Predictive Spatio-Temporal Modelling With Neural Networks

Hong-Bin LIU

College of Science and Engineering

James Cook University

This thesis is submitted for the degree of

Doctor of Philosophy

Discipline of IT

November 2020

I would like to dedicate this thesis to my loving family ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Hong-Bin LIU
November 2020

Acknowledgements

The four years of my PhD study at James Cook University has taken me on a great journey, during which I have grown in every aspect. For example, I have significantly improved my research, communication and critical thinking skills. Most importantly, I have become an independent researcher and a life-long learner.

First and foremost, I offer my sincere gratitude to my principal supervisor, Ickjai Lee. I am thankful for his support, patience, forgiveness, inspiration, passion, foresight, and hard work. Without these, I would not have survived this long journey.

I would also like to thank Hao Wu and Weiwei Sun from Fudan University, China, for their contributions to our ICDM 2019 paper. The ICDM paper was the most significant milestone of my PhD study, and I have learnt a lot from our collaboration.

Over the last four years, I was supported by an Australian Research Training Program (RTP) Scholarship with Stipend. I would like to thank James Cook University and the Australian Government for the generous support.

Lastly, I would like to thank my beloved family, Xiaoyuan Su and Zixi Liu.

Statement of the Contribution of Others

Nature of Assistance	Contribution	Names
Supervision support	Primary, secondary, and secondary supervision, respectively.	Professor Ickjai Lee, Dr Joanne Lee, Dr Trina Myers.
Research support	Drafting, paper / thesis design, research idea brain storming, administration support.	Professor Ickjai Lee.
Proofreading support	Grammar, spelling, punctuation, language.	Professor Ickjai Lee and Wendy Smith (paid external proofreader).
Financial support	Australian Research Training Program (RTP) Scholarship with Stipend 3.5 years.	James Cook University

List of Publications

In This Thesis

Some of the research leading to this thesis has appeared previously in the following publications.

Conference Papers

- **Hong-Bin Liu**, Ickjai Lee: "End-to-end Trajectory Transportation Mode Classification using Bi-LSTM Recurrent Neural Network". – *12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, Nov 2017, Nanjing, China. (ERA Rank B)

- **Hong-Bin Liu**, Hao Wu, Weiwei Sun, Ickjai Lee: "Spatio-Temporal GRU for Trajectory Classification". – *19th IEEE International Conference on Data Mining (ICDM)*, Nov 2019, Beijing, China. (ERA Rank **A**, Core Rank **A***))

- **Hong-Bin Liu**, Ickjai Lee: "Bridging the Gap Between Training and Inference for Spatio-Temporal Forecasting". – *24th European Conference on Artificial Intelligence (ECAI)*, August 2020, Santiago de Compostela, Spain. (ERA Rank **A**, Core Rank **A**))

Journal Articles

- **Hong-Bin Liu**, Ickjai Lee: "Irregular-sampled Trajectory Modelling with Spatio-Temporal GRU". – *Transactions on Intelligent Systems and Technology (TIST)*, Submitted

- **Hong-Bin Liu**, Ickjai Lee: "MPL-GAN: Towards Realistic Meteorological Predictive Learning Using Conditional GAN". – *IEEE Access, vol. 8, pp. 93179-93186, 2020*

Abstract

With advances in location-acquisition technologies such as smart phones and watches that have GPS sensors, Internet of Things, a massive amount of spatio-temporal data has been collected. Mining such ambiguous data facilitates many applications, ranging from transportation management and weather prediction to autonomous driving etc.

Predictive spatio-temporal modelling is a challenging task for the following reasons. The first challenge is due to complex non-linear spatio-temporal dependencies. In spatio-temporal systems, nearby locations have strong correlations; such correlations are represented by sub-sequence time steps. Secondly, spatio-temporal data are collected at spatially discrete locations at temporally discrete intervals, resulting in what is known as data sparsity. Using sparse data to model continuous changes brings further challenges. Thirdly, the real world is dynamic, stochastic and unpredictable, whereas, machine learning methods assume the output is deterministic.

To model the complex non-liner spatio-temporal dependencies, we propose a Recurrent Neural Network based model called Spatio-Temporal GRU to succinctly model the spatio-temporal relationships and correlations embedded in irregularly sampled spatio-temporal trajectories. We propose a novel segmented convolutional weight mechanism to capture short-term local spatial variations and correlations in trajectories, and introduce an additional temporal gate to control the information flow for the temporal interval information.

To solve the modelling difficulty caused by data sparsity, we propose a novel curriculum learning based strategy called Temporal Progressive Growing Sampling to effectively bridge the gap between training and inference phases for spatio-temporal sequence forecasting, by transforming the training process from a fully-supervised approach which utilises all available previous ground-truth values to a less-supervised approach which replaces some of the ground-truth context with generated predictions.

To solve the blurry prediction issue in spatio-temporal sequence prediction, we introduce a Meteorological Predictive Learning GAN model (MPL-GAN) that utilises the conditional GAN along with the predictive learning module to handle the uncertainty in future frame prediction.

We conduct extensive experiments on several real-world datasets for various spatio-temporal prediction tasks, such as travel mode classification, next-location prediction, weather forecasting and meteorological imagery prediction. The results show our proposed models consistently achieve exceptional improvements over state-of-the-art baselines.

Table of contents

List of figures

List of tables

Nomenclature

Roman Symbols

a	A scalar
$\mathbf{a}$	A vector
A	A matrix
$\mathbf{W}$	Weight matrix
$\mathcal{X}$	Input
$\hat{y}$	Output or prediction

Greek Symbols

ϵ	A probability variable
τ	A random variable

Superscripts

(i)	specific layer of matrix (e.g. $\mathcal{H}^i$ denotes hidden state at i layer)

Subscripts

i	row / column of a matrix (e.g. x_i)
ij	specific element of a matrix (e.g. x_{ij})

Other Symbols

$\otimes$	Convolutional operation
$\mathrm{A} \odot \mathrm{B}$	Element-wise (Hadamard) product of A and B

$\mathcal{M}$	A model
$\mathbb{R}$	The real numbers

Acronyms / Abbreviations

BPTT	Back-propagate Through Time
CNN	Convolutional neural network
FNN	Feed-forward neural network
GAN	Generative adversarial network
GRU	Gated recurrent unit
LSTM	Long short-term memory
MAE	Mean absolute error
MLP	Multi-layer Perceptron
MSE	Mean square error
NMT	Neural Machine Translation
NWP	Numerical Weather Prediction
PL	Predictive Learning
POI	Point of interest
ReLU	Rectified linear unit
RMSE	Root mean square error
RNN	Recurrent neural network
ST	Spatio-temporal
STSF	Spatio-temporal Sequence Forecasting
TPG	Temporal progressive growing

e.g. Exempli gratia (“for the sake of an example”)

i.e. i.e. Id est (“it is”)

Chapter 1

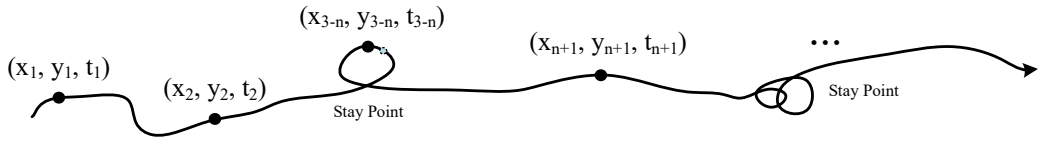
Introduction

1.1 Background and Motivations

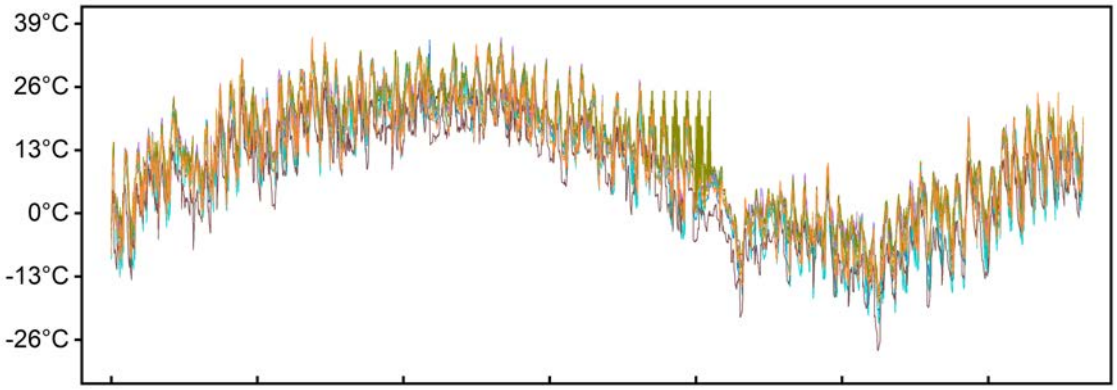
With advances of location-acquisition technologies such as smart phones and watches that have GPS sensors, Internet of things (IOT), a massive amount of spatio-temporal data has been collected. For example, the 'DiDi' platform currently processes an average of 4875 TB+ data points, and receives 106 TB+ vehicle trajectory data points daily [DiD]. Such humongous amounts of data carries interesting people movement patterns. Mining patterns from such data facilitates many applications ranging from transportation management and weather forecasting to autonomous driving etc.

A spatio-temporal dataset consists of both space and time aspects. Predictive spatio-temporal modelling is used to model the space correlations between objects over time to predict the future based on past observations. It is a challenging task to effectively model space and time simultaneously, and previous approaches focus on extracting features from raw data in order to apply machine learning techniques. In recent years, deep learning has dominated in almost every field of machine learning in which it is capable of learning representations from raw data. For example, deep learning gained huge success in large scale image classification by learning complex representations directly from image pixels [He et al., 2016, Krizhevsky et al., 2012]. Similarly, deep learning has also been successful in various natural language processing tasks such as machine translation, sentiment analysis, and question answering [Bordes et al., 2014, Collobert et al., 2011, Devlin et al., 2018, Jean et al., 2014, Sutskever et al., 2014, Vaswani et al., 2017]. Despite the success of deep learning in many fields, it is still a challenging task to adopt deep

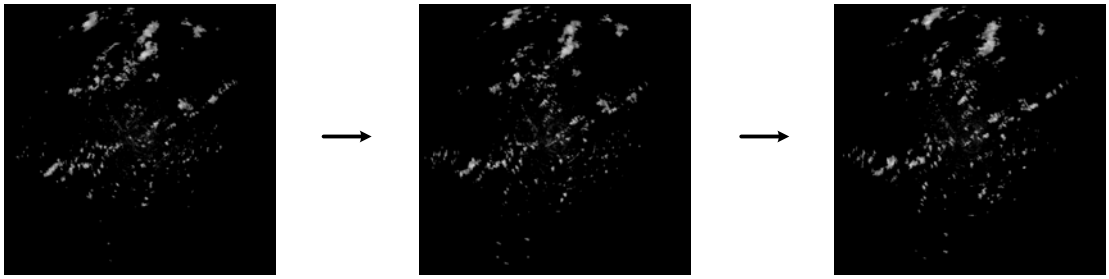
learning into predictive spatio-temporal modelling. In this thesis, we will address several of the challenges of modelling spatio-temporal patterns using deep learning.



(a) A simulated irregular sampled movement trajectory



(b) Air temperatures across 10 weather stations from 2015/3/01 to 2016/6/01, data is sampled every hour.



(c) Radar echo imagery of Hong Kong sampled every 6 minutes.

Fig. 1.1 Three typical spatio-temporal datasets.

In order to introduce the research challenges of predictive spatio-temporal modelling, we first introduce some typical types of spatio-temporal data and their applications.

Firstly, one of the most common types of spatio-temporal data is spatio-temporal trajectory data. Spatio-temporal trajectories consist of various GPS locations points $(\langle x_1, y_1, t_1\rangle, \langle x_2, y_2, t_2\rangle, \ldots, \langle x_t, y_n, t_n\rangle)$, each point records the location $\langle x, y\rangle$ at time t of a moving entity. Figure 1.1a shows a simulated irregularly sampled spatio-temporal trajectory. As shown, a moving object constantly changes speed and direction, and may even remain motionless at a location for a period of time. Irregular sampling, non-linear movement patterns, and uncertainty in measurement makes modelling spatio-temporal trajectories an extremely challenging task. Applications of trajectory modelling include travel mode classification [Liu and Lee, 2017, Liu et al., 2019, Qin et al., 2019], travel time estimation [Wang et al., 2018a, 2014], and next location prediction [Cheng et al., 2013, Monreale et al., 2009, Ye et al., 2010].

Another common type of spatio-temporal data is weather data. Weather is important to our everyday life, and a large number of weather stations are constantly monitoring weather conditions such as temperature, wind speed, rainfall etc. Weather data is normally regularly sampled, for example, Figure 1.1b shows a chart of air temperature across 10 weather stations observed every hour. Therefore, unlike spatio-temporal trajectory data, the time interval is often omitted when constructing the sequence, and each time step of the sequence S is a vector containing measurements across all coordinates. This is denoted as $S = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where $\mathbf{x} \in \mathbb{R}^n$. Applications include weather prediction [Wang et al., 2019a, **?**] and wind / solar power prediction [Bacher et al., 2009, Inman et al., 2013].

Last but not least are spatio-temporal sequences of grid-like coordinates. This is very similar to a weather dataset, except each time step contains a densely observed grid-like matrix instead of a sparsely observed vector. A spatio-temporal sequence is denoted as $S = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n)$, where $\mathcal{X} \in \mathbb{R}^{b \times c}$. For example, Figure 1.1c shows a sequence of radar echo imagery sampled every six minutes. Each frame contains a regular grid of pixels, each pixel represents the radar backscatter for that area on Hong Kong. Applications include precipitation nowcasting [Shi et al., 2015, 2017], urban flow prediction [Zhang et al., 2018] and video prediction [Brock et al., 2019, Karras et al., 2018].

1.2 Research Challenges

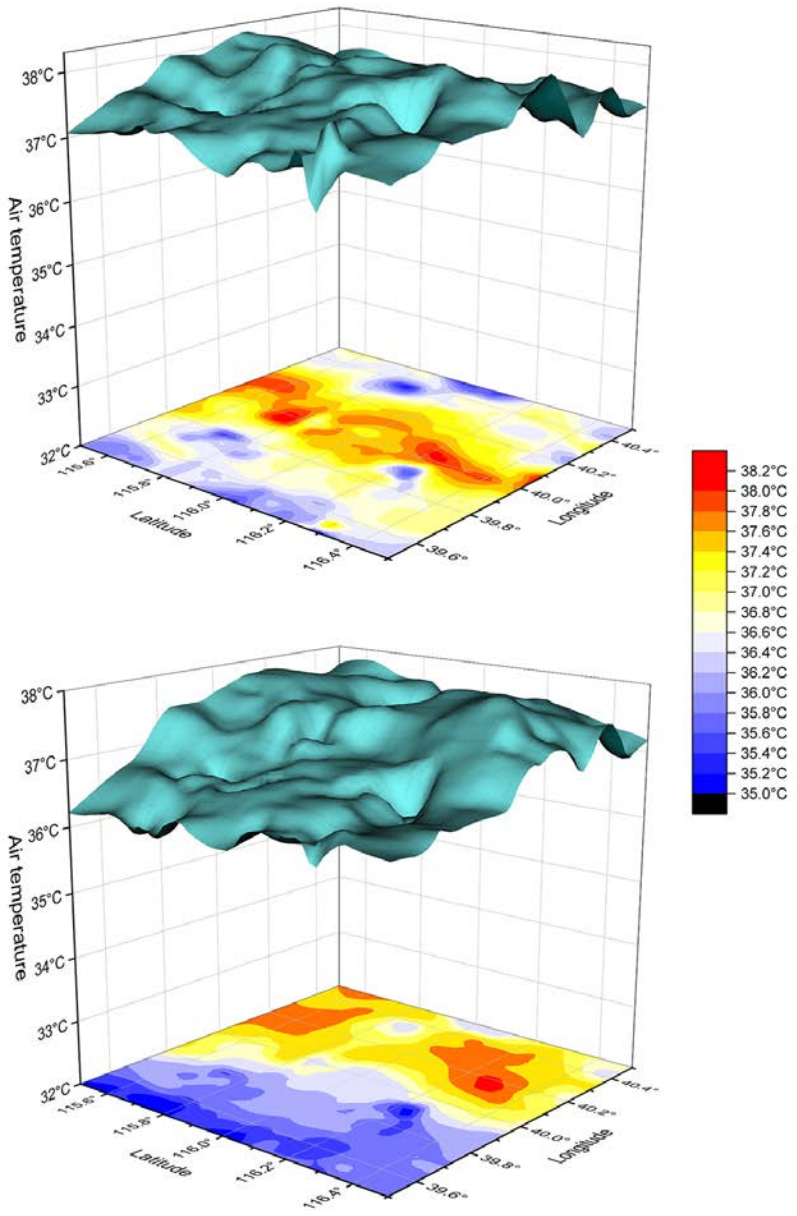


Fig. 1.2 3D simulations of surface air temperature based on observed data from weather stations in Beijing (randomly chosen from two sequential hours). Note: the X-axis represents latitude, the Y-axis represents longitude, and the Z-axis represents the air temperature of the corresponding location (data has been smoothed and interpolated to plot in a 3D graph).

Modelling predictive spatio-temporal data is a challenging task; some of these challenges are shared by different types of spatio-temporal data, whereas, others might be unique to a specific type of spatio-temporal data.

Non-linear spatio-temporal dependency. Let us take the weather data as an example. In the real world, $\mathcal{X}$ constantly, rapidly and dynamically changes at each location over time. For example, the surface air temperature at a location changes over time unceasingly. Moreover, the changing pattern is not static, but dynamic and complicated, that is, it is determined by numerous external factors such as sunshine duration, wind speed, altitude and many other non-obvious environmental factors. Furthermore, a measurement at a location is related to its previous measurements as well as the measurements from nearby locations. The first law of geography [Tobler, 1970] states, "Everything is related to everything else, but near things are more related than distant things." However, both the temporal and spatial correlations are non-linear. Figure 1.2 shows an example of how surface air temperature changes non-linearly over the spatial and temporal axes.

Data sparsity. In real world applications, spatio-temporal data is often collected at a limited number of locations at a regular time interval to capture the spatial and temporal dynamics. That is, data is collected at spatially discrete locations at temporally discrete intervals. For example, Zhang et al. [2018] sampled the city crowd flows into 32×32 grids every 30 minutes. In weather forecasting applications, spatio-temporal sequential data is typically collected by the weather stations across the nation every hour [Ghaderi et al., 2017, Yi et al., 2018]. The underlying reason is that it is practically impossible and unnecessary to collect all the data across all locations in real time. However, data sparsity creates further challenges for deep learning to learn from the raw data.

Uncertainty handling. The real world is dynamic, stochastic and unpredictable, however, machine learning methods assume the output is deterministic. For example, Recurrent Neural Network (RNN) models minimise Euclidean losses such as Mean Absolute Error (MAE) and Mean Square Error (MSE) across the whole sequence. These models make assumptions that the output is deterministic and draw from Gaussian distribution, resulting in blurriness in predicting densely grid-like sequence such as meteorological prediction [Clark et al., 2019, Mathieu et al., 2016, Saito and Saito, 2018, Wang et al.,

2018b]. Uncertainty handling is essential to match the input and prediction distribution to overcome this drawback.

Irregularly sampled. Most spatio-temporal trajectories are irregular where GPS reception is unavailable due to inconsistent weather or environmental conditions, geographical obstacles such as tunnels, device malfunctions, and limited battery issues. Current deep learning models handle regularly sampled sequences such as speech and video using RNN. It remains an unsolved challenge, however, to effectively model irregularly sampled spatio-temporal trajectory in its raw data form [Rehfeld et al., 2011]. As stated previously, spatio-temporal dependencies are non-linear, and non-stationary. Irregularly sampled data creates an additional challenge to model such rapidly changing and dynamic dependencies.

1.3 Thesis Outline

This thesis is organised into seven chapters. The background, motivation and preliminaries are introduced in Chapters 1–3, along with a review of some of the related studies. The methodologies used to address the challenges of predictive spatio-temporal modelling are described in Chapters 4–6. Conclusions are drawn in Chapter 7. Detailed summaries are as follows:

In Chapter 2, we introduce some preliminaries of Deep Learning (DL), including Feed-forward Neural Networks (FNN), Convolution Neural Networks (CNN), Recurrent Neural Networks (RNN), Seq2Seql Model and Scheduled Sampling.

In Chapter 3, we review some major relevant work of predictive spatio-temporal modelling, such as travel mode classification, next-location prediction, and spatio-temporal sequence forecasting etc.

In Chapter 4, we study the difficulty of modelling the spatial dimension and temporal dimension simultaneously for spatio-temporal trajectory. More specifically, we focus on modelling irregularly sampled GPS trajectories based on the travel mode. We propose an RNN based model called Spatio-Temporal GRU to succinctly model the spatio-temporal relationships and correlations embedded in irregularly sampled spatio-temporal trajectories.

In Chapter 5, we study the difficulty of modelling spatial-temporal dependencies for sequence forecasting, and the gap between training and inference phases. We propose a novel curriculum learning based strategy called Temporal Progressive Growing Sampling to effectively bridge the gap between training and inference for spatio-temporal sequence forecasting.

In Chapter 6, we study the blurry predictions of current state-of-the-art deep learning based approaches that optimise the MSE loss. We address this problem by introducing a Meteorological Predictive Learning GAN model (MPL-GAN) that utilises the conditional GAN along with the predictive learning module to handle the uncertainty in future frame prediction.

In Chapter 7, we summarise the study's contributions and limitations, as well as propose some potential future work to address the limitations .

1.4 Contributions and Relative Publications

In this thesis, to address the challenges of predictive spatio-temporal modelling, we make the following main contributions:

- In Chapter 4, we propose an RNN based model called Spatio-Temporal GRU to succinctly model the spatio-temporal relationships and correlations embedded in irregularly sampled spatio-temporal trajectories. We propose a novel segmented convolutional weight mechanism to capture short-term local spatial variations and correlations in trajectories, and introduce an additional temporal gate to control the information flow for the temporal interval information. We present experimental results that demonstrate the superior performance of our proposed method against popular deep learning approaches proposed for spatio-temporal trajectory modelling.

  Relevant publications related to this chapter:

  - **Hong-Bin Liu**, Ickjai Lee: "End-to-end Trajectory Transportation Mode Classification using Bi-LSTM Recurrent Neural Network". – *12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, Nov 2017, Nanjing, China. (ERA Rank B)

  - **Hong-Bin Liu**, Hao Wu, Weiwei Sun, Ickjai Lee: "Spatio-Temporal GRU for Trajectory Classification". – *19th IEEE International Conference on Data Mining (ICDM)*, Nov 2019, Beijing, China. (ERA Rank **A**, Core Rank **A***)
  - **Hong-Bin Liu**, Ickjai Lee: "Irregular-sampled Trajectory Modelling with Spatio-Temporal GRU". – *Transactions on Intelligent Systems and Technology (TIST)*, Submitted

- In Chapter 5, we propose a novel curriculum learning based strategy called Temporal Progressive Growing Sampling to effectively bridge the gap between training and inference for spatio-temporal sequence forecasting, by transforming the training process from a fully-supervised approach which utilises all available previous ground truth values to a less-supervised approach which replaces some of the ground-truth context with generated predictions. To do that we sample the target sequence from midway outputs from intermediate models trained with longer timescales through a carefully designed decaying strategy. Experimental results demonstrate that our proposed method better models long-term dependencies and outperforms baseline approaches on two competitive datasets.

  Relevant publication related to this chapter:

  - **Hong-Bin Liu**, Ickjai Lee: "Bridging the Gap Between Training and Inference for Spatio-Temporal Forecasting". – *24th European Conference on Artificial Intelligence (ECAI)*, August 2020, Santiago de Compostela, Spain. (ERA Rank **A**, Core Rank **A**)

- In Chapter 6, we introduce a Meteorological Predictive Learning GAN model (MPL-GAN) that utilises the conditional GAN along with the predictive learning module to handle the uncertainty in future frame prediction. Experiments on a real-world dataset demonstrate the superior performance of our proposed model. Our proposed model is able to map the blurry predictions produced by traditional MSE loss based predictive learning methods back to their original data distributions, hence, it is able to improve and sharpen the prediction. Specifically, our MPL-GAN achieves an average sharpness of 52.82, which is 14% better than the baseline method. Furthermore, our model correctly detects the meteorological movement patterns that traditional unconditional GANs fail to do.

Relevant publication related to this chapter:

- **Hong-Bin Liu**, Ickjai Lee: "MPL-GAN: Towards Realistic Meteorological Predictive Learning Using Conditional GAN". – *IEEE Access, vol. 8, pp. 93179-93186, 2020*

Chapter 2

Preliminaries of Deep Learning

Deep learning uses multi-layer neural networks that can represent multi-layer abstraction of data of a variety of forms, it has achieved state-of-the-art performance in various areas such as speech recognition, image recognition, machine translation and many other domains [Abdel-Hamid et al., 2014, Druzhkov and Kustikova, 2016, Graves et al., 2013, Schmidhuber, 2015]. In this chapter we will review some of the basic building blocks ranging from Feed-forward Neural Networks (FNN), Convolutional Neural Networks (CNN), to Recurrent Neural Networks (RNN) etc. Along the way, we will introduce the mathematical notations used throughout this thesis.

2.1 Feed-forward Neural Networks

Multi-layer Perceptron (MLP) is the simplest form of FNN that is inspired by neuroscience [Rumelhart et al., 1986]. An MLP consists of an input layer, an output layer and one or multiple hidden layers, each layer containing a variable number of neurons. Figure 2.1 shows an example of a two layer MLP where each layer is typically a vector of which each scalar represents a neuron. Neurons have a pairwise connection between layers. Mathematically, we denote the input of the i-th layer as $h^{(i-1)} \in \mathbb{R}^c$ and the output as $h^{(i)} \in \mathbb{R}^p$, then the transformation between layers is as follows:

$$h^{(i)} = \sigma(\mathbf{W}h^{(i-1)} + \mathbf{b}), \tag{2.1}$$

where $\mathbf{W} \in \mathbb{R}^{c \times p}$ is the weight matrix and $\mathbf{b}$ is the bias term. σ denotes the non-linear function known as the activation function that adds non-linearity to the model; common

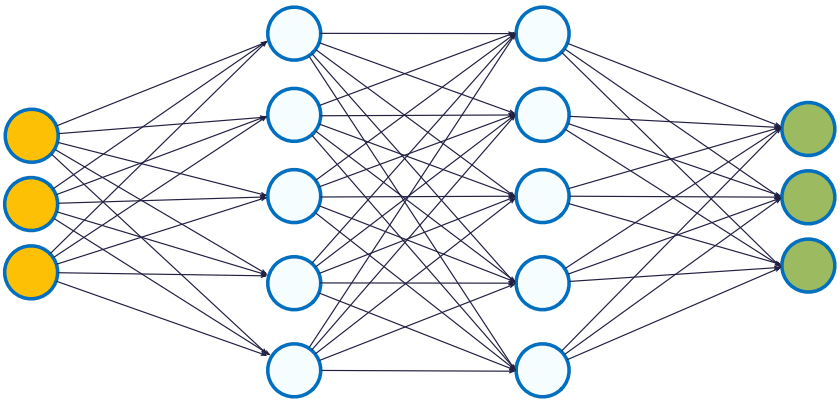


Fig. 2.1 Illustration of a *Multi-layer Perceptron* (MLP) with two hidden layers.

choices of activation function are *Sigmoid*, *Tanh*, and *ReLU*. It has been proven by Hornik et al. [1989] that a multi-layer perceptron with an adequate number of neurons is capable of arbitrarily accurate approximation to any non-linear function.

MLPs had been used in various applications in the 1980s, such as hand-written digit recognition and speech recognition [Khotanzad and Chung, 1998]. To improve the robustness of the model, recent deep learning techniques have been used to try to increase the depth of the layers and the number of parameters as well. However, simply increasing the number of layers does not boost the performance, because the difficulty of parameter tuning increases along with the increasing depth. Pure MLP architecture is outdated in modern deep learning systems, but the concept of fully-connected networks is still being widely used.

2.2 Convolutional Neural Network

A CNN is a type of FNN that was proposed by LeCun et al. [1989] to tackle image processing problems and it achieved state-of-the-art results in handwritten digit recognition in 1989. The recent success of deep CNNs [He et al., 2016, Krizhevsky et al., 2012] reached human's limits on several tasks . Current CNN models typically consist of convolution and pooling layers, followed by fully-connected layers at the end of the network.

Convolution layer. A convolution layer is a linear transformation that preserves spatial information in the input image, defined as:

$$\mathcal{H}_p^{(i)} = \sigma\left(\sum_p \mathbf{W}_p \otimes \mathcal{H}^{(i-1)} + \mathbf{b}_p^{(i-1)}\right),$$

where let $\mathcal{H}^{(i)}, \mathcal{H}^{(i-1)} \in \mathbb{R}^{w \times h \times c}$, w and h represent the input *width* and *height*, c represents the input channel number. Similarly, $\mathbf{W} \in \mathbb{R}^{j \times k \times c \times p}$ is the learnable convolution kernel weight, where j, and k represent the kernel width and height, and p is the output channel number also known as the feature map number. σ denotes the non-liner activation function, $\otimes$ denotes the convolution operation, and $\mathbf{b}$ is the learnable bias weight.

Pooling layer. Pooling layers are commonly known as *max pooling* or *average pooling*. The pooling operation is simple, for each stride of pooling filters, compute the maximum or average over the grid as the output. For instance, given a feature map of $m \times n$, with a common setting 2×2 filter with stride of 2, the output feature map is $\frac{m}{2} \times \frac{n}{2}$. Pooling layers are often placed after convolution layers to reduce the feature map size and hence reduce overfitting.

Use of pooling layers is one of the essential techniques that keeps neural networks going deeper and deeper with greater modelling capacity. The others include *dropout* [Srivastava et al., 2014], and *residual connection* [He et al., 2016] etc. Dropout randomly drops connections between layers during training to prevent overfitting. Residual networks are the basis of the groundbreaking technique developed in 2016 that enables us to train over 1000 layers. Residual connections or skip connections are used in residual networks to form identity connections in order to skip over layers and allow gradients to flow further through the network and hence, allow training of more layers to occur.

It is worth mentioning that convolution layers provide several benefits over fully-connected layers for image processing. First, CNNs have significantly fewer parameters than MLPs with the same model capacity which makes CNNs much easier to train. Second, 2D images need to be flattened to 1D vectors to feed them into an MLP, and during this process, images lose their structure of spatial information. In contrast, CNNs preserve the structure of spatial information that is crucial for image processing. Furthermore, CNNs provide the benefit of so called *shift invariance*, that is the location of an object on an image will not effect the detection result of the object. This is due to two main

reasons. First, as mentioned above, CNNs preserve the structure of spatial information. Second, each CNN kernel slides vertically and horizontally through the entire image to extract features. This important aspect of CNNs inspires researchers to apply CNNs to sequence modelling, the details of which will be provided in the next section.

2.2.1 1D-CNN for Sequence Modelling

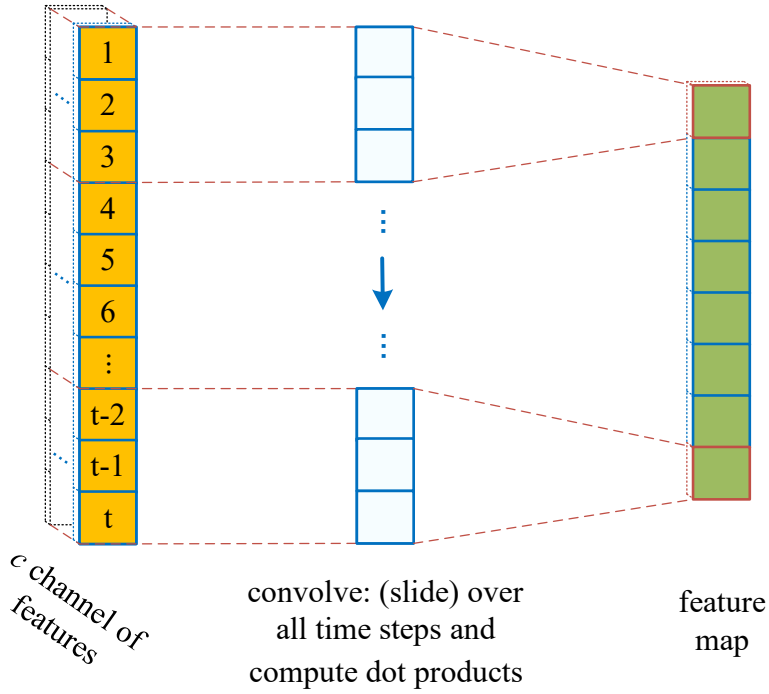


Fig. 2.2 Illustration of a 1D-CNN over a sequence, the kernel size is 3, padding is 1.

While 2D-CNNs are widely used in image processing, 1D-CNNs have also gained tremendous success in sequence modelling, especially in *Natural Language Processing* (NLP), such as machine translation [Gehring et al., 2017] and sentence classification [Kim, 2014]. Transition from a 2D-CNN to a 1D-CNN is simple, a 2D-CNN has a series of two-dimension kernels that slide in both a horizontal and vertical direction, whereas the kernels of a 1D-CNN are vectors and slide over the time dimension. By running this sliding convolve operation over consequential time steps and computing their dot products, the dot products catch the spatial relationship between consequential time

steps. By stacking multiple layers, we can extend the CNN's reception field to a larger range, i.e. to model sequence dependencies of longer range.

2.3 Recurrent Neural Network

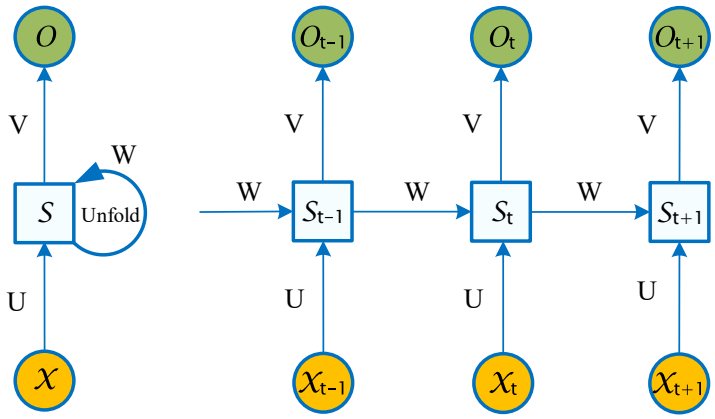


Fig. 2.3 Illustration of an RNN unfolding in time to forward computation.

Even though a 1D-CNN is capable of modelling sequences, it fails to handle variable length sequences. An RNN is designed to handle sequence data of variable length, such as speech and text. While a CNN shares parameters by kernel states, an RNN shares parameters through time steps. An RNN feeds a sequence data forward through time steps by the same shared parameters defined as follows:

$$S_t = f\left(\mathbf{U}\mathcal{X}_t + \mathbf{W}S_{t-1}\right) + \mathbf{b},$$

$$\mathcal{O}_t = \mathbf{V}S_t + \mathbf{c},$$

where, S_t denotes the hidden state at time step t, which is calculated by the previous hidden state S_{h-1} and the input $\mathcal{X}_t$ at time step t. f is the non-linearity function which is usually *tanh*, and $\mathbf{U}$, $\mathbf{W}$, $\mathbf{V}$, $\mathbf{b}$, $\mathbf{c}$ are learnable shared parameters.

With this feed forward though time structure, the RNN naturally handles sequence data of arbitrary length. However, due to gradient explosion and the gradient vanishing problem experienced by vanilla RNN, it is hard to train such a network [Kolen and

Kremer, 2001]. To solve this problem, Hochreiter and Schmidhuber [1997] proposed *Long Short-Term Memory* (LSTM) by including a gating mechanism that controls gradient flows.

$$\begin{cases} i_t = \sigma\left(\mathbf{W}_{xi}\mathcal{X}_t + \mathbf{W}_{hi}h_{t-1} + \mathbf{W}_{ci}c_{t-1} + \mathbf{b}_i\right), \\ f_t = \sigma\left(\mathbf{W}_{xf}\mathcal{X}_t + \mathbf{W}_{hf}h_{t-1} + \mathbf{W}_{cf}c_{t-1} + \mathbf{b}_f\right), \\ c_t = f_t c_{t-1} + i_t \tanh\left(\mathbf{W}_{xc}\mathcal{X}_t + \mathbf{W}_{hc}h_{t-1} + \mathbf{b}_c\right), \\ o_t = \sigma\left(\mathbf{W}_{xo}\mathcal{X}_t + \mathbf{W}_{ho}h_{t-1} + \mathbf{W}_{co}c_t + \mathbf{b}_o\right), \\ h_t = o_t \tanh\left(c_t\right), \end{cases}$$

where i, f, and o denote input, forget, and output gate, respectively, that control gradient flow. c and h are the cell state vector and hidden vector, respectively. By learning the parameters via training the network, it is possible to learn to forget input information from a long range of time steps , or learn what information feeds forward to the next hidden node. Using this gating mechanism, LSTM networks can catch long-term dependencies of different time steps, which improves the significantly performance of modelling sequence data .

Another popular variance is called the Gate Recurrent Unit (GRU) [Cho et al., 2014], which is very similar to LSTM with fewer control gates. We will cover this is detail in Chapter 4.

Vanilla RNN and its variants capture information from previous time steps which means only previous information effects future time steps. However, in many applications, the prediction of $\hat{y}^t$ may depend on future information as well, such as in speech recognition and language modelling. In such tasks, due to the linguistic dependencies, a word prediction may depend on the next few words. To solve such a problem, Schuster and Paliwal [1997] proposed a Bidirectional Recurrent Neural Network (Bi-RNN). A Bi-RNN combines two directional layers, the forward layer accesses the future context, whereas, the backward layer accesses the previous context. Graves [2012] proved that Bi-RNN significantly improves speech recognition performance .

2.3.1 Seq2Seq

The sequence to sequence model (Seq2Seq) was first introduced by Sutskever et al. [2014] to solve Spatial-temporal Sequence Forecasting (STSF) tasks that traditional RNN

approaches could not model due to diverse input lengths and output lengths. Seq2Seq is also known as an encoder-decoder where the encoder encodes the original sequence into a feature vector, then the decoder outputs a target sequence based on the feature vector. Typically, both encoder and decoder are RNNs, and Seq2Seq has been used in sequence modelling tasks like machine translation, speech recognition and recently used for spatio-temporal sequence forecasting tasks [Li et al., 2018, Salinas et al., 2017, Yu et al., 2017].

Despite the Seq2Seq model showing promising performance on spatio-temporal sequence forecasting, the training phase of this model is hard to converge for two reasons. First, Seq2Seq consists of two RNNs and they suffer from the gradient explosion issue and vanish long-term dependencies during training [Pascanu et al., 2013]. LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Cho et al., 2014] have been proposed to overcome such problems, however, RNNs still suffer from training difficulties [Gehring et al., 2017, Pascanu et al., 2013]. Second, in the inference phase, the prediction output $\hat{\mathcal{X}}_t$ at time step t, depends on the prediction output at time step $t-1$. Prediction errors in earlier time steps can lead to bigger error gaps in the generated sequence of later time steps. Bengio et al. [2015] proposed a *curriculum learning* based strategy called *Scheduled Sampling* (SS) that closes the gap between the training phase and the inference phase. We will introduce a scheduled sampling approach to overcome this problem in the next section.

2.3.2 Scheduled Sampling

Before introducing SS, we formulate that the sequence prediction task to predict a sequence of measurements is $\hat{\mathcal{P}} = \left[\hat{\mathcal{X}}_{t+1}, \hat{\mathcal{X}}_{t+2}, \ldots \hat{\mathcal{X}}_{t+k}\right]$, based on the observations of $S = [\mathcal{X}_1, \mathcal{X}_2, \ldots \mathcal{X}_t]$. In the training phase, we cut a length-$(t+k)$ sequence into a length-t sequence S and a length-k sequence $\mathcal{P}$. We use observations S as feature learning for the RNN encoder, and optimise our model based on the decoder output $\hat{\mathcal{P}}$ and the ground-truth $\mathcal{P}$. However, during the inference phase, the ground-truth $\mathcal{P}$ is not available. Therefore, during the inference phase, prediction $\hat{\mathcal{X}}_{t+k}$ relies on the previous output $\hat{\mathcal{X}}_{t+k-1}$. There are two strategies to overcome the difference between the training and inference stages:

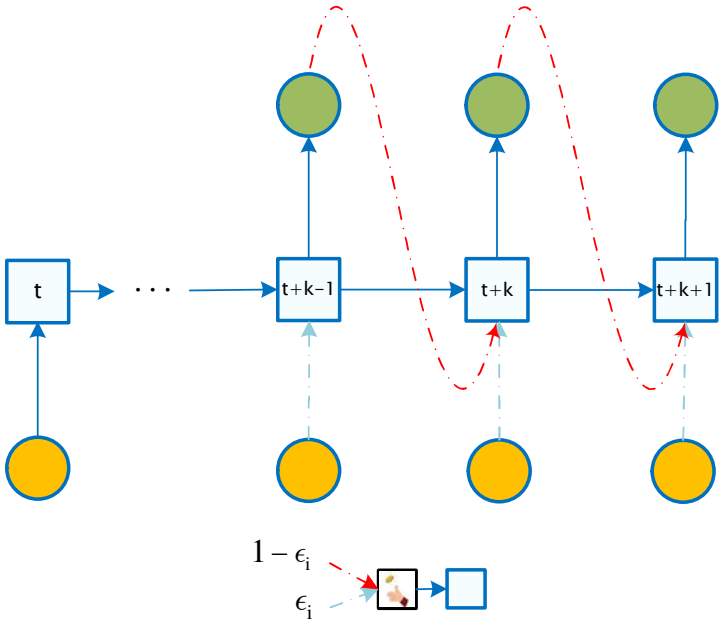


Fig. 2.4 Illustration of Scheduled Sampling: coin flip at each time step of probability ϵ_i to use true sample or otherwise previous output of model itself.

1) During the training phase, use the ground-truth $\mathcal{X}_{t+k}$ as input for the decoder to generate the next prediction $\hat{\mathcal{X}}_{t+k+1}$. Then replace the ground-truth $\mathcal{X}_{t+k}$ with $\hat{\mathcal{X}}_{t+k}$ during the inference phase;

2) Use the same setting during the training and testing phase, i.e. use model generated samples instead of the ground-truth $\mathcal{P}$ as input to the decoder to generate predictions.

While the first strategy benefits the training convergence speed, as every time step of the decoder has the ground-truth value to learn from, it suffers from poor inference performance. Whereas, the second strategy yields better performance, yet suffers from training difficulties as every time step is dependent on previous prediction values [Bengio et al., 2015, 2009, Shi and Yeung, 2018]. To close the gap between the training and inference phase, we adopt the Scheduled Sampling mechanism [Bengio et al., 2015] to start training from strategy 1) and gradually transform to strategy 2). The transformation process is described as follows:

$$
\begin{aligned}
&\forall\, k, 1 \leq k \leq K, \\
&\hat{\mathcal{X}}_{t+k} \sim \mathcal{M}\left(\text{Encoder}(S), \tilde{\mathcal{X}}_{t+1:t+k}; \theta\right), \\
&\tau_{t+k+1} \sim \text{Bernoulli}\left(1, \epsilon_i\right), \\
&\tilde{\mathcal{X}}_{t+k+1} = \left(1 - \tau_{t+k+1}\right) \hat{\mathcal{X}}_{t+k} + \tau_{t+k+1} \mathcal{X}_{t+k}.
\end{aligned}
\tag{2.2}
$$

Here, $\mathcal{M}\left(\text{Encoder}(S), \tilde{\mathcal{X}}_{t+1:t+k}; \theta\right)$ denotes one step prediction based on the encoder and the previous inputs $\tilde{\mathcal{X}}_{t+1:t+k}$. τ_{t+k+1} is a random variable generated by a coin flip following the Bernoulli distribution, where ϵ_i is the probability of $\tilde{\mathcal{X}}_{t+k+1}$ sampling from the ground-truth $\mathcal{X}_{t+k}$, i.e. $1 - \epsilon_i$ is the probability of sampling from the previous prediction $\hat{\mathcal{X}}_{t+k}$. During training, when $\epsilon_i = 1$, $\tilde{\mathcal{X}}_{t+k+1}$ is always sampling from the ground-truth $\mathcal{X}_{t+k}$ which is strategy 1). When $\epsilon_i = 0$, $\tilde{\mathcal{X}}_{t+k+1}$ is always sampling from the previous output $\hat{\mathcal{X}}_{t+k}$ which is strategy 2). Therefore, by controlling ϵ_i gradually decreasing it from 1 to 0, the training goes from strategy 1) to strategy 2) which balances the training speed and inference performance. We will describe the decreasing strategy in the next section.

Chapter 3

Literature Review

In this chapter, we will review some of the typical applications of predictive spatio-temporal trajectory modelling. As mentioned in the introduction, there are three typical types of spatio-temporal datasets. We will group them into two categories by types of dataset: trajectory modelling and spatio-temporal sequence forecasting. Within each category, we will introduce several applications and approaches of both traditional machine learning and deep learning. For more comprehensive surveys, please refer to these two papers, [Shi and Yeung, 2018] and [Wang et al., 2019b]

3.1 Predictive Spatio-Temporal Trajectory Modelling

Predictive spatio-temporal trajectory modelling has a wide range of applications and we review two of the most common tasks: travel mode classification and next location prediction.

3.1.1 Travel Mode Classification

Existing travel mode classification studies can be categorised into two groups: feature based machine learning approaches [Reddy et al., 2010, Shah et al., 2014, Xiao et al., 2017, Zheng et al., 2008a,b, 2010] and deep learning based approaches [Endo et al., 2016a,b, Liu and Lee, 2017, Qin et al., 2019].

Pioneers, Zheng et al. [2008a,b, 2010], proposed several machine learning based approaches to build a classifier based on features extracted from raw GPS logs. More specifically, they extracted nine features such as distance, average velocity, speed,

heading change rate etc. These features were then used to build machine learning based classifiers such as SVM and Random forest. Endo et al. [2016a], Reddy et al. [2010] proposed a Hidden Markov Model (HMM) based approach that works with GPS sensors as well as accelerometer sensors; Shah et al. [2014] took it a step further and combined GPS sensors and GIS information and motion accelerometer sensors; and Xiao et al. [2017] proposed a tree-based ensemble classifier that combines the local and global features extracted from raw trajectories. However, these approaches require heavy feature engineering.

The recent success of deep learning in various applications inspired researchers to apply deep learning AI technologies into this field. Endo et al. [2016a,b] used CNNs to automate the feature extraction process, in particular, they converted raw trajectory data into images and extracted higher-level features from these images by concatenating features like distance and velocity etc. They also utilised deep learning's remarkable feature learning capacity; however, in such transformations from raw trajectory data into images, it is difficult to capture temporal features. Liu and Lee [2017] and Qin et al. [2019] utilised a similar approach to append temporal intervals to spatial aspects directly.

3.1.2 Next Location Prediction

Next location prediction is another hot topic in trajectory modelling. As for travel mode classification, the existing literature on next location prediction can be categorised into two groups: machine learning based approaches [Ye et al., 2010, Yin et al., 2013, Zheng et al., 2009] and deep learning based approaches [Cheng et al., 2013, Fernando et al., 2018, Gao et al., 2015, Kong and Wu, 2018, Monreale et al., 2009, Zhao et al., 2019]. Like travel mode classification, machine learning based methods require extensive feature engineering. On the other hand, deep learning based methods automate the feature extraction, however, they require manually designed architectures. Zhao et al. [2019] proposed a ST RNN which consists of a distance gate and a time gate to model the ST correlations of the next Point of Interest (POI) prediction task. Despite our proposed model sharing a similar temporal gating mechanism, our work is completely different in two ways. First, we introduce a segmented convolutional weight mechanism for spatial modelling, whereas, they use a distance gate. Second, our temporal gate is designed for

modelling various time interval information in trajectory classification, and we focus on the next POI location prediction to evaluate the effectiveness of our proposed time gating mechanism.

3.1.3 Deep Learning for Trajectory Modelling

There are many recent studies in the field of trajectory modelling that utilise deep learning. In particular, several pioneering attempts have been proposed to model trajectories with RNN approaches. Wu et al. [2017] and Gao et al. [2017] tried to model trajectories with multi-layered LSTM. However, in their study, trajectories are modelled by either a series of categorical POIs or a set of road segments which is similar to textual data. Wang et al. [2018a] tried to predict travel time by learning the spatial correlations with a GeoConv layer, but unlike their approach, our approach utilises a 1D-CNN for spatial feature extraction at the cell level. In addition, Yang et al. [2018] proposed a Recurrent-Censored Regression (RCR) model to predict a user's future check-in time at a specific location. Despite several of these attempts using RNN variants for trajectory classification, none of them is particularly designed to handle both the spatial and the temporal aspects of ST trajectories.

3.2 Spatio-temporal Sequence Forecasting

Spatio-temporal sequence forecasting is a well studied research topic, and it is a part of the time series prediction topic. In the literature, traditional machine learning based methods including Support Vector Machine (SVM) [Sapankevych and Sankar, 2009], Gaussian Process (GP) [Flaxman, 2015] and Auto-Regressive Integrated Moving Average (ARIMA) [Box and Jenkins, 1990] have been proposed to deal with this problem.

3.2.1 Deep Learning for a Grid-like Sequence

With recent advances in deep learning models in various domains, the research focus of spatio-temporal sequence forecasting has been redirected to deep models. For example, Shi and Yeung [2018] proposed a Convolutional LSTM Recurrent Neural Network (RNN) for precipitation nowcasting. The main contribution of their work is that it models spatial

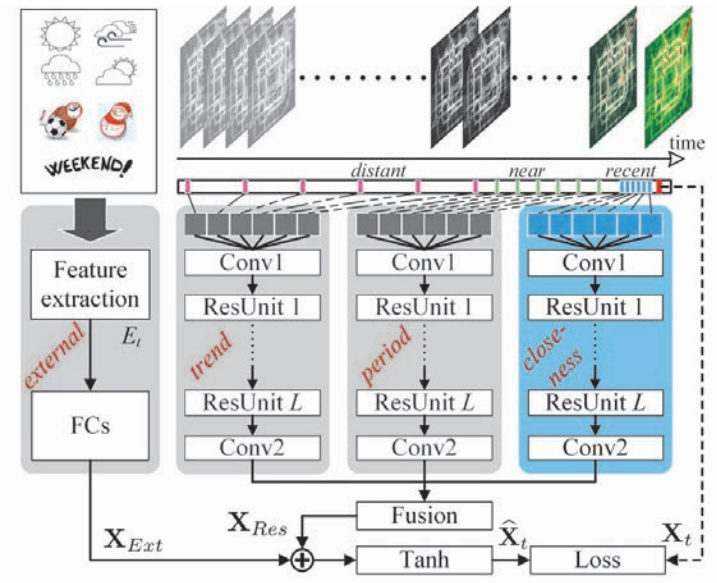


Fig. 3.1 ST-ResNet architecture for Citywide Crowd Flow Prediction. [Zhang et al., 2018]

correlations between two neighbouring time steps by replacing the linear transformation of LSTM with a 2D-CNN, which results in more compact spatial modelling and better performance. Zhang et al. [2018] proposed deep spatio-temporal residual networks to handle the citywide crowd flow prediction problem. Figure 3.1 shows that, temporal closeness, period and trend properties of crowded traffic, as well as external factors such as weather and events are considered and modelled by a fusion network. Similarly, Chen et al. [2018] employed a 3D-CNN network to approach the problem. All of these studies model the spatio-temporal data at a certain timestamp in an image-like format, where each measurement at a location is treated as a pixel of an image. Therefore, modelling a series of spatio-temporal data is the same as modelling videos (a series of images with a regular time interval). Forecasting a spatio-temporal sequence is highly relevant to the problem of video frame prediction. Wang et al. [2017] proposed PredRNN and its followup work PredRNN++ [Wang et al., 2018c] to solve the video frame prediction problem. Their work is based on Seq2Seq with a custom RNN cell called Casual LSTM and a Gradient Highway Unit (GHU).

3.2.2 Deep Learning for Vector-like Sequence

Modelling a spatio-temporal sequence as a grid-like format requires a rich collection of data in a variety of coordinate locations, furthermore, coordinate locations must be in a grid-like format to be compatible, as for city crowd flow prediction. However, not all datasets satisfy these constraints. Another approach is to model spatio-temporal data as a vector, where each measurement of a location is a scalar of a row or column. For example, an LSTM-based model was proposed by Ghaderi et al. [2017] to predict wind speed across 57 measurement locations. To predict weather for 10 weather stations across Beijing City, Wang et al. [2019a] proposed a deep uncertainty Seq2Seq model. Whereas, to predict air quality, Yi et al. [2018] proposed a DeepAir model which consists of a spatial transformation component and a deep distributed fusion network. In another study, Liang et al. [2018] proposed a multi-level attention Seq2Seq model called GeoMAN to model the dynamics of spatio-temporal dependencies. All of these studies above reveal that spatio-temporal correlations are difficult to model, but are crucial for spatial-temporal predictions.

3.2.3 Curriculum Learning Strategy

Venkatraman et al. [2015] proposed a Data As Demonstrator (DAD) model to improve multi-step prediction by feeding paired ground-truth and predicted words to the next step. The gap between a single-step prediction error and a multi-step error in their work is similar to the gap between training and inference in our work. Bengio et al. [2015] further improved the idea by using a Curriculum Learning [Bengio et al., 2009] based approach to close the gap between training and inference by sampling previous ground-truth and previously predicted context, by changing the probability.

3.2.4 Uncertainty Handling for STSF

Last but not least, uncertainty handling is also crucial for spatio-temporal sequence forecasting. Most existing STSF approaches assume that the real-world is deterministic, thus these models generate blurry predictions due to the lack of uncertainty handling. A few recent works have proposed solutions to handle such a problem. Fragkiadaki et al. [2015] proposed a probabilistic forecaster that outputs the parameters of a Gaussian

Mixture Model (GMM) in order to produce non-blurry predictions. Other approaches utilised a Generative Adversarial Network [Goodfellow et al., 2014]. For example, Mathieu et al. [2016] proposed a conditional GAN based uncertainty handling method to generate sharp and realistic video. This approach has also been adopted in the following work [Clark et al., 2019, Saito and Saito, 2018, Wang et al., 2018b]. Besides GAN based methods, a Variational Auto-encoder (VAE) [Kingma and Welling, 2013] is another popular approach for dealing with uncertainties in STSF.

3.2.5 Meteorological Predictive Learning

Optimal flow based methods [Brox et al., 2004, Cheung and Yeung, 2012] have a long history in the meteorological predictive learning literature. With the recent advances in deep learning, Shi et al. [2015] explored the possibility of applying RNNs, and proposed a model called Convolutional LSTM (ConvLSTM) [Shi et al., 2015] and its improved version TrajGRU [Shi et al., 2017] for radar echo imagery prediction. Both approaches tried to optimise the MSE loss. Video frame prediction and traffic flow prediction can be considered to be the same problem. To tackle this problem, PredRNN and its improved version PredRNN++ were proposed by Wang et al. [2017]; these methods did optimise the MSE loss but also shared the issue regarding the prediction getting more blurry over time.

Beyond meteorological imagery predictive learning, neural network based methods have also been used in numerical weather forecasting. For example, Wang et al. [2019a] proposed a Seq2Seq LSTM to predict temperature, wind speed, and relative humidity. An improvement on this method was proposed by Liu and Lee [2020] who introduced a temporal progressive growing schedule sampling strategy. Nevertheless, these approaches suffer from the same long-term prediction accuracy degradation.

3.2.6 GAN for Image and Video Generation

GAN [Goodfellow et al., 2014] has been the most popular generative model since it was first released in 2014. Since then, GAN models have shown their superior abilities especially in image generation, ranging from hand-written digit generation [Goodfellow et al., 2014, Radford et al., 2016] to large scale image set generation [Brock et al., 2019,

Karras et al., 2018]. Recently, researchers tried to push the limits of GAN by generating photo-realistic videos using unconditional GAN [Clark et al., 2019, Mathieu et al., 2016, Saito and Saito, 2018, Tulyakov et al., 2018, Wang et al., 2018b]. Those video GANs aim to produce photo-realistic and temporal coherent videos, and they are used to match the high-dimensional data distribution between the two. Note that those models do not take into account any other considerations, that is, for given initial frames, generated frames do not need to consider the moving entities' direction, speed and other moving information. However, these moving properties play an important role in our study, and, unlike traditional unconditional GANs, our GAN does take into account these properties.

3.3 Summary

We summarised the reviewed methods in the following table.

Table 3.1 Summary of reviewed literature. (TM: Travel Mode Classification; NL: Next-location Prediction; CL: Curriculum Learning; UH: Uncertainty Handling.)

Category	Subcategory	Methods	TM	NL	STSF -Grid	STSF -Vector	CL	UH
Trajectory Modelling	Traditional	[Reddy et al., 2010, Shah et al., 2014, Xiao et al., 2017, Zheng et al., 2008a,b, 2010]	√					
		[Ye et al., 2010, Yin et al., 2013, Zheng et al., 2009]		√				
	DL	[Endo et al., 2016a,b, Liu and Lee, 2017, Qin et al., 2019]	√					
		[Cheng et al., 2013, Fernando et al., 2018, Gao et al., 2015, Kong and Wu, 2018, Monreale et al., 2009, Zhao et al., 2019]		√				
STSF	Traditional	[Box and Jenkins, 1990, Flaxman, 2015, Sapankevych and Sankar, 2009]				√		
	DL	[Chen et al., 2018, Shi and Yeung, 2018, Wang et al., 2017, 2018c]			√			
		[Ghaderi et al., 2017, Liang et al., 2018, Wang et al., 2019a, Yi et al., 2018]				√		
		[Bengio et al., 2015, 2009, Venkatraman et al., 2015]				√	√	
		[Clark et al., 2019, Fragkiadaki et al., 2015, Kingma and Welling, 2013, Mathieu et al., 2016, Saito and Saito, 2018, Wang et al., 2018b]			√			√

3.4 Research Gap

Base on above comprehensive literature review and summary table of reviewed literature, we are able to identify several research gaps in the current literature.

First, none of the existing deep learning models consider both spatial and temporal dimensions simultaneously for the travel mode classification task. They either transform the raw spatio-temporal data into other format such as image resulting missing temporal information, or simply treat temporal as the same as spatial dimension. Furthermore, none of the existing take the in-regularly sampled temporal into account, resulting misleading spatio-temporal feature learning.

Second, researchers simply adapt the Curriculum Learning based approach name scheduled sampling to close the gap between training and inference. Scheduled sampling has been widely used in NLP and many other sequence modelling tasks, however, simply adapt it to spatio-temporal modelling without considering the unique characteristics of spatio-temporal data is not ideal. In other words, there should be some room of improvements if we consider the characteristics of spatio-temporal data when adapting scheduled sampling strategy.

Third, current state-of-the-art deep learning based predictive models mainly optimise the mean square error loss, resulting in blurry spatio-temporal sequence predictions. This is mainly due to the assumption that the data is drawn from the Gaussian distribution which only works on a continuous portion of the image while ignoring isolated small regional areas. Furthermore, current RNN approaches assume that all real-world scenario is predictable. Both issues indicate that uncertainty handling is crucial for producing non-blurry and realistic spatio-temporal sequence predictions.

To fulfil these research gaps, we will introduce several novel methods in the next three chapters to effectively apply deep learning into predictive spatio-temporal modelling.

Chapter 4

Irregularly-sampled Trajectory Modelling with Spatio-temporal GRU

Typically, spatio-temporal trajectories are irregularly sampled, and modelling spatio-temporal trajectories is a hot research topic in geo-computation with many real-world applications, in particular, travel mode classification. In recent years, with the advances in deep learning techniques, many approaches have been proposed to address this research area, in particular, using LSTM based networks for spatio-temporal sequence modelling. Spatio-temporal trajectories have two dimensions, spatial and temporal; however, recent approaches fail to take into account both the spatial dimension and the temporal dimension simultaneously when modelling spatio-temporal trajectories. They either consider the spatial dimension and leave out the temporal dimension, or vice versa. It is crucial to consider both spatial and temporal dimensions for spatio-temporal trajectory modelling to avoid missing any spatio-temporal patterns. In this chapter, we propose an RNN based model called Spatio-Temporal GRU to succinctly model the spatio-temporal relationships and correlations embedded in irregularly sampled spatio-temporal trajectories.

4.1 Introduction

An ST trajectory represents a series of movements for an ST object, and modelling ST trajectories is one of the core processes in trajectory data mining. It leads to diverse real-world applications such as human behaviour analysis, travel mode classification, itinerary recommendation, and animal mobility detection [Lee et al., 2008, Lei et al.,

2011, Zheng, 2015]. As ST trajectories consist of spatial coordinates and corresponding time stamps, the problem of ST trajectory modelling is of great interest in the data mining community. An ST trajectory is a list of ST entries, $(\langle x_1, y_1, t_1\rangle, \langle x_2, y_2, t_2\rangle, \ldots, \langle x_n, y_n, t_n\rangle)$, where $x_i, y_i \in \mathbb{R}^2$ and $t_i \in \mathbb{R}^+$ for $1 \leq i \leq n$ and $t_1 < t_2 < \ldots < t_n$. A regular ST trajectory is when $|t_{j+1} - t_j| = |t_{k+1} - t_k|$ for $\forall\ j, k$ whilst an **irregular ST trajectory** is when $|t_{j+1} - t_j| \neq |t_{k+1} - t_k|$ for $\exists\ j, k$ where $1 \leq j \neq k < n$. Notably, most ST trajectories are irregular due to inconsistent weather or environmental conditions, geographical obstacles such as tunnels, device malfunctions, and limited battery issues where GPS reception is unavailable [Bermingham and Lee, 2018], therefore, we focus on irregular ST trajectories in this chapter. This unique property of irregular ST trajectories creates many unique challenges for ST trajectory modelling.

Figure 4.1 illustrates two ST trajectories with different sampling rates. The top left (black) depicts a walking trajectory with a sampling rate of 30 seconds, while the bottom right (green) shows a driving trajectory with a sampling rate of 10 seconds. Grey dots in the green trajectory represent missing points where data were not collected due to a high-variance in the estimation of speed.

ST trajectory classification involves building a model that can classify ST trajectories into various groups based on underlying features and characteristics. For example, classification of transportation modes from ST trajectories is a popular current trajectory classification problem [Liu et al., 2019, Xiao et al., 2017, Zheng et al., 2010]. Using machine learning approaches, Zheng et al. [2010] aimed to classify human trajectories into four categories based on their travel modes: driving, cycling, walking, and running. Note that typical machine learning-based classification models require a set of features to be extracted from the raw data; and in this study, they first extracted nine features including speed, velocity, and direction, which were used in their machine learning based classifier.

With the recent advances and developments in deep learning, several deep models have been proposed that outperform traditional shallow machine learning models. One of the unique advantages of deep neural networks is that they do not need to perform the feature engineering process, but work well the with raw data. RNN models are the most popular and have become state-of-the-art methods for sequence and time series data among the many deep models. However, traditional RNN-based approaches are

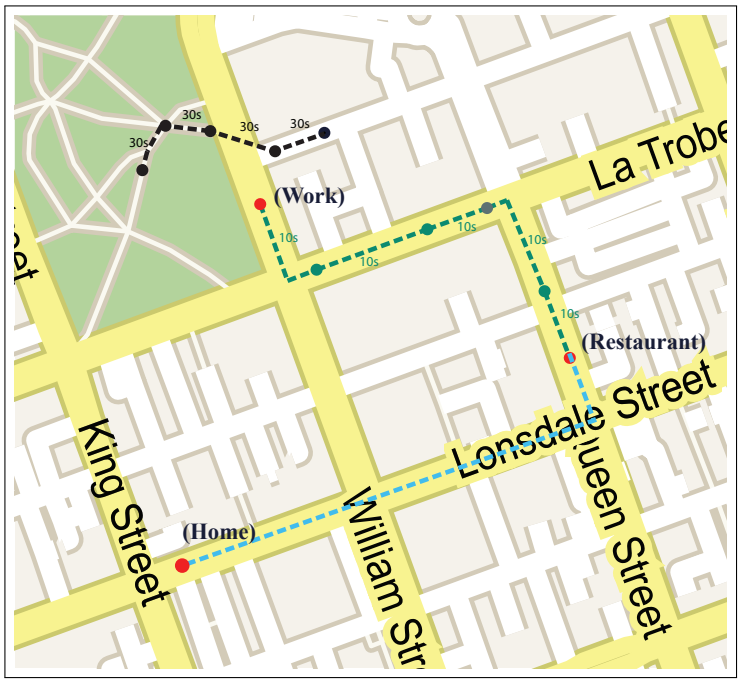


Fig. 4.1 Two trajectory segments with different sampling rates. Top left (black) is a walking trajectory with a sampling rate of 30 seconds and bottom right (green) is a driving trajectory with a sampling rate of 10 seconds. The grey dot represents a missing point which may result in a high-variance in the estimation of speed. The green trajectory shows a user stops at a restaurant after work, then goes home after after dinner; notice that the time intervals between work to restaurant and restaurant to home are different.

designed to model sequence data with regular time intervals, and are not suitable for irregular ST trajectories. For instance, in speech recognition, raw acoustic data are split into equal sized signal frames assuming the same temporal interval, whilst in text data mining, sentences are segmented into words without temporal information. In this study, we consider ST trajectories that are irregularly sampled and have different time intervals. This irregularity of ST trajectories makes them unique, and traditional RNN-based approaches need to be modified to handle the irregularity of ST trajectories as well as their spatiality and temporality.

Many deep neural modes have been proposed for ST trajectory classification tasks. Shah and Romijnders [2016] investigated raw basketball trajectories using an LSTM network to distinguish successful shots from unsuccessful shots. Similarly, Liu and Lee [2017] utilised a bi-LSTM neural network to classify travel modes from raw human

trajectories. Shah and Romijnders [2016] and Liu and Lee [2017] studied the same problem by concatenating the timespan τ to the (x, y). Recently, Shi et al. [2015] realised the importance of temporal interval information in ST trajectories, and Zhu et al. [2017] proposed three Time-LSTM models specifically designed to handle the temporal interval information. However, this approach was limited to the problem of recommendation, and was not designed to be directly applicable to the ST trajectory classification task, as it is not structured to capture local spatial variations. In summary, there have been some studies that have modelled ST trajectories, however to the best of our knowledge, none takes into consideration the unique characteristics of ST trajectories: the irregularity of sampling, simultaneous modelling both the spatial and temporal dimensions, and modelling of local spatial variations (correlations).

To overcome these common drawbacks of traditional approaches, we propose an RNN-based model ST-GRU, specifically designed for irregularly sampled ST trajectory data. The main contributions of this chapter are:

- We propose a segmented convolutional mechanism which improves the computations of a traditional GRU. The mechanism is designed to capture the short-term local correlations of spatial information which is an important factor in ST trajectory classification;

- We propose a temporal gate to the GRU which can better incorporate the temporal interval information into trajectory classification. The temporal gate controls the confidence of input information when faced with irregular inputs due to variances in time intervals;

- We present extensive experiments on two real-world datasets and one synthetic dataset to evaluate the performance of our proposed method. Experimental results reveal that our approach achieves superior ST trajectory classification performance. Ablation studies are also conducted to justify the applicability of our proposed ST modifications.

- We further evaluate the effectiveness of the proposed temporal gate on the next location prediction problem. Experimental results demonstrate significant improvements compared to several compatible baseline methods.

The rest of chapter is organised as follows. In Section 4.2 our proposed Spatio-Temporal GRU model is introduced and the details of our model are discussed. Section 4.3 presents experimental results and the main findings are discussed.

4.2 Spatio-Temporal GRU

We first briefly review the original GRU model and then introduce the proposed Spatio-Temporal GRU model.

4.2.1 Traditional GRU

The GRU model [Cho et al., 2014] is one of the variants of RNN used to solve the gradient vanishing issue of long range dependencies. The computation formula of sequence $X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_n$ at step t can be formalised as follows:

$$\begin{cases} z_t = \sigma\left(\mathbf{W}_{xz} X_t + \mathbf{U}_{hz} h_{t-1} + \mathbf{b}_z\right), \\ r_t = \sigma\left(\mathbf{W}_{xr} X_t + \mathbf{U}_{hr} h_{t-1} + \mathbf{b}_r\right), \\ h_t^{'} = f\left(\mathbf{W}_{xh} X_t + r_t \odot \mathbf{U}_{hh} h_{t-1} + \mathbf{b}_h\right), \\ h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t^{'}, \end{cases}$$

where $z_t, r_t, h_t^{'}$, and h_t represent the update gate, the reset gate, the memory state and the output, respectively. $\mathbf{W}_{x*}$, $\mathbf{U}_{h*}$ and $\mathbf{b}_*$ are parameters of the GRU cell, $\odot$ denotes the element-wise product, and σ and f denote *sigmoid* and *tanh* activation functions, respectively. It has been shown that the GRU model achieves similar results to LSTM in many tasks with fewer parameters and less computational complexity [Cho et al., 2014].

4.2.2 Proposed Spatio-Temporal GRU

Spatial modelling

For ST trajectories, the shape of the trajectory and the local segments of several continuous points usually have a key influence on trajectory classification. If we directly feed one spatial point separately into the network, it will result in the spatial information being sparse and disjoint across time steps. As a solution, we propose a *segmented*

convolutional weight mechanism on all computations corresponding to the spatial information in order to model the local spatial correlations and variations. Thus, instead of using one feature vector $X_t \in \mathbb{R}^C$ as input at each step t, we feed a segment $\mathcal{X}_t \in \mathbb{R}^{L \times C}$ which is composed of L consecutive feature vectors from the trajectory.

Recently, a 1D-CNN has been proven to be capable of handling sequence modelling tasks such as machine translation [Gehring et al., 2017] and sentence classification [Kim, 2014], resulting in great advances in both computation efficiency and accuracy. The key benefit of the 1D-convolution over the recurrent operations is that the convolution operation explicitly captures the local correlations of the input sequence using filters serving as the context window, while recurrent networks can only feed in one element at a time and can model the correlations implicitly in the recurrent cell. Note that, for the task of ST trajectory classification, the objective is usually related to a general attribute of trajectories such as travel modes. Such general attributes are most likely related to the local shape and correlations of a given trajectory. Thus, directly feeding each point into a recurrent network at each time step can make it hard for the network to focus on such local correlations, whereas, such prior knowledge can be naturally incorporated via a 1D-CNN. In addition, traditional recurrent networks can only process input data sequentially in one dimension and will miss two dimensional correlations, i.e. from left to right or from right to left, but convolutions can capture the two dimensional spatial information via a context window around the input, thus they are better suited to ST trajectory mining.

Based on this observation, we use a 1D-convolution to replace all the linear transformation operations. Note that, the input of a recurrent cell is a segment consisting of several contiguous points rather than a single point, and the 1D-convolution is adopted to capture the local correlations for each time step computation. The computational flow of our Spatio-Temporal GRU modified with the segmented convolutional weight mechanism is formally described as follows:

$$
\begin{cases}
\mathcal{Z}_T = \sigma\left(\mathbf{W}_{xz} \otimes [\mathcal{X}_T,\ \mathcal{H}_{T-1}]\right), \\
\mathcal{R}_T = \sigma\left(\mathbf{W}_{xr} \otimes [\mathcal{X}_T,\ \mathcal{H}_{T-1}]\right), \\
\mathcal{H}_T^{'} = f\left(\mathbf{W}_{xh} \otimes \mathcal{X}_T + \mathcal{R}_T \odot (\mathbf{U}_{hh} \otimes \mathcal{H}_{T-1})\right), \\
\mathcal{H}_T = \mathcal{Z}_T \odot \mathcal{H}_{T-1} + (1 - \mathcal{Z}_T) \odot \mathcal{H}_T^{'}.
\end{cases}
\tag{4.1}
$$

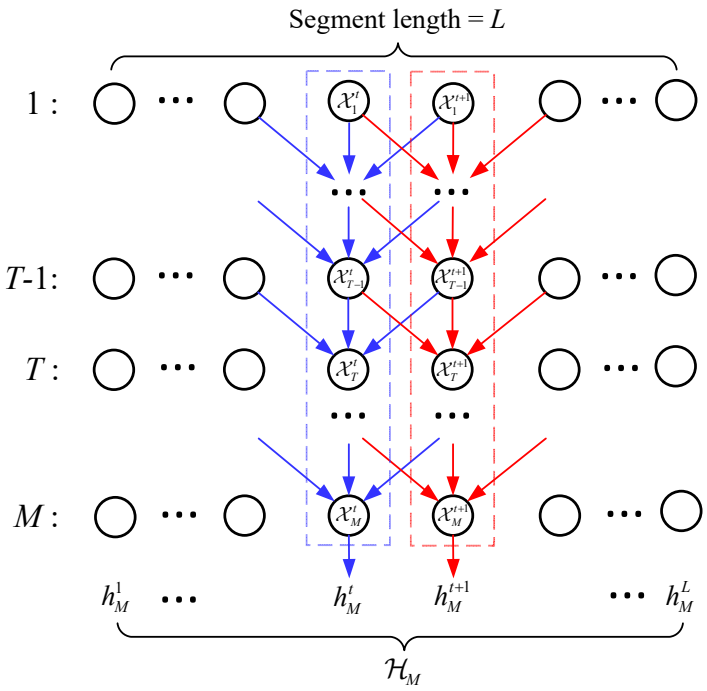


Fig. 4.2 The computation flow of the proposed segmented convolutional weight mechanism with respect to an entire sequence with $M \times L$ time steps. Each row represents a length-L segment $\mathcal{X}_T$ and the superscript (e.g. t in $\mathcal{X}_M^t$) represents the index of the element in each segment.

For clarity, we use the subscript T to indicate the index of the segment and t as the index of the basic step from the original sequence. Here, $\otimes$ denotes the 1D-convolution operation. Note that, the input information $\mathcal{X}_t$ now includes several steps of the sequence, and the linear transformation becomes the 1D-convolution which preserves the step dimension, i.e. L. This results in all gates ($\mathcal{Z}_t$, $\mathcal{R}_t$) becoming $L \times H$ matrices recording L basic steps. The states $\mathcal{H}_t$ and $\mathcal{H}_t^{'}$ also become $L \times H$ matrices to be compatible with the gate computation.

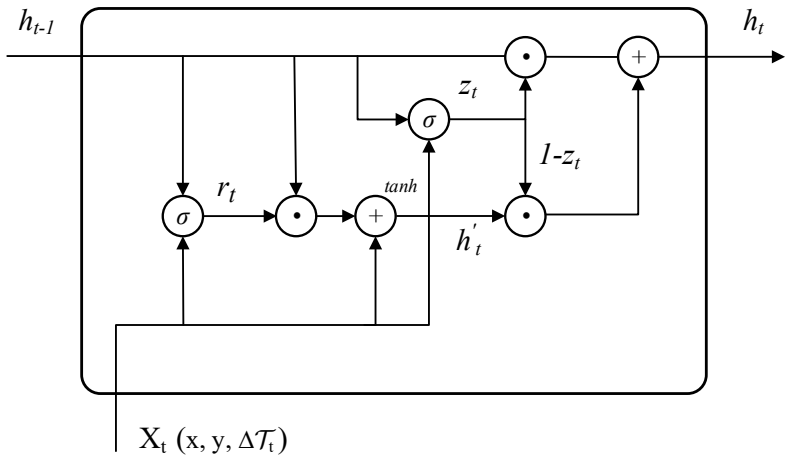
The segmented convolutional weight mechanism makes the gate and state computation of our Spatio-Temporal GRU become locally sensitive. Thus, for each computation, information before and after the current basic step t can all be inferred during the computation. In contrast to traditional recurrent networks, where there is only one element as input and they tend to be sparse in information, the segmented convolutional weight mechanism results in rich and useful information in each GRU step, as the input is composed of several points. This maximises the sequential modelling power of our Spatio-Temporal GRU. Although Reed et al. [2016] attempts to extract local sensitive

features via a 1D-CNN, this is for images and the temporal interval information is ignored. Our proposed segmented convolutional weight mechanism is different and more robust to outliers.

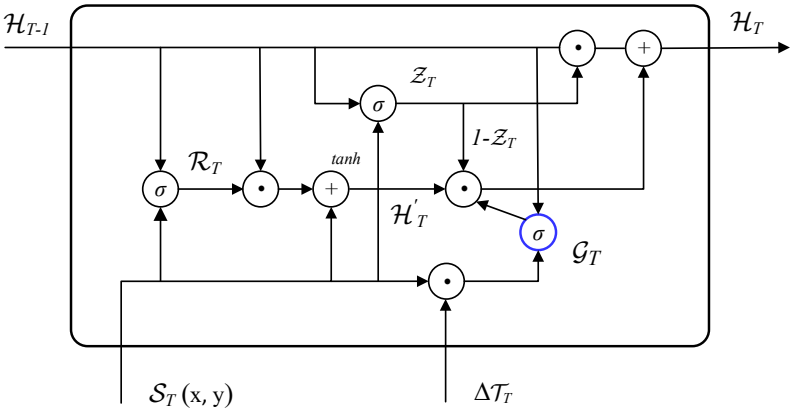
Suppose a sequence can be divided into M segments with L basic time steps. The computational flow is illustrated in Figure 4.2 with arrows indicating the dependencies introduced by the 1D-convolution with a length-3 kernel. We construct L sub-trajectories by sub-sampling the original trajectory with an interval L, i.e. $\{\mathcal{X}_1^1 \rightarrow \mathcal{X}_2^1 \rightarrow \mathcal{X}_T^1\}$, $\{\mathcal{X}_1^2 \rightarrow \mathcal{X}_2^2 \rightarrow \mathcal{X}_T^2\}, \cdots, \{\mathcal{X}_1^L \rightarrow \mathcal{X}_2^L \rightarrow \mathcal{X}_T^L\}$. The superscript indexes the element in each segment, whilst the subscript indexes each segment. Then Eq. (4.1) can be regarded as the computation of L sub-trajectories by referring context information in each time step. For instance, the blue dashed box in Figure 4.2 represents the computation of the t-th sub-trajectory Tr^t by referring context input features for each time step and generates the output h_M^t with respect to Tr^t. The red dashed box represents the sub-trajectory Tr^{t+1} output and the state h_M^{t+1}. The state $\mathcal{H}_M$ is seen as column-wise stacking the states of all L sub-trajectories, i.e. $\mathcal{H}_M = \left[h_M^1, h_M^2, \cdots, h_M^L\right]$. The final prediction can be produced by performing a global pooling process on these L sub-states. Thus, encoding the whole trajectory can be regarded as encoding L sub-trajectories. Even though some outliers lie in certain sub-trajectories, the final result will be averaged by L predictions as a whole which makes the model more robust to outliers as well as preserving the power to capture the local correlations and variations. We will conduct experiments to demonstrate the superiority of our segmented convolutional weight mechanism compared to the feature convolution strategy.

Temporal modelling

As studied in [Bermingham and Lee, 2018, Shi et al., 2015], the interval information (the interval between two consecutive points in a trajectory) plays an important role as it contains the temporal information. In this chapter, we regard the temporal interval information of step t as the relative time interval between the time stamp at step t and step $t-1$, i.e. $\Delta\tau_t = \tau_t - \tau_{t-1}$. A straightforward way to incorporate the temporal interval information is to directly treat the spatial and the temporal interval information as a whole feature vector, i.e. $X_t = (x_t, y_t, \Delta\tau_t)$, and let the weights W_{x*} automatically learn the correspondence between the spatial and temporal features. Such a strategy is



(a) Traditional GRU



(b) Proposed Spatio-Temporal GRU

Fig. 4.3 The computational graph of one step in a traditional GRU and a Spatio-Temporal GRU.

simple but not optimal as the characteristics of spatial and temporal interval information are totally different (and also measurement units are different) and, therefore, it is inappropriate to use a linear transformation to combine the spatial and the temporal interval information. Inspired by the time gate proposed in the Time-LSTM model [Zhu et al., 2017], we extend the GRU model with a temporal gate $\mathcal{G}_T$ with a segmented

convolutional weight mechanism. The computation formulas are:

$$\begin{cases} \mathcal{Z}_T = \sigma\left(\mathbf{W}_{xz} \otimes [\mathcal{S}_T,\ \mathcal{H}_{T-1}]\right), \\ \mathcal{R}_T = \sigma\left(\mathbf{W}_{xr} \otimes [\mathcal{S}_T,\ \mathcal{H}_{T-1}]\right), \\ \mathcal{G}_T = \sigma\left(\mathbf{W}_{xg} \otimes [\mathcal{S}_T,\ \mathcal{T}_T,\ \mathcal{H}_{T-1}]\right), \\ \mathcal{H}_T^{'} = f\left(\mathbf{W}_{xh} \otimes \mathcal{S}_T + \mathcal{R}_T \odot (\mathbf{U}_{hh} \otimes \mathcal{H}_{T-1})\right) \odot \mathcal{G}_T, \\ \mathcal{H}_T = \mathcal{Z}_T \odot \mathcal{H}_{T-1} + (1 - \mathcal{Z}_T) \odot \mathcal{H}_T^{'}. \end{cases}$$

The temporal gate $\mathcal{G}_T$ is decided by the current input spatial features, intervals as well as the historical states $\mathcal{H}_t$, and it is plugged into the input state $\mathcal{H}_T'$. The idea is that we want the temporal gate to control the confidence on the input state of step T, i.e. $\mathcal{H}_T'$ by considering the temporal interval information corresponding to the current position and previous position (which is recorded in $\mathcal{H}_{T-1}$). Take the travel mode classification task as an example. Intuitively, we know the moving speed of the object determines the travel mode. However, if the object stops at a crossroad or the interval suddenly becomes very large, both cases will result in an inaccurate speed estimation. In the first case, the object is actually not moving which will confuse the model as it should be regarded as noise and be filtered out. Whereas, the second case suffers from a large variance in speed estimation which should lower its information confidence for prediction. If the temporal gate is introduced, the input information for the current step will be further controlled by the temporal gate, which may be set to a low value to filter out the inputs that are likely to confuse the decision. We will conduct an ablation study and visualisation to justify the applicability of the proposed temporal gate.

4.2.3 Spatio-Temporal GRU Model for Trajectory Classification

Figure 4.3 illustrates computational graphs for one step in the traditional GRU and the Spatio-Temporal GRU. In this section, we present the details of adopting the Spatio-Temporal GRU model for the ST trajectory classification problem. Given a time interval pre-processed trajectory $Tr = \{(x_1, y_1, \Delta\tau_1), (x_2, y_2, \Delta\tau_2), \cdots, (x_N, y_N, \Delta\tau_N)\}$ containing N points where $\Delta\tau_t = \tau_t - \tau_{t-1}$ for $1 \leq t \leq N$ and $\tau_0 = 0$, we first break them into $M = \lfloor N/L \rfloor$ length-L segments to be compatible as input to our ST-GRU model. We denote $S_T^{'} \in \mathbb{R}^{L\times 2}$ as the spatial coordinates of T-th segment and $\mathcal{T}_T \in \mathbb{R}^L$ to be the corresponding interval. Directly regarding the spatial coordinate $S_T^{'}$ as a feature

may not be an optimal solution as latitude and longitude will then be regarded as independent features. This is not desirable as modelling the correlation between latitude and longitude must be handled by the ST-GRU's recurrent cell. The literature reveals several attempts to adopt embedding techniques into trajectory modelling [Gao et al., 2017, Wu et al., 2017]. However, trajectories are modelled by either a series of categorical POI or road segments which are similar to textual data. As a solution, we perform an additional feature transformation layer on the spatial coordinates to increase the feature dimension before feeding them into our ST-GRU model which can be regarded as a *soft-embedding* of the raw position. Formally, the spatial feature S_T can be computed by $S_T = \text{ReLU}(S_T^{'}\mathbf{W}_s + \mathbf{b}_s) \in \mathbb{R}^{L\times C}$ where $\mathbf{W}_s \in \mathbb{R}^{2\times C}$ and the bias $\mathbf{b}_s \in \mathbb{R}^C$, where C denotes the input spatial feature dimension of ST-GRU.

For the ST-GRU cell, we stack two ST-GRU cells to enlarge the model's capacity. Note that the temporal interval information $\mathcal{T}_T$ will be individually fed into two ST-GRU layers and the states produced by the first layer will be the spatial feature of the second layer. After the whole sequence is traversed, ST-GRU outputs the hidden states of the last segment, i.e. $\mathcal{H}_M \in \mathbb{R}^{L\times H}$ where H denotes the dimension of the hidden states. As introduced in Section 4.2.2, $\mathcal{H}_M$ is composed of the L predictions $\left(h_M^1, h_M^2, \cdots, h_M^L\right)$ generated by L sub-trajectories. We adopt the average global pooling on these L states to aggregate the L states to $h_M \in \mathbb{R}^H$. One fully connected layer with ReLU activation is adopted that follows with a linear Softmax classifier.

4.2.4 Spatio-Temporal GRU Model for Next Location Prediction

To evaluate the effectiveness of the proposed time gating mechanism, we also conduct experiments on the next POI location prediction. In this section, we present the details of adopting the ST-GRU model for next location prediction. Given a time interval preprocessed sequence S of POIs, $S = \{(p_1, \Delta\tau_1), (p_2, \Delta\tau_2), \cdots, (p_N, \Delta\tau_N)\}$ containing N POIs where $\Delta\tau_t = \tau_t - \tau_{t-1}$ for $1 \leq t \leq N$ and $\tau_0 = 0$. Unlike in travel mode classification, spatial information is not considered, instead, the embedding [Mikolov et al., 2013] of the POI takes its place to feed into the ST-GRU cell. Irregular time intervals will be fed into the model in the same manner as discussed in the previous section. However, irregular time intervals here are for the purpose of modelling the long-term and short-term interests of the user's visits to POI locations , which has not been addressed previously.

In regard to the network architecture, we adopt similar settings to those used for travel mode classification. In addition to the spatial feature $S_T^{'}$, we also concatenate embeddings of the user's identification, the date, the day of week, and the POI category.

4.2.5 Implementation and Hyperparameters

In our implementation, the soft-embedding dimension C is set to 16, and the segment length L is set to 10. In our ST-GRU model, all 1D-convolution operators have the kernel size set to 3, and the output channel set to 32 which is the same as in the hidden state dimension H. We train the model by optimising the cross-entropy loss function on the labels with a learning rate of $1e^{-2}$. The model is optimised using the Adam optimiser [Cho et al., 2014] for the first 30 epochs and we select the parameters with the highest accuracy (the parameters are saved at every epoch). The selected parameters are further optimised using the Stochastic Gradient Descent (SGD) optimiser for 50 epochs, with the learning rate set to $1e^{-4}$.

Algorithm 1 depicts the details of our model training. It consists of two modules: basic training with Adam and fine tuning with SGD. We implement our model using Tensorflow 2.0, a well-known deep learning library developed by Google. Our model is trained and evaluated on a server with an Nvidia V100 GPU and an Intel^{R} Xeon^{TM} Gold 5118 CPU @ 2.30GHz (24 cores).

4.3 Experiments – Trajectory Classification

4.3.1 Datasets for Experiments

We evaluate our model using three ST trajectory datasets: the Geolife dataset, the Shanghai taxi dataset, and a synthetically generated dataset. We pre-process all the datasets to split them in a ratio of 7:1:2 to generate corresponding training, validation and test datasets.

Geolife dataset This is a ST trajectory dataset widely used in the data mining community which was collected by Microsoft Asia in 2011 [Zheng et al., 2011]. It has been used for a variety of types of research such as human mobility prediction and travel mode

Algorithm 1: Spatio-Temporal GRU training

Input: A set $\mathcal{T}$ of trajectories, and a set $\mathcal{L}$ of labels;
Output: A trained model M;

Initialize the parameters θ;
// Train with Adam
for $i \in 1, 2, 3...$EPOCH **do**
  while batch $< N_{\text{batches}}$ **do**
    Select a random batch $(\mathcal{T}_{\text{batch}}, \mathcal{L}_{\text{batch}})$;
    Feedforward the batch;
    Get gradient and update weights θ with Adam;
Output weights θ of best performance;

// Fine tune with SGD
Load weights θ;
for $i \in 1, 2, 3...$EPOCH_C **do**
  while batch $< N_{\text{batches}}$ **do**
    Select a random batch $(\mathcal{T}_{\text{batch}}, \mathcal{L}_{\text{batch}})$;
    Feedforward the batch;
    Get gradient and update weights θ with SGD;
Output weights θ of best performance;

classification. This dataset contains 17,621 trajectories of which about 8,000 trajectories contain travel mode labels (of four types: bike, walk, car and bus). The travel mode classification task is run against this dataset.

Shanghai Taxi dataset This dataset comprises of taxi GPS traces in Shanghai city from 01 April to 17 April 17 2015 [Deng and Ji, 2011]. We segment the traces according to the labels of occupied (when the taxi is occupied by a passenger, resulting in about 314 kB of data) and available (taxis without passengers, resulting in about 198 kB of data). We conduct comparative experiments on a binary classification task, that is to correctly classify trips into two categories: having or not having passengers, for this dataset.

Synthetic dataset To further evaluate our model, we generate a set of synthetic trajectories with varying speeds, acceleration, and sampling rates. We generate trajectories with four travel modes (the same as in the Geolife dataset). For car mode, we draw a speed from 30 km/h to 60 km/h with acceleration of 3 m/s^2. For bus mode, we set the speed range to be the same as car mode, except acceleration is set to 1 m/s^2. For bike

mode, the speed range is 10 km/h $\sim$ 15 km/h, and for walking mode the range is 2 km/h $\sim$ 5 km/s. The sampling time interval is randomly drawn from $\{1s, 2s, 5s\}$. We also randomly drop points to simulate GPS uncertainties. We generated 5,000 trajectories for each travel mode, with each trajectory containing 1,000 points (before randomly dropping points).

4.3.2 Overall Evaluation

First of all, we perform an evaluation on the ST trajectory classification task on three datasets. We implement the following baselines for comparative analysis.

Baselines

SVM & Random forest: As proposed in [Zheng et al., 2010], nine features are extracted from the raw trajectory data including speed, velocity, and direction. The extracted features are fed into SVM [Cortes and Vapnik, 1995] and random forest Breiman [2001] classifiers as suggested.

RNN: LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Cho et al., 2014] are two common variants of RNNs. As mentioned in Section 4.2.2, we extend the traditional RNN by adopting an 1D-CNN directly onto the whole sequence to extract local sensitive features [Reed et al., 2016]. This approach is denoted by a prefix *FConv*. We compare all RNN models by feeding in the spatial information (x, y) only and feeding in the ST information $(x, y, \Delta\tau)$.

Conv-LSTM: A convolutional LSTM is proposed in [Shi et al., 2015] for spatial temporal forecasting; Conv-LSTM replaces the matrix transition with a 2D convolution. We implement this by replacing the 2D-CNN with a 1D-CNN, then raw trajectories are fed into the model as we do for the RNN models.

Time-LSTM: A Time-LSTM is proposed in [Zhu et al., 2017] to solve the irregularity of time interval problem for recommendation tasks. We implement the first version of Time-LSTM in their series for comparison.

Table 4.1 Experimental results in terms of classification accuracy (%).

	Geolife	**Shanghai**	**Synthetic**
SVM $(x, y, \Delta\tau)$	86.11%	87.78%	86.90%
Random Forest $(x, y, \Delta\tau)$	86.89%	88.03%	87.28%
CNN $(x, y, \Delta\tau)$	87.08%	83.88%	81.47%
LSTM (x, y)	71.86%	85.53%	69.45%
LSTM $(x, y, \Delta\tau)$	88.39%	90.94%	88.50%
FConv-LSTM $(x, y, \Delta\tau)$	88.44%	90.91%	88.61%
GRU (x, y)	72.04%	85.78%	70.20%
GRU $(x, y, \Delta\tau)$	89.76%	92.03%	91.49%
FConv-GRU $(x, y, \Delta\tau)$	89.86%	91.98%	91.55%
Conv-LSTM $(x, y, \Delta\tau)$	89.85%	91.52%	91.25%
Time-LSTM $(x, y, \Delta\tau)$	83.92%	90.88%	88.78%
ST-GRU $(x, y, \Delta\tau)$	**91.25%**	**93.89%**	**93.21%**

Performance Evaluation

We evaluate our Spatio-Temporal GRU against the baselines discussed above including shallow machine learning based approaches and deep RNN based methods. Table 4.1 displays a summary of our experimental results. It clearly demonstrates that our Spatio-Temporal GRU outperforms all baselines for each of the three ST trajectory datasets used in the evaluation. Note that, we use an one-tail paired t-test as a test of significance in the study, since it is commonly used in data mining and provides more power to detect an effect [McCarroll, 2017].

First, as shown in Table 4.1, RNN based approaches exhibit superior performance over machine learning based approaches, i.e. SVM and Random forest, as well as CNN. This indicates that RNNs are capable of modelling ST sequential trajectory data. Random forest $(x, y, \Delta\tau)$ seems to be the best performer among the machine learning approaches we evaluated. RNN based approaches LSTM $(x, y, \Delta\tau)$ and GRU $(x, y, \Delta\tau)$ outperformed Random forest $(x, y, \Delta\tau)$ for all three datasets. The test of significance in terms of accuracy shows that LSTM $(x, y, \Delta\tau)$ significantly improves the best machine learning based approach under study, Random forest $(x, y, \Delta\tau)$ (p-value = 0.035) with 95% confidence, whilst GRU $(x, y, \Delta\tau)$ significantly improves Random forest $(x, y, \Delta\tau)$ (p-value = 0.008) with 99% confidence.

Second, RNN based approaches with ST information exhibit higher accuracies than those with spatial information, while ignoring the temporal interval information. Comparisons of LSTM/GRU (x, y) and LSTM/GRU $(x, y, \Delta\tau)$ prove that the temporal interval information $\Delta\tau$ plays a crucial role in ST trajectory classification. Disregarding $\Delta\tau$ significantly decreases the overall performance. For example, , without the temporal interval information, classification accuracies for LSTM drop 16.53% on the Geolife dataset, 5.41% on Shanghai dataset, and 19.05% on our synthetic dataset. The test of significance shows that LSTM $(x, y, \Delta\tau)$ significantly improves LSTM (x, y) (p-value = 0.041), and GRU $(x, y, \Delta\tau)$ significantly improves GRU (x, y) (p-value = 0.040) with 95% confidence.

Third, performing 1-D convolutions on the sequence before loading it into the network (i.e. the FConv version) appears to have a negligible effect. On one hand, the largest improvement of 0.11% is observed for FConv-LSTM $(x, y, \Delta\tau)$ over LSTM $(x, y, \Delta\tau)$ for our synthetic dataset, on the other hand, a small decrease is observed for the Shanghai dataset. This justifies our claim that the convolution operation should be computed in GRU cells for better modelling of the spatial correlations which separate the spatial and temporal computations.

Fourth, Time-LSTM $(x, y, \Delta\tau)$ has a time gating mechanism for modelling the temporal interval information. However, note that their custom time gating cell is specifically designed to model long-term and short-term interests to recommend products that a user might be interested in, which is not suitable for trajectory classification. Time-LSTM achieves only 83.92% on the Geolife Dataset which is inferior to LSTM $(x, y, \Delta\tau)$, and notably Time-LSTM $(x, y, \Delta\tau)$ is inferior to GRU $(x, y, \Delta\tau)$ for all three datasets. Conv-LSTM $(x, y, \Delta\tau)$ performs better than Time-LSTM $(x, y, \Delta\tau)$. The test of significance shows that Conv-LSTM $(x, y, \Delta\tau)$ significantly improves Time-LSTM $(x, y, \Delta\tau)$ (p-value = 0.095) with 90% confidence. This implies that the spatial information plays a more important role than the temporal interval information in trajectory classification.

Finally, ST-GRU performs the best for all three datasets. It performs 91.25% for Geolife dataset, 93.89% for Shanghai dataset, and 93.21% for our synthetic dataset. The test of significance shows that ST-GRU $(x, y, \Delta\tau)$ significantly improves the best performer among all the others, Conv-LSTM $(x, y, \Delta\tau)$, (p-value = 0.01) with 99% confidence. This clearly demonstrates the outstanding performance of our proposed model over all other approaches under study.

Table 4.2 Results for ablation study.

	Geolife	Shanghai	Synthetic
GRU $(x, y, \Delta\tau)$	89.76%	92.03%	91.49%
Spatial-GRU $(x, y, \Delta\tau)$	90.03%	92.32%	91.87%
Temporal-GRU $(x, y, \Delta\tau)$	90.59%	92.75%	92.34%
ST-GRU $(x, y, \Delta\tau)$	**91.25%**	**93.89%**	**93.21%**

4.3.3 Ablation Study

To validate the proposed spatial modelling and temporal modelling strategies, we conduct an ablation study on these components. We include two new competitors: Spatial-GRU and Temporal-GRU. Spatial-GRU refers to a modification of GRU to use a segmented convolutional weight mechanism which captures the spatial correlations, whilst Temporal-GRU refers to a modification of GRU implemented with the addition of a temporal gate but without the segmented convolutional weight mechanism. The results listed in Table 4.2 show a boost in performance when using Spatial-GRU or Temporal-GRU compared to the original GRU. This indicates that either the proposed spatial modelling or the temporal modelling modification proposed in this paper does improve performance. Moreover, by combining both the temporal and spatial components, ST-GRU further improvements are gained.

4.3.4 Visualisation of Temporal Gates

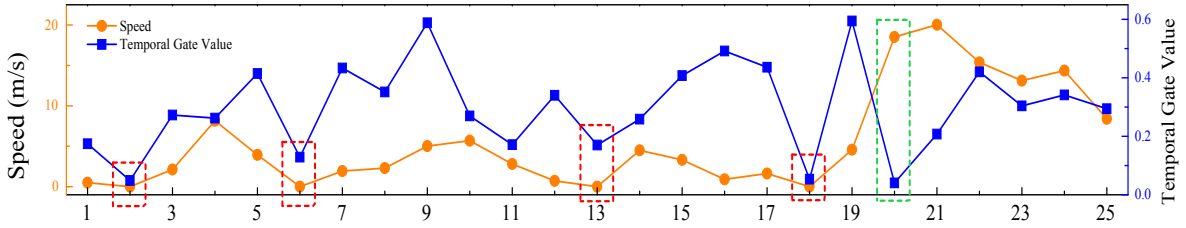


Fig. 4.4 The speed and corresponding temporal gate value for 25 time steps in a trajectory. Note that the temporal gate is a vector and we visualize it by its average value. The example trajectory is from the Geolife dataset with the traffic mode 'car'.

To further understand how the temporal gate works, we design a way to visualise the temporal gate G_T. For the travel mode classification task, speed is an important

factor, therefore, we visualise the speed across all time steps. The temporal gate value is decided by the current spatial position, the interval and the previous states containing the previous position. Supposing the speed information could be related to the gate value, we visualise this in Figure 4.4. We can observe that for the time steps, annotated by red dashed boxes, the speed of the object becomes zero which may complicate the prediction (if the prediction is based on the average speed). From the temporal gate value, we can infer that the gate has learned to restrict the input information of these steps. Interestingly, at step 20 (green box) the speed suddenly increases to a value which seems abnormal compared to the previous speed. The gate detects this phenomenon and tries to ignore this information. But if, after several steps, the speed remains at a relatively high value, the model starts to trust the information and gradually increases the gate value.

4.3.5 Importance of L

As stated earlier, trajectory data is fed into ST-GRU via a set of length L segments. Performance will vary with different length L parameter settings. On one hand, the shorter the value of L, the more difficult it is for the 1D-CNN to catch the spatial correlations. In an extreme condition $L = 1$, ST-GRU becomes Temporal-GRU. On the other hand, the larger the value of L, the shorter the segment sequence will be which reduces the performance of the recurrent cell. We conduct experiments using four parameter settings, $L = \{1, 5, 10, 15\}$ to show the sensitivity of L in regard to performance. Figure 4.5 supports the above-mentioned claim and shows that $L = 10$ achieves the best classification accuracy of 91.25% on the Geolife dataset.

4.4 Experiments – Next-Location Prediction

4.4.1 Data Preparation

To further evaluate the effectiveness of our proposed time gate mechanism, we conduct next-location prediction experiments on two real-world datasets. Foursquare [1] allows users to check-in their location POIs, and NYC and TKY are two openly available large

[1] https://developer.foursquare.com/docs

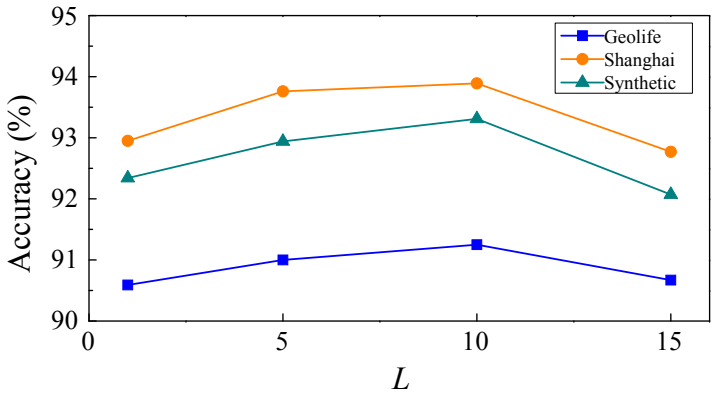


Fig. 4.5 Performance comparison of varying L values.

POI datasets collected by Yang et al. [2015]. NYC contains 227,997 check-in locations of 38,333 POIs in New York City, whereas, TKY contains 572,210 check-in locations of 61,858 POIs in Tokyo City. For each user, we segment each trajectory to the length of 10 POIs, then 80% are randomly selected for the training set, 5% for the validation set for parameter tuning, and 15% for the testing set to evaluate the overall performance.

4.4.2 Overall Evaluation

Table 4.3 Experimental results of next-location prediction on NYC and TKY datasets.

	NYC			TKY		
	Acc@1	Acc@10	Acc@20	Acc@1	Acc@10	Acc@20
LSTM	13.78%	38.57%	43.61%	12.20%	37.63%	44.27%
GRU	14.69%	39.70%	44.68%	13.34%	37.88%	44.20%
ST-GRU	15.31%	39.78%	44.41%	13.59%	38.28%	44.55%

Evaluation Matrix and Baselines

We adopt the accuracy of the top K (Acc@K) matrix, where K is the number of the candidates we select with the highest probabilities. We use K = 1, 10, 20 and report them on both the NYC and TKY datasets. We compare the proposed ST-GRU to the original LSTM and GRU baselines.

Performance Evaluation

As shown in Table 4.3, both the original GRU and the proposed ST-GRU outperform LSTM, which further indicates the GRU family is more suitable for trajectory modelling than LSTM. Among the GRU family, the proposed ST GRU outperforms the original GRU in all matrices excluding Acc@20 on the NYC dataset. ST-GRU performs extremely well on the Acc@1 matrix on both datasets, which demonstrates the proposed time gating mechanism successfully models both long-term and short-term user check-in interests. Recall that time intervals between user check-ins are highly variable, and such interval variance affects the user's check-in interests. Modelling such long-term and short-term interests caused by the time intervals is crucial for this kind of next POI prediction task. Experiments on two real-world POI check-in datasets demonstrate the effectiveness of the proposed ST-GRU for modelling such long-term and short-term interests, resulting in a performance boost over the original GRU.

4.5 Summary

In this chapter, we studied the difficulties of modelling in-regularly sampled spatio-temporal trajectory. To fulfil the research gap identified in Section 3.4, we proposed a novel model called Spatio-Temporal GRU to effectively model the ST correlations and variations for ST trajectory classification. We propose a novel segmented convolutional weight mechanism to capture short-term local spatial variations and correlations in trajectories, and introduce an additional temporal gate to control the information flow for the temporal interval information. Experimental results that demonstrate the superior performance of our proposed method against popular deep learning approaches proposed for spatio-temporal trajectory modelling.

Chapter 5

Bridging the Gap Between Training and Inference for Spatio-temporal Forecasting

Spatio-temporal sequence forecasting is one of the fundamental tasks in spatio-temporal data mining. It facilitates many real world applications such as precipitation nowcasting, citywide crowd flow prediction and air pollution forecasting. Recently, a few Seq2Seq based approaches have been proposed, but one of the drawbacks of Seq2Seq models is that small errors can accumulate quickly along the generated sequence at the inference stage due to the different distributions of training and inference phases. That is because Seq2Seq models minimise single step errors only during training; however, the entire sequence has to be generated during the inference phase which generates a discrepancy between training and inference. In this chapter, we propose a novel curriculum learning based strategy called Temporal Progressive Growing Sampling to effectively bridge the gap between training and inference for spatio-temporal sequence forecasting, by transforming the training process from a fully-supervised approach which utilises all available previous ground-truth values to a less-supervised approach which replaces some of the ground-truth context with generated predictions.

5.1 Introduction

STSF is one of the fundamental tasks in spatio-temporal data mining [Shi and Yeung, 2018]. It facilitates many real world applications such as precipitation nowcasting [Shi et al., 2015], citywide crowd flow prediction [Chen et al., 2018, Zhang et al., 2018] and air

pollution forecasting [Yi et al., 2018]. It is used to predict future values based on a series of past observations. STSF is formally defined as:

Definition 1. Given a length-T matrix sequence $S = [\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_T]$. Each matrix $\mathcal{X}_t \in S$ consists of measurements of coordinates at time-step t. STSF is used to predict a sequence of corresponding measurements of the following k time-steps based on the past observations of S, denoted as $\hat{\mathcal{P}} = \left[\hat{\mathcal{X}}_{T+1}, \hat{\mathcal{X}}_{T+2}, \ldots, \hat{\mathcal{X}}_{T+k}\right]$.

From the recent advances in the Seq2Seq model [Sutskever et al., 2014] in sequence modelling such as Neural Machine Translation (NMT) and speech recognition, researchers have adapted Seq2Seq to model STSF as sequence modelling. In particular, both DCRNN [Li et al., 2018] and PredRNN [Wang et al., 2017] have utilised an RNN-based encoder to encode a source sequence S into a feature matrix which will then be decoded recursively conditioned on previous contexts into the target sequence $\hat{\mathcal{P}}$, with a separate RNN-based decoder. During the training phase, the previous contexts become ground-truth observations. However, during the inference phase, the previous contexts are drawn from the model itself as no ground-truth observations are available for the decoder. The cause of the discrepancy between the training and inference stages is referred to as *exposure bias* [Ranzato et al., 2016]. Small errors caused by this bias quickly accumulate to become a large error along the generated sequence at the inference stage. One intuitive solution is to unify the training and inference phases by using previously generated contexts instead of ground-truth values during training. However, this causes the model more difficulties in the converging process or, at worst, the converging process fails [Bengio et al., 2015, Venkatraman et al., 2015]. Bengio et al. [2015] introduced *Scheduled Sampling* to overcome this problem by gradually transforming between these two strategies, resulting in significant improvements, and this has been widely used in NMT systems. DCRNN and PredRNN adapt Scheduled Sampling into STSF, producing some improvements. However, we argue that simply adopting Scheduled Sampling from NMT to STSF is not ideal even though they are both designed for sequence modelling. This is due to two clear distinctions. First, an NMT system is ideal for the word level classification problem that optimises the cross entropy loss conditioned on the source sequence and previous contexts, whereas, STSF is ideal for the regression problem that optimises a regression loss such as MSE and MAE. Second, two consecutive words in NMT systems are semantically close to each other, and small errors caused by the training bias

could result in a completely different translation. However, two measurements in two consecutive time steps in STSF are geographically close and contiguous to each other, and errors made by previous steps could confuse the local trend of prediction, leading to a bigger gap in the long-term predictions.

Motivated by the above observations, we bridge the gap between training and inference for spatio-temporal forecasting by introducing a novel coarse-to-fine hierarchical sampling method called TPG. The idea is to transform the training process from a fully-supervised approach which utilises all available previous ground-truth to a less-supervised approach which replaces some of the ground-truth contexts with generated predictions. To do that, we also sample the target sequence from midway outputs from intermediate models trained with longer timescales through a carefully designed decaying strategy. By doing so, the model explores the differences between the training and inference phases as well as the intermediate model in order to correct its exposure bias. Experimental results demonstrate our model achieves superior performance as well as faster convergence time on two spatio-temporal sequence forecasting datasets. Experiments also show our proposed method is better at modelling long-term dependencies, hence our method increases the long-term prediction accuracy.

The main contributions of this chapter are summarised as follows:

- We propose a novel temporal progressive growing sampling method to effectively bridge the gap between training and inference for spatio-temporal forecasting. The model is initially trained with a longer time gap, which gradually transforms into a shorter time gap.

- We carefully design a decay strategy of sampling that takes the current index of the sequence into account, which helps the convergence of the training and yields better performance.

- We conduct extensive experiments on two real-world spatio-temporal datasets to evaluate the performance of our proposed method. Experimental results reveal that our approach achieves superior performance on sequence forecasting tasks, and experimental results also show our approach achieves better long-term prediction accuracy.

The rest of this chapter is organised as follows. In Section 5.2 our proposed TPG model is introduced and the details of the model are discussed. Section 5.3 and Section 5.4 present the experimental results of weather prediction, against a Moving MNIST dataset [Srivastava et al., 2015].

5.2 Temporal Progressive Growing Sampling

5.2.1 Seq2Seq and Scheduled Sampling

Seq2Seq [Sutskever et al., 2014] was first introduced to solve complex sequential problems that traditional RNN approaches cannot model, such as diverse input and output lengths. Seq2Seq is also known as an encoder-decoder where the encoder encodes the original sequence into a feature vector, then the decoder outputs a target sequence based on the feature vector and previous contexts. Typically, both encoder and decoder are RNNs, and Seq2Seq has been used for sequence modelling tasks like NMT , speech recognition and recently STSF tasks [Li et al., 2018, Salinas et al., 2017, Yu et al., 2017].

One of the main drawbacks of Seq2Seq models is that small errors can accumulate quickly along the generated sequence at the inference stage due to the different distributions of training and inference phases. That is because RNN models minimise single step errors only during training when all previous ground-truth contexts are available. However, the entire sequence has to be generated during the inference phase which causes a discrepancy between training and inference. To close the gap between training and inference for NMT, Bengio et al. [2009] proposed a sampling strategy called Scheduled Sampling which is explained below.

$$
\begin{aligned}
&\forall\ k, 1 \leq k \leq K, \\
&\hat{\mathcal{X}}_{t+k} \sim \mathcal{M}\left(\text{Encoder}(S), \tilde{\mathcal{X}}_{t+1:t+k}; \theta\right), \\
&\tau_{t+k+1} \sim \text{Bernoulli}\left(1, \epsilon_i\right), \\
&\tilde{\mathcal{X}}_{t+k+1} = \left(1 - \tau_{t+k+1}\right) \hat{\mathcal{X}}_{t+k} + \tau_{t+k+1} \mathcal{X}_{t+k}.
\end{aligned}
\tag{5.1}
$$

Here, $\mathcal{M}\left(\text{Encoder}(S), \tilde{\mathcal{X}}_{t+1:t+k}; \theta\right)$ denotes one step prediction based on the encoder and the previous inputs $\tilde{\mathcal{X}}_{t+1:t+k}$. τ_{t+k+1} is a random variable generated by a coin flip following the Bernoulli distribution, where ϵ_i is the probability of $\tilde{\mathcal{X}}_{t+k+1}$ sampling

from the ground-truth $\mathcal{X}_{t+k}$, i.e. $1-\epsilon_i$ is the probability of sampling from the previous prediction $\hat{\mathcal{X}}_{t+k}$. When $\epsilon_i = 1$, $\tilde{\mathcal{X}}_{t+k+1}$ is always sampling from the ground-truth $\mathcal{X}_{t+k}$. When $\epsilon_i = 0$, $\tilde{\mathcal{X}}_{t+k+1}$ is always sampling from the previous output $\hat{\mathcal{X}}_{t+k}$. During training, the probability ϵ_i is decreased from 1 to 0.

5.2.2 Bridging the Gap with TPG Sampling

While adopting Seq2Seq for NMT to STSF brings significant benefits, it also creates some drawbacks. The discrepancy between training and inference creates a gap in both NMT and STSF systems. In NMT systems, that gap might result in a completely different translation, and in STSF systems, errors at previous steps in the STSF could generate confusion in the local trends. Therefore, adopting the Scheduled Sampling strategy as is to use for STSF is not an ideal solution. Furthermore, spatio-temporal dynamics can be modelled using different sampling rates. For instance, if we assume that we sample data every 1 hour, then we can train a model to estimate the prediction of every 2 hours by feeding the odd index sequence and the even index sequence separately. The current Scheduled Sampling method does not consider this unique characteristic of STSF.

Motivated by this, we propose a TPG sampling strategy to close the gap between training and inference. First, we sub-sample the sequence into two sub-sequences by separating the odd and even indexes (see Fig. 5.1, green represents the odd index inputs and orange represents the even). The idea is to start training a simple model $\mathcal{M}_1(\text{Encoder}(S_{::2}), \theta)$ that takes as input the odd or even index sequence denoted as $S_{::2}$. The benefit of this approach is that the sequence length is cut to half to the original length, which is much easier for both the encoder and decoder to learn. Moreover, if model $\mathcal{M}_1$ is trained with Scheduled Sampling as in Equation 5.1, then the decoder input of model $\mathcal{M}_1$ is used as a sampling source for model $\mathcal{M}_2$ during the transition phase. This brings another advantage in that the model is not only able to explore the differences between training and inference but also the intermediate model trains with data from a longer time period. The detailed transition process is described as follows:

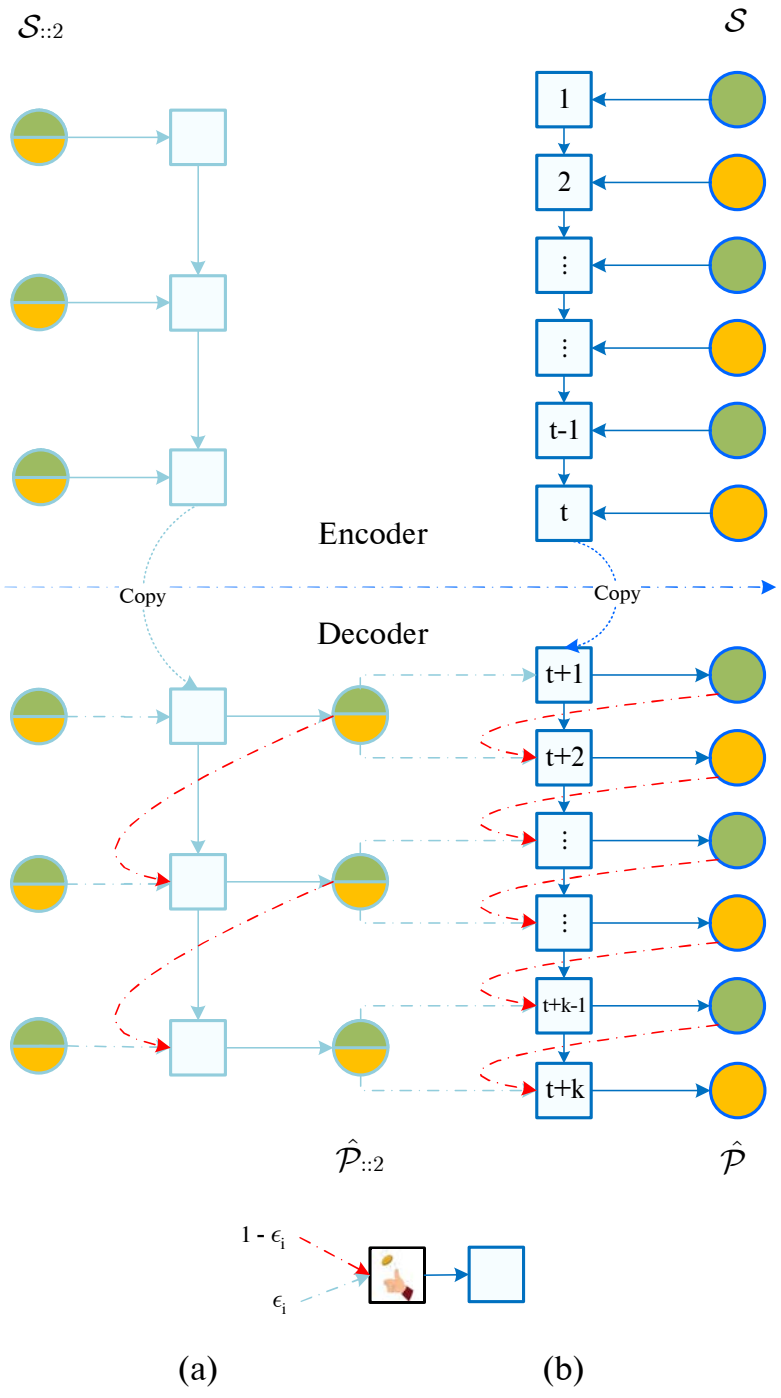


Fig. 5.1 Illustration of our proposed TPG sampling. Green circles represent odd index inputs $\mathcal{X}_{\text{odd}}$ and outputs $\hat{\mathcal{X}}_{\text{odd}}$, orange circles represent even index inputs $\mathcal{X}_{\text{even}}$ and outputs $\hat{\mathcal{X}}_{\text{even}}$. At the start of training model $\mathcal{M}_1$ (a) initially, each sequence is fed with a sub-sequence of green and orange. During the transition to model $\mathcal{M}_2$ (b), the decoder input $\tilde{\mathcal{X}}_{t+\frac{k}{2}::2}$ of model $\mathcal{M}_1$ is used as a source for sampling.

$$
\begin{aligned}
&\forall\ k, 1 \leq k \leq K, \\
&\hat{\mathcal{X}}_{t+\frac{k}{2}::2} \sim \mathcal{M}_1 \left(\text{Encoder}_1(S_{::2}), \tilde{\mathcal{X}}_{t+1:t+\frac{k}{2}::2}; \theta_1\right), \\
&\hat{\mathcal{X}}_{t+k} \sim \mathcal{M}_2 \left(\text{Encoder}_2(S), \tilde{\mathcal{X}}_{t+1:t+k}; \theta_2\right), \\
&\tau_{t+k+1} \sim \text{Bernoulli}\,(1, \epsilon_i), \\
&\tilde{\mathcal{X}}_{t+k+1} = (1 - \tau_{t+k+1})\,\hat{\mathcal{X}}_{t+k} + \tau_{t+k+1}\tilde{\mathcal{X}}_{t+\frac{k}{2}::2}.
\end{aligned}
\tag{5.2}
$$

Here, $\mathcal{X}_{t+\frac{k}{2}::2}$ denotes decoder inputs from model $\mathcal{M}_1$ of the corresponding index of model $\mathcal{M}_2$. Similar to Scheduled Sampling, ϵ_i is the probability that follows the Bernoulli distribution that controls whether $\tilde{\mathcal{X}}_{t+k+1}$ samples are from the previous output $\hat{\mathcal{X}}_{t+k}$ or $\tilde{\mathcal{X}}_{t+\frac{k}{2}::2}$ of $\mathcal{M}_1$. When $\epsilon_i = 1$, $\tilde{\mathcal{X}}_{t+k+1} = \hat{\mathcal{X}}_{t+k}$. During the transition from $\mathcal{M}_1$ to $\mathcal{M}_2$, we decrease ϵ_i from 1 to 0. When $\epsilon_i = 0$, then model $\mathcal{M}_2$ gets solely trained conditioned on its previous output.

Although our proposed Temporal Progressive Growing (TPG) Sampling shares some similarities with Scheduled Sampling, there are two key differences. First, TPG closes the gap between training and inference not only from exploring the bias but also corrects the errors caused by the bias by learning the intermediate model over a longer time scale. Second, we careful design a decaying strategy to work with TPG which is introduced in the next section, and our experiments show significant improvements.

5.2.3 Decay Strategy

Scheduled Sampling decreases ϵ_i during the transition period by the inverse sigmoid function as follows:

$$
\epsilon_i = \frac{\lambda}{\lambda + \exp(i/\lambda)}, \tag{5.3}
$$

where i is the current global batch number, λ is the parameter setting to control the deceasing speed of ϵ_i and, therefore, the convergence speed. Here, the whole sequence shares the same probability to replace the ground-truth with output generated by the model itself, noted ϵ_i towards 0 the greater probability is. However, during our experiment, we observed that STSF model converges from the beginning of the sequence. Replacing ground-truth with the model generated output at the beginning of the sequence increases the convergence difficulty of the whole training process. Therefore, we

propose a decay strategy that takes the current index of the sequence into account when calculating the probability:

$$\epsilon_i^{\upsilon} = \frac{\lambda}{\lambda + \exp(i \times \log(\upsilon)/\lambda)}, \tag{5.4}$$

where υ is the current index of the sequence starting from 2. Therefore, later index input has a higher probability where ground-truth values are replaced than the earlier input at the beginning of the training process. This helps the convergence of the training phase by keeping the previous input as ground truth. However, ϵ_i^{υ} from different indexes will become smaller and eventually equal zero due to the exponential growth towards the end of the training, which is desirable because we want all ground truth to be replaced by model generated outputs when the training is complete.

5.3 Weather Forecasting Experiments

To evaluate our proposed TPG sampling strategy, we first conducted experiments based on a weather forecasting task. Weather forecasting is a typical STSF task that brings many benefits to people's everyday life, as well as to agriculture and many other areas. Experimental results show our method outperforms several baseline methods, as well as the basic Seq2Seq using Scheduled Sampling.

5.3.1 Dataset

AI Challenger weather forecasting is an online competition [1], and the goal is to predict air temperature at 2 metres (t2m), relative humidity at 2 metres (rh2m) and wind speed at 10 metres (w10m) across 10 weather stations in Beijing city. This dataset contains historic observations from 01 March 2015 to 30 October 2018. We use 01 March 2015 to 31 May 2018 for our training set, 01 June 2018 to 28 August 2018 for the test set, and 29 August 2018 to 30 October 2018 for validation purposes. Each node contains nine measurements including t2m, rh2m, w10m and 6 others, as well as 37 other predictions which are generated by Numerical Weather Prediction (NWP).

[1] https://challenger.ai/competition/wf2018

	t2m		rh2m		w10m	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
NWP	2.939	2.249	18.322	13.211	1.813	1.327
ARIMA	3.453	2.564	18.887	14.163	2.436	1.675
Seq2Seq $_{180}$ without sampling	3.149	2.393	16.230	11.712	1.437	1.032
Seq2Seq $_{90}$ + Scheduled Sampling	2.828	2.173	16.023	11.428	1.380	0.962
Seq2Seq $_{180}$ + Scheduled Sampling	3.080	2.334	**14.180**	**10.254**	1.417	1.035
TPG $_{\mathcal{M}_1}$	3.006	2.379	16.639	12.197	2.211	1.360
TPG	**2.611**	**1.984**	14.994	10.623	**1.328**	**0.914**

Table 5.1 Experimental results based on the test set from 01 June 2018 to 28 August 2018. Smaller numbers indicates smaller prediction errors.

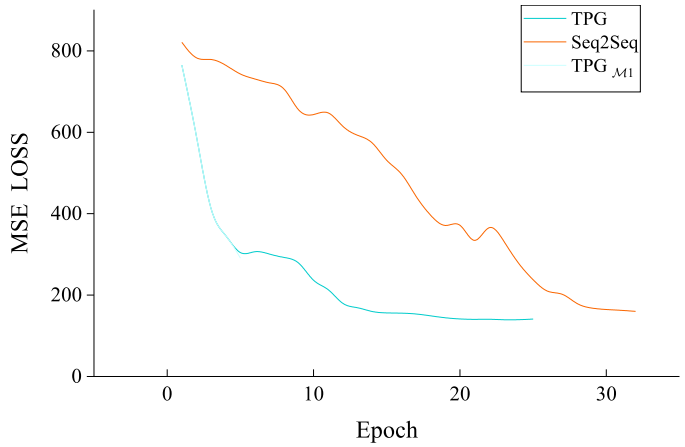


Fig. 5.2 Test set loss comparison during training.

5.3.2 Implementation

Seq2Seq

As mentioned previously, the data is from 10 weather stations, where each station records observations of nine different measurements.The challenge is to model correlations between stations as well as correlations between different measurements. For example, wind speed and air temperature have strong correlations. In fact, every measurement could have some impact on every other measurement; it may be non-obvious but highly dynamic. As stated in the previous section, our approach is based on Seq2Seq which is capable of modelling such correlations with its encoder and decoder architecture. First, we use an LSTM encoder taking an input $S \in \mathbb{R}^{T\times 10\times 9}$ which produces a feature vector

$\mathbf{h} \in \mathbb{R}^C$. Here, T is a hyper-parameter to specify the sequence length we use for feature encoding, 10 is the total number of stations, and 9 means we use all 9 measurements for feature learning. This process is called feature extraction or feature learning. During the computation of the LSTM encoder for each time step, the linear transformation operation takes into account of nine measurements from 10 stations. The overall feature of time steps will then be encoded into one vector space $\mathbf{h}$. Then feature vector $\mathbf{h}$ will be decoded by an LSTM decoder to produce an output $\hat{S} \in \mathbb{R}^{37\times10\times9}$ which is the following 37 hours prediction of the nine measurements. Note that we are predicting only t2m, rh2m and w10m. Therefore, we follow a fully-connected layer to output the final prediction $\hat{\mathcal{P}} \in \mathbb{R}^{37\times10\times3} = \hat{S}\mathbf{W}_s + \mathbf{b}_s$.

TPG

Loss Function

We divide the loss function into three parts:

$$\mathcal{L} = \text{MSE}_{\text{t2m}} + \text{MSE}_{\text{rh2m}} + \text{MSE}_{\text{w10m}}, \tag{5.5}$$

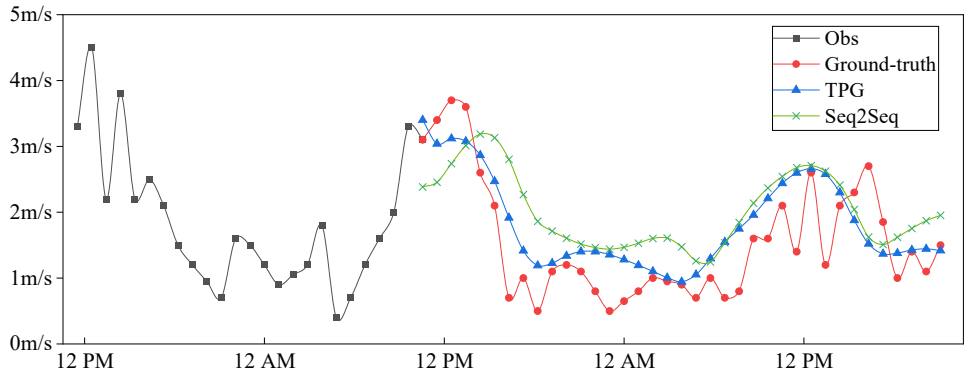
where MSE denotes *mean squared error*: $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{X}_i - \hat{\mathbf{X}}_i\right)^2$. During training, $\mathcal{L}$ is optimised jointly by BPTT [Werbos, 1990] with an Adam optimiser [Kingma and Ba, 2014].

Parameter Settings

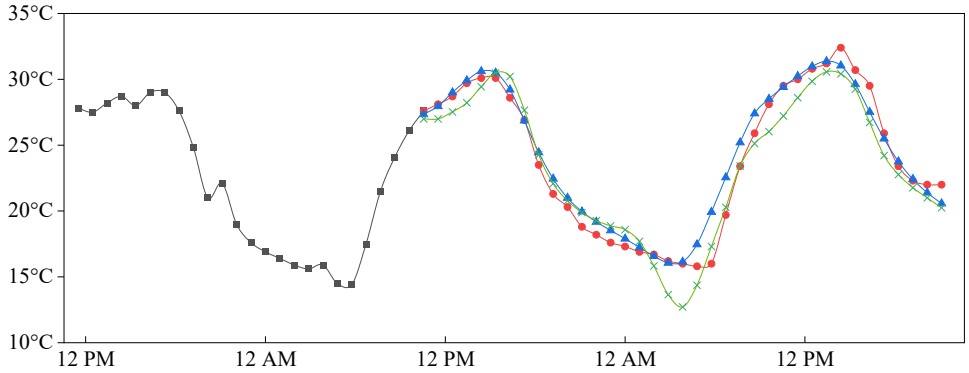
For encoder and decoder, we choose a dimension $C = 90$ for LSTM, whilst for the encoder, T is set to 96 which means we use 4 days made up of 96 hours of historic observations to predict the next 37 hours. For TPG, $\lambda = 3000$ for weather forecasting and $\lambda = 1000$ for Moving MNIST++. The learning rate for Adam is set to $1e^{-2}$.

Experiment Environment

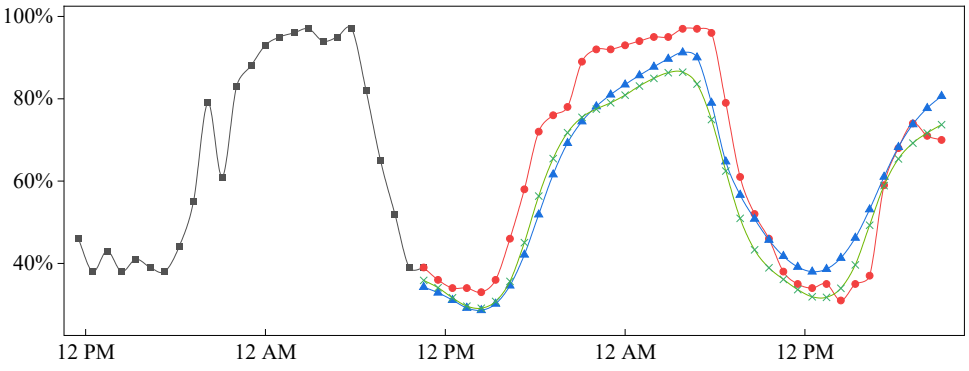
We implement our model using Tensorflow 1.12, a well known deep learning library developed by Google. Our model is trained and evaluated on a server with a Nvidia V100 GPU and an Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz (24 cores).



(a) w10m predictions.



(b) t2m predictions.



(c) rh2m predictions.

Fig. 5.3 Prediction visualisations (based on 30 August 2018) of air temperature (t2m), relative humidity (rh2m) and wind speed (w10m). TPG predictions have fewer outliers and have more accurate long term predictions.

5.3.3 Overall Evaluation

Evaluation Matrix

We report experimental results on each measurement for the test set using *Root Mean Squared Error* (RMSE) and *Mean Absolute Error* (MAE):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{X}_i - \hat{\mathbf{X}}_i\right)^2}. \tag{5.6}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\mathbf{X}_i - \hat{\mathbf{X}}_i\right|. \tag{5.7}$$

Here, n is the number of prediction time steps times the number of locations.

Baselines

NWP: Numerical Weather Prediction [Lynch, 2008] uses mathematical models of the atmosphere and oceans to predict the weather based on current weather conditions.

ARIMA: Auto-Regressive Integrated Moving Average is the most common baseline method for time series predictions [Box and Jenkins, 1990]. The model is trained individually by each station.

Seq2Seq $_{180}$ without sampling: The basic Seq2Seq model with the LSTM dimension is set to 180. Other parameters are set the same as for the TPG model.

Seq2Seq $_{90}$ + Scheduled Sampling: The basic Seq2Seq model with Scheduled Sampling of the LSTM dimension is set to 90. Other parameters are set the same as for theTPG model.

Seq2Seq $_{180}$ + Scheduled Sampling: The basic Seq2Seq model with Scheduled Sampling of the LSTM dimension is set to 180. Other parameters are set the same as for the TPG model.

TPG $_{\mathcal{M}_1}$: Proposed TPG model with λ is set to 500, results reported based on the intermediate model $\mathcal{M}_1$.

TPG: Proposed TPG model with λ is set to 500, results reported based on the final model $\mathcal{M}_2$.

Note that all deep learning based models are trained with the same parameter settings, or otherwise stated above. Models are trained with a training dataset, then, the best models are chosen which demonstrate the best performance based on the validation dataset, and experimental results are reported based on the test dataset.

Performance Evaluation

We compare our TPG model to the baselines listed above including the tradition mathematical model NWP, and machine learning based approaches. Table 5.1 shows a summary of experimental results based on two evaluation matrices. It clearly demonstrates that our TPG model outperforms all baselines for the three measurements under study.

First, Seq2Seq based models outperform traditional approaches including NWP and ARIMA. Seq2Seq based models utilise the LSTM encoder and decoder which effectively model the spatio-temporal dynamics. Seq2Seq $_{90}$ performs better than Seq2Seq $_{180}$ overall, but in particular for t2m and w10m.

Second, Table 5.1 also shows the Seq2Seq model trained with Scheduled Sampling performs better than Seq2Seq without sampling which shows that Scheduled Sampling improves STSF. Our proposed TPG sampling continues to improve the effectiveness of bridging the gap between training and inference.

Third, our proposed TPG model outperforms base Seq2Seq models. The test of significance in terms of both accuracy measures shows that TPG significantly improves Seq2Seq $_{90}$. As shown in Fig. 5.2, our proposed TPG model converges significantly faster than the basic Seq2Seq model. The red line represents the test loss of the Seq2Seq model, whilst the light blue line represents the test loss of model $\mathcal{M}_1$, and the dark blue line represents the test loss of model $\mathcal{M}_2$. Benefiting from shorter sequence lengths, model $\mathcal{M}_1$ quickly converges compared to the original Seq2Seq. The resulting the second stage of the training model $\mathcal{M}_2$, shows faster convergence as well as a better final performance.

To sum up, our proposed TPG model achieves the best overall performance. Specifically, the TPG model converges much faster during training due to the progressive growing training mechanism. Prediction examples also show that TPG model predictions

are more reliable for the outlier predictions and the long range predictions than the baseline approaches and Scheduled Sampling.

5.4 Moving MNIST Experiments

The performance of our proposed TPG model against the weather forecasting dataset shows that our model outperforms other models on a vector-like dataset. To further evaluate the usability and applicability of our proposed method on image-like spatio-temporal sequential datasets, we conduct further experiments on the Moving MNIST dataset [Srivastava et al., 2015].

5.4.1 Dataset

The Moving MNIST dataset was originally created for evaluating video (a series of sequential images) prediction performance; it has since become one of the most common spatio-temporal sequence prediction benchmarks [Shi et al., 2015, Wang et al., 2017, 2018c]. We generate a series of image sequences containing two moving handwritten digits, moving at a different speed and velocity. In addition to the typical setup, we extend the sequence length to 60, that is, 30 for the input sequence and 30 for prediction. We generate 10,000 sequences for training, 3,000 for validation and 5,000 for testing. Experimental results are reported based on the test dataset.

5.4.2 Implementation

Seq2Seq

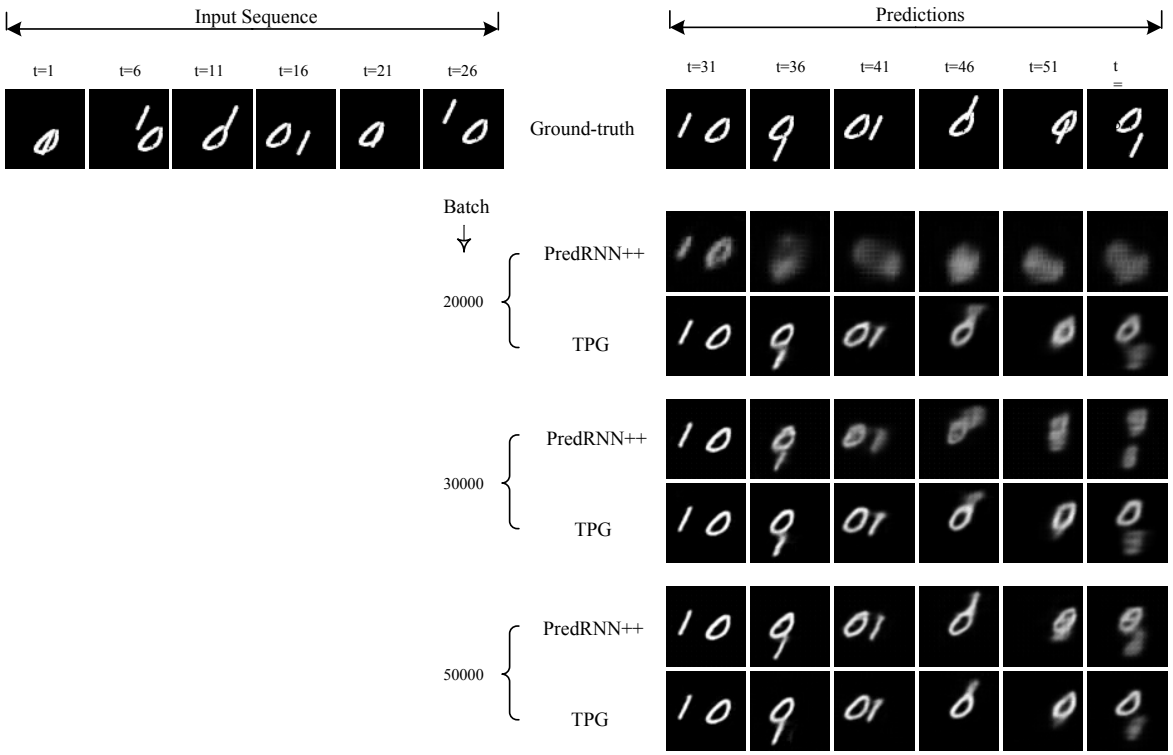
We use open source code[2] provided by PredRNN++ [Wang et al., 2018c] for our base Seq2Seq model for images. PredRNN++ is a strong baseline for video frame prediction which is based on Seq2Seq with a custom RNN cell, called Casual LSTM and a GHU unit.

TPG

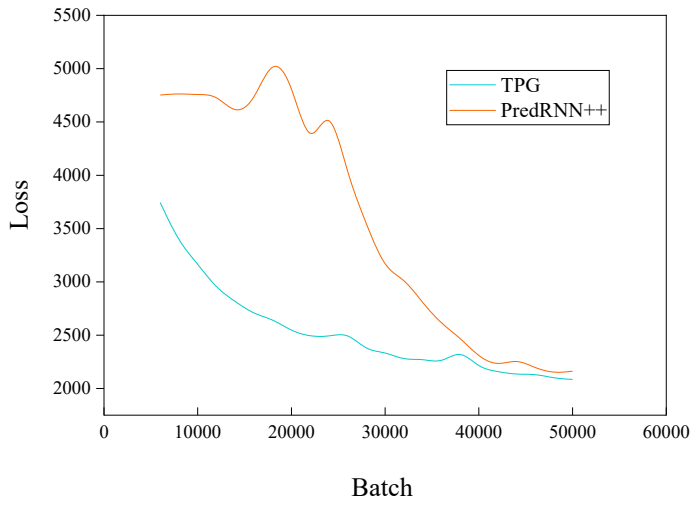
Then we extend the baseline Seq2Seq model with our proposed TPG model. We set $\lambda = 3,000$ and we train the model for 50,000 iterations.

[2] https://github.com/Yunbo426/predrnn-pp

5.4.3 Overall Evaluation



(a) Moving MNIST prediction visualisations during training.



(b) Test set loss during training

Fig. 5.4 A visualisation of training progress: (5.4b) shows our proposed TPG converges faster during training; (5.4a) shows our proposed TPG trains long range predictions faster and provides more accurate long range predictions.

	SSIM ↑	MSE ↓
ConvLSTM [Shi et al., 2015]	0.597	156.2
TrajGRU [Shi et al., 2017]	0.588	163.0
CDNA [Finn et al., 2016]	0.609	142.3
DFN [De Brabandere et al., 2016]	0.601	149.5
VPN [Kalchbrenner et al., 2017]	0.620	129.6
PredRNN [Wang et al., 2017]	0.645	112.2
PredRNN++ without sampling	0.733	91.10
PredRNN++ with SS [Wang et al., 2018c]	0.769	87.74
TPG	**0.811**	**85.41**

Table 5.2 Experimental results of 30 time step predictions: results reported per frame. Baselines reported as in PredRNN++ [Wang et al., 2018c].

Evaluation Matrix

We report results based on two matrices: per-frame Structural Similarity Index Measure (SSIM) [Wang et al., 2004] and MSE. The larger SSIM matrix scores indicate greater similarities between the ground-truth and prediction whilst the smaller MSE matrix values indicate smaller prediction errors.

Baselines

We compare the performance of our approach against several baseline methods as in PredRNN++ [Wang et al., 2018c] including ConvLSTM [Shi et al., 2015], TrajGRU [Shi et al., 2017], CDNA [Finn et al., 2016], DFN [De Brabandere et al., 2016], VPN [Kalchbrenner et al., 2017] and PredRNN [Wang et al., 2017].

Performance Evaluation

Table 5.2 shows a summary of experimental results in regard to the two evaluation matrices. It clearly demonstrates that our TPG model outperforms all baselines reported in PredRNN++ [Wang et al., 2018c]. Specifically, TPG achieves the best SSIM score of 0.811 and MSE score of 85.41 per frame. Moreover, Fig. 5.4 shows the training progress. From Fig. 5.4b, we can see our proposed TPG converges much faster and more smoothly than other baselines. Also from the prediction visualisations of Fig. 5.4a we can see our proposed TPG model provides more accurate long term predictions. Specifically, at batch

20,000, moving hand-written digits start to be recognisable at time step 41, whereas, predictions via PredRNN++ are still blurry. This could be because with the curriculum of model $\mathcal{M}_1$, $\mathcal{M}_2$ can learn longer ranges as well as higher order dynamics, whereas, with the regular Seq2Seq model, a later time step has to wait for the earlier time step to converge, due to the nature of RNNs.

5.5 Summary

In this chapter, we studied the discrepancy between the training and inference of RNN models, and the problems of adapting scheduled sampling to close the gap between training and inference for STSF. We propose a novel sampling strategy called TPG that sample the target sequence from midway outputs from intermediate models trained with longer timescales through a carefully designed decaying strategy. Experimental results demonstrate that our proposed method better models long term dependencies and outperforms baseline approaches on two different datasets.

Chapter 6

MPL-GAN: Towards Realistic Meteorological Predictive Learning Using Conditional GAN

Meteorological imagery prediction is an important and challenging problem for weather forecasting. It can also be seen as a video frame prediction problem that estimates future frames based on observed meteorological imagery. Despite it being a widely-investigated problem, it is still far from being solved. Current state-of-the-art deep learning based approaches mainly optimise the mean square error loss, resulting in blurry predictions. In this chapter, we address this problem by introducing a MPL-GAN model that utilises the conditional GAN along with the predictive learning module to handle the uncertainty in future frame prediction. Experiments on a real-world dataset demonstrate the superior performance of our proposed model. Our proposed model is able to map the blurry predictions produced by traditional mean square error loss based predictive learning methods back to their original data distributions, hence, it is able to improve and sharpen the prediction. In particular, our MPL-GAN model achieves an average sharpness of 52.82, which is 14% better than the baseline method. Furthermore, our model correctly detects the meteorological movement patterns that traditional unconditional GANs fail to do.

6.1 Introduction

Weather forecasting is one of the main applications of meteorological prediction. It is important for our daily life as well as industrial and agricultural production. Common

uses include precipitation nowcasting [Shi et al., 2015, 2017], streamflow prediction [Fu et al., 2020, Hadi et al., 2019, Yaseen et al., 2018, 2019a,b], wind speed simulation [Bokde et al., 2019], radiation estimation [Hai et al., 2020], and temperature forecasting [Liu and Lee, 2020, Wang et al., 2019a]. Numerous techniques have been proposed to more accurately predict weather measurements including Numerical Weather Prediction (NWP), radar map based methods, and satellite imagery based methods.

Recently, with the advances of deep learning techniques, researchers have adopted RNN based methods to improve the traditional approaches to address this challenging problem. For example, Shi et al. [2015] formulated the precipitation nowcasting problem into a spatio-temporal sequence forecasting model, and proposed an LSTM-based model called ConvLSTM for radar echo map prediction. A Seq2Seq-LSTM based model [Wang et al., 2019a] was proposed to improve NWP performance through historical observations. A study by Rüttgers et al. [2019] proposed an adversarial model to predict cyclone trajectories from satellite imagery. These studies reveal that radar and satellite imagery play more and more important roles in meteorological prediction, not only because they are more robust, but they also provide end-users with more sequential information and better visualisations of the history from the current to the predicted atmosphere. However, these approaches share some common drawbacks: they do not generalise well in real-world meteorological datasets, especially for long-term predictions. To be more specific, the pioneering work, ConvLSTM, produces blurry radar image predictions which get worse as the time step moves forward. These meteorological images do not appear to be realistic but are blurry, resulting in unpleasant visualisations. These drawbacks are mainly due to two reasons. First, these models optimise Euclidean losses such as MAE and MSE across the overall length of sequential meteorological images. A few studies introduced various models with MAE and MSE, but produced blurry images [Mathieu et al., 2016, Pathak et al., 2016]. This is mainly due to the assumption that the data is drawn from the Gaussian distribution which only works on a continuous portion of the image while ignoring isolated small regional areas. Second, due to the nature of RNN architecture, small errors quickly accumulate to become large errors along the generated sequence, because of the gap between training and inference [Bengio et al., 2015, Venkatraman et al., 2015]. These two issues indicate that it is crucial to include an uncertainty handling procedure in order to generate realistic meteorological predictions.

Meanwhile, the video frame prediction can be modelled as a spatio-temporal sequence forecasting problem. For instance, Wang et al. [2017] extracted a sequence of images from video frames, and proposed an encoder-decoder RNN based model called PredRNN [Wang et al., 2017] and its improved version PredRNN++ [Wang et al., 2018c]. However, these models suffer from the same drawback as ConvLSTM and produce blurry predictions. Recently, *Generative Adversarial Network* (GAN) [Goodfellow et al., 2014] was used to handle uncertainties in video frame prediction [Clark et al., 2019, Mathieu et al., 2016, Saito and Saito, 2018, Wang et al., 2018b]. GAN models match two distributions using one generator and one discriminator playing the minmax game, where the generator learns to generate samples to fool the discriminator and the discriminator learns to distinguish these fake samples. These unconditional GAN based models are able to produce realistic looking videos by learning a high dimensional distribution of complex datasets. However, these models are not suitable for meteorological predictive learning even though they are able to produce realistic looking and temporally coherent video frames. This is because these generated video frames do not model the real-world meteorological changes given by the source of meteorological image frames. Note that meteorological prediction needs to consider the moving entities' (pixel wise) direction, speed, rotation acceleration and other relevant information.

To sum up, on one hand RNN based meteorological predictive models with MAE and MSE produce blurry predictions. On the other hand, GAN based models are able to generate realistic looking video frames but fail to catch the actual atmospheric movement as they miss local variations and patterns. In this work, we propose a Conditional GAN based model called Meteorological Predictive Learning GAN (in short MPL-GAN) that optimises both the regression loss and GAN loss, and aims to generate realistic meteorological predictions. By optimising regression loss, we aim to model real-world atmosphere imagery movement which is crucial for weather forecasting. The GAN loss is used to estimate the data distribution to deal with the uncertainty to produce non-blurry predictions.

Our main contributions are summarised as follows:

- To the best of our knowledge, this is the first model that combines regression loss with GAN loss to generate realistic meteorological predictions that provide better visualisations than existing models.

- We conduct extensive experiments on a real-world radar imagery dataset. Experimental results demonstrate our model generates non-blurry predictions even in the long term, while it catches real-world atmosphere changes.
- We provide an extensive experiment analysis to show that the pure GAN model without the predictive learning module fails to catch the actual atmospheric movement, which demonstrates the effectiveness of our proposed MPL-GAN model in detecting meteorological changes.

The rest of the chapter is organised as follows. Our proposed Meteorological Predictive Learning GAN model is introduced in Section 6.2 and the details of our model are discussed, whilst Section 6.3 presents experimental results and the main findings are discussed.

6.2 Proposed MPL-GAN

In this section, we will describe our proposed MPL-GAN model that aims to produce realistic looking meteorological imagery. Figure 6.1 shows the overall architecture of our proposed model that contains a predictive learning model to generate predictions and a Conditional GAN module to map those predictions back to photo-realistic distributions. First, we define meteorological predictive learning as follows:

Definition 2. Given a length-k matrix sequence $\mathcal{M}_1^k \equiv [\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k]$, where each matrix $\mathcal{X}_t \in \mathcal{M}$ represents the meteorological imagery at time-step t. Meteorological predictive learning is used to predict a sequence of corresponding meteorological imagery of following K time-steps based on the past frames of $\mathcal{M}$, denoted as $\hat{\mathcal{P}}_1^K \equiv \left[\hat{\mathcal{X}}_1, \hat{\mathcal{X}}_2, \ldots, \hat{\mathcal{X}}_K\right]$.

Note, meteorological imagery carries important weather information such as rainfall, temperature, and wind speed etc.

6.2.1 Predictive Learning

To model meteorological changes, we adopt encoder-decoder ConvLSTM [Shi et al., 2015] as a predictive learning module. As investigated by previous studies [Shi et al., 2015,

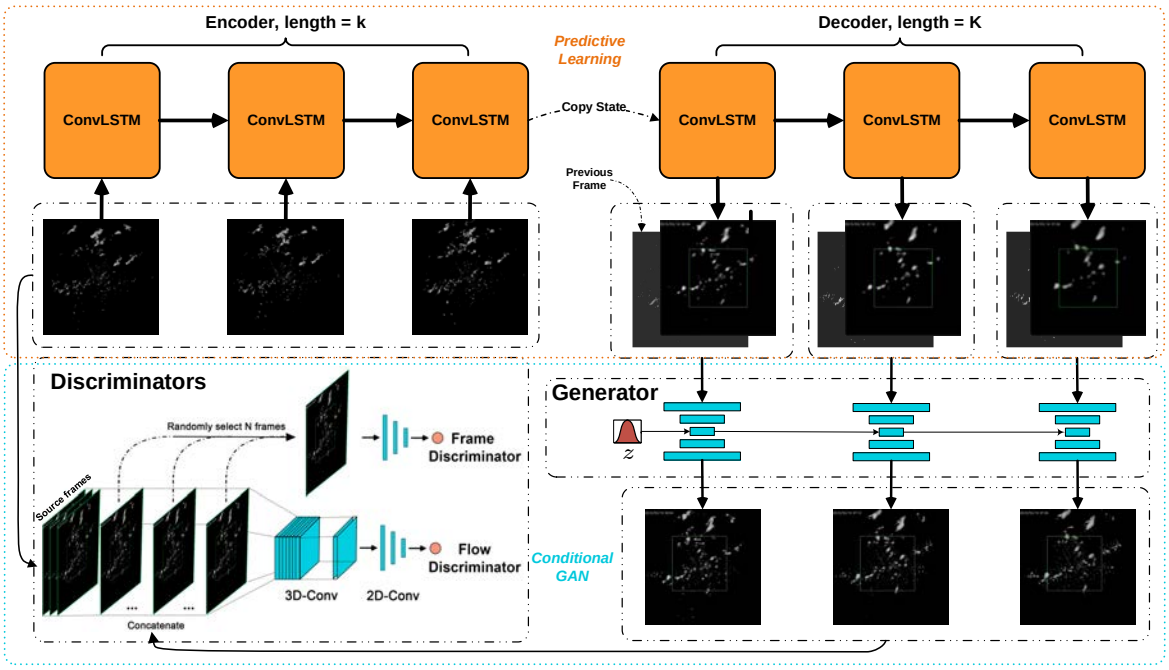


Fig. 6.1 Architecture of our proposed model MPL-GAN. The orange boxes indicates the predictive learning modules that model the real-world meteorological movement patterns. We use ConvLSTM for evaluation purposes, but that can be replaced by other predictive learning approaches if needed. The blue box concludes the Conditional GAN module which consists of the conditional generator and two discriminators $\mathcal{D}_{\mathrm{Fr}}$ and $\mathcal{D}_{\mathrm{Fl}}$.

2017], predictive learning aims to capture the local spatio-temporal pattern movement such as rotation and scaling. As mentioned earlier, none of the existing GAN-based next frame prediction models are suitable for meteorological predictive learning as these models do not capture the real-world meteorological changes. Furthermore, these GAN models without a predictive learning module are not able to produce long-term predictions. For example, a study by [Mathieu et al., 2016] could only produce a maximum of two frames for future video predictions. In contrast, our MPL-GAN model generates the next prediction conditional on previous ground-truth and current predictive output with conditional GAN. With the help of a predictive learning module we managed to generate more than 10 frames of non-blurry and realistic meteorological imagery predictions, and yet still model the real-world atmospheric changes. This demonstrates that the predictive learning module is crucial for modelling meteorological changes. Note that, we use ConvLSTM for evaluation purposes in this study, but it can be replaced by any other advanced model such as TrajGRU [Shi et al., 2017] and PredRNN++ [Wang et al., 2018c].

On one hand, the predictive learning module is required for modelling meteorological changes, but on the other hand, naive predictive learning models suffer from the blurry image issue and they need to be specially refined for meteorological change analysis. In the next section, we will introduce the Conditional GAN model to solve the blurry problem caused by the traditional naive predictive learning module.

6.2.2 Conditional GAN

A GAN [Goodfellow et al., 2014] attempts to learn a mapping function $\mathcal{G}$ to map a random noise vector z to an image $\mathcal{X}$, $\mathcal{G} : z \rightarrow \mathcal{X}$. In our settings, we aim to map the blurry prediction produced by ConvLSTM to the original non-blurry distribution. Let $\hat{\mathcal{P}}_1^K$ denote the generated sequence of ConvLSTM, $\mathcal{M}_1^K$ denote the observed ground-truth frames, and our goal is to train a conditional Generator $\mathcal{G} : \{z, \hat{\mathcal{P}}_1^K\} \rightarrow \mathcal{M}_1^K$.

Conditional Generator

As the prediction sequence is generated recursively by the ConvLSTM, we train the Conditional Generator $\mathcal{G} : \{z, \hat{\mathcal{X}}_t\} \rightarrow \mathcal{X}_t$ per frame along with the ConvLSTM time steps, instead of taking the whole generated sequence to train the GAN generator, where $\hat{\mathcal{X}}_t$ denotes the ConvLSTM output at time t and $\mathcal{X}_t$ is the ground-truth frame at time t. However, when the ConvLSTM prediction gets more blurry in the later time steps, it is harder for the Conditional GAN to map the conditional distributions between the two. To solve this problem, we train the generator along with conditioning the previous frame $\mathcal{X}_{t-1}$, i.e. $\mathcal{G} : \{z, \mathcal{X}_{t-1}, \hat{\mathcal{X}}_t\} \rightarrow \mathcal{X}_t$. Ideally, we should train the generator conditioning on the current ground-truth frame $\mathcal{X}_t$ and current ConvLSTM prediction frame $\hat{\mathcal{X}}_t$. We use the previous frame instead of the current frame based on the observation that two consecutive meteorological frames are very similar in terms of data distribution. They even look very similar since the atmosphere normally changes gradually and slowly. Moreover, during the inference stage, none of the previous and current ground-truth frames would be available. Then, we can replace the previous ground-truth frame $\mathcal{X}_{t-1}$ with the generator output of the previous time step during the inference phase, i.e. $\tilde{\mathcal{X}}_t = \mathcal{G}(z, \tilde{\mathcal{X}}_{t-1}, \hat{\mathcal{X}}_t)$. We can also think the other way around, since the ground-truth frames are not available during the inference phase. However, we make an assumption that the output distribution of our generator perfectly matches the actual data distribution, then

the data distribution can be carried forward recursively from the last known ground-truth frame to the predicting frame. Therefore, we use the ground-truth frame $\mathcal{X}_{t-1}$ instead of $\tilde{\mathcal{X}}_{t-1}$ during the training phase for training stability, then replace $\mathcal{X}_{t-1}$ with $\tilde{\mathcal{X}}_{t-1}$ during the inference phase.

Frame Discriminator

We randomly select N frames from amongst the K time steps to train the frame discriminator $\mathcal{D}_{\text{Fr}}$. The $\mathcal{D}_{\text{bFr}}$ outputs *1* for the true frame $\mathcal{X}_t$ and outputs *0* for the fake frame $\tilde{\mathcal{X}}_t$. Then, we train the frame discriminator by optimising the minmax game defined in the original GAN. We use Hinge Loss [Lim and Ye, 2017] for $\mathcal{L}_{\mathcal{D}_{\text{Fr}}}$ defined as follows:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_{\text{Fr}}} = \mathbb{E}[\max(0, 1 - \mathcal{D}_{\text{Fr}}(\mathcal{X}_t)) \\ + \mathbb{E}[\max(0, 1 + \mathcal{D}_{\text{Fr}}(\tilde{\mathcal{X}}_t))].\end{aligned} \tag{6.1}$$

Flow Discriminator

A frame discriminator aims to ensure the generator produces samples matching the actual data distribution, i.e. to ensure the samples produced look realistic. In addition, similar to the video discriminator proposed by [Clark et al., 2019], we propose a flow discriminator $\mathcal{D}_{\text{Fl}}$ to ensure the generator produces temporal coherent frames. Similarly, $\mathcal{D}_{\text{Fl}}$ outputs *1* for the real sequence $\mathcal{M}_1^K$ and outputs *0* for the generator sequence $(\mathcal{M}_1^k; \tilde{\mathcal{P}}_k^K)$, then we concatenate the initial source sequence $\mathcal{M}_1^k$ and the conditional generated sequence $\tilde{\mathcal{P}}_k^K$. $\mathcal{L}_{\mathcal{D}_{\text{Fl}}}$ is defined as follows:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_{\text{Fl}}} = \mathbb{E}[\max(0, 1 - \mathcal{D}_{\text{Fl}}(\mathcal{M}_1^K)) \\ + \mathbb{E}[\max(0, 1 + \mathcal{D}_{\text{Fl}}((\mathcal{M}_1^k, \tilde{\mathcal{P}}_k^K)))].\end{aligned} \tag{6.2}$$

6.2.3 Training

Again, a predictive learning module is essential for modelling the real-world meteorological movement patterns, and the conditional GAN is used to map the blurry predictions generated by predictive learning back to non-blurry image distributions. Therefore, we divide the training process into two stages. First, we start training the predictive learning module, and when that is almost stable, then we start the training of the GAN module.

Predictive Learning Training

Based on the settings of ConvLSTM [Shi et al., 2015] and TrajGRU [Wang et al., 2017], we train our Predictive Learning (PL) module by minimising the balance of MSE and MAE losses (B-MSE-MAE) with Stochastic Gradient Descent (SGD) and Back-propagation Through Time (BPTT) [Werbos, 1990]. We train the B-MSE-MAE loss until it becomes stable before we start training the Conditional GAN, so that the Conditional GAN learns the stable data distribution. However, we continue to train the PL module along with the GAN module even if the loss does not decrease. The logic behind this is to give the GAN variances of distribution to make the GAN more robust.

Conditional GAN Training

Training GAN models requires training both the generator and discriminator by optimising the minimax game [Goodfellow et al., 2014], to the point where the generator learns to fool the discriminator with generated fake samples and the discriminator learns to identify true and fake samples. We follow in the same spirit and extend this to training one conditional generator and two separate discriminators. The losses of Frame Discriminator $\mathcal{D}_{\text{Fr}}$ and Flow Discriminator are defined in Equation 6.1 and Equation 6.2. Now we define the loss function for conditional generator as follows:

$$\mathcal{L}_{\mathcal{G}} = -\mathbb{E}[\mathcal{D}_{\text{Fr}}(\mathcal{G}(z, \mathcal{X}_{t-1}, \hat{\mathcal{X}}_t)) + \mathcal{D}_{\text{Fl}}(\mathcal{G}^{\phi}(z, \mathcal{X}_{t-1}, \hat{\mathcal{X}}_t))], \tag{6.3}$$

where ϕ denotes a process of applying the generator recursively with the ConvLSTM time steps to generate a sequence flow of frames. Therefore, our overall optimisation goal is to minimise $\mathcal{L}_{\mathcal{D}_{\text{Fr}}}$ and $\mathcal{L}_{\mathcal{D}_{\text{Fl}}}$ which maximises the probability of the discriminators identifying fake frames and fake sequences; and minimises $\mathcal{L}_{\mathcal{G}}$ to maximise the probability of the generator producing samples that the discriminators think are true.

$$\min_{\mathcal{G}} \max_{\mathcal{D}_{\text{Fr}}} \mathcal{L}_{\mathcal{D}_{\text{Fr}}}(\mathcal{G}, \mathcal{D}_{\text{Fr}}) + \min_{\mathcal{G}} \max_{\mathcal{D}_{\text{Fl}}} \mathcal{L}_{\mathcal{D}_{\text{Fl}}}(\mathcal{G}, \mathcal{D}_{\text{Fl}}). \tag{6.4}$$

Note, N random frames will be selected for training $\mathcal{D}_{\text{Fr}}$ N times for each training batch b, whereas $\mathcal{D}_{\text{Fl}}$ will be trained once for each b. Moreover, the gradient of $\mathcal{G}$ will be back-propagated multiple times recursively with operation ϕ when training $\mathcal{G}$ with $\mathcal{D}_{\text{Fl}}$. This makes it extremely difficult to train the GAN. Following the principle of [Wang

et al., 2018b] and [Karras et al., 2018], we down-sample each frame of the sequence to overcome the training difficulties.

6.3 Experiment

In this section, we briefly describe the dataset used and provide experimental results.

6.3.1 Dataset

We use the HKO-7 [Shi et al., 2015, 2017] radar echo imagery dataset to evaluate our proposed MPL-GAN model. The radar echo imagery is recorded every 6 minutes, therefore, there are 240 frames per day. Each frame contains 480×480 pixels that cover a 512 km $\times$ 512 km area. We sample data into sequences of the length of 15 frames by a sliding window, 5 for the encoder and 10 for the decoder. From the total number of 993 days of data, we randomly select 80% for the training set, 5% for the validation set, and 15% for the testing set. Unlike the original experimental settings of ConvLSTM and TrajGRU, where they try to predict the pixel value and report the precipitation prediction based on that, we focus on imagery frame prediction that is realistic for better visualisation.

6.3.2 Implementation and Parameters

Predictive Learning

We use ConvLSTM as the predictive learning module. We implement a three layer ConvLSTM encoder-decoder with the following parameters for each layer: $[3 \times 3 - 64, 3 \times 3 - 192, 3 \times 3 - 192]$.

Conditional GAN

Training the GAN is challenging, thus we carefully choose our architecture for the generator and discriminators. For the generator, our architecture is somewhat similar to the PG-GAN [Karras et al., 2018]. To match the resolution of the generated samples, we up-sample the original resolution from 480×480 to 512×512. We randomly select $N = 2$ frames to train the Frame Discriminator $\mathcal{D}_{\text{Fr}}$. The 3D-Conv of Flow Discriminator consists of three layers set to the following parameters: $[3 \times 3 \times 3 - 64, 3 \times 3 \times 3 - 16, 3 \times 3 \times 3 - 1]$.

We implement our model using PyTorch 1.4, a well known deep learning library developed by Facebook. Our model is trained and evaluated on a server with a Nvidia V100 32GB GPU and an Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz (24 cores). We train our model using the Adam optimiser [Kingma and Ba, 2014] with a learning rate of $1e^{-4}$ for the ConvLSTM and $1e^{-3}$ for the generator and discriminators. Batch size is set to 2 due to the resource consumption of the conditional GAN model. For all models including baseline methods, we train 100,000 batches and select the best model based on the minimum MSE against the validation set. For the MPL-GAN model, we train the PL module for 10,000 batches before training the conditional GAN. All experimental results are reported based on the test dataset.

6.3.3 Overall Evaluation

Baselines

To evaluate the effectiveness of our proposed model, we compare our model to two baseline methods:

- PL with MSE. To show that the PL with MSE produces blurry predictions, we compare our model to the pure ConvLSTM without the conditional GAN module [Shi et al., 2015].
- PG-GAN. We extend the PG-GAN [Karras et al., 2018] from image generation to sequence generation with the same architecture of our conditional GAN module which has a Frame discriminator and a Flow discriminator. This is also very similar to the DVD-GAN [Clark et al., 2019].

Model	Sharpness ↑
PL with MSE [Shi et al., 2015]	46.48
PG-GAN [Karras et al., 2018]	51.27
MPL-GAN (ours)	52.82

Table 6.1 Experimental results based on the test data, averaged by 10 prediction time steps (a larger number indicates sharper prediction imagery).

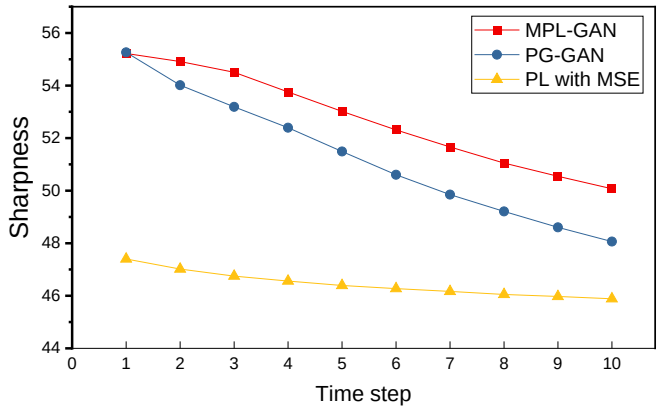


Fig. 6.2 Sharpness of 10 prediction time steps based on the test dataset.

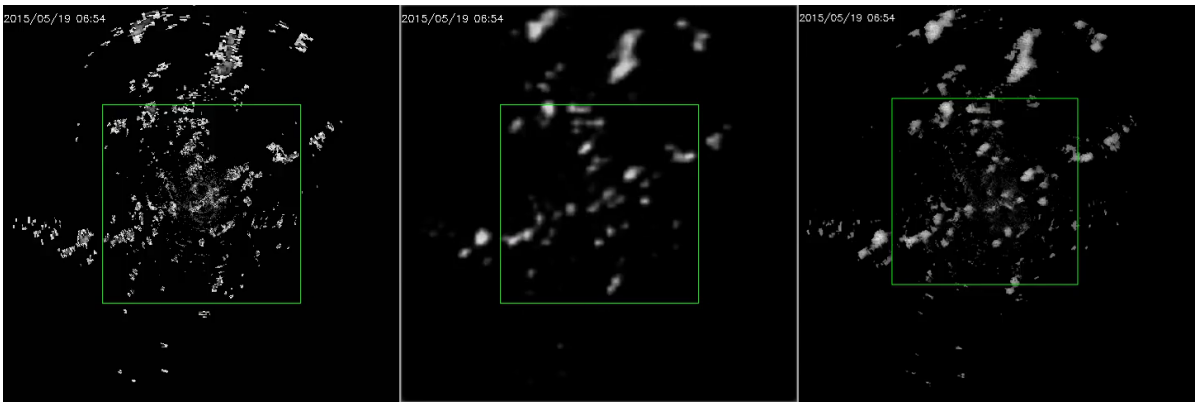


Fig. 6.3 Future meteorological prediction samples. *Left*: ground truth. *Middle*: PL with MSE. *Right*: MPL-GAN (ours). *Click to view the animations with Adobe Acrobat Reader.*

Evaluation Matrix

We use the sharpness measure based on the gradient of two images defined in [Mathieu et al., 2016] as follows:

$$\text{Sharp.} = 10\log_{10}\frac{\max_{\tilde{X}}^2}{\frac{1}{N}\left(\sum_i\sum_j\left|(\nabla_i X+\nabla_j X)-(\nabla_i\tilde{X}+\nabla_j\tilde{X})\right|\right)}, \tag{6.5}$$

where, X is the ground-truth frame and $\tilde{X}$ is the output frame; $\nabla_i X = \left|X_{i,j} - X_{i-1,j}\right|$ and $\nabla_j X = \left|X_{i,j} - X_{i,j-1}\right|$; $\max_{\tilde{X}}$ is the maximum possible value of the image intensities. We report the average sharpness of the test dataset across K frames in the table as well as the individual frame evaluations as a line chart as shown in Figure 6.2.

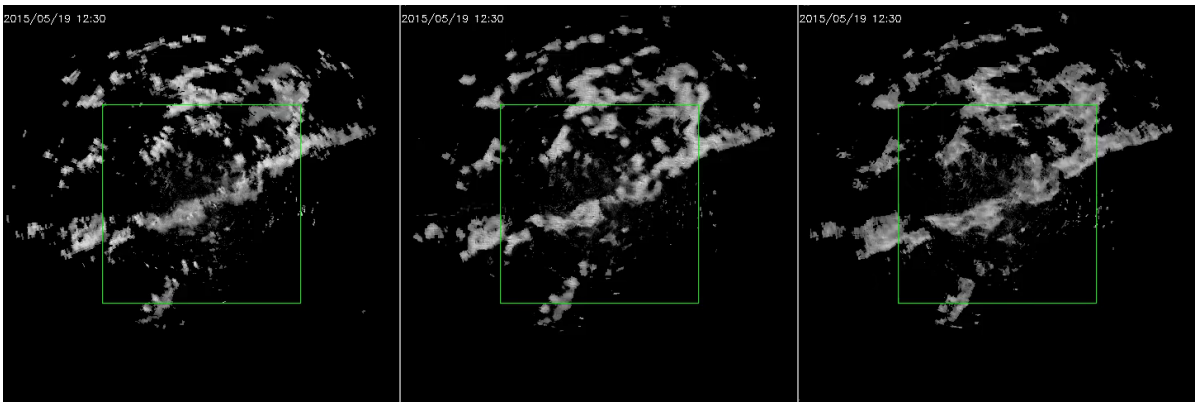


Fig. 6.4 Meteorological prediction movement of our MPL-GAN and PG-GAN. *Left*: ground truth. *Middle*: MPL-GAN (ours). *Right*: PG-GAN. *Click to view the animations with Adobe Acrobat Reader.*

Experiment Analysis

We conduct both quantitative and qualitative evaluations against two baseline approaches. As shown in Table 6.1, our proposed MPL-GAN model achieves an overall average sharpness of 52.82; PG-GAN achieves a similar result of 51.27; whereas PL with MSE achieves only 46.48. This shows that GAN based models are able to generate sharper meteorological imagery predictions compared to PL with MSE. This is because GAN models can handle uncertainties of future frames. A per frame quantitative comparison in Figure 6.2 indicates that GAN based models not only beat PL with MSE in the average score but also in the long-term predictions. MPL-GAN and PG-GAN achieve a similar score in the first frame, however, PG-GAN drops performance quickly in the long term.

Besides quantitative evaluation, we visualise a sample prediction sequence in Figure 6.3 (please view the animation by clicking the figure using Adobe Acrobat Reader). As shown in the animation, both PL with MSE and MPL-GAN catch the real-world meteorological movement patterns. However, PL with MSE generates blurry predictions which continue to get more blurry over time, especially in the long term. Whereas, our proposed MPL-GAN model continues to generate realistic looking and sharp predictions. Furthermore, if we look at small regions of the prediction frames, PL with MSE tends to omit small areas as a result of MSE loss, whereas, our model has more regional details due to the incorporation of uncertainty handling in the GAN.

Furthermore, the whole experiment aims to find out whether the GAN model is able to solve the blurry prediction problem caused by PL with MSE. In fact, we can see our proposed model as PL with MSE plus the advanced version of PG-GAN that has two discriminator heads. As stated previously, GAN based models are able to produce sharper predictions compared to PL with MSE. However, as samples in Figure 6.4 show, PG-GAN is not able to model the meteorological movements. More specifically, the first generated frame of PG-GAN looks very similar to the first generated frame of our model MPL-GAN, but the later frames are just an expansion of the first frame which is obviously not the real-world scenario. On the other hand, with the constraint of the predictive learning module, our proposed model MPL-GAN continues to generate realistic looking and diverse meteorological frames that catch the real-world meteorological movement pattern. We summarise the findings above in Table 6.2.

Model	Sharpness	Learns Met. Patt.
PL with MSE	Blurry	√
PG-GAN	Sharp	×
MPL-GAN (ours)	Sharp	√

Table 6.2 Performance comparison.

In summary, the quality of meteorological imagery prediction is of crucial importance in weather forecasting, and in monitoring climate change. Figure 6.3 clearly depicts that MPL-GAN produces a quality prediction result identifying both global trends and local variations, whilst the imagery produced by PL with MSE is too blurry and coarse and missing local variations and details. Hence, PL with MSE is less useful in practice since it misses many localised patterns, and results in inaccurate predictions, however, MPL-GAN is practically useful for forecasting weather and monitoring local and global climate change as evidenced in Figure 6.3 and Figure 6.4.

6.4 Summary

In this chapter, we studied the blurry problem of meteorological imagery prediction, and proposed a novel GAN based model called MPL-GAN to deal with the blurry problem. Our model utilised a conditional GAN to map the blurry predictions back to their original non-blurry data distributions. Both quantitive and qualitative analysis demonstrate that our model is able to produce sharper and more realistic-looking meteorological imagery predictions than baseline methods. Quantitive visualisations also showed that our proposed model is able to catch the real-world meteorological movement patterns with the constraint of the predictive learning module.

Chapter 7

Conclusion

7.1 Summary of Contribution

In this thesis, we studied the difficulties of predictive spatio-temporal modelling and proposed several models to overcome these challenges.

The main contributions are summarised as follows:

- First, we introduced a Spatio-Temporal GRU to model ST correlations and variations for ST trajectory classification. We introduced a novel approach using a 1D-CNN in the segmented convolutional weight mechanism to model short-term spatial correlations between neighbouring locations, along with the time gating mechanism for ST correlations within RNN cells. The experimental results demonstrated the effectiveness of the Spatio-Temporal GRU for ST correlations, and proved the superior performance of our proposed model over existing state-of-the-art approaches. We also provided ablation studies to demonstrate the applicability and flexibility of the Spatio-Temporal GRU, and presented a visualisation of 'car' transportation mode from the Geolife dataset to illustrate the ability of the proposed temporal gate. The temporal gate is able to filter out some input information which could confuse or complicate the model. Last but not least, we conducted next-location experiments on two real-world user check-in datasets. These experiments demonstrated the effectiveness of the proposed ST-GRU with time-gating mechanism to model long-term and short-term check-in interests.

- Second, we proposed Temporal Progressive Growing Sampling to bridge the gap between training and inference phases for Seq2Seq based STSF systems. This was

achieved by transforming the training process from a fully-supervised approach which utilises all available previous ground-truth to a less-supervised approach which replaces the ground-truth contexts with generated predictions. We also sampled the target sequence from midway outputs from the intermediate model trained with longer timescales with a carefully designed decaying strategy. The experiments on two datasets demonstrated that our proposed method achieves superior performance compared to all baseline methods, and also exhibits fast convergence speed as well as better long-term accuracy. Two different types of datasets used in experiments prove the novelty and applicability of our proposed method.

- Third, we proposed the MPL-GAN model to solve the blurry prediction problem of predictive learning methods such as ConvLSTM and TrajGRU. We utilised a conditional GAN to handle this problem by mapping the blurry predictions generated by predictive learning methods back to their original non-blurry data distributions. To do that, we recursively applied a conditional generator conditioning on the previous output of itself and the current output of the predictive learning module. Through the novel design of Frame Discriminator and Flow Discriminator, the generator learns to produce temporally coherent and realistic frames. The experiments on a real-world radar echo dataset demonstrated that our proposed MPL-GAN model not only produces sharp and realistic looking meteorological predictions, but also models the real-world meteorological movement patterns with the constraint of the predictive learning module.

7.2 Future work

7.2.1 Continuation of the Work of this Thesis

Even though we have achieved some positive results, there are still some limitations we have not yet explored due to the time limitation. We summarise some potential directions to extend the current work.

- **ST-GRU.** Three possible directions for future studies are as follows. First, an extensive ablation study along with a hyper-parameter optimisation study is required

to further validate the applicability and utility of the Spatio-Temporal GRU model. Second, experimental tests with a variety of spatio-temporal trajectories for diverse classification problems are required to fully verify the effectiveness of the Spatio-Temporal GRU. model Third, an investigation into different convolutions, to systematically model both spatial proximity and temporal proximity, would further enhance the current Spatio-Temporal GRU model.

- **TPG.** Two possible directions for future studies are as follows. First, an extensive ablation study along with a hyper-parameter optimisation study is required to further validate the applicability and utility of the TPG model. Second, we can extend the progressive idea to spatial and temporal dimensions simultaneously for spatio-temporal sequence predictive learning.

- **MPL-GAN.** Although our model is able to generate non-blurry predictions, there is room to improve the prediction accuracy. Since the GAN model brings uncertainties, improving the sharpness of prediction, but with deteriorating accuracy, our future work will investigate this problem, and evaluate our proposed model using various real-world datasets.

7.2.2 Unsupervised Pre-training

BERT is the most successful language representation model that pushes the limits of several natural language tasks including question answering and language inference [Devlin et al., 2018]. The magic happens in the Transformer architecture and the powerful unsupervised pre-training technique. There is a lot of unlabelled text data all over the internet, and the BERT model takes advantage of these unlabelled data to build deep bidirectional representations using unsupervised tasks like Masked LM or Next Sentence Prediction (NSP). As mentioned in the introduction, a humongous amount of spatio-temporal data is being captured and stored everyday. Most of the captured data is unlabelled due to technical limitations or privacy restriction. If an unsupervised pre-training technique like BERT can take take advantage of such huge amounts of unlabelled data to build up the underlying representation of the data, and then fine-tune the model for supervised predictive tasks would gain significant benefit.

References

Didi brain. https://www.didiglobal.com/science/brain. (Accessed on 05/21/2020).

Ossama Abdel-Hamid, Abdel-Rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 22(10):1533–1545, October 2014. ISSN 2329-9290. doi: 10.1109/TASLP.2014.2339736. URL http://dx.doi.org/10.1109/TASLP.2014.2339736.

Peder Bacher, Henrik Madsen, and Henrik Aalborg Nielsen. Online short-term solar power forecasting. *Solar energy*, 83(10):1772–1783, 2009.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1171–1179, Cambridge, MA, USA, 2015. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969239.2969370.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553380. URL http://doi.acm.org/10.1145/1553374.1553380.

Luke Bermingham and Ickjai Lee. A probabilistic stop and move classifier for noisy gps trajectories. *Data Mining and Knowledge Discovery*, 32(6):1634–1662, Nov 2018.

Neeraj Dhanraj Bokde, Andrés Feijóo, Nadhir Al-Ansari, and Zaher Mundher Yaseen. A comparison between reconstruction methods for generation of synthetic time series applied to wind speed simulation. *IEEE Access*, 7:135386–135398, 2019.

Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*, 2014.

George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., San Francisco, CA, USA, 1990. ISBN 0816211043.

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=B1xsqj09Fm.

Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In Tomás Pajdla and Jiří Matas, editors, *Computer Vision - ECCV 2004*, pages 25–36, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24673-2.

Cen Chen, Kenli Li, Sin G Teo, Guizi Chen, Xiaofeng Zou, Xulei Yang, Ramaseshan C Vijay, Jiashi Feng, and Zeng Zeng. Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 893–898, Nov 2018. doi: 10.1109/ICDM.2018.00107.

Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *Twenty-Third international joint conference on Artificial Intelligence*, 2013.

P Cheung and HY Yeung. Application of optical-flow technique to significant convection nowcast for terminal areas in hong kong. In *The 3rd WMO International Symposium on Nowcasting and Very Short-Range Forecasting (WSN12)*, pages 6–10, 2012.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv e-prints*, art. arXiv:1406.1078, June 2014.

Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial Video Generation on Complex Datasets. *arXiv e-prints*, art. arXiv:1907.06571, Jul 2019.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12(null):2493–2537, November 2011. ISSN 1532-4435.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3): 273–297, September 1995. ISSN 0885-6125. doi: 10.1023/A:1022627411411. URL https://doi.org/10.1023/A:1022627411411.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 667–675, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9. URL http://dl.acm.org/citation.cfm?id=3157096.3157171.

Zhongwei Deng and Minhe Ji. Spatiotemporal structure of taxi services in shanghai: Using exploratory spatial data analysis. In *2011 19th International Conference on Geoinformatics*, pages 1–5, June 2011.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

PN Druzhkov and VD Kustikova. A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26(1):9–15, Jan 2016. ISSN 1555-6212. doi: 10.1134/S1054661816010065. URL https://doi.org/10.1134/S1054661816010065.

Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Jotaro Ikedo. Classifying spatial trajectories using representation learning. *International Journal of Data Science and Analytics*, 2(3-4):107–117, 2016a.

Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Akihisa Kawanobe. Deep feature extraction from trajectories for transportation mode estimation. In James Bailey, Latifur Khan, Takashi Washio, Gill Dobbie, Joshua Zhexue Huang, and Ruili Wang, editors, *Advances in Knowledge Discovery and Data Mining*, pages 54–66, Cham, 2016b. Springer International Publishing. ISBN 978-3-319-31750-2.

Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Soft+hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection. *Neural networks*, 108:466–478, 2018.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 64–72, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9. URL http://dl.acm.org/citation.cfm?id=3157096.3157104.

Seth R. Flaxman. *Machine learning in space and time*. PhD thesis, 2015.

Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

Minglei Fu, Tingchao Fan, Zi'ang Ding, Sinan Q Salih, Nadhir Al-Ansari, and Zaher Mundher Yaseen. Deep learning data-intelligence model based on adjusted forecasting window scale: Application in daily streamflow simulation. *IEEE Access*, 8:32632–32651, 2020.

Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. In *Twenty-ninth AAAI conference on Artificial Intelligence*, 2015.

Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. Identifying human mobility via trajectory embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1689–1695, 2017. doi: 10.24963/ijcai.2017/234. URL https://doi.org/10.24963/ijcai.2017/234.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/gehring17a.html.

Amir Ghaderi, Borhan M Sanandaji, and Faezeh Ghaderi. Deep forecast: Deep learning-based spatio-temporal forecasting. In *The 34th International Conference on Machine Learning (ICML), Time series Workshop*, 2017.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012. ISBN 978-3-642-24796-5. doi: 10.1007/978-3-642-24797-2. URL https://doi.org/10.1007/978-3-642-24797-2.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013. doi: 10.1109/ICASSP.2013.6638947.

Sinan Jasim Hadi, Sani Isah Abba, Saad Sh Sammen, Sinan Q Salih, Nadhir Al-Ansari, and Zaher Mundher Yaseen. Non-linear input variable selection approach integrated with non-tuned data intelligence model for streamflow pattern simulation. *IEEE Access*, 7: 141533–141548, 2019.

Tao Hai, Ahmad Sharafati, Achite Mohammed, Sinan Q Salih, Ravinesh C Deo, Nadhir Al-Ansari, and Zaher Mundher Yaseen. Global solar radiation estimation and climatic variability analysis using extreme learning machine based predictive model. *IEEE Access*, 8:12026–12042, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, July 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL http://dx.doi.org/10.1016/0893-6080(89)90020-8.

Rich H Inman, Hugo TC Pedro, and Carlos FM Coimbra. Solar forecasting methods for renewable energy integration. *Progress in energy and combustion science*, 39(6): 535–576, 2013.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1771–1779. JMLR.org, 2017. URL http://dl.acm.org/citation.cfm?id=3305381.3305564.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk99zCeAb.

Alireza Khotanzad and C Chung. Application of multi-layer perceptron neural networks to vision problems. *Neural Computing & Applications*, 7(3):249–259, Sep 1998. ISSN 1433-3058. doi: 10.1007/BF01414886. URL https://doi.org/10.1007/BF01414886.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1181. URL http://aclweb.org/anthology/D14-1181.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, art. arXiv:1412.6980, December 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. Wiley-IEEE Press, 2001. ISBN 9780470544037. doi: 10.1109/9780470544037.ch14. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5264952.

Dejiang Kong and Fei Wu. Hst-lstm: A hierarchical spatial-temporal long-short term memory network for location prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2341–2347. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/324. URL https://doi.org/10.24963/ijcai.2018/324.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541.

Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. Traclass: Trajectory classification using hierarchical region-based and trajectory-based clustering. *Proc. VLDB Endow.*, 1 (1):1081–1094, August 2008. ISSN 2150-8097. doi: 10.14778/1453856.1453972. URL http://dx.doi.org/10.14778/1453856.1453972.

Po-Ruey Lei, Tsu-Jou Shen, Wen-Chih Peng, and Jiunn Su. Exploring spatial-temporal trajectory model for location prediction. In *2011 IEEE 12th International Conference on Mobile Data Management*, volume 1, pages 58–67, June 2011. doi: 10.1109/MDM.2011.61.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJiHXGWAZ.

Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3428–3434. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/476. URL https://doi.org/10.24963/ijcai.2018/476.

Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv e-prints*, art. arXiv:1705.02894, May 2017.

Hong-Bin Liu and Ickjai Lee. End-to-end trajectory transportation mode classification using bi-lstm recurrent neural network. In *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pages 1–5, Nov 2017. doi: 10.1109/ISKE.2017.8258799.

Hong-Bin Liu and Ickjai Lee. Bridging the gap between training and inference for spatio-temporal forecasting. In *Proceedings of the Twenty-Fourth European Conference on Artificial Intelligence*, ECAI'20. IOS Press, 2020.

Hongbin Liu, Hao Wu, Weiwei Sun, and Ickjai Lee. Spatio-temporal GRU for trajectory classification. In Jianyong Wang, Kyuseok Shim, and Xindong Wu, editors, *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pages 1228–1233. IEEE, 2019.

Peter Lynch. The origins of computer weather prediction and climate modeling. *J. Comput. Phys.*, 227(7):3431–3444, March 2008. ISSN 0021-9991. doi: 10.1016/j.jcp.2007.02.034. URL http://dx.doi.org/10.1016/j.jcp.2007.02.034.

Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.05440.

Danny McCarroll. *Simple Statistical Tests for Geography*. Chapman and Hall/CRC Press, 2017.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–646, 2009.

Razvan Pascanu, Tomas Mikolov, and Yoshua 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, pages III–1310–III–1318. JMLR.org, 2013. URL http://dl.acm.org/citation.cfm?id=3042817.3043083.

Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

Yanjun Qin, Haiyong Luo, Fang Zhao, Chenxing Wang, Jiaqi Wang, and Yuexia Zhang. Toward transportation mode recognition using deep convolutional and long short-term memory recurrent neural networks. *IEEE Access*, 7:142353–142367, 2019.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06434.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06732.

Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks*, 6(2):1–27, February 2010.

Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–58, 2016.

Kira Rehfeld, Norbert Marwan, Jobst Heitzig, and Jürgen Kurths. Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics*, 18(3):389–404, 2011.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

Mario Rüttgers, Sangseung Lee, Soohwan Jeon, and Donghyun You. Prediction of a typhoon track using a generative adversarial network and satellite images. *Scientific Reports*, 9(1):6057, 2019. doi: 10.1038/s41598-019-42339-y. URL https://doi.org/10.1038/s41598-019-42339-y.

Masaki Saito and Shunta Saito. TGANv2: Efficient Training of Large Models for Video Generation with Multiple Subsampling Layers. *arXiv e-prints*, art. arXiv:1811.09245, Nov 2018.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *arXiv e-prints*, April 2017.

Nicholas I Sapankevych and Ravi Sankar. Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, 4(2):24–38, May 2009. ISSN 1556-603X. doi: 10.1109/MCI.2009.932254.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. doi: 10.1016/j.neunet.2014.09.003. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997. ISSN 1053-587X. doi: 10.1109/78.650093. URL http://dx.doi.org/10.1109/78.650093.

Rahul C. Shah, Chieh-yih Wan, Hong Lu, and Lama Nachman. Classifying the mode of transportation on mobile phones using gis information. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, page 225–229, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329682. doi: 10.1145/2632048.2632109. URL https://doi.org/10.1145/2632048.2632109.

Rajiv Shah and Rob Romijnders. Applying Deep Learning to Basketball Trajectories. *ArXiv e-prints*, August 2016.

Xingjian Shi and Dit-Yan Yeung. Machine Learning for Spatiotemporal Sequence Forecasting: A Survey. *arXiv e-prints*, art. arXiv:1808.06865, Aug 2018.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 802–810, Cambridge, MA, USA, 2015. MIT Press.

Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Deep learning for precipitation nowcasting: A benchmark and a new model. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*,

pages 5617–5627. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7145-deep-learning-for-precipitation-nowcasting-a-benchmark-and-a-new-model.pdf.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 843–852. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045209.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Waldo R Tobler. A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46:234–240, 1970. ISSN 00130095, 19448287. URL http://www.jstor.org/stable/143141.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing motion and content for video generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1526–1535, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 3024–3030. AAAI Press, 2015. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=2888116.2888137.

Bin Wang, Jie Lu, Zheng Yan, Huaishao Luo, Tianrui Li, Yu Zheng, and Guangquan Zhang. Deep uncertainty quantification: A machine learning approach for weather forecasting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, pages 2087–2095, New York, NY, USA, 2019a. ACM. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330704. URL http://doi.acm.org/10.1145/3292500.3330704.

Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. When will you arrive? estimating travel time based on deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2018a. URL https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16657.

Senzhang Wang, Jiannong Cao, and Philip S Yu. Deep learning for spatio-temporal data mining: A survey. *arXiv preprint arXiv:1906.04928*, 2019b.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1144–1156. Curran Associates, Inc., 2018b. URL http://papers.nips.cc/paper/7391-video-to-video-synthesis.pdf.

Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 25–34, 2014.

Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 879–888. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6689-predrnn-recurrent-neural-networks-for-predictive-learning-using-spatiotemporal-lstms.pdf.

Yunbo Wang, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and Philip S Yu. PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5123–5132, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018c. PMLR. URL http://proceedings.mlr.press/v80/wang18b.html.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004. ISSN 1057-7149. doi: 10.1109/TIP.2003.819861.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct 1990. ISSN 0018-9219. doi: 10.1109/5.58337.

Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. Modeling trajectories with recurrent neural networks. In *Proceedings of the Twenty-Sixth International*

Joint Conference on Artificial Intelligence, IJCAI-17, pages 3083–3090, 2017. doi: 10.24963/ijcai.2017/430. URL https://doi.org/10.24963/ijcai.2017/430.

Zhibin Xiao, Yang Wang, Kun Fu, and Fan Wu. Identifying Different Transportation Modes from Trajectory Data Using Tree-Based Ensemble Classifiers. *ISPRS International Journal of Geo-Information*, 6(3):57–22, March 2017.

Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, Jan 2015. ISSN 2168-2232. doi: 10.1109/TSMC.2014.2327053.

Guolei Yang, Ying Cai, and Chandan K Reddy. Spatio-temporal check-in time prediction with recurrent neural network based survival analysis. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2976–2983. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/413. URL https://doi.org/10.24963/ijcai.2018/413.

Zaher Yaseen, Minglei Fu, Chen Wang, Hanna wan mohtar, Ravinesh Deo, and Ahmed El-shafie. Application of the hybrid artificial neural network coupled with rolling mechanism and grey model algorithms for streamflow forecasting over multiple time horizons. *Water Resources Management*, 32, 01 2018. doi: 10.1007/s11269-018-1909-5.

Zaher Mundher Yaseen, Wan Hanna Melini Wan Mohtar, Ameen Mohammed Salih Ameen, Isa Ebtehaj, Siti Fatin Mohd Razali, Hossein Bonakdari, Sinan Q Salih, Nadhir Al-Ansari, and Shamsuddin Shahid. Implementation of univariate paradigm for streamflow simulation using hybrid data-driven model: Case study in tropical region. *IEEE Access*, 7:74471–74481, 2019a.

Zaher Mundher Yaseen, Sadeq Oleiwi Sulaiman, Ravinesh C. Deo, and Kwok-Wing Chau. An enhanced extreme learning machine model for river flow forecasting: State-of-the-art, practical applications in water resource engineering area and future research direction. *Journal of Hydrology*, 569:387 – 408, 2019b. ISSN 0022-1694.

Mao Ye, Peifeng Yin, and Wang-Chien Lee. Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 458–461, 2010.

Xiuwen Yi, Junbo Zhang, Zhaoyuan Wang, Tianrui Li, and Yu Zheng. Deep distributed fusion network for air quality prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 965–973, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3219822. URL http://doi.acm.org/10.1145/3219819.3219822.

Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. Lcars: a location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 221–229, 2013.

Rose Yu, Stephan Zheng, Anima Anandkumar, and Yisong Yue. Long-term Forecasting using Tensor-Train RNNs. *arXiv e-prints*, art. arXiv:1711.00073, Oct 2017.

Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, Xiuwen Yi, and Tianrui Li. Predicting city-wide crowd flows using deep spatio-temporal residual networks. *Artificial Intelligence*, 259:147 – 166, 2018. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2018.03.002. URL http://www.sciencedirect.com/science/article/pii/S0004370218300973.

Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S. Sheng, and Xiaofang Zhou. Where to go next: A spatio-temporal gated network for next poi recommendation. In *AAAI Conference on Artificial Intelligence*, 2019.

Yu Zheng. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.*, 6(3): 29:1–29:41, May 2015. ISSN 2157-6904. doi: 10.1145/2743025. URL http://doi.acm.org/10.1145/2743025.

Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, pages 312–321, New York, NY, USA, 2008a. ACM. ISBN 978-1-60558-136-1. doi: 10.1145/1409635.1409677. URL http://doi.acm.org/10.1145/1409635.1409677.

Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. Learning transportation mode from raw gps data for geographic application on the web. In *WWW 2008*. WWW 2008, April 2008b. URL https://www.microsoft.com/en-us/research/publication/learning-transportation-mode-from-raw-gps-data-for-geographic-application-on-the-web/.

Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800, 2009.

Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. Understanding transportation modes based on GPS data for web applications. *ACM Transactions on the Web*, 4(1):1–36, January 2010.

Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li. *Geolife GPS trajectory dataset - User Guide*, July 2011.

Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. What to do next: Modeling user behaviors by time-lstm. In *Proceedings of the Twenty-Sixth*

International Joint Conference on Artificial Intelligence, IJCAI-17, pages 3602–3608, 2017. doi: 10.24963/ijcai.2017/504. URL https://doi.org/10.24963/ijcai.2017/504.