^{(t)}\right)^2\right)\right) \tag{4.23}$$

where $\left\{\boldsymbol{\mu}_{\text{prior}}^{(t)}, \boldsymbol{\sigma}_{\text{prior}}^{(t)}\right\} = \varphi_{\text{DSI}}^{\text{prior}}\left(\mathcal{H}_{t-1}, \epsilon_t^{1,\ldots,K}\right)$ denote the parameters of semi-implicit prior distributution conditioned on the previous hidden state tensor and random noise vectors fed at the each layer of $\varphi_{\text{DSI}}^{\text{prior}}$. The joint distribution factorization in DSI-VarConvLSTM is formalised as

$$\begin{aligned} p_{DSI}\left(\mathcal{X}_{\leq T}, \mathbf{z}_{\leq T}\right) &= \prod_{t=1}^{T} p\left(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t}\right) p\left(\mathbf{z}_t \mid \mathbf{z}_{<t}, \mathcal{X}_{<t}\right) \\ &= \prod_{t=1}^{T} p\left(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right) g_{\Omega}\left(\mathbf{z}_t \mid \mathcal{H}_{t-1}\right) \end{aligned} \tag{4.24}$$

where the parameters of the likelihood distribution $p\left(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right)$ are modeled with $\varphi_{\text{DSI}}^{\text{decoder}}$.

4.2.3 Learning with Semi-Implicit Priors

Here, we assume that approximate posterior $q_\phi(\boldsymbol{z}_t)$ is explicit. Derivation of VLB for DSI-VarConvLSTM has similar steps with semi-implicit posterior case, except the prior is the only semi-implicit distribution. The KL divergence between the variational posterior and the likelihood of DSI-VarConvLSTM is written as

$$\begin{aligned}
&\mathrm{KL}(q(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}) \,||\, p(\mathcal{X}_{\leq T}, \mathbf{z}_{\leq T})) = \int q(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}) \log\left(\frac{p(\mathcal{X}_{\leq T}, \mathbf{z}_{\leq T})}{q(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T})}\right) d\mathbf{z}_{\leq T} \\
&= \int \sum_{t=1}^{T} \left(q(\mathbf{z}_{\leq T} \mid \mathcal{X}_{\leq T}) \log\left(\frac{p(\mathbf{z}_t \mid \mathcal{X}_{<t}, \mathbf{z}_{<t})\, p(\mathcal{X}_t \mid \mathbf{z}_{\leq t}, \mathcal{X}_{<t})}{q(\mathbf{z}_t \mid \mathcal{X}_{\leq t})}\right)\right) d\mathbf{z}_{\leq T} \\
&= \sum_{t=1}^{T} \left(\int q(\mathbf{z}_{\leq t} \mid \mathcal{X}_{\leq t}) \log\left(\frac{g_\Omega(\mathbf{z}_t \mid \mathcal{H}_{t-1},)\, p(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1})}{q_\phi(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})}\right) d\mathbf{z}_{\leq t}\right)
\end{aligned} \tag{4.25}$$

After interchanging integration and summation, we write (4.11) as the sum of expectations. Then, with the help of distribution factorizations and seperation of the log-likelihood the divergence, we get:

$$= \sum_{t=1}^{T} \left(\mathbb{E}_{\mathbf{z}_t \sim q_\phi(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})} \log p(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1})\right) - \sum_{t=1}^{T} \left(\mathbb{E}_{\mathbf{z}_t \sim q_\phi(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})} \log\left(\frac{q_\phi(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})}{g_\Omega(\mathbf{z}_t \mid \mathcal{H}_{t-1})}\right)\right) \tag{4.26}$$

Next,we express SIVI prior as a discrete mixture:

$$\begin{aligned}
&= \sum_{t=1}^{T} \left(\mathbb{E}_{\mathbf{z}_t \sim q_\phi(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})} \log p(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1})\right) \\
&\quad - \sum_{t=1}^{T} \mathbb{E}_{\mathbf{z}_t \sim q_\phi(\psi_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})} \log \mathbb{E}_{\zeta^{1,\ldots,K} \sim p_\theta(\zeta)} \frac{q_\phi(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1})}{\frac{1}{K}\sum_{k=1}^{K} p_\theta(\mathbf{z}_t \mid \zeta^K)}
\end{aligned} \tag{4.27}$$

Next, using the Jensen's inequailty for logarithm function yields:

$$
\begin{aligned}
&= \sum_{t=1}^{T} \left(\mathbb{E}_{\mathbf{z}_t \sim q_\phi(\mathbf{z}_t | \mathcal{X}_t, \mathcal{H}_{t-1})} \log p\left(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right) \right) \\
&\quad - \sum_{t=1}^{T} \mathbb{E}_{\mathbf{z}_t \sim q_\phi(\psi_t | \mathcal{X}_t, \mathcal{H}_{t-1})} \mathbb{E}_{\zeta^{1,\ldots,K} \sim p_\theta(\zeta)} \log \frac{q_\phi\left(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right)}{\frac{1}{K} \sum_{k=1}^{K} p_\theta\left(\mathbf{z}_t \mid \zeta^K\right)} \\
&\geq \sum_{t=1}^{T} \left(\mathbb{E}_{\mathbf{z}_t \sim q_\phi(\mathbf{z}_t | \mathcal{X}_t, \mathcal{H}_{t-1})} \log p\left(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right) \right) \\
&\quad - \sum_{t=1}^{T} \mathbb{E}_{\zeta^{1,\ldots,K} \sim p_\theta(\zeta)} \mathbb{E}_{\mathbf{z}_t \sim q_\phi(\psi_t | \mathcal{X}_t, \mathcal{H}_{t-1})} \log \frac{q_\phi\left(\mathbf{z}_t \mid \mathcal{X}_t, \mathcal{H}_{t-1}\right)}{\frac{1}{K} \sum_{k=1}^{K} p_\theta\left(\mathbf{z}_t \mid \zeta^K\right)} = \underline{\mathcal{L}}_{\text{DSI-VarConvLSTM}}
\end{aligned}
\tag{4.28}
$$

which concludes our derivation of variational lower bound only using semi-implicit priors.

To derive the upper bound, SIVI uses the concavity of logarithm function. However, it is impossible to use this trick here because of the expectation with respect to the prior. In SIVI, we could upper bound the cross-entropy because we did not sample from the prior using the mixing samples. Consider the variational approximation to the KL-divergence [110]:

$$
\begin{aligned}
\mathrm{KL}(q_\phi(z) \,\|\, p_\theta(z)) &= 1 + \sup_{g : \mathrm{dom}\, z \to \mathbb{R}} \left\{ \mathbb{E}_{q_\phi(z)} g(z) - \mathbb{E}_{p_\theta(z)} e^{g(z)} \right\} \\
&\geq 1 + \sup_{\eta} \left\{ \mathbb{E}_{q_\phi(z)} g(z, \eta) - \mathbb{E}_{p_\theta(z)} e^{g(z,\eta)} \right\}.
\end{aligned}
\tag{4.29}
$$

The VUB using semi-implicit priors becomes:

$$
\overline{\mathcal{L}}_\eta^p = \mathbb{E}_{q_\phi(z)} \log p(x \mid z) - 1 - \mathbb{E}_{q_\phi(z)} g(z, \eta) + \mathbb{E}_{p_\theta(z)} e^{g(z,\eta)} \tag{4.30}
$$

4.2.4 Learning in the Doubly Semi-Implicit Setting

In this subsection, we consider using semi-implicit distributions both for posterior and prior. Thanks to the doubly semi-implicit framework, the joint variational lower and upper bounds are obtained just by inducing semi-implicit hierarchy for the posterior distribution as in the SI-VarConvLSTM case. Let K_1 and K_2

be the number of samples required to construct semi-implicit posterior $q_\phi(\psi)$ and semi-implicit prior respectively. Then, the joint VLB for the doubly semi-implicit case becomes

$$\begin{aligned}\underline{\mathcal{L}}^{q,p}_{K_1,K_2} = \sum_{t=1}^{T} &\mathbb{E}_{\mathbf{z}_t\sim q_\phi(\mathbf{z}_t|\mathcal{X}_t,\mathcal{H}_{t-1})} \log p\left(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right) \\ &- \mathbb{E}_{\psi^{0..K_1}\sim q_\phi(\psi)}\mathbb{E}_{\mathbf{z}_t\sim q_\phi(\mathbf{z}_t|\mathcal{X}_t,\mathcal{H}_{t-1})} \log \frac{1}{K_1+1}\sum_{k=0}^{K_1} q_\phi\left(\mathbf{z}_t \mid \psi^k\right) \\ &+ \mathbb{E}_{\zeta^{1,\ldots,K_2}\sim p_\theta(\zeta)}\mathbb{E}_{\mathbf{z}_t\sim q_\phi(\psi_t|\mathcal{X}_t,\mathcal{H}_{t-1})} \log \frac{1}{K_2}\sum_{k=1}^{K_2} p_\theta\left(\mathbf{z}_t \mid \zeta^K\right)\end{aligned} \tag{4.31}$$

The joint VUB becomes,

$$\overline{\mathcal{L}}^{q,p}_{\eta} = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{z}_t\sim q_\phi(\mathbf{z}_t|\mathcal{X}_t,\mathcal{H}_{t-1})} \log p\left(\mathcal{X}_t \mid \mathbf{z}_t, \mathcal{H}_{t-1}\right) - 1 - \mathbb{E}_{q_\phi(z)} g(z,\eta) + \mathbb{E}_{p_\theta(z)} e^{g(z,\eta)}. \tag{4.32}$$

The lower bound $\underline{\mathcal{L}}^{q,p}_{K_1,K_2}$ is non-decreasing in both K_1 and K_2, and is asymptotically exact:

$$\underline{\mathcal{L}}^{q,p}_{K_1,K_2} \leq \underline{\mathcal{L}}^{q,p}_{K_1+1,K_2}, \qquad \underline{\mathcal{L}}^{q,p}_{K_1,K_2} \leq \underline{\mathcal{L}}^{q,p}_{K_1,K_2+1}, \tag{4.33}$$

$$\lim_{K_1,K_2\to\infty} \underline{\mathcal{L}}^{q,p}_{K_1,K_2} = \mathcal{L}. \tag{4.34}$$

4.2.5 (Doubly) Semi-Implicit Stochastic Layers

The encoder neural network of VAE transforms the observations to Gaussian parameters in a deterministic way. Its corresponding codes are forced to follow a Gaussian distribution, no matter how powerful the deep neural networks are. However, the Gaussian assumption is often too restrictive to model skewness, kurtosis, and multimodality because it is a single-mode distribution. To this end, rather than using a single deterministic encoder with one stochastic layer, we add many stochastic layers as needed, thanks to SIVI as long as the first stochastic layer is reparameterizable and has an analytic PDF. The layers added after are reparameterizable and simple to sample from. More specifically, we construct our encoder network as a deep neural network with M layers whose pre-activations

are concatenated with a reparameterizable noise $\epsilon_t \sim q_t(\epsilon)$. After M layers, we obtain the statistic ℓ_M and use it to generate the Gaussian parameters, as shown below:

$$\begin{aligned} &q_\phi(\mathbf{z}_t \mid \mathcal{X}_t, \mu, \Sigma) = \mathcal{N}(\mu(\mathcal{X}_t, \phi), \Sigma(\mathcal{X}_t, \phi)) \\ &\mu(\mathcal{X}_t, \phi) = f\left(\ell_M, \mathcal{X}_t; \phi\right), \Sigma(\mathcal{X}_t, \phi) = g\left(\ell_M, \mathcal{X}_t; \phi\right) \\ &\ell_t = T_t\left(\ell_{t-1}, \epsilon_t, \mathcal{X}_t; \phi\right), \epsilon_t \sim q_t(\epsilon), t = 1, \ldots, M \end{aligned} \tag{4.35}$$

This moves the encoder's variational distribution beyond a simple Gaussian form since the encoder network's Gaussian statistics are now random variables. The semi-implicit layer for the semi-implicit variational posterior can be modified for a semi-implicit prior as follows:

$$\begin{aligned} &p_\theta(\mathbf{z}_t \mid \mathcal{X}_t, \mu, \zeta) = \mathcal{N}(\mu(\mathcal{X}_t, \zeta), \Sigma(\mathcal{X}_t, \zeta)) \\ &\mu(x, \zeta) = f\left(\ell_M, \mathcal{X}_t; \zeta\right), \Sigma(\mathcal{X}_t, \zeta) = g\left(\ell_M, x\mathcal{X}_t; \zeta\right) \\ &\ell_t = T_t\left(\ell_{t-1}, \epsilon_t, \mathcal{X}_t; \zeta\right), \epsilon_t \sim q_t(\epsilon), t = 1, \ldots, M \end{aligned} \tag{4.36}$$

For ConvLSTM layers, injecting noise to each pre-activation implies that we add the noise as a channel to our tensor $\mathcal{X}_t$. Besides, using semi-implicit layers also serves the same purpose of φ^{prior} and φ^{encoder}, as we learn the distribution through a composition of multiple semi-implicit stochastic layers, learmable feature extractors specifically guided for distribution learning.

In this chapter, we proposed extensions to our *VarConvLSTM* algorithm, namely *SI-VarConvLSTM* and *DSI-VarConvLSTM*. These algortihms employ semi-implicit variational inference, which models posterior and prior distributions as a mixture of explicit variational posteriors mixed via neural networks. In fact, with this setting, it is also possible to model a semi-implicit likelihood, but we did not consider it since our main target is to estimate the future mobile traffic in base station networks, which mainly exhibits Gaussian and interpretable for practical uses. In the next chapter, we present our experimental results showing that incorporating uncertainty via deep variational generative models improves spatio-temporal forecasting systems' prediction capabilities.

Chapter 5

Experimental Results and Discussions

In this chapter, we present our algorithms' performance on spatio-temporal grid sequence forecasting datasets. First, we describe our datasets. Then, we give brief details on the baselines chosen for each dataset and hyperparameter configurations for all algorithms. After elaborating on evaluation metrics, we discuss our results.

The chapter is organized as follows: In Section 5.1, we describe the datasets used in this thesis's experiments. In Section 5.2, we give brief details on baselines and their configurations for each dataset. Section 5.3 briefly discusses our evaluation criteria for each dataset. Finally, in Section 5.4, we elaborate on the results for each dataset.

5.1 Datasets

5.1.1 Benchmarking Spatiotemporal Sequence Datasets

- **Sequential MNIST** [111]: We first evaluate our proposed generative models with a common sequence modeling benchmarking dataset, *Sequential MNIST*, which is the task of MNIST digit generation in a sequential manner. First, the MNIST dataset is binarized as in [111] and then seperated into two subsets: 60,000 training samples and 10,000 testing samples [112].

- **BikeNYC** [113]: This a dataset of spatio-temporal bicycle trajectories obtained from the NYC Bike system, from 1st Apr. 2014 - 30th Sept. 2014. Trip data are composed of trip duration, start-end IDs, and start-end times. Among the data, the last ten days are chosen as testing data, and the others as training data. The goal set for this dataset is to predict the citywide crowd flow.

- **TaxiBJ** [113]: Like BikeNYC, TaxiBJ is also a spatio-temporal trajectory data collected with the same goal, except the crowd we want to measure is the taxicab GPS data and meteorology data collected in Beijing from four distinct time intervals: 1st Jul.2013 - 30th Oct. 2013, 1st Mar. 2014 - 30th Jun. 2014,1st Mar. 2015 - 30th Jun. 2015, 1st Nov. 2015 - 10th Apr.2016. We reserved the data from the last four weeks for the testing data and the remains for the training data. For this thesis, we do not use external factors like holidays and meteorology as our proposed algorithms are not designed to effectively use the external factors, which is a future work to integrate.

Table 5.1: Summary statistics for BikeNYC & TaxiBJ Datasets [113] (holidays include adjacent weekends).

Dataset	**TaxiBJ**	**BikeNYC**
Data type	Taxi GPS	Bike rent
Location	Beijing	New York
Time Span	7/1/2013 - 10/30/2013 3/1/2014 - 6/30/2014 3/1/2015 - 6/30/2015 11/1/2015 - 4/10/2016	4/1/2014 - 9/30/2014
Time interval	30 minutes	1 hour
Grid map size	(32, 32)	(16, 8)
Trajectory data		
Average sampling rate (s)	~ 60	\
# taxis/bikes	34,000+	6,800+
# available time interval	22,459	4,392
External factors (holidays and meteorology)		
# holidays	41	20
Weather conditions	16 types (*e.g.*, Sunny, Rainy)	\
Temperature / °C	$[-24.6, 41.0]$	\
Wind speed / mph	$[0, 48.6]$	\

5.1.2 Turkcell Dataset

To measure the proposed algorithms, we prepared a large-scale mobile traffic dataset extracted from Turkcell servers. The dataset is three months long with 1 hour time interval between consecutive data points. We had 200 base stations at first, but we filtered antenna to keep 140 of them since these stations are located at the region center. The rest of the base stations were sparsely located and receiving weak traffic, so considering those stations would increase the grid graph size, the data sparsity, and the computation time. For this dataset, we only used downlink traffic. The data collection was conducted under the supervision of my co-advisor, Dr. Salih Ergut. The data are fully-anonymized.

Deep learning models employing convolution operation forces us to have regularly rectangular graph-structured data such as tensors, grids, and sequences. However, Turkcell's mobile traffic networks are sampled from an irregularly distributed base station network that does not form a structured data inherently. To resolve this problem, we first construct a regular grid whose total number is equal to the total number of base stations in the network and then construct a one-to-one mapping from antennas to the newly constructed grid points. This mapping

is constructed via the Hungarian mapping algorithm [114], which minimizes the spatial displacement while preserving the geographical correlations. Finally, before feeding the measurements to deep learning models, we apply standardization to fasten the convergence and, therefore, to get superior performance.

5.2 Baselines & Parameter Configurations

We implement the proposed deep forecasting algorithms and baseline deep learning forecasters on PyTorch [115] and Tensorflow [116] respectively. We train the deep learning models on NVIDIA RTX2080 GPUs (2944 cores). The parameter selection procedure was done via 10-fold cross-validation. We select a hyper-parameter set that attains the lowest validation set error among all possible hyperparameter combinations as the best set. In the following subsections, we describe our baselines and parameter configurations for each dataset.

We employ a Gated Recurrent Unit (GRU) [117] as the RNN module of ConvLSTM blocks for our proposals over all datasets. All of the proposed models were trained for 1000 epochs using the Adam optimizer [118] with a learning rate of 0.001 and the mini-batch size of 32. We used cyclic annealing [119] as the KL annealing in ELBO tto impose the prior regularization term gradually and to avoid posterior collapse. All activation functions in neural network models are determined to be ReLU activation functions [120]. We explain other hyperparameter choices in the following subsections.

5.2.1 Sequential MNIST

In the Sequential MNIST dataset, we consider the use the results of the algortihms **DBN**: [121] , **NADE** [122] , **DRAW** [123] , **PixelVAE** [124] , **PixelRNN** [125] , **Z-Forcing** [112] , **VRNN** [106].

We used only one ConvLSTM layer for this dataset to show that the proposed

variational recurrent models are superior to the other models. For this purpose, we had one ConvLSTM layer of 32 3×3 filters for each proposed model. For the semi-implicit layers, due to binarized the dataset, we injected Bernoulli noise as an additional channel to the each layer's inputs.

5.2.2 BikeNYC & TaxiBJ

Like in Sequential MNIST, we use the results from [113, 2] directly. To ensure a fair comparison between the proposed algorithms and the baseline for these datasets, we did not include the external factors as they play a huge role in the predictive performance. We aim to verify that our models can model data distribution without any external factors. Besides, the dominant method for these datasets is ResNet-based approaches. They are not only notoriously difficult to train but also hard to scale to distributed settings, which Turkcell plans to use the proposed traffic prediction system in a federated setting as future work. Here, the classical baselines are **Historical Average (HA)** [126], **ARIMA** [127], and **Vector Autoregression** [128] . Zhang et al. uses three different neural network models: One is called **ST-ANN** [113] , a fully connected neural network fed by extracted spatiotemporal features, **DeepST** [2] a CNN based algorithm which focus on different temporal dependencies and external factor deploying different CNNs for each exogenous variable, and **STResNet** [113], an extended version of DeepST algorithm using ResNet architecture [61].

For *VarConvLSTM*, constructing a two ConvLSTM layers with 3×3 filters 64 channels for feature extractor φ^{prior} was helpful. For φ^{encoder}, we used a three ConvLSTM layers with 3x3 filters and 32 channels. For this dataset, three layered SI-VarConvLSTM and DSI-VarConvLSTM with 45 3x3 filters were chosen after a cross-validation procedure. For the semi-implicit layers, we injected standard Gaussian noise as an additional channel to each semi-implicit layers' inputs.

5.2.3 Turkcell Dataset

- **Linear Regression**: We employed a linear regression model with Elastic Net Regularization [129] implemented via scikit-learn package [130] in Python.
- **ARIMA**: We implemented it using statsmodels [131] package. Its parameters were p = 3,d = 1, and q = 2.
- **HW-ExpS**: Holt-Winters Exponential smoothing was also implemented with statsmodels [131] package. Its parameters were $\alpha = 0.9$, $\beta = 0.1$, and $\gamma = 0.001$.

These methods above were used to forecast network traffic before this project began. Thus, the parameters of these models were not tuned. Then, we trained the deep learning models for a maximum of 1000 epochs using the Adam optimizer [118] with a learning rate of 0.001 learning rate at the mini-batch size of 32.

- **MLP**: After 10-fold cross-validation, the best model was determined to be the one with five hidden layers, 200 hidden units each having a ReLU hidden activation function, where the possible number of hidden layers lies between 1 and 15 and the possible number of hidden neurons lies between 1 and 500. We trained the MLP for 50 epochs.
- **CNN**: Using the same range for hidden layers and neurons, we trained a CNN with ten vanilla convolutional layers, each having a ReLU hidden activation function with batch normalization layers [132], followed by a fully-connected layer. Each layer was configured with 3×3 filters and 32 channels.
- **LSTM**: We used 5 LSTM layers each having 100 hidden neurons.
- **ConvLSTM**: To fairly compare with LSTM, we trained ConvLSTM network with the same number of hidden layers of LSTM, plus we had 3×3 filters and 32 channels.

For *VarConvLSTM*, we used the same hyperparameters with ConvLSTM to compare them reasonably. It is sensible since *VarConvLSTM* is the variational extension to the *ConvLSTM*. One caveat is that 5 layers of ConvLSTM translates to feature extractors φ^{prior} and φ^{encoder} each with 4 hidden layers of ConvLSTM before reparameterization. For this dataset, five-layered SI-VarConvLSTM and six-layered DSI-VarConvLSTM with 48 3x3 filters and 32 3x3 filters were chosen after the cross-validation procedure. For the semi-implicit layers, we injected standard Gaussian noise as an additional channel to the inputs of each layer.

5.3 Prediction Metrics

We present the necessary evaluation metrics for each dataset in this thesis.

- **Sequential MNIST**: We used negative log-likelihood metric to measure the performance. Therefore, the lower the negative log-likelihood, the better the generative modeling of the model is.
- **BikeNYC & TaxiBJ**:

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\widetilde{d}_t - d_t\right)^2} \tag{5.1}$$

  where $\widetilde{d}_t$ are the predicted values, d_t are the corresponding ground truth values, T denotes the total number of measurement points over space and time. This metric is also used in [113].

- **Turkcell Dataset**: For this dataset, we considered relative metrics: Normalised Root Mean Square Error (NRMSE) and Mean Absolute Percentage Error(MAPE) given below:

$$\text{NRMSE} = \frac{1}{\bar{d}}\sqrt{\sum_{t=1}^{T}\frac{\left(\widetilde{d}_t - d_t\right)^2}{T}} \tag{5.2}$$

$$\text{MAPE} = \frac{1}{T}\sum_{t=1}^{T} 100\left|\frac{\widetilde{d}_t - d_t}{d_t}\right| \tag{5.3}$$

where $\bar{d}$ is the prediction mean. NRMSE and MAPE are frequently used for the comparison between data sets or models with different scales. The smaller the NRMSE and/or MAPE, the more accurate the predictions of the model are. The reason why we did not use RMSE is that we want to measure prediction accuracy, and initial experiments have shown that RMSE was not interpretable for our purposes due to the range mismatch in the training and the test dataset.

5.4 Results & Discussions

5.4.1 Results on Sequential MNIST Dataset

The performance comparison for this dataset is provided in Table 5.2. Proposed models are superior to the rest for two reasons: First, we harness convolutional blocks to take advantage of regular grid graph-structured data, which other methods do not employ. This can be seen from the outperformance of VarConvLSTM compared to VRNN. Second, other competing methods always use a parametric posterior and a fixed prior, limiting the possible distributions to model the image data. While PixelRNN has seven layers attains 79.20 NLL and semi-implicit VarConvLSTM methods attain at worst 70 NLL, it shows the importance of nonparametric prior and posteriors distributions. Though increment seems to be small with one stochastic layer, it implies that performance can be further improved by stacking more semi-implicit stochastic layers.

Table 5.2: Comparison of the negative log-likelihood (NLL) between various algorithms for Sequential MNIST. Depending on the nature of the algortihm, we report either exact NLL, approximate NLL (with $\approx$ sign) or the VLB (with $\leq$ sign).

Model	**NLL**
DBN [121]	$\approx$84.55
NADE [122]	88.33
DRAW [123]	$\leq$80.97
PixelVAE [124]	$\approx$79.02
PixelRNN$_{\text{(7-layers)}}$ [125]	79.20
Z-Forcing$_{\text{(1-layer)}}$ [112]	$\leq$ 80.60
VRNN$_{\text{(1-layer)}}$ [106]	$\leq$ 74.15
VarConvLSTM	**71.72**
SI-VarConvLSTM	**70.52**
DSI-VarConvLSTM	**68.03**

5.4.2 Results on BikeNYC/TaxiBJ Dataset

Table 5.3 & 5.4 summarizes the results of our model and other baselines on BikeNYC & TaxiBJ datasets respectively. It is worth to note that the proposed algortihms do not receive any external factors because we wanted to show that semi-implicit variational deep models' capability to model complex distributions. This choice is just to measure the algorithms' pure performance only. External factors would help for semi-implicit models too.

Table 5.3: Comparisons with baselines on BikeNYC. The results of ARIMA, SARIMA, VAR and 4 DeepST variants are taken from [2]. Notice that our models do not use any external factors of BikeNYC dataset.

Model	RMSE
ARIMA [113]	10.07
SARIMA [113]	10.56
VAR [113]	9.92
VarConvLSTM	**8.90**
DeepST-C [2]	8.39
SI-VarConvLSTM	**7.634**
DSI-VarConvLSTM	**6.58**
STResNet [113]	6.33

Table 5.4: RMSE results on TaxiBJ dataset. Notice that our models do not use any external factors of TaxiBJ dataset.

Model	RMSE
HA [113]	57.69
ARIMA [113]	22.78
SARIMA [113]	26.88
VAR [113]	22.88
ST-ANN [113]	19.57
DeepST [2]	18.18
VarConvLSTM	**19.40**
SI-VarConvLSTM	**17.45**
DSI-VarConvLSTM	**16.9**
STResNet [113]	16.69

Proposed models outperform linear autoregressive models and some configurations of DeepST models easily, except VarConvLSTM, which was expected. This

is because VarConvLSTM cannot represent multimodalities due to Gaussian distributions, and receiving no side-information would not help shape the posterior distribution. Transitioning into semi-implicit posteriors is significant in RMSE because this shows that semi-implicit posterior's multimodality helped us perform robust predictions. With DSI-ConvLSTM, we observe that a semi-implicit prior is much better than having a Gaussian prior. However, STResNet is still the best model thanks to the ResNet structure and utilization of external factors, but the gap between STResNet's and DSI-VarConvLSTM's performance is small. Again our aim is to show that implicit posterior and priors are powerful even when there is no side information.

5.4.3 Results on Turkcell Dataset

For the Turkcell dataset, we analyze the performance for 1-step, 5-step, and 10-step ahead predictions. They correspond to 1-hour, 5-hour, and 10-hour ahead predictions. We present the results within one standard deviation obtained by running the experiments 100 times and averaging all the results.

Table 5.5: 1-step ahead prediction evaluation (in Mbps)

Model	NRMSE	MAPE
Linear Regression	0.40 ± 0.04	35.6% ± 0.01%
ARIMA	0.36 ± 0.07	33.5% ± 0.21%
ExpS	0.35 ± 0.09	33.7% ± 0.25%
MLP	0.37 ± 0.04	30.3% ± 0.015%
CNN	0.33 ± 0.06	29.57% ± 0.002%
LSTM - RNN	0.35 ± 0.09	31.7% ± 0.005%
ConvLSTM	0.345 ± 0.05	32.7% ± 0.005%
VarConvLSTM	**0.33 ± 0.10**	**32.9% ± 0.3%**
SI-VarConvLSTM	**0.29 ± 0.12**	**30.68% ± 0.35%**
DSI-VarConvLSTM	**0.27 ± 0.09**	**30.55% ± 0.25%**

Table 5.6: 5-step ahead prediction evaluation (in Mbps)

Model	NRMSE	MAPE
Linear Regression	1.23 ± 0.004	77.8% ± 0.05%
ARIMA	0.65 ± 0.25	73.5% ± 0.21%
ExpS	0.77 ± 0.08	75.67% ± 0.25%
MLP	0.84 ± 0.06	76.8% ± 0.5%
CNN	0.64 ± 0.07	52.3% ± 0.002%
LSTM - RNN	0.70 ± 0.11	55.7% ± 0.4%
ConvLSTM	0.72 ± 0.05	42.7% ± 0.07%
VarConvLSTM	**0.72 ± 0.14**	**42.6% ± 0.15%**
SI-VarConvLSTM	**0.60 ± 0.12**	**36.2% ± 0.1%**
DSI-VarConvLSTM	**0.65 ± 0.10**	**32.55% ± 0.05%**

Table 5.7: 10-step ahead prediction evaluation (in Mbps)

Model	NRMSE	MAPE
Linear Regression	2.90 ± 0.04	95.6% ± 0.01%
ARIMA	2.35 ± 0.09	93.5% ± 0.21%
ExpS	1.95 ± 0.32	93.7% ± 0.25%
MLP	1.25 ± 0.09	80.3% ± 0.015%
CNN	0.85 ± 0.09	65.7% ± 0.002%
LSTM - RNN	0.95 ± 0.09	71.7% ± 0.005%
ConvLSTM	0.84 ± 0.09	46.7% ± 0.005%
VarConvLSTM	**0.85 ± 0.12**	**46.9% ± 0.3%**
SI-VarConvLSTM	**0.705 ± 0.12**	**40.68% ± 0.35%**
DSI-VarConvLSTM	**0.67 ± 0.09**	**33.55% ± 0.25%**

We observe that all models' metrics are very close to each other at Table 5.5. However, as we increase the prediction horizon, linear models stumble a lot on this task because they handle each time-series independently. MLP and LSTM models' performance also gets worse since they do not have the right structure to capture the spatiotemporal structure of the data. CNN's performance is a

bit better than LSTM thanks to capturing the spatial information, but it still is worse than ConvLSTM's performance. Therefore, harnessing both spatial and temporal dependencies is a must for the Turkcell dataset.

VarConvLSTM has similar performance with ConvLSTM except that the prediction variance is larger than the latter. The stochasticity in the *VarConvLSTM* and is the desired property since our aim is to incorporate uncertainty in the wireless networks showing that injecting stochasticity allows us to broaden our possibilities other than a single point estimate. Formulating posterior and prior as nonparametric distributions have reduced the prediction error significantly compared to *VarConvLSTM* and showed the importance of using nonparametric distributions for the Bayesian inference scheme over the Turkcell dataset.

In this chapter, we introduced our results over well-known benchmarking datasets and a curated dataset from Turkcell. Results indicate that proposed variational generative models are capable of modeling data-driven complex prior and posterior distributions efficiently. In the next and the final chapter of this thesis, we summarize our findings and present some future directions.

Chapter 6

Conclusion

We studied spatiotemporal sequence forecasting of high-dimensional grid structured graphs. In Chapter 3, we introduced the spatio-temporal extension of variational autoencoder algorithm, called VarConvLSTM, to bestow implicit regularization, robustness against out-of data distribution, and uncertainty quantification for predictions to resolve overconfident estimates. We constructed isotropic Gaussian latent vectors for each space-time series and a time-dependent prior. The prior structure is a Gaussian with time-dependent statistics, a function of the hidden state tensor from the previous time step. To train the VarConvLSTM, we derive the variational lower bound for this architecture and maximize the lower bound. In Chapter 4, we extended the variational inference scheme to allow arbitrarily complex prior and posterior distributions via semi-implicit and doubly semi-implicit variational inference. We modeled posterior as a semi-implicit distribution by first transforming noise into variational posterior parameters through a succession of stochastic layers, then sampling from the variational posterior; we could obtain samples from an implicit distribution. This approach was also used to generate semi-implicit priors to remove the need to select standard Gaussian priors. We derived variational lower bounds required for training semi-implicit VarConvLSTM's with different configurations: only semi-implicit posterior, only semi-implicit prior, and both semi-implicit prior and posterior. In Chapter 5, we showed that proposed algorithms are applicable for spatiotemporal forecasting

tasks, including space-time mobile traffic forecasting over Turkcell's base station networks, which is a simulation study for 5G network traffic. Proposed algorithms outperformed the popular methods on the SequentialMNIST dataset thanks to the capability of attaining arbitrarily complex prior and posterior distributions. For TaxiBJ/BikeNYC datasets, despite having no access to the external factors like weather and temperature, proposed algorithms have a very close performance to the ResNet based approaches. This observation designates that our models are computationally efficient and could surpass ResNet based algorithms if they processed exogenous features. Finally, on the Turkcell dataset, a real-world mobile traffic dataset, we observed that proposed methods could replace Turkcell's current standards, mainly ARIMA and LSTM. As future work, at Turkcell, we are still working on generative models for dynamic graphs and its Bayesian extensions to forecast traffic values with dynamic graphs. We hope that this future work will lift the restriction brought by grid-structured data and leads to efficient use of exogenous variables like special days, weather, and the correlation between the base stations.

Bibliography

[1] M. K. Titsias and F. J. R. Ruiz, "Unbiased Implicit Variational Inference," *arXiv:1808.02078 [cs, stat]*, Feb. 2019. arXiv: 1808.02078.

[2] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPACIAL '16, (New York, NY, USA), pp. 1–4, Association for Computing Machinery, Oct. 2016.

[3] "Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper."

[4] F. Giust, L. Cominardi, and C. J. Bernardos, "Distributed mobility management for future 5G networks: overview and analysis of existing approaches," *IEEE Communications Magazine*, vol. 53, pp. 142–149, Jan. 2015. Conference Name: IEEE Communications Magazine.

[5] N. Wang, E. Hossain, and V. K. Bhargava, "Backhauling 5G small cells: A radio resource management perspective," *IEEE Wireless Communications*, vol. 22, pp. 41–49, Oct. 2015. Conference Name: IEEE Wireless Communications.

[6] M. Agiwal, A. Roy, and N. Saxena, "Next Generation 5G Wireless Networks: A Comprehensive Survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016. Conference Name: IEEE Communications Surveys Tutorials.

[7] A. Gupta and R. K. Jha, "A Survey of 5G Network: Architecture and Emerging Technologies," *IEEE Access*, vol. 3, pp. 1206–1232, 2015. Conference Name: IEEE Access.

[8] B. Krithikaivasan, Y. Zeng, K. Deka, and D. Medhi, "ARCH-Based Traffic Forecasting and Dynamic Bandwidth Provisioning for Periodically Measured Nonstationary Traffic," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 683–696, June 2007. Conference Name: IEEE/ACM Transactions on Networking.

[9] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, "Long-term forecasting of Internet backbone traffic: observations and initial models," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 2, pp. 1178–1188 vol.2, Mar. 2003. ISSN: 0743-166X.

[10] R. Li, Z. Zhao, X. Zhou, J. Palicot, and H. Zhang, "The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice," *IEEE Communications Magazine*, vol. 52, pp. 234–240, June 2014. Conference Name: IEEE Communications Magazine.

[11] H.-W. Kim, J.-H. Lee, Y.-H. Choi, Y.-U. Chung, and H. Lee, "Dynamic bandwidth provisioning using ARIMA-based traffic forecasting for Mobile WiMAX," *Computer Communications*, vol. 34, pp. 99–106, Jan. 2011.

[12] D. Tikunov and T. Nishimura, "Traffic prediction for mobile network using Holt-Winter's exponential smoothing," in *2007 15th International Conference on Software, Telecommunications and Computer Networks*, pp. 1–5, Sept. 2007.

[13] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114 [cs, stat]*, May 2014. arXiv: 1312.6114.

[14] A. Graves, "Generating Sequences With Recurrent Neural Networks," *arXiv:1308.0850 [cs]*, June 2014. arXiv: 1308.0850.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[16] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019. Conference Name: IEEE Communications Surveys Tutorials.

[17] C. Zhang, M. Fiore, and P. Patras, "Multi-Service Mobile Traffic Forecasting via Convolutional Long Short-Term Memories," *arXiv:1905.09771 [cs, eess]*, May 2019. arXiv: 1905.09771.

[18] C. Zhang and P. Patras, "Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Mobihoc '18, (New York, NY, USA), pp. 231–240, Association for Computing Machinery, June 2018.

[19] S. U. Dar, M. Yurt, L. Karacan, A. Erdem, E. Erdem, and T. Çukur, "Image Synthesis in Multi-Contrast MRI With Conditional Generative Adversarial Networks," *IEEE Transactions on Medical Imaging*, vol. 38, pp. 2375–2388, Oct. 2019. Conference Name: IEEE Transactions on Medical Imaging.

[20] M. Yurt, S. U. H. Dar, A. Erdem, E. Erdem, and T. Çukur, "mustGAN: Multi-Stream Generative Adversarial Networks for MR Image Synthesis," *arXiv:1909.11504 [cs, eess]*, Sept. 2019. arXiv: 1909.11504.

[21] D. Shen, G. Wu, and H.-I. Suk, "Deep Learning in Medical Image Analysis," *Annual Review of Biomedical Engineering*, vol. 19, no. 1, pp. 221–248, 2017. _eprint: https://doi.org/10.1146/annurev-bioeng-071516-044442.

[22] C. Cao, F. Liu, H. Tan, D. Song, W. Shu, W. Li, Y. Zhou, X. Bo, and Z. Xie, "Deep Learning and Its Applications in Biomedicine," *Genomics, Proteomics & Bioinformatics*, vol. 16, pp. 17–32, Feb. 2018.

[23] Y.-W. Chen and L. C. Jain, eds., *Deep Learning in Healthcare: Paradigms and Applications*. Springer, 1st ed. 2020 edition ed., Nov. 2019.

[24] Y. Huang and Y. Chen, "Autonomous Driving with Deep Learning: A Survey of State-of-Art Technologies," *arXiv:2006.06091 [cs]*, July 2020. arXiv: 2006.06091.

[25] V. G. Goecks, "Human-in-the-Loop Methods for Data-Driven and Reinforcement Learning Systems," *arXiv:2008.13221 [cs, stat]*, Aug. 2020. arXiv: 2008.13221.

[26] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial," *arXiv:1710.02913 [cs, math]*, June 2019. arXiv: 1710.02913.

[27] A. T. Z. Kasgari, W. Saad, and M. Debbah, "Human-in-the-Loop Wireless Communications: Machine Learning and Brain-Aware Resource Management," *arXiv:1804.00209 [cs, eess, math]*, July 2019. arXiv: 1804.00209.

[28] J. Nixon, M. Dusenberry, G. Jerfel, T. Nguyen, J. Liu, L. Zhang, and D. Tran, "Measuring Calibration in Deep Learning," *arXiv:1904.01685 [cs, stat]*, Aug. 2020. arXiv: 1904.01685.

[29] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in *International Conference on Machine Learning*, pp. 1321–1330, PMLR, July 2017. ISSN: 2640-3498.

[30] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift," *arXiv:1906.02530 [cs, stat]*, Dec. 2019. arXiv: 1906.02530.

[31] C. Rose and R. Yates, "Location uncertainty in mobile networks: a theoretical framework," *IEEE Communications Magazine*, vol. 35, pp. 94–101, Feb. 1997. Conference Name: IEEE Communications Magazine.

[32] S. K. Das and C. Rose, "Coping with Uncertainty in Mobile Wireless Networks," in *Emerging Location Aware Broadband Wireless Ad Hoc Networks*

(R. Ganesh, S. L. Kota, K. Pahlavan, and R. Agustí, eds.), pp. 189–204, Boston, MA: Springer US, 2005.

[33] A. Kristiadi, M. Hein, and P. Hennig, "Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks," in *International Conference on Machine Learning*, pp. 5436–5446, PMLR, Nov. 2020. ISSN: 2640-3498.

[34] A. G. Wilson and P. Izmailov, "Bayesian Deep Learning and a Probabilistic Perspective of Generalization," *arXiv:2002.08791 [cs, stat]*, Apr. 2020. arXiv: 2002.08791.

[35] A. G. Wilson, "The Case for Bayesian Deep Learning," *arXiv:2001.10995 [cs, stat]*, Jan. 2020. arXiv: 2001.10995.

[36] R. M. Neal, *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics, New York: Springer-Verlag, 1996.

[37] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, (Madison, WI, USA), pp. 681–688, Omnipress, June 2011.

[38] R. M. Neal, "MCMC using Hamiltonian dynamics," *arXiv:1206.1901 [physics, stat]*, June 2012. arXiv: 1206.1901.

[39] H. Ritter, A. Botev, and D. Barber, "A Scalable Laplace Approximation for Neural Networks," Feb. 2018.

[40] T. P. Minka, "Expectation propagation for approximate Bayesian inference," in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, UAI'01, (San Francisco, CA, USA), pp. 362–369, Morgan Kaufmann Publishers Inc., Aug. 2001.

[41] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, pp. 6402–6413, Curran Associates, Inc., 2017.

[42] E. Lobacheva, N. Chirkova, M. Kodryan, and D. P. Vetrov, "On Power Laws in Deep Ensembles," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[43] P. Izmailov, W. J. Maddox, P. Kirichenko, T. Garipov, D. Vetrov, and A. G. Wilson, "Subspace Inference for Bayesian Deep Learning," in *Uncertainty in Artificial Intelligence*, pp. 1169–1179, PMLR, Aug. 2020. ISSN: 2640-3498.

[44] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, "Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, pp. 8789–8798, Curran Associates, Inc., 2018.

[45] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational Inference: A Review for Statisticians," *Journal of the American Statistical Association*, vol. 112, pp. 859–877, Apr. 2017.

[46] M. J. Wainwright and M. I. Jordan, "Graphical Models, Exponential Families, and Variational Inference," *FNT in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2007.

[47] G. E. Hinton and D. van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the sixth annual conference on Computational learning theory*, COLT '93, (New York, NY, USA), pp. 5–13, Association for Computing Machinery, Aug. 1993.

[48] Y. Gal and Z. Ghahramani, "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference," *arXiv:1506.02158 [cs, stat]*, Jan. 2016. arXiv: 1506.02158.

[49] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *International Conference on Machine Learning*, pp. 1050–1059, PMLR, June 2016. ISSN: 1938-7228.

[50] I. Osband, "Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout," p. 5.

[51] J. Shi, S. Sun, and J. Zhu, "Kernel Implicit Variational Inference," Feb. 2018.

[52] S. Mohamed and B. Lakshminarayanan, "Learning in Implicit Generative Models," Oct. 2016.

[53] F. Huszár, "Variational Inference using Implicit Distributions," *arXiv:1702.08235 [cs, stat]*, Feb. 2017. arXiv: 1702.08235.

[54] D. Molchanov, V. Kharitonov, A. Sobolev, and D. Vetrov, "Doubly Semi-Implicit Variational Inference," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2593–2602, PMLR, Apr. 2019. ISSN: 2640-3498.

[55] M. Yin and M. Zhou, "Semi-Implicit Variational Inference," *arXiv:1805.11183 [cs, stat]*, May 2018. arXiv: 1805.11183.

[56] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song, and Y. Li, "Big Data Driven Mobile Traffic Understanding and Forecasting: A Time Series Approach," *IEEE Transactions on Services Computing*, vol. 9, pp. 796–805, Sept. 2016. Conference Name: IEEE Transactions on Services Computing.

[57] P. R. Winters, "Forecasting Sales by Exponentially Weighted Moving Averages," *Management Science*, vol. 6, no. 3, pp. 324–342, 1960. Publisher: INFORMS.

[58] A. Furno, M. Fiore, and R. Stanica, "Joint spatial and temporal classification of mobile traffic demands," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, (Atlanta, GA), pp. 1–9, IEEE, May 2017.

[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, May 2017.

[60] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language Modeling with Gated Convolutional Networks," *arXiv:1612.08083 [cs]*, Sept. 2017. arXiv: 1612.08083.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015.

[62] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997.

[63] T. Ergen and E. Ceyani, "A highly efficient recurrent neural network architecture for data regression," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2018.

[64] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," *Advances in Neural Information Processing Systems*, vol. 28, pp. 802–810, 2015.

[65] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," *arXiv:1412.0767 [cs]*, Oct. 2015. arXiv: 1412.0767.

[66] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, May 2017.

[67] K. Xu, M. Zhang, J. Li, S. S. Du, K.-i. Kawarabayashi, and S. Jegelka, "How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks," *arXiv:2009.11848 [cs, stat]*, Nov. 2020. arXiv: 2009.11848.

[68] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," in *International Conference on Machine Learning*, pp. 1613–1622, PMLR, June 2015. ISSN: 1938-7228.

[69] D. Krueger, C.-W. Huang, R. Islam, R. Turner, A. Lacoste, and A. Courville, "Bayesian Hypernetworks," *arXiv:1710.04759 [cs, stat]*, Apr. 2018. arXiv: 1710.04759.

[70] C. Louizos and M. Welling, "Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors," *arXiv:1603.04733 [cs, stat]*, June 2016. arXiv: 1603.04733.

[71] C. Louizos and M. Welling, "Multiplicative Normalizing Flows for Variational Bayesian Neural Networks," *arXiv:1703.01961 [cs, stat]*, June 2017. arXiv: 1703.01961.

[72] L. Zhu and N. Laptev, "Deep and Confident Prediction for Time Series at Uber," *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 103–110, Nov. 2017. arXiv: 1709.01907.

[73] Y. Gal and Z. Ghahramani, "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks," *Advances in Neural Information Processing Systems*, vol. 29, pp. 1019–1027, 2016.

[74] M. W. Dusenberry, G. Jerfel, Y. Wen, Y.-A. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, and D. Tran, "Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors," *arXiv:2005.07186 [cs, stat]*, Aug. 2020. arXiv: 2005.07186.

[75] S. Sun, C. Chen, and L. Carin, "Learning Structured Weight Uncertainty in Bayesian Neural Networks," in *Artificial Intelligence and Statistics*, pp. 1283–1292, PMLR, Apr. 2017. ISSN: 2640-3498.

[76] N. Pawlowski, A. Brock, M. C. H. Lee, M. Rajchl, and B. Glocker, "Implicit Weight Uncertainty in Neural Networks," *arXiv:1711.01297 [cs, stat]*, May 2018. arXiv: 1711.01297.

[77] K. Shridhar, F. Laumann, and M. Liwicki, "Uncertainty Estimations by Softplus normalization in Bayesian Convolutional Neural Networks with Variational Inference," *arXiv:1806.05978 [cs, stat]*, May 2019. arXiv: 1806.05978.

[78] K. Shridhar, F. Laumann, and M. Liwicki, "A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference," *arXiv:1901.02731 [cs, stat]*, Jan. 2019. arXiv: 1901.02731.

[79] Kingma, D.P., Welling, Max, Mooij, Joris, and Intelligent Sensory Information Systems (IVI, FNWI), *Variational inference & deep learning*. PhD thesis.

[80] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised Learning with Deep Generative Models," *Advances in Neural Information Processing Systems*, vol. 27, pp. 3581–3589, 2014.

[81] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improving Variational Inference with Inverse Autoregressive Flow," *arXiv:1606.04934 [cs, stat]*, Jan. 2017. arXiv: 1606.04934.

[82] D. P. Kingma, T. Salimans, and M. Welling, "Variational Dropout and the Local Reparameterization Trick," *arXiv:1506.02557 [cs, stat]*, Dec. 2015. arXiv: 1506.02557.

[83] S. Sun, G. Zhang, J. Shi, and R. Grosse, "Functional Variational Bayesian Neural Networks," *arXiv:1903.05779 [cs, stat]*, Mar. 2019. arXiv: 1903.05779.

[84] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, (Atlanta, GA, USA), pp. III–1310–III–1318, JMLR.org, June 2013.

[85] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *arXiv:1409.3215 [cs]*, Dec. 2014. arXiv: 1409.3215.

[86] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," in *International Conference on Machine Learning*, pp. 2048–2057, PMLR, June 2015. ISSN: 1938-7228.

[87] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An Introduction to Variational Methods for Graphical Models," in *Learning in Graphical Models* (M. I. Jordan, ed.), pp. 105–161, Dordrecht: Springer Netherlands, 1998.

[88] T. Salimans, D. P. Kingma, and M. Welling, "Markov Chain Monte Carlo and Variational Inference: Bridging the Gap," *arXiv:1410.6460 [stat]*, May 2015. arXiv: 1410.6460.

[89] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Ann. Math. Statist.*, vol. 22, pp. 79–86, Mar. 1951. Publisher: Institute of Mathematical Statistics.

[90] A. A. Alemi, B. Poole, I. Fischer, J. V. Dillon, R. A. Saurous, and K. Murphy, "Fixing a Broken ELBO," *arXiv:1711.00464 [cs, stat]*, Feb. 2018. arXiv: 1711.00464.

[91] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," Nov. 2016.

[92] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661.

[93] D. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," in *International Conference on Machine Learning*, pp. 1530–1538, PMLR, June 2015. ISSN: 1938-7228.

[94] D. P. Kingma and P. Dhariwal, "Glow: Generative Flow with Invertible 1x1 Convolutions," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, pp. 10215–10224, Curran Associates, Inc., 2018.

[95] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP," *arXiv:1605.08803 [cs, stat]*, Feb. 2017. arXiv: 1605.08803.

[96] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked Autoregressive Flow for Density Estimation," *arXiv:1705.07057 [cs, stat]*, June 2018. arXiv: 1705.07057.

[97] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing Flows for Probabilistic Modeling and Inference," *arXiv:1912.02762 [cs, stat]*, Dec. 2019. arXiv: 1912.02762.

[98] E. G. Tabak and C. V. Turner, "A Family of Nonparametric Density Estimation Algorithms," *Comm. Pure Appl. Math.*, vol. 66, pp. 145–164, Feb. 2013.

[99] A. Hyvärinen and P. Pajunen, "Nonlinear Independent Component Analysis: Existence and Uniqueness Results," *Neural Networks*, vol. 12, no. 3, pp. 429–439, 1999.

[100] V. I. Bogachev, A. V. Kolesnikov, and K. V. Medvedev, "Triangular transformations of measures," *Sb. Math.*, vol. 196, p. 309, Apr. 2005. Publisher: IOP Publishing.

[101] M. Sugiyama, T. Suzuki, and T. Kanamori, *Density Ratio Estimation in Machine Learning*. Cambridge: Cambridge University Press, 2012.

[102] R. Ranganath, D. Tran, and D. M. Blei, "Hierarchical Variational Models," *arXiv:1511.02386 [cs, stat]*, May 2016. arXiv: 1511.02386.

[103] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Auxiliary Deep Generative Models," *arXiv:1602.05473 [cs, stat]*, June 2016. arXiv: 1602.05473.

[104] F. V. Agakov and D. Barber, "An Auxiliary Variational Method," in *Neural Information Processing* (D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, N. R. Pal, N. Kasabov, R. K. Mudi, S. Pal, and S. K. Parui, eds.), vol. 3316, pp. 561–566, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. Series Title: Lecture Notes in Computer Science.

[105] T. M. Cover and J. A. Thomas, *Elements of Information Theory 2nd Edition*. Hoboken, N.J: Wiley-Interscience, 2 edition ed., July 2006.

[106] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio, "A Recurrent Latent Variable Model for Sequential Data," *arXiv:1506.02216 [cs]*, Apr. 2016. arXiv: 1506.02216.

[107] M. Fraccaro, S. r. K. Sø nderby, U. Paquet, and O. Winther, "Sequential Neural Models with Stochastic Layers," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, pp. 2199–2207, Curran Associates, Inc., 2016.

[108] A. J. Kurdila and M. Zabarankin, *Convex Functional Analysis*. Systems & Control: Foundations & Applications, Birkhäuser Basel, 2005.

[109] J. M. Tomczak and M. Welling, "VAE with a VampPrior," *arXiv:1705.07120 [cs, stat]*, Feb. 2018. arXiv: 1705.07120.

[110] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization," *IEEE Transactions on Information Theory*, vol. 56, pp. 5847–5861, Nov. 2010. Conference Name: IEEE Transactions on Information Theory.

[111] R. Salakhutdinov and I. Murray, "On the quantitative analysis of deep belief networks," in *Proceedings of the 25th international conference on Machine learning - ICML '08*, (Helsinki, Finland), pp. 872–879, ACM Press, 2008.

[112] A. Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio, "Z-Forcing: Training Stochastic Recurrent Networks," *arXiv:1711.05411 [cs, stat]*, Nov. 2017. arXiv: 1711.05411.

[113] J. Zhang, Y. Zheng, and D. Qi, "Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction," *arXiv:1610.00081 [cs]*, Jan. 2017. arXiv: 1610.00081.

[114] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics*, vol. 2, pp. 83–97, Mar. 1955.

[115] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner,

L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *arXiv:1912.01703 [cs, stat]*, Dec. 2019. arXiv: 1912.01703.

[116] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

[117] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv:1412.3555 [cs]*, Dec. 2014. arXiv: 1412.3555.

[118] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017. arXiv: 1412.6980.

[119] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin, "Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing," *arXiv:1903.10145 [cs, stat]*, June 2019. arXiv: 1903.10145.

[120] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, JMLR Workshop and Conference Proceedings, June 2011. ISSN: 1938-7228.

[121] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "MADE: Masked Autoencoder for Distribution Estimation," *arXiv:1502.03509 [cs, stat]*, June 2015. arXiv: 1502.03509.

[122] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle, "Neural Autoregressive Distribution Estimation," *arXiv:1605.02226 [cs]*, May 2016. arXiv: 1605.02226.

[123] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "DRAW: A Recurrent Neural Network For Image Generation," *arXiv:1502.04623 [cs]*, May 2015. arXiv: 1502.04623.

[124] I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville, "PixelVAE: A Latent Variable Model for Natural Images," *arXiv:1611.05013 [cs]*, Nov. 2016. arXiv: 1611.05013.

[125] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks," Jan. 2016.

[126] J. Y. Campbell and S. B. Thompson, "Predicting the Equity Premium Out of Sample: Can Anything Beat the Historical Average?," Tech. Rep. w11468, National Bureau of Economic Research, July 2005.

[127] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. 2016. OCLC: 915507780.

[128] J. Stock and M. W. Watson, "Vector Autoregressions," *Journal of Economic Perspectives*, vol. 15, no. 4, pp. 101 – 116, 2001.

[129] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. _eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2005.00503.x.

[130] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[131] S. Seabold and J. Perktold, "Statsmodels: Econometric and Statistical Modeling with Python," (Austin, Texas), pp. 92–96, 2010.

[132] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, (Lille, France), pp. 448–456, JMLR.org, July 2015.